Specification of Dynamic Probabilistic Secure Distributed Systems

Pierre Civit

Thèse préparée au LIP6
Sorbonne Université

École Doctorale Informatique, Télécommunications et Électronique
ED130 - EDITE de Paris

Équipe Networks and Performance Analysis (NPA)

Spécification des Systèmes Distribués Dynamiques Probabilistes Sécurisés

(Specification of Dynamic Probabilistic Secure Distributed Systems)

par Pierre Civit

Thèse de doctorat d'Informatique

- dirigée par Maria Potop-Butucaru
- présentée et soutenue publiquement le 15 décembre 2022
- devant un jury composé de

Damien VERGNAUD, Professeur, Sorbonne Université, Président du jury
Paul ATTIE, Professeur, Augusta University, Rapporteur
Stephan MERZ, Chercheur senior, INRIA Nancy, Rapporteur
Hagit ATTIYA, Professeure, Technion University, Examinatrice
Vincent GRAMOLI Professeur, University of Sydney, Examinateur
Rachid GUERRAOUI Professeur, EPFL, Examinateur
Achour MOSTEFAOUI Professeur, Nantes Université, Examinateur
Maria POTOP-BUTUCARU, Professeure, Sorbonne Université, Directrice de thèse

Spécification des Systèmes Distribués Dynamiques Probabilistes Sécurisés

(Specification of Dynamic Probabilistic Secure Distributed Systems)

par Pierre Civit

Résumé *(en français)*

Cette thèse propose un modèle hiérarchique naturel pour les systèmes distribués dynamiques probabilistes. Le modèle étend de manière intuitive les systèmes de transition d'états étiquetés capturant aussi simplement que possible l'intuition d'un objet se déplaçant d'un état à un autre. Le modèle comprend 3 ingrédients essentiels : (1) une opération de composition parallèle, notée $||$, permettant de représenter un nouvel objet $\mathcal{A}||\mathcal{B}$ issu de l'interaction entre deux objets $\mathcal{A}$ et $\mathcal{B}$, (2) une relation de préordre $\leq$, où $\mathcal{A} \leq \mathcal{B}$ signifie que l'objet $\mathcal{A}$ implémente l'objet $\mathcal{B}$ au sens d'une sémantique observationnelle, (3) la propriété de composabilité pour $\leq$, c'est-à-dire $\mathcal{A} \leq \mathcal{B} \Longrightarrow \mathcal{C}||\mathcal{A} \leq \mathcal{C}||\mathcal{B}$ ($\leq$ est une précongruence pour $||$), (4) une structure hiérarchique, c'est-à-dire qu'un système X, composé d'objets interagissant les uns avec les autres et pouvant rejoindre et quitter le système dynamiquement, est lui aussi un objet du modèle. De plus, la thèse discute des conditions nécessaires et suffisantes pour obtenir (5) La monotonicité (avec $\leq$) de la création/destruction dynamique d'objets, c'est-à-dire que si (i) $\mathcal{A} \leq \mathcal{B}$ et (ii) $X_{\mathcal{A}}$ et $X_{\mathcal{B}}$ ne diffèrent que par le fait que $X_{\mathcal{A}}$ crée et détruit dynamiquement l'objet $\mathcal{A}$ au lieu de créer et détruire dynamiquement l'objet $\mathcal{B}$ comme le fait $X_{\mathcal{B}}$, alors (iii) $X_{\mathcal{A}} \leq X_{\mathcal{B}}$. Un tel résultat permet une méthodologie modulaire de conception et de raffinement basée uniquement sur la notion de comportement observable de l'extérieur. Le modèle est décliné en plusieurs variantes : asynchrone, temporelle, bornée. Ces différentes variantes permettent de modéliser des systèmes distribués complexes faisant des hypothèses de synchronie ou d'adversaire borné, typiquement une "blockchain".

Mots-clés

Automates, Algorithmique distribuée, Verification Formelle, Systèmes Dynamiques

---

Abstract *(in english)*

This thesis proposes a natural hierarchical model for dynamic probabilistic distributed systems. The model extends in an intuitive way the labeled transition systems that best capture the intuition of an object moving from one state to another. The model consists of 3 essential ingredients : (1) a parallel composition operation, noted $||$, allowing to represent a new object $\mathcal{A}||\mathcal{B}$ resulting from the interaction between two objects $\mathcal{A}$ and $\mathcal{B}$, (2) a pre-order relation $\leq$, where $\mathcal{A} \leq \mathcal{B}$ means that the object $\mathcal{A}$ implements the object $\mathcal{B}$ in the sense of an observational semantics, (3) the composability property for $\leq$, that is $\mathcal{A} \leq \mathcal{B} \Longrightarrow \mathcal{C}||\mathcal{A} \leq \mathcal{C}||\mathcal{B}$ ($\leq$ is a precongruence for $||$), (4) a hierarchical structure, i.e. a system X, composed of objects interacting with each other and able to join and leave the system dynamically, is also an object of the model. Furthermore, the thesis discusses the necessary and sufficient conditions to obtain (5) the monotonicity (with $\leq$) of dynamic creation/destruction of objects, i.e., if (i) $\mathcal{A} \leq \mathcal{B}$ and (ii) $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ differ only by the fact that $X_{\mathcal{A}}$ dynamically creates and destroys the object $\mathcal{A}$ instead of dynamically creating and destroying the object $\mathcal{B}$ as $X_{\mathcal{B}}$ does, then (iii) $X_{\mathcal{A}} \leq X_{\mathcal{B}}$. Such a result allows a modular design and a refinement methodology based only on the notion of externally observable behavior. The model is declined in several variants : asynchronous, timed, and bounded. These different variants allow us to model complex distributed systems making assumptions of synchrony or bounded adversary, typically a "blockchain".

Keywords

Automata, Distributed Computing, Formal Verification, Dynamic Systems

Acknowledgements

This thesis would never have started without the guidance of several admirable mentors who reinforced my inclination for algorithmic logic. In this sense, I thank Jérôme and Éric for having worked to create a fulfilling environment around them so that students like me could discover the beauty of their discipline. I thank Vincent for having trusted me by welcoming me within the CSRG where I could benefit from his experience, as well as that of Seth who very quickly joined the project at the embryonic stage ; the latter becoming the first step of a long and fruitful collaboration. Thank you both. I cemented the idea of pursuing a thesis during my unforgettable experience in the famous "lab" led by Khaled. In addition to having learned a lot on a technical level, I found a wonderful friend, to whom I could continuously confide during these last 3 years. Thank you for everything.

During the 3 years of my thesis, I had the chance to work under Maria's supervision. Maria has been invariably available, with all the sensitivity, benevolence, intelligence, and humor that characterize her. I can't count the number of times I have been able to unblock myself by exposing my problems to her. Thank you for your high standards and your scientific rigor which pushed me to give the best of myself. Thank you for having trusted me and for having given me so much freedom. Thank you for encouraging me to go and meet other researchers (at their expense) to exchange and understand other points of view. Thank you for your unconditional support and encouragement that kept me afloat.

I would like to express my gratitude to the members of the Jury for their time and kindness. Thank you, Stephan and Damien, for agreeing, at a very early stage, to form my monitoring committee with all your affability, before accepting a second mandate. Thank you to Achour for hosting me in Nantes and for the countless discussions, sometimes technical, sometimes less technical, but always with your unchanging joviality. Thanks to Hagit for all those numerous and demanding discussion sessions, questioning the most technical and minor details always with interest. Thanks to Paul for accepting to undergo such treatment and for welcoming me with all his thoughtfulness. Your encouragement touched me a lot. Thank you to Vincent for this rich collaboration and for having invited me a second time to the CSRG, whose team I warmly greet. Thank you to Rachid for having welcomed me, not without infallible France, in his team, by giving me the benefit of all his empathy and all his experience, which were decisive elements at key moments. Thanks to all the young researchers of the DCL for integrating me among you. I am particularly grateful to Jovan and Manuel for making all those sessions in front of all those whiteboards so pleasant. More generally, I would like to thank all my co-authors for all these meetings, mainly made of elucidations, but which always end up converging as if by magic towards a result.

Among my partners, Jovan deserves a special mention since he had to endure more than anyone else the discomfort of having to listen patiently to my vague explanations of uncertain and shaky solutions to poorly posed problems before he could finally take charge of the situation. The interposed screens have only slightly lessened this punishment for a crime he did not commit. Thank you sincerely for making this expedition so great.

Thank you to all my fellow researchers for giving up your time to welcome me without counting the cost and always with enriching discussions. Let me thank Petr, Matthieu, Anca, Frédéric, Alessia, Corentin, Victor, and Ran. Wherever I went, I always felt at home.

I thank all those who have made LIP6 so friendly during this whole period through countless coffees,

meals, and corridor discussions inside and outside the Jussieu campus. A special thought to those who did not hesitate to lend me a hand to use the computer tools for administrative tasks, despite all my ill will.

Thank you to all my friends in Nancy, Toulouse, and Paris for all the joie de vivre that you bring to me on a daily basis, coordinating perfectly to share this terribly thankless task. I will never be grateful enough for all the happiness you give me.

I thank my family for their tenderness and unfailing support and especially my parents, Françoise and Thierry, for all they have done for me. Thank you both for your tireless dedication and for the role models you continue to be for me.

Finally, my deep and sincere gratitude goes to my life partner, Alexandra, for all her love and support during this whole adventure, which would have been simply impossible without her. The tumultuous journey usually promised turned out to be a sweet sunny walk in your company. Thank you for everything.

Publications

During my PhD thesis, my co-authors and myself have published the following contributions:

1. About Dynamic Automata:
   - Brief Announcement : Composable Dynamic Secure Emulation. P. Civit and M. Potop-butucaru. 34th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), 2022.
   - Dynamic Probabilistic Input Output Automata. P. Civit and M. Potop-butucaru. 36th International Symposium on Distributed Computing (DISC), 2022.
2. About Byzantine Distributed Tasks:
   - Polygraph: Accountable Byzantine Agreement. P. Civit, S. Gilbert and V. Gramoli. 41st IEEE International Conference on Distributed Computing Systems (ICDCS), 2021.
   - As easy as ABC: Optimal (A)ccountable (B)yzantine (C)onsensus is easy! P. Civit, S. Gilbert, V. Gramoli, R. Guerraoui, and J. Komatovic. 2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2022.

     (Best Paper Award)
   - Crime and Punishment in Distributed Byzantine Decision Tasks. P. Civit, S. Gilbert, V. Gramoli, R. Guerraoui, J. Komatovic and A. Seredinschi. 42nd IEEE International Conference on Distributed Computing Systems (ICDCS), 2022.

     (Best Paper Award)
   - Byzantine Consensus is $\Theta(n^2)$: The Dolev-Reischuk Bound is Tight even in Partial Synchrony! P. Civit, M. A. Dzulfikar, S. Gilbert, V. Gramoli, R. Guerraoui, J. Komatovic and M. Vidigueira. 36th International Symposium on Distributed Computing (DISC), 2022.

     (Best Student Paper Award)

This manuscript only focus on Dynamic Automata and does not cover Byzantine Distributed Tasks. However, a summary of the work on Byzantine Distributed Tasks can be found in appendix A.

Symbols

LTS	Labeled Transition System
Signature	A triple (in, out, int) of sets of input, output, and internal actions
IOA	Input Output Automata, a special kind of LTS equiped with a signature
Probabilistic transition	A triple (s, a, η) where s is a state, a is an action and η is a probability distribution over states, the final state being chosen according to η
PIOA	Probabilistic IOA, a generalization of IOA with probabilistic transitions
Dynamic signature	A mapping from states to signatures
SIOA	Signature IOA, a generalization of IOA with a dynamic signature
PSIOA	Probabilistic Signature IOA, an IOA that has both a dynamic signature and probabilistic transitions
Configuration	A finite set of PSIOA together with a local state for each PSIOA
PCA	Probabilistic Configuration Automata, a PSIOA equipped with a homomorphism from states to configurations which respects PSIOA creation and destruction, respects signatures, and respects probability distributions
$\mathcal{A}$	PSIOA with id $\mathcal{A}$
$Q_{\mathcal{A}}$	States of $\mathcal{A}$
$(Q_{\mathcal{A}}, \mathcal{F}_{Q_{\mathcal{A}}})$	State space of $\mathcal{A}$, $\mathcal{F}_{Q_{\mathcal{A}}}$ being a sigma-field over $Q_{\mathcal{A}}$
$\mathcal{P}rob(Q_{\mathcal{A}}, \mathcal{F}_{Q_{\mathcal{A}}})$	Set of probabilistic measures over $(Q_{\mathcal{A}}, \mathcal{F}_{Q_{\mathcal{A}}})$
$\bar{q}_{\mathcal{A}}$	Start state of $\mathcal{A}$
$sig(\mathcal{A})$	Signature of $\mathcal{A}$, maps each state to a signature
$\widehat{sig}(\mathcal{A})$	Signature of $\mathcal{A}$, maps each state to the union of actions of the signature $sig(\mathcal{A})$
$in(\mathcal{A})$	Input actions of $\mathcal{A}$, maps each state to a set of input actions
$out(\mathcal{A})$	Output actions of $\mathcal{A}$, maps each state to a set of output actions
$int(\mathcal{A})$	Internal actions of $\mathcal{A}$, maps each state to a set of internal actions
$ext(\mathcal{A})$	External actions of $\mathcal{A}$, maps each state $q \in Q_{\mathcal{A}}$ to the pair $(in(\mathcal{A})(q), out(\mathcal{A}))(q))$
$\widehat{ext}(\mathcal{A})$	External actions of $\mathcal{A}$, maps each state $q \in Q_{\mathcal{A}}$ to $in(\mathcal{A})(q) \cup out(\mathcal{A}))(q)$
$loc(\mathcal{A})$	Local actions of $\mathcal{A}$, maps each state $q \in Q_{\mathcal{A}}$ to the pair $(out(\mathcal{A}))(q), int(\mathcal{A}))$
$\widehat{loc}(\mathcal{A})$	Local actions of $\mathcal{A}$, maps each state $q \in Q_{\mathcal{A}}$ to $out(\mathcal{A}))(q) \cup int(\mathcal{A})$
$acts(\mathcal{A})$	Universal set of actions of $\mathcal{A}$, i.e. $\bigcup_{q \in Q_{\mathcal{A}}} \widehat{sig}(\mathcal{A})$
$D_{\mathcal{A}}$	Probabilistic transitions of $\mathcal{A}$ (a subset of $Q_{\mathcal{A}} \times acts(\mathcal{A}) \times \mathcal{P}rob(Q_{\mathcal{A}}, \mathcal{F}_{Q_{\mathcal{A}}})$)
$Steps(\mathcal{A})$	Steps of $\mathcal{A}$, i.e. triplet $(q, a, q') \in Q_{\mathcal{A}} \times acts(\mathcal{A}) \times Q_{\mathcal{A}}$ s.t. $\exists (q, a, \eta) \in D_{\mathcal{A}}$, $\eta(q') > 0$
$Frags(\mathcal{A})$	Execution fragments of $\mathcal{A}$, i.e. the set of alternating sequences $q^0 a^1 q^1 a^2 q^2 ...$ of states and actions, starting with a state and not finishing with an action, such that each $(q^{i-1}, a^i, q^i) \in Steps(\mathcal{A})$
$Frags^*(\mathcal{A})$	Finite execution fragments of $\mathcal{A}$
$Frags^\omega(\mathcal{A})$	Infinite execution fragments of $\mathcal{A}$
$Execs(\mathcal{A})$	Executions of $\mathcal{A}$, i.e. execution fragments starting with start state
$Execs^*(\mathcal{A})$	Finite executions of $\mathcal{A}$
$Execs^\omega(\mathcal{A})$	Infinite executions of $\mathcal{A}$

$trace_{\mathcal{A}}(\alpha)$	The trace of the execution α of automaton $\mathcal{A}$, i.e. the projection of α over its external actions only
$Traces(\mathcal{A})$	Traces of $\mathcal{A}$
$Traces^*(\mathcal{A})$	Finite traces of $\mathcal{A}$
$Traces^\omega(\mathcal{A})$	Infinite traces of $\mathcal{A}$
$Reachable(\mathcal{A})$	Reachable states of $\mathcal{A}$
C_α	The set of all executions with α as prefix (called the cone of α)
$fstate(\alpha)$	First state of execution α
$lstate(\alpha)$	Last state of finite execution α
$states(\alpha)$	Set of states that occur in the execution α
$actions(\alpha)$	Set of actions that occur in the execution α
$\restriction$	Projection for states, executions
$\|\|$	Parallel composition
$\times$	Cartesian product, also overloaded as the composition operator for signatures
$\otimes$	Product of measures, also overloaded as product of σ-algebra
Q_{conf}	Set of configurations
$auts(C)$	The PSIOA that occur in configuration C
$map(C)$	The mapping from each automata of $auts(C)$ to its current state in C
$sig(C)$	Signature of configuration C, i.e., the product of the current signatures of the PSIOA in C
$in(C)$	Input actions of configuration C, i.e., the input actions of $sig(C)$
$out(C)$	Output actions of configuration C, i.e., the output actions of $sig(C)$
$int(C)$	Internal actions of configuration C, i.e., the internal actions of $sig(C)$
$config(X)$	Maps each state q of PCA X to a configuration
$created(X)(q)(a)$	The PSIOA created by the execution of action a in state q of PCA X
$hidden\text{-}actions(X)$	Maps each state q of PCA X to the hidden actions of PCA X at state q ($hidden\text{-}actions(X) \subseteq out(config(X)(q))$)
$\eta_1 \overset{c}{\leftrightarrow} \eta_2$	c is a measure-preserving bijection between distributions η_1 and η_2
$\Phi[\mathcal{B}/\mathcal{A}]$	The set of PSIOA that results from replacing $\mathcal{A}$ by $\mathcal{B}$ in the set of PSIOA Φ
$C \lhd_{\mathcal{AB}} C'$	C and C' are the same configurations modulo $\mathcal{A}$ in C being replaced by $\mathcal{B}$ in C'
$X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$	$X_{\mathcal{A}}$, $X_{\mathcal{B}}$ corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$ i.e., $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ differ only by the fact that $X_{\mathcal{A}}$ dynamically creates and destroys the automaton $\mathcal{A}$ instead of dynamically creating and destroying the automaton $\mathcal{B}$ as $X_{\mathcal{B}}$ does
$X \setminus \{\mathcal{A}\}$	PCA X deprived of $\mathcal{A}$, i.e. , $\mathcal{A}$ is removed from every configuration of X
$qR_{conf}q'$	The states q and q' are mapped to the same configuration (q, q' can be states of different PCA's that are implicit)
$qR_{conf}^{\setminus\{\mathcal{A}\}}q'$	The states q and q' are mapped to the same configuration if we ignore $\mathcal{A}$
$qR_{strict}q'$	The states q and q' are mapped to the same components (configuration, hidden actions, created automata for each action) of their respective PCA's (may be the same or different)
$qR_{strict}^{\setminus\{\mathcal{A}\}}q'$	The states q and q' are mapped to the same components of their PCA's if we ignore $\mathcal{A}$
$pot\text{-}out(X)(\mathcal{A})(q)$	The (potential) output actions of $\mathcal{A}$ in $config(X)(q)$: the actual output actions if $\mathcal{A}$ is in the configuration, and the empty set otherwise
$\tilde{\mathcal{A}}^{sw}$	Simpleton wrapper of $\mathcal{A}$: A PCA that encapsulates $\mathcal{A}$ without changing any behavior
σ	Scheduler: entity that resolves pure non-determinism
ϵ_σ	Measure of probability on $Execs(\mathcal{A})$ generated by the scheduler σ

$env(\mathcal{A})$	Set of environments of $\mathcal{A}$
$f_{(.,.)}$	Insight function: a function parametrized by a pair $(\mathcal{E}, \mathcal{A})$ where $\mathcal{A}$ is an automaton (PCA or PSIOA) and $\mathcal{E} \in env(\mathcal{A})$, that captures the ability of the environment to infer information about the behavior of $\mathcal{A}$. The domain of $f_{(\mathcal{E},\mathcal{A})}$ is $Execs(\mathcal{E}\|\mathcal{A})$.
$f\text{-}dist_{(\mathcal{E},\mathcal{A})}(\sigma)$	Measure of probability on $f(Execs(\mathcal{E}\|\mathcal{A}))$ generated by scheduler σ for $\mathcal{E} \in env(\mathcal{A})$
$\sigma\, B^{\leq\varepsilon}_{(\mathcal{E},\mathcal{A},\mathcal{B}),f}\, \sigma'$	σ and σ' are balanced schedulers iff $\forall \zeta \in range(f_{(\mathcal{E},\mathcal{A})}) \cup range(f_{(\mathcal{E},\mathcal{B})})$, $\lvert f\text{-}dist_{(\mathcal{E},\mathcal{A})}(\sigma)(\zeta) - f\text{-}dist_{(\mathcal{E},\mathcal{B})}(\sigma')(\zeta)\rvert \leq \varepsilon$
$\leq_{\varepsilon}^{S,f}$	Implementation relation w.r.t. scheduler schema S, insight-function f, and approximation ε
$\mathcal{A} \leq_{\varepsilon}^{S,f} \mathcal{B}$	$\forall \mathcal{E} \in env(\mathcal{A}) \cap env(\mathcal{B})$, $\forall \sigma \in S(\mathcal{E}\|\mathcal{A})$, $\exists \sigma' \in S(\mathcal{E}\|\mathcal{B})$, $\sigma B^{\leq\varepsilon}_{(\mathcal{E},\mathcal{A},\mathcal{B}),f} \sigma'$
$proj_{(.,.)}$	$proj_{(.,.)}$ is an insight function such that for each automaton K, $\forall \mathcal{E} \in env(K)$, $\forall \alpha \in Execs(\mathcal{E}\|K)$, $proj_{(\mathcal{E},K)}(\alpha) = \alpha \upharpoonright \mathcal{E}$.That is, $proj_{(\mathcal{E},K)}$ produces the projection of an execution of $K\|\mathcal{E}$ onto the environment $\mathcal{E}$
$\mathtt{p}$	Abbreviation for $proj_{(.,.)}$
$\alpha \equiv^{cr}_{\mathcal{A}} \alpha'$	Executions α and α' of PCA X differ only on states and internal actions of the PSIOA $\mathcal{A}$, where $\mathcal{A}$ may occur in configurations of X.
S_o	The scheduler schema of creation-oblivious schedulers
$\leq_0^{S_o,\mathtt{p}}$	Exact implementation relation w.r.t. scheduler schema S_o and insight-function $\mathtt{p}$.
$\mathcal{A} \leq_0^{S_o,\mathtt{p}} \mathcal{B}$	$\forall \mathcal{E} \in env(\mathcal{A}) \cap env(\mathcal{B})$, $\forall \sigma \in S_o(\mathcal{E}\|\mathcal{A})$, $\exists \sigma' \in S_o(\mathcal{E}\|\mathcal{B})$, $\sigma B^{\leq 0}_{(\mathcal{E},\mathcal{A},\mathcal{B}),\mathtt{p}} \sigma'$

Contents

Abstract ii

Acknowledgements iii

Publications v

Symbols vii

1. Introduction 1
- 1.1. Complex Systems . . . 2
  - 1.1.1. Modeling . . . 2
  - 1.1.2. Randomization . . . 4
  - 1.1.3. Dynamicity . . . 4
- 1.2. Thesis Overview (Short Version) . . . 6
- 1.3. Thesis Overview (Long Version) . . . 8
  - 1.3.1. Chapter 2: Mathematical Preliminaries . . . 8
  - 1.3.2. Chapter 3: Dynamic Probabilistic Automata . . . 8
  - 1.3.3. Chapter 4: Road to monotonicity . . . 11
  - 1.3.4. Chapter 5: Extension with time . . . 13
  - 1.3.5. Chapter 6: Extension with simulation-based security . . . 13
  - 1.3.6. Summary . . . 15

2. Preliminaries 21
- 2.1. Sets and Functions . . . 22
  - 2.1.1. Sets and intervals . . . 22
  - 2.1.2. Functions . . . 22
- 2.2. Trajectories . . . 24
  - 2.2.1. Basics . . . 24
  - 2.2.2. Time shift, prefix, suffix, concatenation . . . 24
  - 2.2.3. Deteministic Trajectories . . . 24
  - 2.2.4. Trajectories sets composition and isomorphism . . . 25
- 2.3. Probability Theory . . . 26
  - 2.3.1. Set Algebra . . . 26
  - 2.3.2. Measure Spaces . . . 27
  - 2.3.3. Measurable Function . . . 27
  - 2.3.4. Product of Measure Spaces . . . 28
  - 2.3.5. Discrete probability space . . . 28
  - 2.3.6. Measure Spaces Isomorphism . . . 28
  - 2.3.7. Integration . . . 28
  - 2.3.8. Carathéodory's Extension Theorem . . . 29
- 2.4. Turing Machines . . . 30
  - 2.4.1. Deterministic Turing Machine . . . 30
  - 2.4.2. Probabilistic Turing Machine . . . 31

2.4.3. Encoding . . . 32
2.4.3.1. Encoded Sets . . . 32
2.4.3.2. Encoding of Turing Machines . . . 34
2.4.4. Decision . . . 34
2.5. Labeled Transition System (LTS) . . . 35
2.5.1. Simple Labeled Transition System . . . 35
2.5.2. I/O Automata . . . 35
2.5.3. PIOA . . . 36
2.5.4. DIOA . . . 36
2.5.5. TIOA . . . 36
2.5.6. DPIOA . . . 36
2.5.7. DPTIOA . . . 36
2.5.8. Overwiev of the I/O Automata framework . . . 36

3. Dynamic Probablistic I/O Automata 39
3.1. Probabilistic Signature Input/Output Automata (PSIOA) . . . 40
3.1.1. Action Signature . . . 40
3.1.2. PSIOA . . . 41
3.1.3. Local composition . . . 41
3.1.4. Hiding operator . . . 42
3.1.5. Action renaming . . . 43
3.2. Probabilistic Configuration Automata . . . 45
3.2.1. configuration . . . 45
3.2.2. probabilistic configuration automata (PCA) . . . 47
3.3. A Toy Example: Specification of a Dynamic Reliable Broadcast . . . 49
3.3.1. Modelisation of Byzantine Reliable Broadcast . . . 49
3.3.2. Static Consensus . . . 50
3.3.3. Dynamic Byzantine Reliable Broadcast . . . 52
3.4. Executions, reachable states, partially-compatible automata . . . 65
3.4.1. Executions, reachable states, traces . . . 65
3.4.2. PSIOA and PCA composition . . . 66
3.5. Toolkit for configurations & PCA closure under composition . . . 68
3.6. Scheduler, measure on executions, implementation . . . 74
3.6.1. General definition and probabilistic space $(Frags(\mathcal{A}), \mathcal{F}_{Frags(\mathcal{A})}, \epsilon_{\sigma,\mu})$ . . . 74
3.6.2. Implementation . . . 77
3.7. Introduction on PCA corresponding w.r.t. PSIOA $\mathcal{A}$, $\mathcal{B}$ to introduce monotonicity . . . 82
3.7.1. An informal statement of the theorem . . . 82
3.7.2. Naive correspondence between two PCA . . . 83
3.7.3. Conservatism: the additional assumption for relevant definition of correspondence w.r.t. $\mathcal{A}, \mathcal{B}$ . . . 84
3.7.4. Corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$ . . . 86
3.7.5. Creation-oblivious scheduler . . . 86
3.8. Summary . . . 88

4. Monotonicty of dynamic creation/destruction of PSIOA with implementation 89
4.1. Executions-matching . . . 92
4.1.1. PSIOA executions-matching and semantic equivalence . . . 92
4.1.2. PCA-matching execution . . . 104
4.1.3. Digest . . . 108

4.2. Projection . . . 109
4.2.1. Projection on Configurations . . . 109
4.2.2. $\mathcal{A}$-fairness assumption, motivated by our definition of PCA deprived of an internal PSIOA: $X \setminus \{\mathcal{A}\}$ . . . 113
4.2.3. $Y = X \setminus \{\mathcal{A}\}$ is a PCA if X is $\mathcal{A}$-fair . . . 115
4.3. Reconstruction . . . 119
4.3.1. Simpleton wrapper : $\tilde{A}^{sw}$ . . . 119
4.3.2. Partial-compatibility of $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})$ and $\tilde{\mathcal{A}}^{sw}$ . . . 119
4.3.3. Executions-matching from X to $(X \setminus \{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw}$ . . . 125
4.3.4. Composition and projection are commutative . . . 128
4.4. PCA corresponding w.r.t. PSIOA $\mathcal{A}$, $\mathcal{B}$ . . . 131
4.5. Top/Down corresponding classes . . . 138
4.5.1. Creation-oblivious scheduler . . . 138
4.5.2. Tools: proxy function, creation-explicitness, classes . . . 140
4.5.3. Homomorphism between simple classes . . . 141
4.5.4. Decomposition, pasting-friendly functions . . . 144
4.5.5. Creation oblivious scheduler applied to decomposition . . . 148
4.5.6. Monotonicity of implementation . . . 150
4.6. Task schedule . . . 154
4.6.1. Task-schedulers for PIOA . . . 154
4.6.2. Discussion on adaptation of task-structure in dynamic setting . . . 154
4.6.3. task-schedule for dynamic setting . . . 158
4.6.4. Why a task-scheduler is not creation-oblivious ? . . . 159
4.7. Monotonicity of Tenacious Task-Implementation . . . 160
4.7.1. schedule notations . . . 161
4.7.2. Tenacious Implementation . . . 162
4.7.3. sub-classes according to a task schedule . . . 162
4.7.4. Preparation before monotonicity . . . 170
4.7.5. Tenacious Implementation Monotonicity . . . 171
4.8. Summary . . . 176

5. Dynamic Probablistic Timed I/O Automata **177**
5.1. Probabilistic Timed Signature I/O Automata (PTSIOA) . . . 178
5.2. Probabilistic Timed Configuration Automata . . . 181
5.2.1. Measurable space on configurations . . . 181
5.2.2. Probabilistic Timed Configuration Automata . . . 183
5.3. Executions . . . 186
5.4. Global composition . . . 188
5.4.1. PTSIOA composition . . . 188
5.4.2. Configuration composition . . . 189
5.5. Measure over executions, scheduler, implementation . . . 191
5.5.1. Ring on Executions and Traces Definition . . . 191
5.5.2. Probability Measure . . . 196
5.5.3. Implementation . . . 197
5.6. Local Scheduler . . . 198
5.7. Summary . . . 200

6. Dynamic Secure Emulation **201**
6.1. Polynomial-Bounded automata: formalizing implementation by computational indistinguishability . . . 204
6.1.1. time-bounded automata . . . 204
6.1.2. Boundedness preservation by usual operators . . . 205
6.1.3. Bounded Scheduling . . . 207
6.1.4. Implementation . . . 208
6.2. Security Layer . . . 211
6.2.1. Structured dynamic I/O Automata . . . 211
6.2.2. Adversary for structured automata . . . 212
6.2.3. Dynamic Secure-Emulation . . . 213
6.3. Summary . . . 217

Conclusion **218**

Appendices **223**
A. Second Part of the thesis: Byzantine Distributed Tasks . . . 225

List of Figures **229**

Bibliography **231**

Chapter 1

Introduction

Overview

1.1. Complex Systems **2**
1.1.1. Modeling 2
1.1.2. Randomization 4
1.1.3. Dynamicity 4
1.2. Thesis Overview (Short Version) **6**
1.3. Thesis Overview (Long Version) **8**
1.3.1. Chapter 2: Mathematical Preliminaries 8
1.3.2. Chapter 3: Dynamic Probabilistic Automata 8
1.3.3. Chapter 4: Road to monotonicity 11
1.3.4. Chapter 5: Extension with time 13
1.3.5. Chapter 6: Extension with simulation-based security 13
1.3.6. Summary 15

1.1. Complex Systems

For reasons of performance and resilience, to meet the needs of different geographical locations, to allow the governance of multiple users without a trusted entity, etc., modern systems are often distributed, i.e. they consist of a number of physically distributed components that interact with each other. These applications are dynamic in several respects. They involve a dynamic set of users, the number of which is not known in advance, who may join and leave the system during its life cycle. These users will exchange messages in a network whose topology will change over time, usually due to mobile failures. During the execution, a part of the protocol can be downloaded and incorporated into computation on the fly. In the meantime, a distributed algorithm uses randomization and sophisticated cryptography (whose foundations lie down to a probabilistic computational model) to circumvent impossibility results and/or enable scalability for enormous groups of processes. The infrastructures executing these protocols follow instructions that sometimes cannot hold in less than 1 million lines of code, making their implementation particularly error-prone, while simple tests cannot cover the immense number of possibilities. Even in some simple toy examples, the (hyper)properties resulting from the interaction of a few components can be counter-intuitive. Therefore, it is desirable to have a mathematical model that is:

1. expressive enough to handle both dynamicity, randomization, and other ingredients that make distributed systems successful, such as (i) different forms of synchrony (ii) the notion of security against a polynomially bounded adversary (iii) non-sequential scheduling.
2. simple enough to be understood by a programmer with a background in distributed computing
3. based on already known and proven tools

Finally, to avoid overwhelming monolithic reasoning, a model must 4. allow for modular design, in a sense elaborated in the next subsections.

1.1.1. Modeling

In the introduction of his famous book "Communicating Sequential Processes" [Hoa85], Tony Hoare invites us to "think about objects in the world around us, which act and interact with us and with each other in accordance with some characteristic pattern of behaviour." Of course, there are infinite ways to formalize these objects. However, if we fix the expressiveness and simplicity of the frameworks, some of them are more practical than others.

What are the desirable properties of a formalism?

- (composition) Each pair $(\mathcal{A}, \mathcal{B})$ of entities of the model that interact with each other can be composed to obtain a new entity, noted $\mathcal{A}||\mathcal{B}$. Of course, this composition has to capture the interaction in an intuitive way. It is convenient if the composition operator is both associative and commutative and returns an entity of the same nature as $\mathcal{A}$ and $\mathcal{B}$.
- (implementation) We want to be able to express in the model that a concrete, potentially very sophisticated, object $\mathcal{A}$ is meeting the specification of an, usually simple, abstract object $\mathcal{B}$, noted $\mathcal{A} \leq \mathcal{B}$. This relationship is usually called implementation (a.k.a. realization or emulation) and is supposed to be a preorder (transitive and reflexive).
- (substitutability) Assume two objects $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively implement objects $\mathcal{B}_1$ and $\mathcal{B}_2$, i.e. $\mathcal{A}_1 \leq \mathcal{B}_1$ and $\mathcal{A}_2 \leq \mathcal{B}_2$. We want to be able to deduce that their interaction implements the interaction of $\mathcal{B}_1$ and $\mathcal{B}_2$, i.e. $\mathcal{A}_1||\mathcal{A}_2 \leq \mathcal{B}_1||\mathcal{B}_2$. Such a property allows modular reasoning, where (1) a sophisticated specification $\mathcal{B}$ can be decomposed into several sub-specifications $\mathcal{B}_1, ..., \mathcal{B}_n$ (2) we can find an implementation $\mathcal{A}_i$ of each sub-specification $\mathcal{B}_i$ and finally (3) all the concrete implementations can be composed all together to implement $\mathcal{B}$.

One of the most well-known implementation relationships for non-probabilistic distributed systems is trace inclusion. The idea is to characterize the specification of a system by its set of external behaviors. To illustrate this concept, Lamport describes the famous example of the biscuit machine [Lam]. The specification of a biscuit machine describes how the machine should behave, e.g. when a coin is inserted and a button is pressed, a cookie is delivered. Such a description does not specify how the machine checks the value of the coin. A natural tool of description is the notion of trace, where a trace of an execution is the projection onto its external actions. Typically, if a concrete machine executes a very sophisticated algorithm to check the validity of the coin, the details of this run are hidden by the trace operator that just says that (i) a coin has been inserted, (ii) a cookie has been delivered and (iii) the change has been given. We do not know if the machine has measured the mass or the dimensions of the coin. In addition, since this specification allows (ii) the cookie to be delivered and (iii) the change to be given in any order, a correct implementation could always (iii) deliver the cookie first, then (ii) give change. Hence, a natural definition of implementation is trace inclusion. A concrete object $\mathcal{A}_1$ is said to refine another abstract object $\mathcal{A}_2$, noted $\mathcal{A}_1 \leq_T \mathcal{A}_2$, if $\mathit{Traces}(\mathcal{A}_1) \subseteq \mathit{Traces}(\mathcal{A}_2)$, with $\mathit{Traces}(\mathcal{A})$ representing the set of traces of an automaton.

Let us note that the non-determinism about the order of different actions like (ii) delivering the cookie and (iii) giving the change is at the core of distributed computing. This non-determinism is sometimes called *pure non-determinism* (to underline the difference with the *probabilistic non-determinism*, a.k.a. randomization). Pure non-determinism captures our ignorance about the relative speeds of different entities to perform their respective computation. Most distributed systems are designed following the guideline that no matter how the pure non-determinism is resolved, the system does what it is supposed to do. Thus, the refinement relationship $\leq_T$ can be interpreted as: "for any way of resolving the pure non-determinism in the real world, there exists a way of resolving the pure non-determinism in the ideal world, such that the external behaviors are the same."

A safety property that says nothing bad ever happens (e.g. the machine never gives more change than it should) can be seen as a set of traces (a.k.a. trace property). The refinement relationship $\leq_T$ ensures the preservation of trace properties with composition: $\mathcal{A}_1 \leq_T \mathcal{A}_2 \implies \mathcal{A}_1||\mathcal{B} \leq_T \mathcal{A}_2||\mathcal{B}$. This is very good news: if the machine is composed of different sub-modules, we can delegate the concrete implementation of these sub-modules to different teams and finally merge them to obtain an implementation of the entire specification, without questioning the validity of this composition.

Among the various approaches to model systems, Lamport acknowledges two major branches which, for lack of better names, he calls the Hatfields (Hs) and the McCoys (Ms) [1] [Lam].

(Hs): CSP [Hoa85], process algebra such as CCS [Mil80], ...

(Ms): Temporal logic [Lam94], Unity [CM88], I/O automata [LT87], ...

Roughly speaking, the Hatfields follow the tradition of structured operational semantics, which first formally describe how the individual steps of a computation occur, before proving the desired properties from derivations of logical statements. On their side, the McCoys follow the tradition of denotational semantics, formalizing the meanings of distributed systems by constructing mathematical objects (called denotations) that describe the meanings of what is a programm or what a program does. Lamport confesses that he "does not really know what the fundamental difference between the Hatfields and McCoys is" and thinks that "the two approaches are good for abstracting different aspects of a real system". In addition, some strong links have already been exhibited (e.g. [Vaa91, Seg93, DS95]).

Extending previous automata formalisms [Tut87, Seg95b, KLSV06, Mit07, AL16, CCK+07], this thesis is clearly a descendant of McCoy's family, where a specification consists of an abstract program written

1. The Hatfield–McCoy feud involved two rural American families in the 19th century. The quarrel has entered the lexicon of American folklore as a metonymy for any rival party that argues bitterly.

in some form of abstract programming language. "An obvious advantage of specifying a system as an abstract program is that while few programmers are familiar with temporal logic, they are all familiar with programs" [Lam93]. A priori, this approach does not have a precisely defined language, with formal semantics, and proof rules, but some works filled the gap [Mül98, GL00, BGL02, LKLM05, ALL+06, ALL+08].

1.1.2. Randomization

The resolution of pure non-determinism can be conceptually delegated to an abstract entity called the *scheduler*. This scheduler gives so much trouble to computer scientists that it is often called a demon. Randomization is an incredibly powerful way of mitigating the power of this demoniac entity: it allows circumventing impossibility results [FLP85, BO83], helps to increase the scalability of a system [GKM+19, CGMV18, CKS20], and is at the core of the foundational layer of simulation-based security [PW00, Can01, Can20, KTR20].

However, $\leq_T$ does not preserve [GHW11] *hyperproperties* [CS08] (e.g. security properties), which are sets of sets of traces, potentially associated with a probability. It might be a problem for the termination of algorithm or the privacy of data that relies on probabilistic protocols [GHW11]. A more sophisticated implementation relationship has been introduced by Roberto Segala (undeniably a member of the McCoys): trace distribution precongruence, noted $\leq_{DC}$ [Seg95a, LSV03]. A concrete object $\mathcal{A}_1$ is said to strongly observationally refines another abstract object $\mathcal{A}_2$, noted $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2$, if, for every entity $\mathcal{E}$ that can interact with both $\mathcal{A}_1$ and $\mathcal{A}_2$, for every way of resolving pure non-determinism in the first world by a certain scheduler, there exists a way of resolving pure non-determinism in the second world by another scheduler, such that every event is observed with the same probability by $\mathcal{E}$ in the two situations. This implementation relationship is composable, i.e. $\mathcal{A}_1 \leq_{DC} \mathcal{A}_2 \implies \mathcal{A}_1||\mathcal{B} \leq_{DC} \mathcal{A}_2||\mathcal{B}$, and deals with hyperproperties.

Even if $\leq_{DC}$ is equivalent to a relationship called strong linearizability [AE19] which cannot be ensured both asynchronously and deterministically for every object [AEW21], it makes sense to focus on $\leq_{DC}$ as an implementation relationship since (1) randomized or partially synchronous implementation can exist (e.g. via consensus [BO83, DNS88]), (2) restriction to subclasses of schedulers can be studied, (3) approximated version of the implementation can be ensured [AEW22, ML07a], and (4) the reasoning for different types of implementation relationships might be similar.

1.1.3. Dynamicity

Real-life systems are not always *static*, i.e. composed of a fixed set of entities with a fixed topology. It seems to be fair to say they are sometimes *dynamic*, i.e. their components (including communication links) may be created or destroyed during the system's execution. For example, the cells of a living being are created and destroyed, new companies are initiated while others go bankrupt in the same economy, computers of a network crash and have to be replaced, etc.

Some frameworks have been proposed to model, specify, and verify dynamic systems. They mostly have the form of discrete state machines [AL16, Kap09, Zim03, FHN+11, KPH15, Kur15] sometimes enriched with continuous steps capturing the passage of time, during which the variables can evolve (e.g. according to some algebraic differential equations) [KSPL06, Pla10, Pla11, Pla12, CSS10].

These models can be used to handle a high variety of use cases such as self-organization of dynamic systems [ADGR05, ADPBR10], file systems [SADADJ05], glue code [DGHK05], air-traffic control [KSPL06], distributed car control verification [Pla10], flight collision avoidance maneuver [Pla11],

peer-to-peer networks [DGV06], reconfigurable systems [YSSY15], actor-based programming [KPH15], etc.

In 2008, the famous Bitcoin protocol [Nak08] has made the study of dynamic systems fashionable again. Indeed, its anonymous author known under the pseudonym "Nakamoto", popularized a "simple" technique to handle Sybil attacks in an anonymous network, where a single malicious machine would like to masquerade as many [AJ05]. This technique, called Proof-of-Work, depends on the assumption that correct users possess a certain fraction of the total computational power of the system. This idea has been modified into a multitude of different variants, generically called proof-of-X, whose resiliency depends on the assumption that correct users possess a certain fraction of a physical quantity X that cannot be duplicated for free. With such a mechanism, it is possible to implement anonymous and decentralized applications without fearing that a malicious minority pretends to represent a majority and behaves to tackle the desired properties of the system. The most well-known example is certainly the double-spending attack, where an agent uses the same currency for two different transactions. A huge literature followed. Most of them address the dynamicity of the membership [SKM17, SK17, GKK$^+$20, KT20, LTWW22, DZ22], following a long tradition of churn-based protocol analysis [DW93, MRT$^+$05, BBKR09, BBR11, BN11, KLW11, BBS11, BKLW12, IRS$^+$13, KLW13, BPPb$^+$16, KW19, AKSW20, AKSW22], while a few also consider the dynamicity of the network [BNS22] or the proof-of-X mechanism [GKO$^+$20, Ter22, SW21]. In this paradigm, some new sophisticated security tasks emerged, such as dynamic secret sharing [BDLO15, ZZM$^+$19, GKM$^+$20].

The challenge of specifying dynamic systems has already been addressed by the Hatfields and the McCoys. Robin Milner, one of the most famous Hatfields, proposed π-calculus, offering an abundance of operators, a rich algebraic structure, and a well-developed fixed point theory.

On their side, the McCoys, represented by Paul Attie and Nancy Lynch, proposed a (non-probabilistic) hierarchical model where a system X, composed of objects interacting with each other and able to join and leave the system dynamically, is also an object of the model, which in turn can be used to build a system of a higher layer [AL16]. They proved (1) the stability of parallel composition $||$ in their model (if X and Y are objects of the model, then $X||Y$ is an object of the model) and (2) the composability of $\leq_T$ for dynamic systems. Furthermore, they proved (under technical minor assumptions) another theorem of substitutivity: (3) the dynamic creation/destruction of objects is monotonic with $\leq_T$, i.e. if (i) $\mathcal{A} \leq_T \mathcal{B}$ and (ii) $X_\mathcal{A}$ and $X_\mathcal{B}$ differ only by the fact that $X_\mathcal{A}$ dynamically creates and destroys the object $\mathcal{A}$ instead of dynamically creating and destroying the object $\mathcal{B}$ as $X_\mathcal{B}$ does (we note $X_\mathcal{A} \underset{A,B}{\triangledown} X_\mathcal{B}$), then (iii) $X_\mathcal{A} \leq_T X_\mathcal{B}$.

Dynamicity and Randomization Substitutivity results save computer scientists a lot of hassle. To describe, analyze, and verify complex systems rigorously, one needs an appropriate mathematical foundation, handling both dynamicity [AL16] and randomization [Seg95b]. To be practical, such a model has to naturally extend well-established ones, allowing the use of proven tools. Such a formalism should clarify if issues coming from dynamicity and issues coming from randomization are (1) independent or (2) can lead to new types of problems. This thesis gives answers in the direction of the first case (1), namely for composability of implementation relationship, and others in the direction of the second case (2), namely for monotonicity of creation/destruction of dynamic objects. Also, the model should allow us to express formally what we mean when we claim that a problem is solved by a dynamic protocol (evolving in a highly dynamic setting, using sophisticated cryptographic tools and randomized subroutines, under some synchrony assumptions) and should give us greater confidence to answer to the question: "Are we not missing something?". Here again, we believe that this thesis takes a step in this direction.

1.2. Thesis Overview (Short Version)

The chapter 2 lays down mathematical preliminaries. The sections that it contains should not be critical to read the thesis. Its first section 2.1, contains basic notions of set and function notations. Its second section 2.2, describes trajectories that are only used in chapter 5. Its third section 2.3 recalls the foundations of probability theories, but the thesis can be read without a deep understanding of it, since it only reminds the foundations to guarantee that we can properly define probabilities and integrations on the set of executions of a system. The section 2.4 recalls a few formal definitions related to Turing Machines that are not used outside of subsections 6.1.2 and 6.1.1 that provide sound computational foundations for simulation-based security. Here again, an intuitive understanding of Turing Machines is enough to use the framework of Chapter 6. Its last section 2.5 gives the definition of labeled transition systems (LTS) and explains why different variations of I/O automata formalism can be seen as particular cases of LTS with very few additional constraints and how to trivially move from I/O automata to LTS and vice versa.

Chapter 3 describes our new model for dynamic probabilistic systems, which, we believe, achieves a good tradeoff between expressiveness and simplicity. Basically, an entity of the model is a kind of LTS with (i) actions that lead to probabilistic distributions on states instead of directly to some states and (ii) a signature that can evolve during its life. The dynamicity is captured by (ii) and the fact that (iii) it is possible to build a new object X from a dynamic set of existing objects $\mathcal{A}_1, ..., \mathcal{A}_i, ...$ interacting with each other and able to join and leave the system dynamically. In that case, we can see the new object X as a member of a certain layer n for some integer n, while the objects $\mathcal{A}_1, ..., \mathcal{A}_i, ...$ belong to the layer $n-1$. To formalize the construction, each state q of X is associated with a "configuration", which is a photography of the system at the state q, i.e. a pair $(\mathbf{A}, \mathbf{S})$ where $\mathbf{A}$ contains the objects of the layer $n-1$ at state q and $\mathbf{S}$ represents the current states of the elements of $\mathbf{A}$. The elements of the layer 0 are called PSIOA for probabilistic signature input/output automata, while the elements of a higher layer are called PCA for probabilistic configuration automata. A dynamic probabilistic input/output automata (DPIOA) is either a PSIOA or a PCA. The closure of the model under parallel compositions, (noted $||$) is proved in section 3.5 in theorem 2 page 72. The inherent non-determinism of the model is resolved by traditional schedulers, that (potentially randomly) trigger the next action among the enabled ones. A classic notion of observational semantics is used, with a family of implementation relationships of the form $\leq_{\varepsilon}^{S,f}$. An implementation relationship captures the idea that a concrete real system is meeting the specification of an idealized system. Here, an object $\mathcal{A}$ implements another object $\mathcal{B}$, noted $\mathcal{A} \leq_{\varepsilon}^{S,f} \mathcal{B}$, if for every external distinguisher $\mathcal{E}$ that can interact with both $\mathcal{A}$ and $\mathcal{B}$, for every way (in S) of resolving the non-determinism in the $\mathcal{A}$-world, there exists a way (in S) of resolving the non-determinism in the $\mathcal{B}$-world, s.t. every possible observation via a function f is observed with the same probability in the two worlds, plus or minus ε. The well-established theory of probabilistic automata allows us to easily obtain good substitutability properties for the implementation relationships, which are shown to be a precongruence for parallel composition in theorem 5 page 80. Finally, we state the monotonicity of dynamic creation/destruction with a particular, but very natural, implementation relationship $\leq_0^{S_o,\mathfrak{p}}$ in theorem 7 page 87, i.e. if (1) $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ differ only by the fact that $X_{\mathcal{A}}$ dynamically create/destroy $\mathcal{A}$ instead of $\mathcal{B}$ as $X_{\mathcal{B}}$ does, noted $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$, and (2) $\mathcal{A}$ implements $\mathcal{B}$ (in the sense of $\mathcal{A} \leq_0^{S_o,\mathfrak{p}} \mathcal{B}$), then (3) $X_{\mathcal{A}}$ implements $X_{\mathcal{B}}$ (in the sense of $X_{\mathcal{A}} \leq_0^{S_o,\mathfrak{p}} X_{\mathcal{B}}$).

The chapter 4 contains a detailed modular homomorphic-based proof of theorem 7 stated a second time in theorem 23 page 150. First, we define in section 4.1 the notion of "executions-matching" to capture the idea that two automata have the same "comportment" [2] along with some corresponding executions.

2. The term is deliberately chosen to be vague.

Basically an executions-matching from a PSIOA $\mathcal{A}$ to a PSIOA $\mathcal{B}$ is a morphism connecting their respective sets of executions, preserving some properties. When the executions-matching is a bijection, we say $\mathcal{A}$ and $\mathcal{B}$ are semantically equivalent (they differ only syntactically). Second, we defined the notion of a PCA $X_{\mathcal{A}}$ deprived of a PSIOA $\mathcal{A}$, noted $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})$. Such an automaton corresponds to the intuition of a similar automaton where $\mathcal{A}$ is systematically removed from the underlying configuration of the original PCA. Thereafter we show that under technical minor assumptions $X_{\mathcal{A}} \setminus \{\mathcal{A}\}$ and $\tilde{\mathcal{A}}^{sw}$ are "partially-compatible" where $\tilde{\mathcal{A}}^{sw}$ and $\mathcal{A}$ are semantically equivalent. In fact, $\tilde{\mathcal{A}}^{sw}$ is the "simpleton wrapper" of $\mathcal{A}$, that is a PCA that only owns $\mathcal{A}$ in its attached underlying configuration. The notion of partial compatibility corresponds to an extension of the usual compatibility condition on LTS, where the compatibility revolves around reachable states only. Then we show that there is an (incomplete) execution-matching from $X_{\mathcal{A}}$ to $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw}$. The domain of this executions-matching is the set of executions where $\mathcal{A}$ is not (re-)created before the very last action. After this, we always try to reduce any reasoning on $X_{\mathcal{A}}$ (resp. $X_{\mathcal{B}}$) on a reasoning on $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw}$ (resp. $(X_{\mathcal{B}} \setminus \{\mathcal{B}\})||\tilde{\mathcal{B}}^{sw}$). We show that under certain reasonable technical assumptions, encapsulated in the definition of $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$, $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})$ and $(X_{\mathcal{B}} \setminus \{\mathcal{B}\})$ are semantically-equivalent. We note Y an arbitrary PCA semantically-equivalent to $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})$ and $(X_{\mathcal{B}} \setminus \{\mathcal{B}\})$. Finally, a reasoning on $\mathcal{E}||X_{\mathcal{A}}$ (resp. $\mathcal{E}||X_{\mathcal{B}}$) can be reduced to a reasoning on $\mathcal{E}'||\tilde{\mathcal{A}}^{sw}$ (resp. $\mathcal{E}'||\tilde{\mathcal{B}}^{sw}$) with $\mathcal{E}' = \mathcal{E}||Y$. Since $\tilde{\mathcal{A}}^{sw}$ implements $\tilde{\mathcal{B}}^{sw}$, we have already some results on $\mathcal{E}'||\tilde{\mathcal{A}}^{sw}$ and $\mathcal{E}'||\tilde{\mathcal{B}}^{sw}$ and so on $\mathcal{E}||X_{\mathcal{A}}$ and $\mathcal{E}||X_{\mathcal{B}}$. However, this reduction, represented in figure 1.9b, is valid only for the subset of executions without the creation of neither $\mathcal{A}$ nor $\mathcal{B}$ before the very last action. Ideally, we would like to decompose an "aggregated" class of perception, with an arbitrary number of creations/destructions of $\mathcal{A}$ (resp. $\mathcal{B}$)), into "atomic" classes of perception without creation/destruction of $\mathcal{A}$ (resp. $\mathcal{B}$)) before the last action. Some technical precautions have to be taken to be allowed to paste these fragments together to finally say that $\mathcal{A}$ implements $\mathcal{B}$ implies $X_{\mathcal{A}}$ implements $X_{\mathcal{B}}$. In fact, such a pasting is generally not possible for a perfect-information online scheduler, that can trigger an action taking into account the whole history. This observation motivated us to introduce the class S_o of schedulers to manipulate independent atomic classes of perception. This allowed us to prove monotonicity of dynamic creation/destruction with implementation relationship for this class of schedulers only: if (1) $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$, and (2) $\mathcal{A} \leq_0^{S_o,\mathtt{p}} \mathcal{B}$, then (3) $X_{\mathcal{A}} \leq_0^{S_o,\mathtt{p}} X_{\mathcal{B}}$. We also discuss a special case of task-schedulers [CCK+18], a very user-friendly class of fully-offline schedulers, that allows straightforward oblivious fair scheduling. Surprisingly, a naive adaptation of task-schedulers, to the dynamic paradigm is not a subset of S_o. We show how a more sophisticated adaptation of task-based implementation relationship, called tenacious implementation, noted $\leq_0^{ten}$, can allow obtaining monotonicity of dynamic creation/destruction of automata. This result is stated in theorem 24 page 174: if (1) $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$, and (2) $\mathcal{A} \leq_0^{ten} \mathcal{B}$, then (3) $X_{\mathcal{A}} \leq_0^{ten} X_{\mathcal{B}}$.

The chapters 5 and 6 explain how to extend the model with the notion of time and simulation-based security respectively, using pre-existing well-established variations of the I/O Automata framework. The associated models are still stable by composition and the new implementation relationships are still precongruences.

In the end, we obtain a framework that

1. is expressive enough to model non-sequential scheduling, dynamicity, randomness, time, synchrony, limited computational resources, and simulation-based security,
2. enjoys a sound modular design based on a classic observational semantic,
3. naturally extends the well-established framework of labeled transition systems,
4. is as simple as possible for a fixed expressiveness.

1.3. Thesis Overview (Long Version)

1.3.1. Chapter 2: Mathematical Preliminaries

The chapter 2 lays down mathematical preliminaries. The sections that it contains should not be critical to read the thesis. Its first section 2.1, contains basic notions of set and function notations. Its second section 2.2, describes trajectories that are only used in chapter 5. Its third section 2.3 recalls the foundations of probability theory, but the thesis can be read without a deep understanding of it, since it only gives the foundations to guarantee that we can properly define probabilities and integrations on the set of executions of a system. The section 2.4 recalls a few formal definitions related to Turing Machines that are not used outside of subsections 6.1.2 and 6.1.1 that provide sound computational foundations for simulation-based security. Here again, an intuitive understanding of Turing Machines is enough to use the framework of Chapter 6. Its last section 2.5 gives the definition of labeled transition systems (LTS) and explains why different variations of I/O automata formalism can be seen as simple particular cases of LTS and how to trivially move from I/O automata to LTS and vice versa.

1.3.2. Chapter 3: Dynamic Probabilistic Automata

Chapter 3 describes our new model for dynamic probabilistic systems, which, we believe, achieves a good tradeoff between expressiveness and simplicity.

Probabilistic Signature Input/Output Automata (PSIOA) The section 3.1 defines the notion of probabilistic signature Input/Output automata (PSIOA). A PSIOA $\mathcal{A}$ is an automaton (an LTS) that can move from one *state* to another through *actions*. The set of states of $\mathcal{A}$ is then denoted $Q_{\mathcal{A}}$, while we note $\bar{q}_{\mathcal{A}} \in Q_{\mathcal{A}}$ the unique start state of $\mathcal{A}$. At each state $q \in Q_{\mathcal{A}}$ some actions can be triggered in its signature $sig(\mathcal{A})(q)$. Such an action leads to a new state with a certain probability. The measure of probability triggered by an action a in a state q is denoted $\eta_{(\mathcal{A},q,a)}$. The model aims to allow the composition of several automata (noted $\mathcal{A}_1||...||\mathcal{A}_n$) to capture the idea of an interaction between them. That is why a signature is composed of three categories of actions: the input actions, the output actions, and the internal actions. In practice, the input action of an automaton potentially aims to be the output action of another automaton and vice-versa. Hence an automaton can influence another one through shared action. The entire system is formalized by the automaton issued from the composition of the automata of the system.

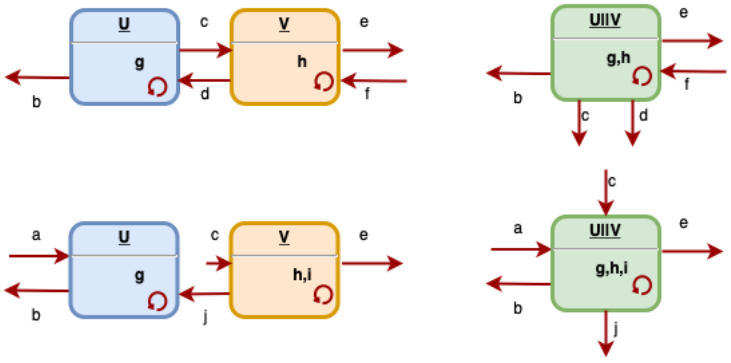


Figure 1.1. – A representation of 2 automata U and V and their composition at 2 different states with 2 different signatures

After this, we can speak about an execution of an automaton, which is an alternating sequence of states and actions. We can also speak about a trace of an automaton, which is the projection of an execution on its external actions uniquely. This is a classic way to speak about what can we observe from an outside point of view.

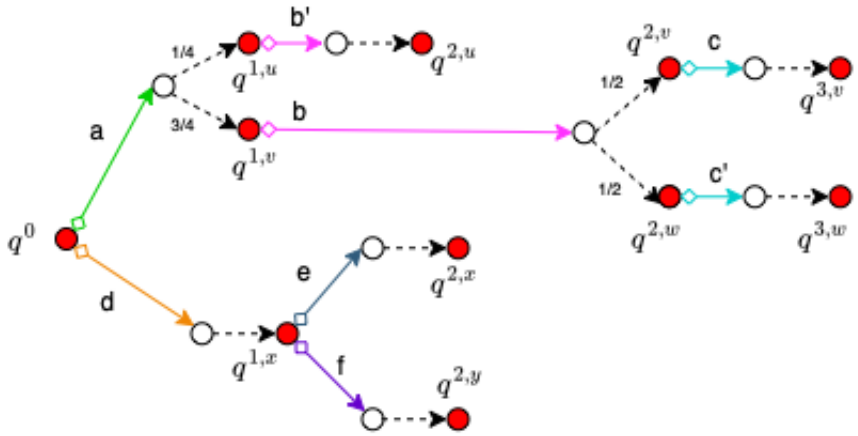


Figure 1.2. – A tree of possible executions for a PSIOA $\mathcal{A}$.

A priori, nothing allows to know which action has to be triggered between e and f after execution $q^0, d, q^{1,x}$

Scheduler We remarked in the example of figure 1.2, that inherent non-determinism has to be solved to be able to define a measure of probability over the executions. This is the role of the scheduler, which is a function σ that (consistently) maps a finite execution fragment to a discrete sub-probability distribution over the set of discrete transitions of the concerned PSIOA $\mathcal{A}$. Loosely speaking, the scheduler σ decides (probabilistically) which transition to take after each finite execution fragment α. Since this decision is a discrete sub-probability measure, it may be the case that σ chooses to halt after α with non-zero probability.

A scheduler σ generates a measure of probability over the set of executions. Hence, when a scheduler is made explicit, we can state the probability that an event append or that a property holds in temporal logic. The set of schedulers for the automaton $\mathcal{A}$ is denoted $schedulers(\mathcal{A})$. We denote by $\epsilon_\sigma : Execs(\mathcal{A}) \to [0,1]$ the execution distribution generated by the scheduler σ.

Environment, external behavior, implementation Now it is possible to define the crucial concept of implementation that captures the idea that an automaton $\mathcal{A}$ is meeting the specification of another automaton $\mathcal{B}$. To do so, we use an environment $\mathcal{E}$ which takes on the role of a "distinguisher" for $\mathcal{A}$ and $\mathcal{B}$. The set of environments of the automaton $\mathcal{A}$ is denoted $env(\mathcal{A})$. The information used by an environment to attempt a distinction between two automata $\mathcal{A}$ and $\mathcal{B}$ is captured by a function $f_{(.,.)}$ that we call *insight function*. In the literature, we very often deal with (i) $f_{(\mathcal{E},\mathcal{A})} = trace_{(\mathcal{E},\mathcal{A})}$, the trace function or (ii) $proj_{(\mathcal{E},\mathcal{A})} : \alpha \in Execs(\mathcal{E}||\mathcal{A}) \mapsto \alpha \upharpoonright \mathcal{E}$, the function that maps every execution to its projection on the environment. The philosophy of the two approaches is the same.

For any insight function $f_{(.,.)}$, we denote by $f\text{-}dist_{\mathcal{E},\mathcal{A}}(\sigma)$ the image measure of ϵ_σ under $f_{(\mathcal{E},\mathcal{A})}$. Intuitively, the probability measure $f\text{-}dist_{\mathcal{E},\mathcal{A}}(\sigma)$ gives the probability of any observation that can be made by the environment during its interaction with $\mathcal{A}$, when the pure non-determinism has been resolved by the scheduler σ. From here, this is classic to define the f-external behaviour of $\mathcal{A}$, denoted $ExtBeh^f_{\mathcal{A}} : \mathcal{E} \in env(\mathcal{A}) \mapsto \{f\text{-}dist_{\mathcal{A},\mathcal{E}}(\sigma) | \sigma \in schedulers(\mathcal{E}||\mathcal{A})\}$. Such an object capture all

the possible measures of probability on the possible external perceptions of the interaction of the concerned automaton $\mathcal{A}$ and an arbitrary environment $\mathcal{E}$. Finally we can say that $\mathcal{A}$ f-implements $\mathcal{B}$ if $\forall \mathcal{E} \in env(\mathcal{A}) \cap env(\mathcal{B}), ExtBeh^{f}_{\mathcal{A}}(\mathcal{E}) \subseteq ExtBeh^{f}_{\mathcal{B}}(\mathcal{E})$, i.e. for any "distinguisher" $\mathcal{E}$ for $\mathcal{A}$ and $\mathcal{B}$, for any way of resolving the non-determinism of $\mathcal{A}$-world, there exists a way of resolving the non-determinism of $\mathcal{B}$-world, such that the probability to observe a situation is the same in the two worlds. This is a way to formalize that there is no way to distinguish $\mathcal{A}$ from $\mathcal{B}$. (see figure 1.3).

However, the correctness of an algorithm may be based on some specific assumptions about the scheduling policy that is used. Thus, in general, we are interested only in a subset of $schedulers(\mathcal{E}||\mathcal{A})$. A function S that maps any automaton to a subset of its schedulers is called a *scheduler schema*. Among the most noteworthy examples are the fair schedulers, the off-line, a.k.a. oblivious schedulers, defined in opposition with the online schedulers. So, we note $ExtBeh^{f,S}_{\mathcal{A}} : \mathcal{E} \in env(\mathcal{A}) \mapsto \{f\text{-}dist_{\mathcal{A},\mathcal{E}}(\sigma) | \sigma \in S(\mathcal{E}||\mathcal{A})\}$ where S is a scheduler schema and we say that $\mathcal{A}$ f-implements $\mathcal{B}$ according to a scheduler schema S, noted $\mathcal{A} \leq_0^{S,f} \mathcal{B}$, if $\forall \mathcal{E} \in env(\mathcal{A}) \cap env(\mathcal{B}), ExtBeh^{f,S}_{\mathcal{A}}(\mathcal{E}) \subseteq ExtBeh^{f,S}_{\mathcal{B}}(\mathcal{E})$. In the remaining, we will have a great interest in two certain classes of oblivious schedulers, i.e. i) the class S_o of schedulers that do not take into account the internal past lives of sub-automata before their last destruction to trigger an action and ii) the task-scheduler: a user-friendly off-line scheduler already introduced in [CCK+18]. It is possible to formally define an ε-approximated version of the implementation relationship, noted $\leq_{\varepsilon}^{S,f}$. The intuition is that an environment cannot distinguish the real world from the ideal world with a probability greater than $\frac{1}{2} + \varepsilon$.

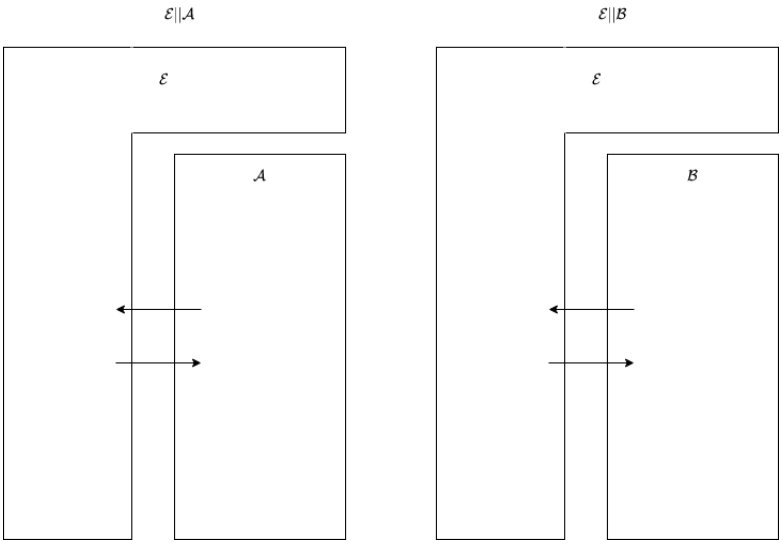


Figure 1.3. – An environment $\mathcal{E}$, which is nothing more than a PSIOA compatible with both $\mathcal{A}$ and $\mathcal{B}$, tries to distinguish $\mathcal{A}$ from $\mathcal{B}$.

Probabilistic Configuration Automata (PCA) The section 3.2 introduces the notion of probabilistic configuration automata (PCA). (see figure 1.4). A PCA corresponds to a PSIOA, but each state is mapped to an underlying *configuration* $C = (\mathbf{A}, \mathbf{S})$ which is a pair constituted by a set $\mathbf{A}$ of PSIOA and the current states of each member of the set (with a mapping function $\mathbf{S} : \mathcal{A} \in \mathbf{A} \mapsto q_{\mathcal{A}} \in Q_{\mathcal{A}}$). The idea is that the composition of the attached set can change during the execution of a PCA, which allows us to formalize the notion of dynamicity, that is the potential creation and potential destruction of a PSIOA in a dynamic system. Some particular precautions have to be taken to make it consistent.

We state key results for a sound modular design.

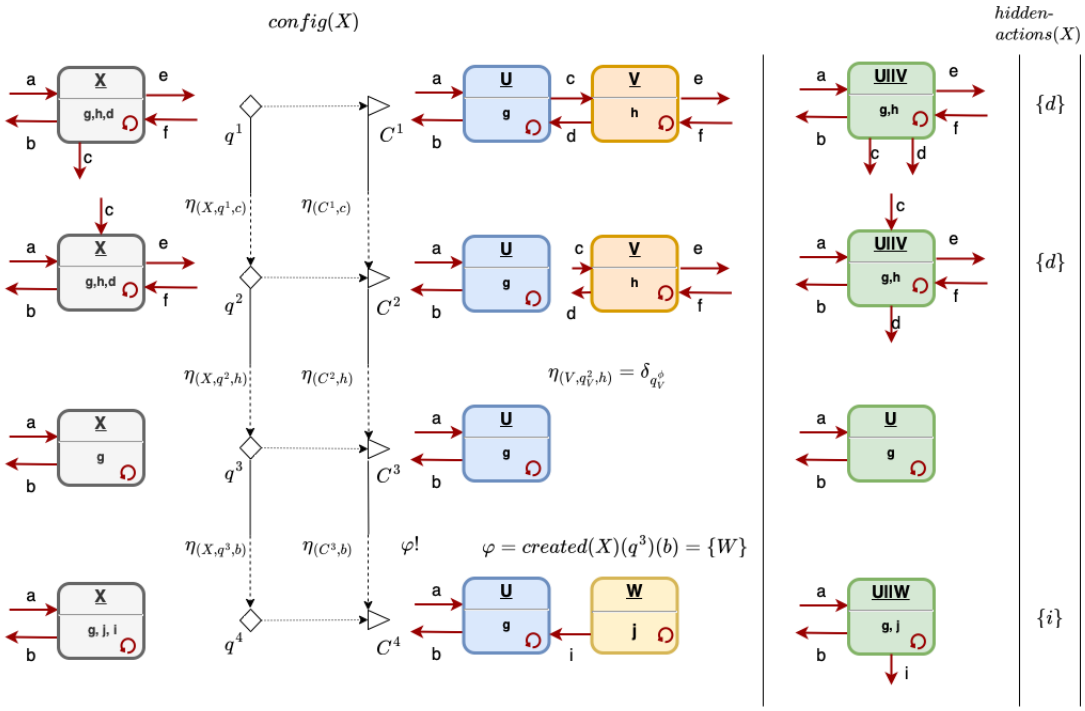


Figure 1.4. – PCA execution fragment

1. We show (theorem 2 page 72) the closure of the set of PCA under parallel composition
2. We show (theorem 5 page 80) that implementation relationship is a precongruence for parallel composition, i.e. $\mathcal{A} \leq_\varepsilon^{S,f} \mathcal{B} \implies \mathcal{A}||\mathcal{C} \leq_\varepsilon^{S,f} \mathcal{B}||\mathcal{C}$. $\leq_\varepsilon^{S,f}$ (for any scheduler schema, any $\varepsilon \in [0,1]$, any perception function f, which is an insight function with an additional property that captures the idea that an environment $\mathcal{E}$ does not have a greater power of distinction than another environment $\mathcal{E}||\mathcal{E}'$ for some arbitrary $\mathcal{E}'$).
3. We show (theorem 7 page 87) that, under certain technical conditions, dynamic creation/destruction is monotonic with respect to external behavior inclusion, i.e. if a system X dynamically creates/destroys automaton $\mathcal{A}$ instead of dynamically creating/destroying automaton $\mathcal{B}$ and the external behaviors of $\mathcal{A}$ are respective subsets of the external behaviors of $\mathcal{B}$, then the set of external behaviors of the overall system is possibly reduced, but not increased.

The results enable a design and refinement methodology based solely on the notion of externally visible behavior, that is independent of specific methods of establishing external behavior inclusion. It permits the refinement of components and subsystems in isolation from the entire system.

A dynamic probabilistic input/output automata (DPIOA) is either a PSIOA or a PCA.

1.3.3. Chapter 4: Road to monotonicity

The chapter 4 contains a detailed modular homomorphic-based proof of theorem 7 of monotonicity stated a second time in theorem 23 page 150. To do so, we develop different mathematical tools.

Executions-matching First, we define in section 4.1, the notion of executions-matching (see figure 1.5) to capture the idea that two automata have the same "comportment" [3] along with some corre-

3. The term is deliberately chosen to be vague.

sponding executions. Basically an execution-matching from a PSIOA $\mathcal{A}$ to a PSIOA $\mathcal{B}$ is a morphism $f^{ex} : Execs'_{\mathcal{A}} \to Execs(\mathcal{B})$ where $Execs'_{\mathcal{A}} \subseteq Execs(\mathcal{A})$. This morphism preserves some properties along the pair of matched executions: signature, transition, ... in such a way that for every pair $(\alpha, \alpha') \in Execs(\mathcal{A}) \times Execs(\mathcal{B})$ s.t. $\alpha' = f^{ex}(\alpha)$, $\epsilon_\sigma(\alpha) = \epsilon_{\sigma'}(\alpha')$ for every pair of scheduler (σ, σ') (so-called *alter ego*) that are "very similar" in the sense they take into account only the "structure" of the argument to return a sub-probability distribution, i.e. $\alpha' = f^{ex}(\alpha)$ implies $\sigma(\alpha) = \sigma'(\alpha')$. When the executions-matching is a bijection function from $Execs(\mathcal{A})$ to $Execs(\mathcal{B})$, we say $\mathcal{A}$ and $\mathcal{B}$ are semantically-equivalent (they differ only syntactically).

A PCA $X_{\mathcal{A}}$ deprived of a PSIOA $\mathcal{A}$ Second, we define in section 4.2 the notion of a PCA $X_{\mathcal{A}}$ deprived of a PSIOA $\mathcal{A}$ noted $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})$. Such an automaton corresponds to the intuition of a similar automaton where $\mathcal{A}$ is systematically removed from the configuration of the original PCA (see figure 1.6a and 1.6b).

Reconstruction: $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw}$ Thereafter we show in section 4.3 that under technical minor assumptions $X_{\mathcal{A}} \setminus \{\mathcal{A}\}$ and $\tilde{\mathcal{A}}^{sw}$ are composable where $\tilde{\mathcal{A}}^{sw}$ and $\mathcal{A}$ are semantically equivalent in the sense loosely introduced in the section 1.3.3. In fact, $\tilde{\mathcal{A}}^{sw}$ is the simpleton wrapper of $\mathcal{A}$, which is a PCA that only owns $\mathcal{A}$ in its attached configuration (see figure 1.7). Let us note that if $\mathcal{A}$ implements $\mathcal{B}$, then $\tilde{\mathcal{A}}^{sw}$ implements $\tilde{\mathcal{B}}^{sw}$.

Then we show that there is an (incomplete) execution-matching from $X_{\mathcal{A}}$ to $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw}$ (see figure 1.8). The domain of this executions-matching is the set of executions where $\mathcal{A}$ is not (re-)created.

After this, we always try to reduce any reasoning on $X_{\mathcal{A}}$ (resp. $X_{\mathcal{B}}$) on a reasoning on $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw}$ (resp. $(X_{\mathcal{B}} \setminus \{\mathcal{B}\})||\tilde{\mathcal{B}}^{sw}$).

Corresponding PCA We show in section 4.4 that, under certain reasonable technical assumptions (captured in the definition of corresponding PCA w.r.t. $\mathcal{A}$, $\mathcal{B}$), $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})$ and $(X_{\mathcal{B}} \setminus \{\mathcal{B}\})$ are semantically-equivalent. We can note Y an arbitrary PCA semantically-equivalent to $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})$ and $(X_{\mathcal{B}} \setminus \{\mathcal{B}\})$. Finally, a reasoning on $\mathcal{E}||X_{\mathcal{A}}$ (resp. $\mathcal{E}||X_{\mathcal{B}}$) can be reduced to a reasoning on $\mathcal{E}'||\tilde{\mathcal{A}}^{sw}$ (resp. $\mathcal{E}'||\tilde{\mathcal{B}}^{sw}$) with $\mathcal{E}' = \mathcal{E}||Y$. Since $\tilde{\mathcal{A}}^{sw}$ implements $\tilde{\mathcal{B}}^{sw}$, we have already some results on $\mathcal{E}'||\tilde{\mathcal{A}}^{sw}$ and $\mathcal{E}'||\tilde{\mathcal{B}}^{sw}$ and so on $\mathcal{E}||X_{\mathcal{A}}$ and $\mathcal{E}||X_{\mathcal{B}}$. However, these results are a priori valid only for the subset of executions without the creation of neither $\mathcal{A}$ nor $\mathcal{B}$ before the very last action). This reduction is represented in figures 1.9a and 1.9b.

Cut-paste execution fragments creation at the endpoints The reduction, which is roughly described in figures 1.9a and 1.9b, holds only for executions fragments that do not create the automata $\mathcal{A}$ and $\mathcal{B}$ after their destruction (or at the very last action). Some technical precautions have to be taken to be allowed to paste these fragments together to finally say that $\mathcal{A}$ implements $\mathcal{B}$ implies $X_{\mathcal{A}}$ implements $X_{\mathcal{B}}$. In fact, such a pasting is generally not possible for a perfect-information online scheduler, that can trigger an action taking into account the whole history. This observation motivated us to introduce the class S_o of schedulers that outputs (randomly) a transition without taking into account the triggered internal actions and the visited states of a sub-automaton $\mathcal{A}$ preceding its last destruction.

We prove the monotonicity of dynamic creation/destruction with external behavior inclusion for the schema of schedulers in S_o in Section 4.5. This result is encapsulated in Theorem 23 page 150: if (1) $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$, and (2) $\mathcal{A} \leq_0^{S_o,\mathtt{p}} \mathcal{B}$, then (3) $X_{\mathcal{A}} \leq_0^{S_o,\mathtt{p}} X_{\mathcal{B}}$. Figure 1.10 represents the issue with schedulers absent from S_o. We also discuss a special case of task-schedulers [CCK+18], a very user-friendly class

of fully offline schedulers, that allows straightforward oblivious fair scheduling. Surprisingly, a naive adaptation of task-schedulers, to the dynamic paradigm is not a subset of S_o. We show how a more sophisticated adaptation of task-based implementation relationship, called tenacious implementation, noted $\leq_0^{ten}$, can allow obtaining monotonicity of dynamic creation/destruction of automata. This result is stated in theorem 24 page 174: if (1) $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$, and (2) $\mathcal{A} \leq_0^{ten} \mathcal{B}$, then (3) $X_{\mathcal{A}} \leq_0^{ten} X_{\mathcal{B}}$.

1.3.4. Chapter 5: Extension with time

Previous chapters 3 and 4 deal with "asynchronous" systems. Based on these results, it is impossible to express something like "two events are separated by at most a time t", or apply synchrony assumptions to prove the correctness of an algorithm. In chapter 5, we fill this gap by extending PSIOA (resp. PCA) with a set of trajectories to obtain probabilistic timed signature input/output automata (PTSIOA-(resp. probabilistic timed configuration automata (PTCA)), following the well-established theory of timed I/O automata [KLSV06]. It would have been possible to use the "old-fashioned recipe for real-time" [AL92] as in [Seg95b], where actions can additionally contain real numbers to capture the time elapsing. But, we would miss the opportunity to use the theory of timed I/O automata [KLSV03, KLSV06], which is ready to use. For example, it provides some results to deal with liveness under fairness assumptions (through notions of feasibility, responsiveness, progressiveness,...) that might be more intuitive to express than in [AL92] and benefits of enabled tools [ALL$^+$06, ALL$^+$08] to support system development with, for example, specification simulators, code generators, model checking, and theorem proving support for analyzing specification, etc. Also, it has been extended with a (continuous) probabilistic setting [Mit07, ML07b], yielding Probabilistic Timed I/O automata (PTIOA). Hence, the extension of the formalism to Probabilistic Timed Signature I/O automata (PTSIOA) becomes an easy adaptation of PTIOA. We keep the continuous probabilistic setting of Mitra and Lynch [ML07b], since (1) their results are easily adaptable to our dynamic framework, and (2) it would be convenient to potentially specify hybrid systems where we would like to model the evolution of some physical quantities (force, mass, velocity, ...) with real numbers, that might be dependent of some algebraic differential equations, typically for collision avoidance systems [KSPL06, Pla10, Pla11].

The big picture does not change significantly. Dynamic timed automata have to verify a new constraint of trajectory preservation, that requires a strong link between the set of trajectories of the dynamic automata and the ones of the associated configurations. Theorem 26 shows that the composition of two pre-PTCA is a pre-PTCA itself, where a pre-PTCA is a PTCA that does not necessarily verify a certain constraint of measurability, noted $\mathbf{M}$. Theoretically, it is possible that the composition of two PTCA with continuous state space does not verify $\mathbf{M}$, which enables the definition of a probability measure over its set of executions. However, such an example would be very artificial and should not correspond to classic systems. The results of substitutability of the implementation relationship do not change and do not require additional sophisticated treatment.

1.3.5. Chapter 6: Extension with simulation-based security

The chapter 6 explains how to extend the model of chapters 3 and 4 with the notion of simulation-based security. Indeed, these two latter chapters deal with "unbounded" systems, in the sense that nothing prevents, a priori, from performing steps that would require a huge amount of computational resources, which is assumed to be infeasible by classic cryptographic tools, assuming a (usually polynomially) bounded adversary using a limited amount of computational resources. The achieved security properties are then of the form, "for any adversary with a bounded computational power, the probability of violating the security is negligible". Even if we would like to use a symbolic model,

where operations are seen as functions in a space of symbolic (formal) expressions, its soundness will rely on a randomized computational model. The most popular approach is simulation-based security.

In the simulation-based security, a.k.a. "real-ideal paradigm", the objective of the adversary is defined only with respect to an "idealized" game, which can be seen as the specification of the task we want to solve. Then, the security properties will be stated in the following form: for any adversary, there exists a "simulator" (which is another "ideal" adversary), so that the "adversary in the real game" and the "simulator in the ideal game" cannot be distinguished with more than a negligible probability by an external observer (our environment which plays the role of a "distinguisher"). The idea is to say that any attack that (i) can be made against the real game (ii) can also be made against the ideal game, while we are satisfied with such an ideal world (ii). Such a paradigm has many interests. First, the absence of an explicitly defined goal for the adversary prevents one from missing any subtle attacks that might occur. Second, it allows for the correct definition of certain properties whose alternative definitions might be cumbersome (e.g., the property of "not learning something more than something").

In this chapter 6, we extend DPIOA with simulation-based security as Canetti et al. [CCK$^+$07] do with static PIOA. The resulting model has some interests compared to classic frameworks for composable simulation-based security, such as UC [CCK$^+$18] or IITM [KTR20], where entities are modeled as interactive Turing machines (ITMs).

simplicity and abstraction In practice, though entities in the system are defined by interactive Turing machines (ITMs), a precise description would require too many low-level machine details. Hence the entities are usually described using some type of pseudo-code, where it remains unknown how it is supposed to be translated into a Turing Machine. According to Hofheinz and Shoup, "as long as we restrict ourselves to models that are polynomial-time equivalent to Turing machines, none of these details matter"[HS15]. In fact, detailed execution fragments of a system of ITMs seem to be a non-necessary intricacy, that, combined with the sophistication of distributed algorithms, can lead to an excessively hard-to-follow analysis. It seems that the overuse of the simulation-based security paradigm, expressed with ITMs, has participated in the respective misunderstanding between the distributed computing and secure multiparty computing (MPC) communities. Labeled Transition Systems (LTS) such as I/O automata allow not specify the exact computation of the state transition. Furthermore, their manipulation is closer to the human intuition and classic informal reasoning that often appear in distributed computing.

non-sequential scheduling The second interest is that sequential scheduling models are commonly used in many cryptography frameworks (UC, IITM, etc), where multiple processes must not be active simultaneously at any given point in time, and the active process activates the next process by producing and sending some message. Besides being a counter-intuitive modeling, sequential scheduling can artificially introduce some constraints in the ordering of events, and so artificially restrict the power of the adversary in comparison to the real world. For example, Canetti et al. [CCLP07] proved that a beacon protocol preserves security under sequential scheduling (ITM) but not under non-sequential scheduling, while Yoneyama [Yon10] showed that the beacon protocol verifies indifferentiable security (where public channels have to be considered as well as private channels) under the sequential scheduling but not under the non-sequential scheduling.

time The ITMs-based frameworks are inherently asynchronous. It is possible to add a notion of time and/or synchrony, assuming the access to an ideal functionality $\mathcal{F}_{CLOCK}$ that allows different parties to synchronize [KMTZ13], but the result is not trivial to handle and does not necessarily correspond to the human intuition of synchrony. Our model might propose something simpler. We see 3 different

ways of adding time. First, the "old-fashioned recipe for real-time" [AL92], as used in [Seg95b]. Second, we can use trajectories with a discrete-time axis. In this case, one unit of time corresponds to the greatest common divisor of the clock periods (the reverse of clock rates) of all the processors. Then, we can easily limit the number of steps per unit of time of an object of the model, following a similar approach to that of Canetti et al. [CCK$^+$08]. Third, we can use trajectories with a continuous time axis ($\mathbb{T} = \mathbb{R}$) but we need to take precautions not to artificially give extra computational power. Hence, we would restrict the set of continuous variables, to a particular set with type included in $\mathbb{N}$, the so-called clocks set. Hence, the restriction of a trajectory onto non-clock variables is a constant function, while the restriction of a trajectory to clock variables is a simple function, where the number of different values that can be taken is again defined by the greatest common divisor of the clock periods (the reverse of clock rates) of all the processors.

structure In order to analyze cryptographic protocols Canetti & al. [CCK$^+$07] extends probabilistic Input/Output automata, with the notion of "structures", which classifies communications into two categories: those with a distinguisher environment and those with an adversary". Then, they are able to define a new implementation relationship, called "secure emulation" in the same way as traditional simulation-based security ([KDMR08, CCLP07, KTR20, RKC22] discuss the relationship between different notions of simulation-based security). The composability of secure-emulation is then "easily" derived from the composability of classic implementation relationship for PIOA. Even though the formalism proposed in [CCK$^+$07] has been already used in the verification of various cryptographic protocols [CCLP07, CMP07, YKO07, JMMS10, Yon18], this formalism does not allow the modeling of systems that "can dynamically create new protocol instances at run time". For example, we would like to cover blockchain systems where sub-chains can be created or destroyed at run time [RPG19]. Hence, on top of dynamic probabilistic I/O automata described in detail in chapters 3 and 4, we propose an extension of the composable secure-emulation of Canetti et al. [CCK$^+$07] to dynamic settings. Our framework, composable dynamic secure emulation, enjoys the composability properties of secure emulation of [CCK$^+$07]. In terms of dynamicity, our work has closed a modeling problem left open by the work of Canetti et al. [CCK$^+$07]. That is, our framework allows the modeling of environments (or structures) that can dynamically create new protocol instances at run time.

Most of the results of this chapter are straightforward adaptations of their "static" counterparts in [CCK$^+$07]. The main source of modification comes from (1) dealing with compatibility at every reachable state instead of compatibility at each element of the Cartesian product of respective states and (2) handling general schedulers instead of task-schedulers only. For sake of completeness, we repeat the results with the required small modifications in the proofs.

1.3.6. Summary

We proposed a framework that

1. is expressive enough to model non-sequential scheduling, dynamicity, randomness, time/synchrony, limited computational resources, and simulation-based security,
2. enjoys a sound modular design based on a classic observational semantic, where (1) the implementation relationship is a precongruence for parallel composition and (2) dynamic creation/destruction of automata is monotonic with the implementation relationship,
3. naturally extends the well-established framework of labeled transition systems,
4. is, we believe, as simple as possible for the fixed expressiveness.

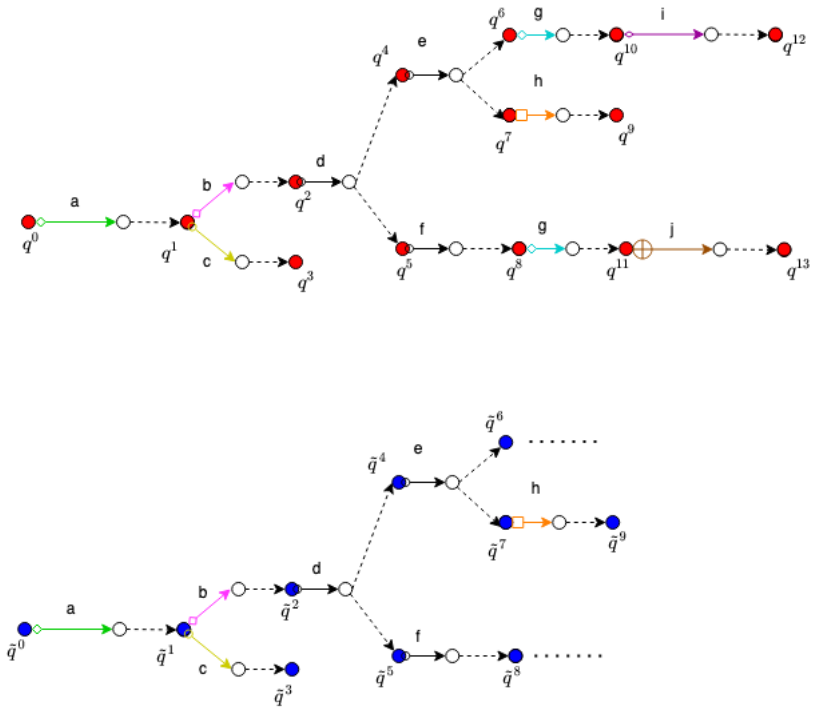
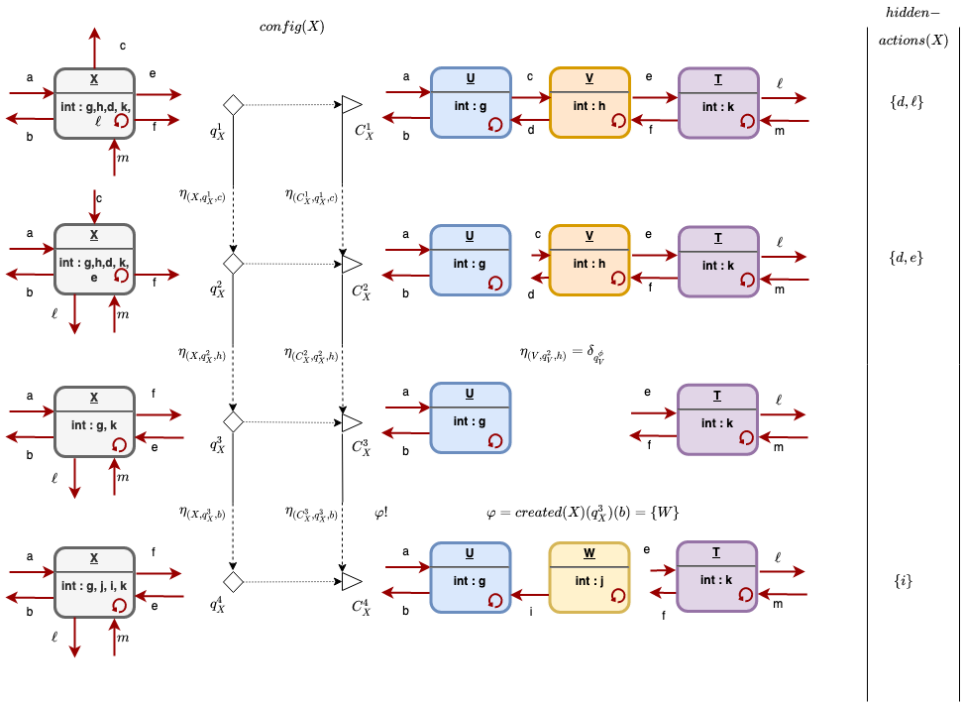
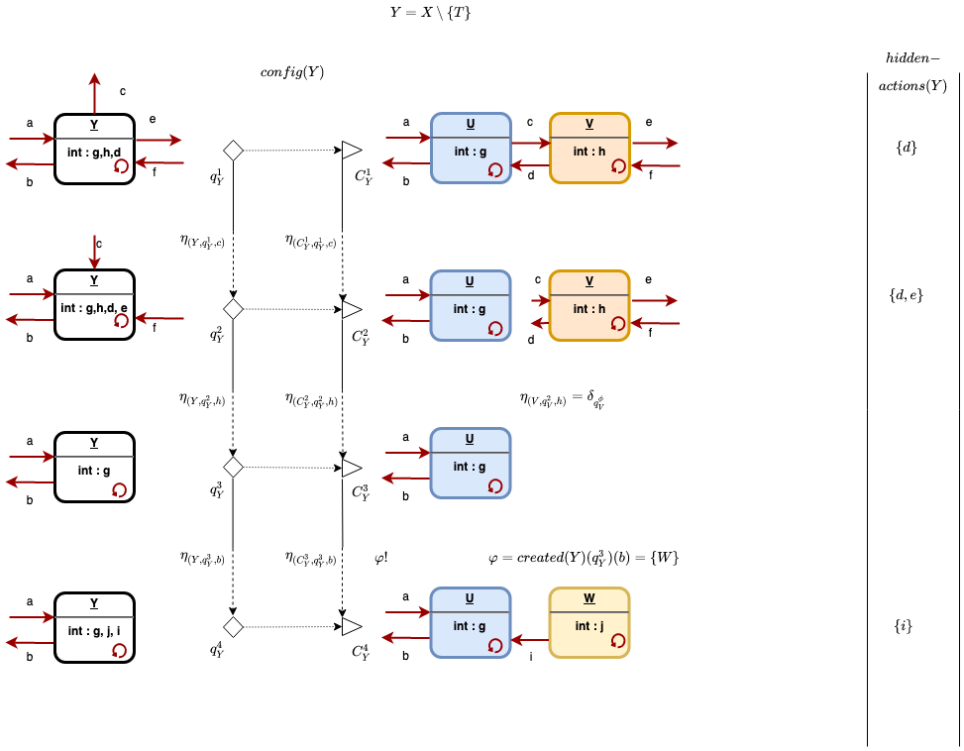


Figure 1.5. – The respective executions tree of two automata $\mathcal{A}$ and $\mathcal{B}$ are linked by an executions-matching.



(a) Projection on PCA, part 1/2



(b) Projection on PCA, part 2/2

The figure represents the PCA $Y = X \setminus \{T\}$ while the original PCA X is represented in figure 1.6a. We can see that the sub-automaton T (in purple in figure 1.6a) has been systematically removed from the configurations attached to the states visited by Y.

Figure 1.6. – PCA deprived of a sub-PSIOA

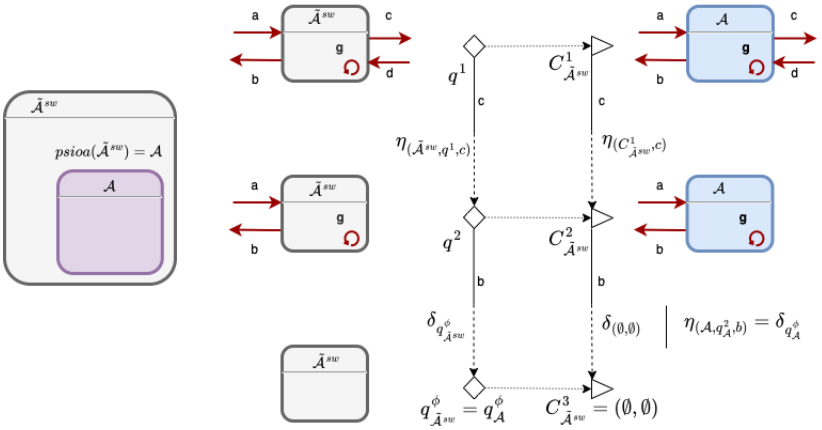


Figure 1.7. – Simpleton Wrapper

The figure represents the simpleton wrapper $\tilde{\mathcal{A}}^{sw}$ of an automaton $\mathcal{A}$. The automaton $\tilde{\mathcal{A}}^{sw}$ is a PCA that only encapsulates one unique sub-automaton which is $\mathcal{A}$. We can confuse $\mathcal{A}$ and $\tilde{\mathcal{A}}^{sw}$ without impact. Intuitively, we can see $\tilde{\mathcal{A}}^{sw}$ as a wrapper of $\mathcal{A}$ that does not provide anything.

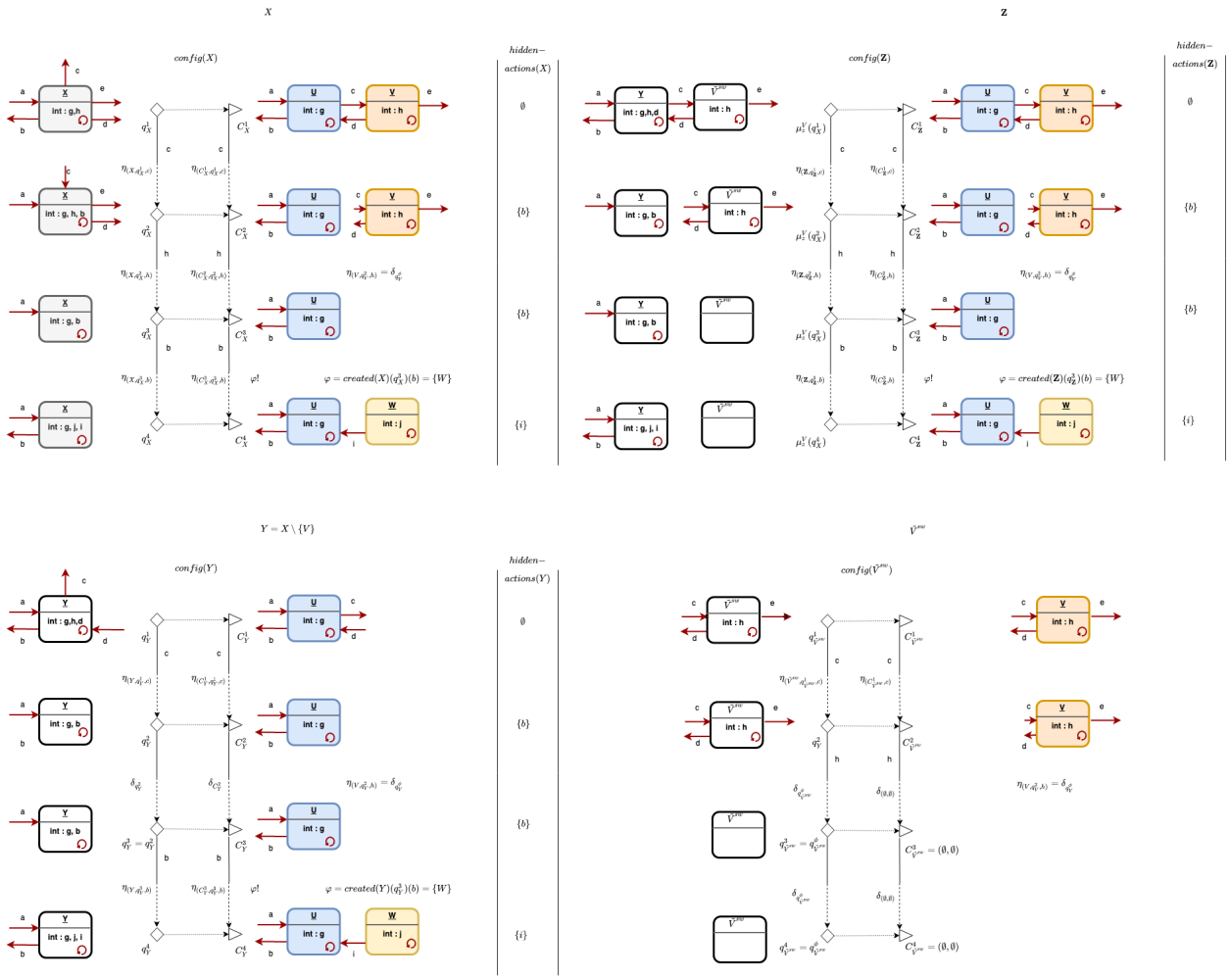
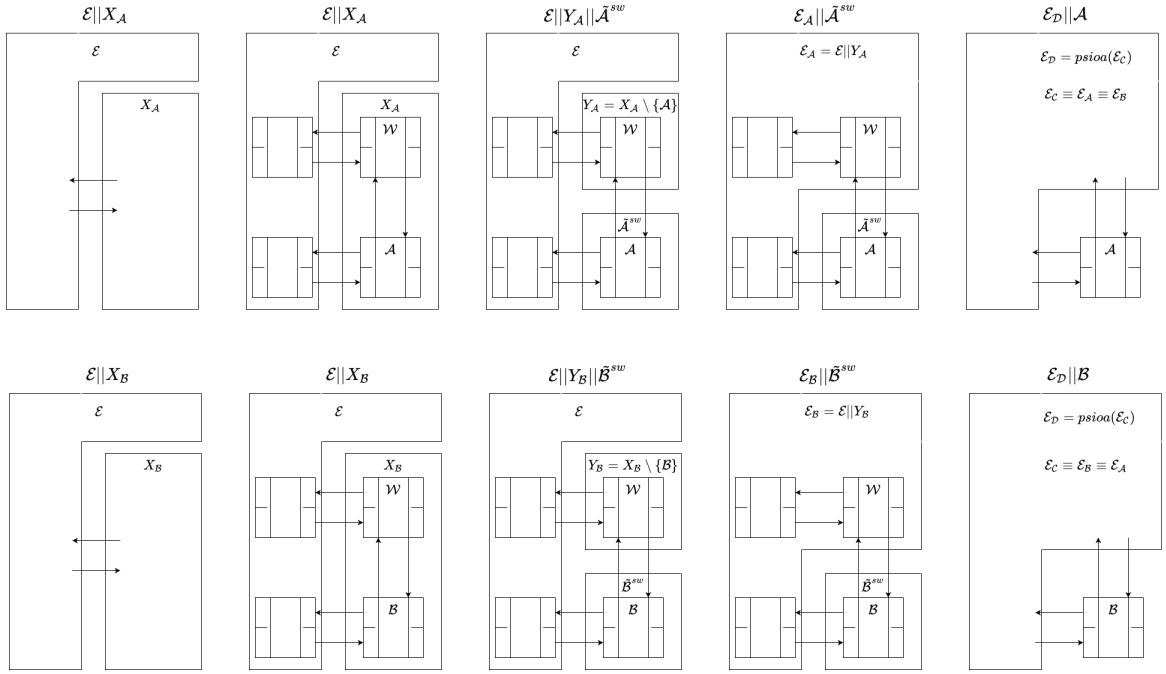


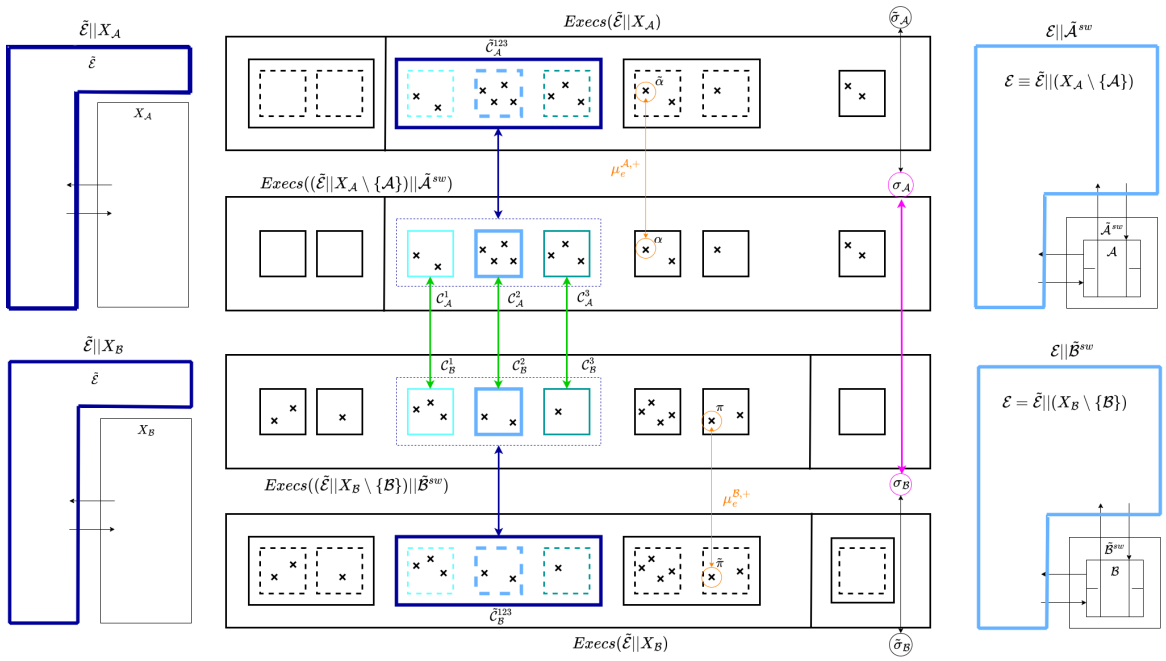
Figure 1.8. – Reconstruction

The figure shows the similarities between two PCA X and $Z = (X \setminus \{V\})||\tilde{V}^{sw}$



(a) Monotonicity reduction

The figure represents successive steps to reduce the problem of an environment $\mathcal{E}$ that tries to distinguish two PCA $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ (represented in the first column) to a problem of an environment $\mathcal{E}_{\mathcal{D}}$ that tries to distinguish the automata $\mathcal{A}$ and $\mathcal{B}$ (represented at last column).



(b) Homomorphism based proof

Figure 1.9. – homomorphism-based-proof

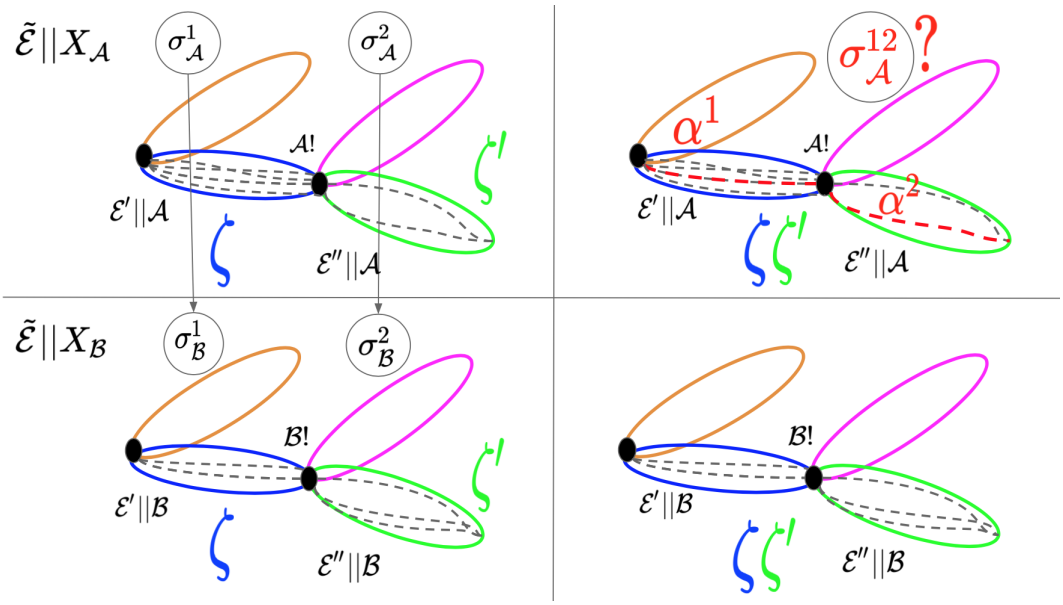


Figure 1.10. – The necessity of a scheduler in S_o.

The reduction described before holds only for the set of executions that do not create either $\mathcal{A}$ or $\mathcal{B}$ before the last action (represented on the left). What if the scheduler $\sigma^{12}_{\mathcal{A}}$ breaks the independence of probabilities between executing α^1 and executing α^2 after α^1 ? In that case, we cannot cut-paste the different reductions and the monotonicity of implementation does not hold, i.e. there is no reason there exists a scheduler counterpart $\sigma^{12}_{\mathcal{B}}$ s.t. that observing $\zeta^\frown\zeta'$ (represented in blue and green) has the same probability to occur in $\mathcal{A}$-world and in $\mathcal{B}$-world.

Chapter 2

Preliminaries

Overview

2.1. Sets and Functions . . . **22**
2.1.1. Sets and intervals . . . 22
2.1.2. Functions . . . 22
2.2. Trajectories . . . **24**
2.2.1. Basics . . . 24
2.2.2. Time shift, prefix, suffix, concatenation . . . 24
2.2.3. Deteministic Trajectories . . . 24
2.2.4. Trajectories sets composition and isomorphism . . . 25
2.3. Probability Theory . . . **26**
2.3.1. Set Algebra . . . 26
2.3.2. Measure Spaces . . . 27
2.3.3. Measurable Function . . . 27
2.3.4. Product of Measure Spaces . . . 28
2.3.5. Discrete probability space . . . 28
2.3.6. Measure Spaces Isomorphism . . . 28
2.3.7. Integration . . . 28
2.3.8. Carathéodory's Extension Theorem . . . 29
2.4. Turing Machines . . . **30**
2.4.1. Deterministic Turing Machine . . . 30
2.4.2. Probabilistic Turing Machine . . . 31
2.4.3. Encoding . . . 32
2.4.4. Decision . . . 34
2.5. Labeled Transition System (LTS) . . . **35**
2.5.1. Simple Labeled Transition System . . . 35
2.5.2. I/O Automata . . . 35
2.5.3. PIOA . . . 36
2.5.4. DIOA . . . 36
2.5.5. TIOA . . . 36
2.5.6. DPIOA . . . 36
2.5.7. DPTIOA . . . 36
2.5.8. Overwiev of the I/O Automata framework . . . 36

2.1. Sets and Functions

2.1.1. Sets and intervals

We use standard set theory.

Definition 1 (operations on sets)**.** *We use the usual symbols* $\in, \subset, \cup, \cap, \setminus, \times$ *for classic notions of set theory:"is an element of", "is a subset of", union, intersection, set difference, and Cartesian product. When universal set* Ω *is clear in the context, the complement of a set* S *is denoted by* $S^c \triangleq \Omega \setminus S$*. The union of a collection* $\{S_i\}_{i\in I}$ *of pairwise disjoint sets indexed by a set* I *is written as* $\biguplus_{i\in I} S_i$*. The power set of a set* S*, is denoted* $\mathcal{P}(S)$*.*

classic sets We note:

- $\mathbb{R}$ the set of real numbers, $\mathbb{R}^{\geq 0} \triangleq \mathbb{R} \setminus \{x \in \mathbb{R} | x < 0\}$, $\mathbb{R}^{>0} \triangleq \mathbb{R}^{\geq 0} \setminus \{0\}$
- $\mathbb{N}$ the set of natural integers, $\mathbb{N}^{>0} \triangleq \mathbb{N} \setminus \{0\}$
- $\mathbb{Q}$ the set of rational numbers, $\mathbb{Q}^{\geq 0} \triangleq \mathbb{Q} \setminus \{x \in \mathbb{R} | x < 0\}$, $\mathbb{Q}^{>0} \triangleq \mathbb{Q}^{\geq 0} \setminus \{0\}$

intervals

- $\forall i, j \in \mathbb{N}$, $i \leq j$, we denote by $[i : j] \triangleq \{k \in \mathbb{N} | i \leq k \leq j\}$ and $sup([i : j]) \triangleq j$.
- $\forall x, y \in \mathbb{R}$, $x \leq y$, we denote by
  - $[x, y] \triangleq \{z \in \mathbb{R} | x \leq z \leq y\}$, which is said to be a closed interval, right-closed, and left-closed.
  - $]x, y] \triangleq [x, y] \setminus \{x\}$, which is said to be a right-closed and left-open interval.
  - $[x, y[\triangleq [x, y] \setminus \{y\}$, which is said to be a left-closed and right-open interval.
  - $]x, y[\triangleq [x, y] \setminus \{x, y\}$, which is said to be an open interval, right-open, and left-open.
  - $sup([x, y]) \triangleq sup([x, y[) \triangleq sup(]x, y]) \triangleq sup(]x, y[) \triangleq y$, the supremum of a fixed interval.

sequences The set of finite (resp. infinite) sequences of elements of a set Σ is noted Σ^* (resp. Σ^ω).

2.1.2. Functions

Let f be a function. The domain and the range of a function f are denoted by $dom(f)$ and $range(f)$, and we use the usual notation to describe such a function, i.e. $f : \left\{ \begin{array}{lcl} dom(f) & \rightarrow & range(f) \\ x & \mapsto & f(x) \end{array} \right.$.

Definition 2 (restriction of a function)**.** *Let* f *be a function. For a set* $S \subseteq dom(f)$*, we write* $f\lceil S$ *for the restriction of* f *to* S*, i.e.* $f\lceil S : \left\{ \begin{array}{lcl} S & \rightarrow & range(f) \\ x & \mapsto & f(x) \end{array} \right.$.

Definition 3 (indicator function)**.** *Let* Ω *be a set clear in the context and* $S \subseteq \Omega$*. The indicator function of* S *is the function* $\mathbb{1}_S : \left\{ \begin{array}{lcl} \Omega & \rightarrow & \{0, 1\} \\ x & \mapsto & \left\{ \begin{array}{ll} 1 & \text{if } x \in S \\ 0 & \text{otherwise} \end{array} \right. \end{array} \right.$.

Definition 4 (parallel composition of functions)**.** *Let* $n \in \mathbb{N}$*. If* $\forall i \in [1:n]$*,* $f_i : \left\{ \begin{array}{ccc} J & \to & Q_i \\ x & \mapsto & f_i(x) \end{array} \right.$*,*

then we note $f_1||...||f_n : \left\{ \begin{array}{ccc} J & \to & Q_1 \times ... \times Q_n \\ x & \mapsto & (f_1(x), ..., f_n(x)) \end{array} \right.$*.*

Definition 5 (coordinate)**.** *If* $f : \left\{ \begin{array}{ccc} J & \to & Q_1 \times ... \times Q_n \\ x & \mapsto & (f_1(x), ..., f_n(x)) \end{array} \right.$*, then,* $\forall i \in [1:n]$*, we write*

$f \downarrow Q_i : \left\{ \begin{array}{ccc} J & \to & Q_i \\ x & \mapsto & f_i(x) \end{array} \right.$*.*

2.2. Trajectories

Most of the definitions in subsection can be found in the theory of timed input/output automata [KLSV03, KLSV06] and in its partial extension to probabilistic setting [Mit07, ML07b]. They are not used outside the Chapter 5. A trajectory is used to model the evolution of a collection of states over an interval of time.

Definition 6 (Time Axis)**.** *We define* $\mathbb{T} = \mathbb{R}^{\geq 0} \cup \{\infty\}$ *to be the time axis (*$\mathbb{R}^{\geq 0}$ *can be replaced by* $\mathbb{N}$*).*

2.2.1. Basics

Definition 7 (Trajectory)**.** *Let* Q *be a set. A* trajectory *in* Q *is a function* $\tau : J \to Q$*, where* J *is a left closed interval in* $\mathbb{T}$ *with left endpoint* 0*. The set of trajectories in* Q *is noted* $trajs(Q)$*. A trajectory* τ *with* $dom(\tau) = \{0\}$ *, and* $\tau(0) = q$*, is called the* point trajectory *at* q *and is written as* $\wp(q)$*. A trajectory* τ *is* finite *if* $dom(\tau)$ *has finite length. It is* closed *if it is finite and* $dom(\tau)$ *is right closed. The first state of* τ*, noted* $fstate(\tau)$ *is* $\tau(0)$ *and the limit time of* τ*,* $\tau.ltime$*, is* $sup\{dom(\tau)\}$*. If* τ *is closed then the limit state of* τ*, noted* $lstate(\tau)$*, is* $\tau(\tau.ltime)$*. Given a set of trajectories* $\mathcal{T}$*, we denote the subset of trajectories starting from* q *by* $\mathcal{T}(q)$*, i.e.* $\mathcal{T}(q) \triangleq \{\tau \in \mathcal{T} | fstate(\tau) = q\}$*.*

2.2.2. Time shift, prefix, suffix, concatenation

Definition 8 (Time shift)**.** *Given a trajectory* τ *and* $t \in \mathbb{T}$*, the* time shifted function $(\tau + t) : (dom(\tau) + t) \to Q$ *is defined as* $(\tau + t)(t') \triangleq \tau(t' - t)$*, for each* $t' \in \{u + t | u \in dom(\tau)\}$*.*

Definition 9 (prefix of a trajectory)**.** *Given two trajectories* τ_1 *and* τ_2*,* τ_1 *is a* prefix *of* τ_2*, written as* $\tau_1 \leq \tau_2$*, if* $\tau_1 = \tau_2 \lceil dom(\tau_1)$ *(in the sense of definition 2). The relation* $\leq$ *is obviously an order (*$\leq$ *is reflexive, antisymmetric, and transitive).*

Definition 10 (suffix of a trajectory)**.** *Given two trajectories* τ_1 *and* τ_2*,* τ_1 *is a* suffix *of* τ_2 *if* $\tau_1 = (\tau_2 \lceil [t, \infty]) - t$*, for some* $t \in dom(\tau_2)$*.*

Definition 11 (concatenation of trajectories)**.** *If* τ_1 *is a closed trajectory with* $\tau_1.ltime = t$ *and* $fstate(\tau_2) = lstate(\tau_1)$*, then the function* $\tau_1 ^\frown \tau_2 : dom(\tau_1) \cup (dom(\tau_2) + t) \to X$ *is defined as* $\tau_1(t)$ *if* $t \leq u$ *and* $\tau_2(t - u)$ *otherwise.*

2.2.3. Deteministic Trajectories

Definition 12 (deterministic set of trajectories)**.** *A set of trajectories* $\mathcal{T}$ *is* deterministic *if for every* $q \in Q$*,* $\tau_1, \tau_2 \in \mathcal{T}(q)$*, either* $\tau_1 \leq \tau_2$ *or* $\tau_2 \leq \tau_1$*, i.e. if* $(\mathcal{T}(q), \leq)$ *is a total order.*

Definition 13 (maximal trajectory)**.** *Let* $\mathcal{T}$ *be a set of trajectories. A trajectory* τ *in* $\mathcal{T}$ *is said to be* maximal *if it is the supremal element of* $\mathcal{T}(fstate(\tau))$*. Obviously, a set of trajectories of the form* $\mathcal{T}(q)$ *contains at most one maximal trajectory.*

Definition 14 (maxtime and maxstate)**.** *Let* Q *be a set, let* $\mathcal{T}$ *be a set of trajectories in* Q*. We define:*

$$maxtime_{\mathcal{T}} : \left\{ \begin{array}{ccc} Q & \to & \mathbb{T} \\ q & \mapsto & \left\{ \begin{array}{ll} \tau.ltime & \text{if there exists a closed maximal trajectory } \tau \in \mathcal{T}(q) \\ \infty & \text{otherwise} \end{array} \right. \end{array} \right. .$$

Similarly, we define:

$$maxstate_{\mathcal{T}} : \begin{cases} Q & \to & Q \\ q & \mapsto & \begin{cases} lstate(\tau) & \text{if there exists a closed maximal trajectory } \tau \in \mathcal{T}(q) \\ q & \text{otherwise} \end{cases} \end{cases} .$$

2.2.4. Trajectories sets composition and isomorphism

The following definitions are used to define the constraint of trajectory preservation for PTCA, which are automata that model dynamic probabilistic timed systems.

Definition 15 (Trajectories with same domain)**.** *Let Q_1, Q_2 be sets. Let $(\mathcal{T}_1, \mathcal{T}_2) \in trajs(Q_1) \times trajs(Q_2)$. We note $\mathcal{T}_1 \circledast \mathcal{T}_2 = \{(\tau_1, \tau_2) \in \mathcal{T}_1 \times \mathcal{T}_2 | dom(\tau_1) = dom(\tau_2)\}$.*

Definition 16 (Composition of sets of trajectories)**.** *Let Q_1, Q_2 be sets. Let $(\mathcal{T}_1, \mathcal{T}_2) \in trajs(Q_1) \times trajs(Q_2)$. We note $\mathcal{T}_1 || \mathcal{T}_2 \triangleq \{(\tau_1 || \tau_2) | (\tau_1, \tau_2) \in \mathcal{T}_1 \circledast \mathcal{T}_2\} = \{\tau \in trajs(Q_1 \times Q_2) |\ \tau \downarrow Q_i \in \mathcal{T}_i \text{ for } i \in \{1, 2\}\}$ (in the sense of definitions 4 and 5).*

Definition 17 (Homomorphism between sets of trajectories)**.** *Let Q_1, Q_2 be sets. Let $(\mathcal{T}_1, \mathcal{T}_2) \in trajs(Q_1) \times trajs(Q_2)$. Let $u : \mathcal{T}_1 \to \mathcal{T}_2$, $v : Q_1 \to Q_2$. We note $\mathcal{T}_1 \overset{u,v}{\rightsquigarrow} \mathcal{T}_2$ if u is a bijection from $\mathcal{T}_1$ to $\mathcal{T}_2$ such that for every pair (τ_1, τ_2) with $\tau_1 = u(\tau_2)$: (i) $dom(\tau_1) = dom(\tau_2) \triangleq J$ and (ii) $\forall t \in J$, $v(\tau_1(t)) = \tau_2(t)$.*

2.3. Probability Theory

We use standard measure theory, like in [Dud04]. An intuitive understanding of probability theory is sufficient to read the thesis. However, we will define probability over the uncountable set of executions of our different automata. Hence, some precautions are required to rigorously define such a probabilistic space. The Chapter 5 extends a lot of concepts of the Chapter 3 with uncountable sets instead of countable sets. Such an extension also requires some additional precautions, but here again, it is possible to read the thesis assuming that all probabilistic spaces and all operators on them stay well-defined.

2.3.1. Set Algebra

The σ-algebra is the key object for rigorously defining measurable space. Some weaker algebraic structures are useful for construction perspectives.

Definition 18 (semi-ring)**.** *A* semi-ring *on a set Ω is a set $\mathscr{C} \subseteq \mathcal{P}(\Omega)$ such that:*

- $\emptyset \in \mathscr{C}$
- $A, B \in \mathscr{C} \implies A \cap B \in \mathscr{C}$
- $A, B \in \mathscr{C} \implies \exists n \in \mathbb{N}^{>0}, (C_i)_{i\in[1:n]} \in \mathscr{C}^{[1:n]}, A \setminus B = \biguplus_{i=1}^{n} C_i$

Definition 19 (ring)**.** *A* ring *on a set Ω is a set $\mathscr{C} \subseteq \mathcal{P}(\Omega)$ such that:*

- $\emptyset \in \mathscr{C}$
- $A, B \in \mathscr{C} \implies A \cap B \in \mathscr{C}$
- $A, B \in \mathscr{C} \implies A \setminus B \in \mathscr{C}$

It is easy to show that a semi-ring on a set Ω is a ring on a set Ω.

Definition 20 (semi-field)**.** *A* semi-field *on a set Ω is a semi-ring on Ω that contains Ω.*

Definition 21 (field)**.** *A* field *on a set Ω is a ring on Ω that contains Ω.*

Definition 22 (σ-algebra)**.** *A* σ-algebra *(sometimes noted sigma-algebra when σ is already used) on Ω is a set $\mathscr{C} \subseteq \mathcal{P}(\Omega)$ such that:*

- $\emptyset, \Omega \in \mathscr{C}$
- $A \in \mathscr{C} \implies \Omega \setminus A \in \mathscr{C}$
- $(A_i)_{i\in\mathbb{N}} \in \mathscr{C}^{\mathbb{N}} \implies \bigcup_{i\in\mathbb{N}} A_i \in \mathscr{C}$

It is easy to show that a σ-algebra is a field stable by countable union (and so by countable intersection).

Definition 23 (*field*($\mathscr{C}$))**.** *The field generated by a family of sets $\mathscr{C}$, denoted by field($\mathscr{C}$) is the smallest field that contains $\mathscr{C}$. The family $\mathscr{C}$ is called a generator for field($\mathscr{C}$).*

Definition 24 (*sigma*($\mathscr{C}$))**.** *The σ-field generated by a family of sets $\mathscr{C}$, denoted by sigma($\mathscr{C}$) is the smallest σ-field that contains $\mathscr{C}$. The family $\mathscr{C}$ is called a generator for sigma($\mathscr{C}$).*

A trivial property of a generator $\mathscr{C}$ is $\mathit{sigma}(\mathscr{C}) = \mathit{sigma}(\mathit{field}(\mathscr{C}))$.

The field generated by a family of sets can be obtained following a simple procedure.

Proposition 1 (field generation)**.** *Let Ω be a set. Let $\mathscr{C} \subseteq \mathcal{P}(\Omega)$.*

1. *Let $F_1(\mathscr{C})$ be the family containing $\emptyset, \Omega$ and every $C \subseteq \Omega$ such that $C \in \mathscr{C}$ or $(\Omega \setminus C) \in \mathscr{C}$.*
2. *Let $F_2(\mathscr{C})$ be the family containing all finite intersections of elements of $F_1(\mathscr{C})$.*
3. *Let $F_3(\mathscr{C})$ be the family containing all finite unions of disjoint elements of $F_2(\mathscr{C})$.*

Then field$(\mathscr{C}) = F_3(\mathscr{C})$.

This procedure will be used to construct a σ-algebra for our set of executions in Section 3.6.

2.3.2. Measure Spaces

Definition 25 (measure)**.** *Let $\mathscr{C}$ be a set of sets with $\emptyset \in \mathscr{C}$. We call* measure *on $\mathscr{C}$, any map $\eta : \mathscr{C} \to [0, \infty]$ with the following properties:*

- $\eta(\emptyset) = 0$
- $(A_i)_{i\in\mathbb{N}} \in \mathscr{C}^{\mathbb{N}}, A = \biguplus_{i\in\mathbb{N}} A_i, A \in \mathscr{C} \implies \eta(A) = \sum_{i\in\mathbb{N}} \eta(A_i)$

Let us note that if $\mathscr{C}$ is a σ-algebra on a set Ω, the precondition $A \in \mathscr{C}$ is necessarily verified in the second item.

Definition 26 (measurable space)**.** *A* measurable space *is a pair $(\Omega, \mathcal{F}_\Omega)$ where $\mathcal{F}_\Omega$ is a σ-algebra over Ω.*

Definition 27 (measure space)**.** *A* measure space *is a triplet $(\Omega, \mathcal{F}_\Omega, \eta)$ where η is a measure on $\mathcal{F}_\Omega$ and $(\Omega, \mathcal{F}_\Omega)$ is a measurable space. In this case, we say that η is a measure on $(\Omega, \mathcal{F}_\Omega)$.*

Definition 28 (probability and sub-probability)**.** *A* probability measure *(resp.* sub-probability measure*) on a measurable space $(\Omega, \mathcal{F}_\Omega)$ is a measure on $(\Omega, \mathcal{F}_\Omega)$ s. t. $\eta(\Omega) = 1$ (resp. $\eta(\Omega) \leq 1$). A* probability space *(resp.* sub-probability space *) is a measure space $(\Omega, \mathcal{F}_\Omega, \eta)$ such that η is a probability (resp. sub-probability) measure on $(\Omega, \mathcal{F}_\Omega)$. The set Ω is called the sample space, and the elements of $\mathcal{F}_\Omega$ are called events. The set of probability (resp. sub-probability) measures over $(\Omega, \mathcal{F}_\Omega)$ is denoted by $\mathcal{P}rob(\Omega, \mathcal{F}_\Omega)$ resp $Sub\mathcal{P}rob(\Omega, \mathcal{F}_\Omega)$.*

2.3.3. Measurable Function

Definition 29 (measurable function)**.** *Let $(\Omega, \mathcal{F}_\Omega)$ and $(\Omega', \mathcal{F}_{\Omega'})$ be two measurable spaces. A function $f : \Omega \to \Omega'$ is said to be a* measurable function *from $(\Omega, \mathcal{F}_\Omega)$ to $(\Omega', \mathcal{F}_{\Omega'})$ if $\forall E' \in \mathcal{F}_{\Omega'}$, $f^{-1}(E) \in \mathcal{F}_\Omega$.*

The two next propositions are classic results in measure theory [Dud04].

Proposition 2. *Let $(\Omega, \mathcal{F}_\Omega)$ and $(\Omega', \mathcal{F}_{\Omega'})$ be two measurable spaces. If f is a measurable function from $(\Omega, \mathcal{F}_\Omega)$ to $(\Omega', \mathcal{F}_{\Omega'})$ and η is a measure on $\mathcal{F}_\Omega$, then the function $g : E' \in \mathcal{F}_{\Omega'} \mapsto \eta(f^{-1}(E'))$ is a measure on $\mathcal{F}_{\Omega'}$, called the* image measure *of η under f.*

Proposition 3. *Let $(\Omega, \mathcal{F}_\Omega)$ and $(\Omega', \mathcal{F}_{\Omega'})$ be two measurable spaces and let $\mathscr{C}$ be a generator of $\mathcal{F}_{\Omega'}$. Let $f : \Omega \to \Omega'$. Then f is measurable if $\forall C \in \mathscr{C}$, $f^{-1}(C) \in \mathcal{F}_\Omega$.*

2.3.4. Product of Measure Spaces

Definition 30 (Product of Measure Spaces). *Let $(\Omega_1, \mathcal{F}_{\Omega_1}, \eta_1)$ and $(\Omega_2, \mathcal{F}_{\Omega_2}, \eta_2)$ be to measure spaces. The product measure space $(\Omega_1, \mathcal{F}_{\Omega_1}, \eta_1) \otimes (\Omega_2, \mathcal{F}_{\Omega_2}, \eta_2)$ is the measure space $(\Omega_1 \times \Omega_2, \mathcal{F}_{\Omega_1} \otimes \mathcal{F}_{\Omega_2}, \eta_1 \otimes \eta_2)$, where $\mathcal{F}_{\Omega_1} \otimes \mathcal{F}_{\Omega_2} = sigma(\{A \times B | (A, B) \in \mathcal{F}_{\Omega_1} \times \mathcal{F}_{\Omega_2}\})$ is the smallest σ-algebra generated by sets of the form $A \times B$ for all $A \in \mathcal{F}_{\Omega_1}, B \in \mathcal{F}_{\Omega_2}$ and $\eta_1 \otimes \eta_2$ is the unique measure s.t. for every $C_1 \in \mathcal{F}_{\Omega_1}, C_2 \in \mathcal{F}_{\Omega_2}$, $\eta_1 \otimes \eta_2(C_1 \times C_2) = \eta_1(C_1) \cdot \eta_2(C_2)$.*

2.3.5. Discrete probability space

Definition 31 (countable set). *A set Ω is countable if there exists a bijection $bij : \Omega \to \mathbb{N}$. A set is uncountable if it is not countable.*

Definition 32 (Discrete measure). *A discrete measure on a set Ω is a measure η on $(\Omega, \mathcal{P}(\Omega))$, such that, for each $C \subset \Omega, \eta(C) = \sum_{c \in C} \eta(\{c\})$.*

Definition 33 (Discrete probability). *We define $Disc(\Omega)$ (resp. $SubDisc(\Omega)$) to be the set of discrete probability (resp. sub-probability) measures on Ω.*

Definition 34 (Dirac). *Given set Ω and a subset $C \subset \Omega$, the Dirac measure δ_C is the discrete probability measure on Ω that assigns probability 1 to C. For each element $s \in \Omega$, we note δ_s for $\delta_{\{s\}}$. In the sequel, we often omit the set notation when we denote the measure of a singleton set.*

2.3.6. Measure Spaces Isomorphism

It is convenient to exclude events of zero-probability in our reasoning. Hence, the notion of support that represents the set of events with non-zero measure is useful.

Definition 35 (support). *Let $(\Omega, \mathcal{F}_\Omega, \eta)$ be a measure space. The support of η, is the set noted $supp(\eta) \triangleq \overline{\{F \in \mathcal{F}_\Omega | \eta(F) \neq 0\}}$, where the bar denotes the set closure.*

In this thesis, to be as close as possible to the notation used in related works, we omit the singleton notation and we note $\omega \in supp(\eta)$ to mean $\omega \in \Omega$ s.t. $\{\omega\} \in supp(\eta)$.

The next definition is used in Chapter 3 to define (in definition 68) PCA, which are automata that model dynamic probabilistic systems. Then, this definition is amply in chapters 3 and 4 used during the thesis to deduce a wide range of intermediate results.

Definition 36 (Measure Spaces Isomorphism). *Let $(\Omega_1, \mathcal{F}_{\Omega_1}, \eta_1)$ and $(\Omega_2, \mathcal{F}_{\Omega_2}, \eta_2)$ be to measure spaces. Let $f : \Omega \to \Omega_2$. We note $\eta_1 \overset{f}{\leftrightarrow} \eta_2$ if the following is verified: (1) the restriction $\tilde{f}$ of f to $supp(\eta_1)$ is a bijection from $supp(\eta_1)$ to $supp(\eta_2)$ and (2) $\forall s \in supp(\eta_1)$, $\eta_1(s) = \eta_2(f(s))$.*

2.3.7. Integration

The notion of integration is not used outside Chapter 5.

Definition 37 (simple function). *A simple function $\hat{f}$ on a measureable space $(S, \mathcal{F}_S)$ is a function of the form $\sum_{i \in [1:n]} c_i \mathbb{1}_{C_i}$ with $\forall i \in [1:n], c_i \in \mathbb{R} \wedge C_i \in \mathcal{F}_S$ where $\mathbb{1}_{C_i}$ is the indicator function of set $C_i \triangleq \{s \in dom(\hat{f}) | \hat{f}(s) = c_i\}$. It means the range of a simple function consists of only finitely many points: $range(\hat{f}) = \{c_1, ..., c_n\}$.*

Definition 38 (integration of a simple function)**.** *Let η be a measure on $(S, \mathcal{F}_S)$, let $\hat{f} = \sum_{i \in [1:n]} c_i \mathbb{1}_{C_i}$ be a simple function on $(S, \mathcal{F}_S)$ and $E \in \mathcal{F}_S$, then $\int_E \hat{f} \, \mathrm{d}\eta = \sum_{i \in [1:n]} c_i \cdot \eta(C_i \cap E)$.*

Definition 39 (integration of a measurable function)**.** *Consider a measureable space $(S, \mathcal{F}_S)$ and a measure η on $(S, \mathcal{F}_S)$. If $f : S \to \mathbb{R}^{\geq 0}$ is measurable and $E \in \mathcal{F}_S$ the Lebesgue integral of f over E is defined as $\int_E f \, \mathrm{d}\eta = \sup_{\hat{f} \leq f} \int_E \hat{f} \, \mathrm{d}\eta$ where the supremum is take over simple functions $\hat{f}$. If the measure η is clear in the context we note $\int_E f = \int_E f \, \mathrm{d}\eta$.*

2.3.8. Carathéodory's Extension Theorem

The next theorem [Dud04] says that it is possible to define a probability measure η on a measurable space $(\Omega, \mathcal{F}_\Omega)$ by specifying η only on a generator of $\mathcal{F}_\Omega$.

Theorem 1 (Carathéodory's Extension Theorem)**.** *Consider a set Ω, a semi-ring $\mathscr{C}$ on Ω and a measure η' on $\mathscr{C}$. There exists a unique measure η extending η' on the σ-field generated by $\mathscr{C}$ (noted sigma($\mathscr{C}$)).*

2.4. Turing Machines

This section aims to provide a formal definition of Turing Machine. This notion is not used outside of the Chapter 6 that handles simulation-based security which is necessarily based on a computational model. However, a vague understanding of Turing Machines is largely enough to use the framework of the Chapter 6. Indeed, those notions are only used to define what is a bounded automaton and a few high level lemma to show that the composition of two automata stays bounded. Thereafter, those notions are encapsulated in the definition of bounded-automata.

Turing machines are an abstract model of computation. They provide a precise, formal definition of what it means for a function to be computable in a bounded fashion.

2.4.1. Deterministic Turing Machine

Definition 40 (Deterministic Turing Machine)**.**

A deterministic Turing machine M is a tuple $(\Sigma, Q, \bar{q}, F, D, \delta)$, such that:

- *Σ the finite alphabet of symbols that can be read on the tape of the deterministic Turing machine. The two special elements $\triangleleft, \sqcup \in \Sigma$ represent the start and end of the tape. In Chapter 6, we require that $\Sigma = \{0, 1, \triangleleft, \sqcup\}$.*
- *Q a finite set of states*
- *$\bar{q} \in Q$ the start state*
- *$F = \{halt, yes, no\} \subset Q$ the finite set of final states. Intuitively, the machine halts if it is in a state in F and the interpreted returned value is either "yes" (resp. "no") if the machine is in state yes (resp. no) or the bit string $\bar{x}$ on the tape (without special symbols $\triangleleft, \sqcup$) if the machine is in state halt*
- *$D = \{\leftarrow, \rightarrow\}$ the possible ways to move in relation to the tape. Intuitively, triggering $\leftarrow$ (resp. $\rightarrow$ means moving to the left (resp. right) of the tape.*
- *$\delta : Q \times \Sigma \to Q \times \Sigma \times D$ the transition function where $\delta(q, s) = (q', s', d)$ intuitively means that when the machine is in state q and read the symbol s, it writes the symbol s' on its tape, moves in relation to the tape according to d, and jumps into state q'. It is assumed that the machine never tries to overwrite the leftmost symbol on its tape nor to move to the left of it, that is $\forall q \in Q, \delta(q, \triangleleft) = (q', s', d)$ implies $d \neq \leftarrow$ and $s' = \triangleleft$. The set of Turing Machines is noted TMs*

Definition 41 (Configuration of deterministic Turing Machine)**.** *A configuration of a deterministic Turing machine $M = (\Sigma, Q, \bar{q}, F, D, \delta)$ is a triple $(x, q, k) \in \Sigma^* \times Q \times \mathbb{N}$ where:*

- *$x = x_0, x_1, ...x_{n-1}$ denotes the string on the tape, where n denotes he length of x, noted $|x|$. The string x is required to begin with $\triangleleft$ and end with $\sqcup$. We note $\bar{x} = x_1, ..., x_{n-2}$ the string written in the tape without the special symbols $\triangleleft, \sqcup$.*
- *q denotes the machine's current state,*
- *k denotes the position of the machine on the tape, which is required to satisfy $0 \leq k < |x|$.*

The set of configurations of a deterministic Turing Machine M is noted $Confs(M)$.

Definition 42 (Step of Turing Machine)**.** *A step of a deterministic Turing Machine $M = (\Sigma, Q, \bar{q}, F, D, \delta)$ captures the jump from one configuration to another one, respecting the constraint imposed by the transition function. Thus, a step of M is a pair $((x, q, k), (x', q', k')) \in Confs(M)^2$, such that M jump from the state q to the state q', i.e. the equality $\delta(q, x_k) = (q', s', d)$ holds for some s' and d, and verifies the following constraints, whose associated relation is noted $(x, q, k) \xrightarrow{s', d} (x', q', k')$:*

- *the string x' is obtained from x by changing x_k to s', and also appending $\sqcup$ to the end of x, if the machine overwrites on the end of the tape, i.e.:*
  - *if $k < |x| - 1$ or $k = |x| - 1 \wedge s' = \sqcup$, then $x' = x_0, x_1, ..., x_{k-1}, s', x_{k+1}, ..., x_{n-1}$*
  - *if $k = |x| - 1 \wedge s' \neq \sqcup$, then $x' = x_0, ..., x_{n-2}, s', \sqcup$*
- *the machine moves in relation to the tape according to d, i.e. $k = k + 1$ if $d = \rightarrow$ and $k = k - 1$ if $d = \leftarrow$.*

The set of steps of a Turing Machine M is noted $Steps(M)$.

Definition 43 (Computation of a Turing Machine)**.** *A computation of a deterministic Turing Machine $M = (\Sigma, Q, \bar{q}, F, D, \delta)$ is a (potentially infinite) sequence of configurations $\alpha = C^0, ..., C^i, ... \in Confs(M)^* \cup Confs(M)^\omega$ such that:*

- *The first configuration $C^0 = (q^0, x^0, k^0)$ is a valid starting configuration, i.e. $q^0 = \bar{q}$ and $k^0 = 0$.*
- *each pair of consecutive configuration is a step of M, i.e. $\forall i \in \mathbb{N}, 0 \leq i < |\alpha| - 1$, $(C^i, C^{i+1}) \in Steps(M)$.*
- *We can remark that previous constraint implictly implies that $\forall i, 0 \leq i < |\alpha| - 1$, $C^i = (q^i, x^i, k^i)$ is not a final configuration i.e. $q^i \notin F$.*
- *if $|\alpha| \neq \infty$, we note $f = |\alpha| - 1$ and the last configuration $C^f = (q^f, x^f, k^f)$ is a valid final configuration, i.e. $q^f \in F$. In that case, we say that M halts in time $|\alpha| - 1$.*
  - *If $q^f = halt$, we say that M returns $\bar{x}^f$, i.e. M returns the bit string written in the tape without the special symbols $\triangleleft, \sqcup$.*
  - *If $q^f = yes$, we say that M returns "yes" and recognizes x^0.*
  - *If $q^f = no$, we say that M returns "no" and does not recognize x^0.*

  We note $returned(\alpha)$ the returned value of α if α is finite and $\nearrow$ otherwise.

The set of computations of a deterministic Turing Machine for a fixed entry x^0 is noted $Computations(M, x_0)$. It is easy to remark that the computation of a deterministic Turing Machine is unique if we fix x^0, i.e. $Computations(M, x_0)$ is a singleton. Hence, we note $returned(M, x^0)$ the value $returned(\alpha)$ for the unique computation α of $Computations(M, x_0)$.

2.4.2. Probabilistic Turing Machine

Definition 44 (Probabilistic Turing Machine)**.**

A probabilistic Turing machine M is a tuple $(\Sigma, Q, \bar{q}, F, D, \delta_0, \delta_1)$, such that both $(\Sigma, Q, \bar{q}, F, D, \delta_0, \delta_1)$ and $(\Sigma, Q, \bar{q}, F, D, \delta_0)$ are deterministic Turing Machines. The intuition is that the machine is equipped with an additional read-only tape with symbols in $\{0, 1\}$ that provides an external source of randomness, used by the machine to choose between δ_0 and δ_1 with probability $1/2$.

A lot of equivalent alternative definitions are possible, for example we could have a machine $M' = (\Sigma, Q, \bar{q}, F, D, \delta_{01})$, where $(\Sigma, Q, \bar{q}, F, D)$ is defined as in definition 44, with $\delta : Q \times \Sigma \times \{0, 1\} \rightarrow Q \times \Sigma \times D$ such that $\forall (q, s) \in Q \times \Sigma$, $\delta_{01}(q, s, 0) = \delta_0(q, s)$ and $\delta_{01}(q, s, 1) = \delta_1(q, s)$. The intuition is more or less the same, the next random bit of the read-only random tape is used as an argument of δ_{01}.

Another alternative definition would be a tuple $M'' = (\Sigma, Q, \bar{q}, F, D, \delta_p)$, where $(\Sigma, Q, \bar{q}, F, D)$ is defined as in definition 44 and $\delta_p : Q \times \Sigma \rightarrow Disc(Q \times \Sigma \times D)$ yields a discrete probabilistic distribution over $Q \times \Sigma \times D$. However, such a definition is less convenient to define encoding.

The definition of a configuration of a probabilistic Turing Machine is the same as the one of a deterministic Turing Machine.

Definition 45 (Configuration of probabilistic Turing Machine)**.** *A configuration of a probabilistic Turing machine $M = (\Sigma, Q, \bar{q}, F, D, \delta_0, \delta_1)$ is a configuration of $M = (\Sigma, Q, \bar{q}, F, D, \delta_0)$ in the sense of definition 45. Again, we note Confs(M) the set of configurations of M.*

The definition of a step of a probabilistic Turing Machine is very close to the one of a deterministic Turing Machine.

Definition 46 (Step of probabilistic Turing machine)**.** *A step of a probabilistic Turing Machine $M = (\Sigma, Q, \bar{q}, F, D, \delta_0, \delta_1)$ captures a possible jump from one configuration to another one, respecting the constraint imposed by the transition functions. Thus, a step of M is a pair $((x, q, k), (x', q', k')) \in Confs(M)^2$, such that there exists $(q', s', d) \in \{\delta_0(q, x_k), \delta_1(q, x_k)$ for some s' and d, and verifies $(x, q, k) \xrightarrow{s', d} (x', q', k')$ (in the sense of definition 42) as for a deterministic Turing machine.*

The definition of a computation of a probabilistic Turing Machine is the same as the one of a deterministic Turing Machine.

Definition 47 (Computation of a probabilistic Turing Machine)**.** *A computation of a probabilistic Turing Machine $M = (\Sigma, Q, \bar{q}, F, D, \delta_0, \delta_1)$ is a (potentially infinite) sequence of configurations $\alpha = C^0, ..., C^i, ... \in Confs(M)^* \cup Confs(M)^\omega$ such that:*

- *The first configuration $C^0 = (q^0, x^0, k^0)$ is a valid starting configuration, i.e. $q^0 = \bar{q}$ and $k^0 = 0$.*
- *each pair of consecutive configuration is a step of M, i.e. $\forall i \in \mathbb{N}, 0 \leq i < |\alpha| - 1$, $(C^i, C^{i+1}) \in Steps(M)$.*
- *We can remark that previous constraint implictly implies that $\forall i, 0 \leq i < |\alpha| - 1$, $C^i = (q^i, x^i, k^i)$ is not a final configuration i.e. $q^i \notin F$.*
- *if $|\alpha| \neq \infty$, we note $f = |\alpha| - 1$ and the last configuration $C^f = (q^f, x^f, k^f)$ is a valid final configuration, i.e. $q^f \in F$. In that case, we say that M halts in α in time $|\alpha| - 1$.*
  - *If $q^f = halt$, we say that M returns $\bar{x}^f$, i.e. M returns the bit string written in the tape without the special symbols $\triangleleft, \sqcup$.*
  - *If $q^f = yes$, we say that M returns "yes" and recognizes x^0.*
  - *If $q^f = no$, we say that M returns "no" and does not recognize x^0.*

  We note returned(α) the returned value of α if α is finite and $\nearrow$ otherwise.

The set of computations of a probabilistic Turing Machine for a fixed entry x^0 is noted Computations(M, x_0). However, if we fix the input x^0, this computation is not necessarily unique. Thus, we note returned(M, x^0) the set of values $\{returned(\alpha) | \alpha \in Computations(M, x_0)\}$ for the unique computation α of Computations(M, x_0).

Moreover, we say that a probabilistic Turing Machine M returns a value in time at most b for an input x^0, if for every computation of M starting with configuration $(\bar{q}, x^0, 0)$, M returns a value in time at most b.

2.4.3. Encoding

2.4.3.1. Encoded Sets

Definition 48 (encoding)**.** *An encoding of a set Q is injective function $\langle . \rangle^Q$:* $\left\{ \begin{array}{ccc} Q & \to & \{0,1\}^* \\ q & \mapsto & \langle q \rangle^Q \end{array} \right.$.

An encoded set $(Q, \langle . \rangle^Q)$ is a set Q equipped with an encoding of Q.

When the set in clear in the context, we note $\langle . \rangle$ for $\langle . \rangle^Q$.

Definition 49 (encoding for integers)**.** *We note* $\langle . \rangle^{\mathbb{N}} : \begin{cases} \mathbb{N} & \to & \{0,1\}^* \\ n & \mapsto & b_{k-1}...b_1b_0 \text{ s.t. } n = \sum_{i \in [0:k-1]} b_i \cdot 2^i \end{cases}$.

Definition 50 (bounded encoded set)**.** *Let* $b \in \mathbb{N}$*. An encoded set* $(Q, \langle . \rangle^Q)$ *is* b*-bounded if* $\forall q \in Q$*,* $|\langle q \rangle^Q| \leq b$*.*

Proposition 4. *For every finite set* Q *of size* $|Q| \leq 2^k$*, there exists an encoding of* Q*, with* k *bits only.*

Proof. We note $Q = \{q_1, ..., q_{|Q|}\}$. We can choose $\langle . \rangle^Q$, such that $\forall n \in [1 : |Q|]$, $\langle q_n \rangle^Q = \langle n \rangle^{\mathbb{N}}$. □

The next function is a classic simple tool to avoid ambiguity when we "compose" several encodings.

Definition 51 (double)**.** *The function double:* $\{0,1\}^* \to \{0,1\}^*$ *is the function that maps each sequence of* k*-bits* $b_{k-1}b_{k-2}...b_0$ *to the sequence of bits* $b_{k-1}b_{k-1}b_{k-2}b_{k-2}...b_0b_0$ *where each bit is repeated twice.*

The idea is two write bit-strings of the form $\underline{b} = double(\underline{b}^1)01double(\underline{b}^2)$ such that there is a unique way to extract $\underline{b}^1$ and $\underline{b}^2$ from $\underline{b}$ since we can browse the bit-string 2 by 2 until reaching the bit-string 01.

This well-known trick allows us to encode easily a Turing Machine. Also it allows obtaining the two following trivial propositions that will be used in subsection 6.1.2 to show that the composition of two bounded automata stay bounded with a linear factor.

Proposition 5. *Let* $b_1, b_2 \in \mathbb{N}$*. Let* $(Q_1, \langle . \rangle^{Q_1})$ *be a* b_1*-bounded encoded set and* $(Q_2, \langle . \rangle^{Q_2})$ *be a* b_2*-bounded encoded set. Let* $Q_{12} = Q_1 \times Q_2$*. Let* $\langle . \rangle^{Q_{12}} : \begin{cases} Q_1 \times Q_2 & \to & \{0,1\}^* \\ (q_1, q_2) & \mapsto & double(\langle q_1 \rangle^{Q_1})01double(\langle q_2 \rangle^{Q_2}) \end{cases}$*. Then* $(Q_{12}, \langle . \rangle^{Q_{12}})$ *is a* $2 \cdot (b_1 + b_2 + 1)$*-bounded encoded set.*

Proof. Let $q_{12} = (q_1, q_2), q'_{12} = (q'_1, q'_2) \in Q_{12}$, $\langle q_{12} \rangle^{Q_{12}} = \langle q'_{12} \rangle^{Q_{12}} = \underline{b}$. Because of property of double function, there exists a unique sequence of bits 01 in $\underline{b}$. Thus $\underline{b} = \underline{b}_1 01 \underline{b}_2$ with $\langle q_1 \rangle^{Q_1} = \langle q'_1 \rangle^{Q_1} = \underline{b}_1$ and $\langle q_2 \rangle^{Q_2} = \langle q'_2 \rangle^{Q_2} = \underline{b}_2$. The injectivity of both $\langle . \rangle^{Q_1}$ and $\langle . \rangle^{Q_2}$ terminates the proof. □

In the same way:

Proposition 6. *Let* $b_1, b_2 \in \mathbb{N}$*. Let* $(Q_1, \langle . \rangle^{Q_1})$ *be a* b_1*-bounded encoded set and* $(Q_2, \langle . \rangle^{Q_2})$ *be a* b_2*-bounded encoded set. Let* $Q_{12} = Q_1 \cup Q_2$*.*

Let $\langle . \rangle^{Q_{12}} : \begin{cases} Q_1 \cup Q_2 & \to & \{0,1\}^* \\ q & \mapsto & \begin{cases} 0\langle q \rangle^{Q_1} \text{ if } q \in Q^1 \\ 1\langle q \rangle^{Q_2} \text{ otherwise} \end{cases} \end{cases}$.

Then $(Q_{12}, \langle . \rangle^{Q_{12}})$ *is a* $(max(b_1, b_2) + 1)$*-bounded encoded set.*

Proof. Let $q, q' \in Q_{12}$, $\langle q \rangle^{Q_{12}} = \langle q' \rangle^{Q_{12}} = b'\underline{b}$ with $b' \in \{0,1\}$ and $\underline{b} \in \{0,1\}^*$. If $b' = 0$, then $\langle q \rangle^{Q_1} = \langle q' \rangle^{Q_1} = \underline{b}$ and by injectivity of $\langle . \rangle^{Q_1}$, $q = q'$. If $b' = 1$, then $\langle q \rangle^{Q_2} = \langle q' \rangle^{Q_2} = \underline{b}$ and by injectivity of $\langle . \rangle^{Q_2}$, $q = q'$. □

2.4.3.2. Encoding of Turing Machines

The practical feasibility of a concrete Turing Machine might depends of its size in a sense that remains to be defined. Such a notion of size is usually defined by the number of bits of the string x returned by an encoding function $encoding : TMs \to \{0,1\}^*$. Since, this encoding function is not unique, we gives a possible example, which is supposed to be a reasonable tradeoff between simplicity and efficiency.

We stress that we can choose any (potentially more efficient) alternative encoding for this definition.

Definition 52 (encoding (arbitrary))**.** *Let $M = (\Sigma, Q, \bar{q}, F, D, \delta)$ be a Turing Machine where Q, Σ and D are finite encoded sets with $|Q| = i$, $|\Sigma| = j$ and $|D| = 2$. Let (q, s, q', s', d) such that $\delta(q, s) = (q', s', d)$. The bit-string representation of (q, s, q', s', d) according to M, knowing i, j is the sequence of bits $\langle q, s, q', s', d\rangle_{i,j}^{M} \triangleq \langle q\rangle\langle s\rangle\langle q'\rangle\langle s'\rangle\langle d\rangle$. The bit-string representation of M, knowing i, j is the sequence of bits $encoding_{i,j}(M) =$*

$\langle q_{\ell_1}, s_{\ell_1}, q'_{\ell_1}, s'_{\ell_1}, d_{\ell_1}\rangle_{i,j}^{M}$

$\langle q_{\ell_2}, s_{\ell_2}, q'_{\ell_2}, s'_{\ell_2}, d_{\ell_2}\rangle_{i,j}^{M}$

...

$\langle q_{\ell_L}, s_{\ell_L}, q'_{\ell_L}, s'_{\ell_L}, d_{\ell_L}\rangle_{i,j}^{M}$

where $(q_\ell, s_\ell, q'_\ell, s'_\ell, d_\ell)$ denotes the ℓ-th line in the transition table of M, representing the $L \leq |Q| \cdot |\Sigma|$ possible transitions (q, s, q', s', d) such that $\delta(q, s) = (q', s', d)$.

The bit string representation of a deterministic Turing Machine M is the sequence of bits $encoding(M) \triangleq double(\langle i\rangle^{\mathbb{N}})$ 01 $double(\langle j\rangle^{\mathbb{N}})$ 01 $encoding_{i,j}(M)$ with $i = \lceil \log_2(|Q|)\rceil$ and $j = (\lceil \log_2(|\Sigma|)\rceil$

The bit string representation of a probabilistic Turing Machine $M = (\Sigma, Q, \bar{q}, F, D, \delta_0, \delta_1)$ is the sequence of bits $encoding(M) \triangleq$ 0 $ones(i)$ 0 $ones(j)$ 0 $encoding_{i,j}(M_0)$ $encoding_{i,j}(M_1)$ with $i = \lceil \log_2(|Q|)\rceil$ and $j = (\lceil \log_2(|\Sigma|)\rceil$, $M_0 = (\Sigma, Q, \bar{q}, F, D, \delta_0)$ and $M = (\Sigma, Q, \bar{q}, F, D, \delta_1)$.

Definition 53 (bit-representation-size)**.** *Let M be a (deterministic or probabilistic) Turing Machine. Its bit-representation size is the number of bits of its encoding $encoding(M)$ for the encoding function encoding described above.*

Definition 54. *Let $b \in \mathbb{N}$. A (deterministic or probabilistic) Turing Machine M is b-bounded if*

- *its bit-representation size $s \leq b$ and*
- *for every input $x^0 \in \{0,1\}^*$, for every computation $\alpha \in Computations(M, x^0)$, $|\alpha| \leq b$.*

2.4.4. Decision

The next definitions make formal the definitions 208 and 207 of subsection 6.1.1.

Definition 55 (decision)**.** *Let M be a (deterministic or probabilistic) Turing Machine M. Let $P : \{0,1\}^* \to \{true, false\}$ be a predicate and $X \subseteq \{0,1\}^*$ be a set of bit-strings. We say that M decides P (resp. given an element in X, M decides P) if for every input $x^0 \in \{0,1\}^*$ (resp. $x^0 \in X$), for every computation $\alpha \in Computations(M, x^0)$, M halts in α, returns either "yes" or "no" and $returned(\alpha) =$ "yes" $\iff P(x^0) = true$.*

Definition 56 (recognizable set)**.** *Let $b \in \mathbb{N}$. Let $(S, \langle . \rangle^S)$ be an encoded set. A Turing Machine M recognizes S if M decides the predicate $P : x \in \{0,1\}^* \to [\exists s \in S, \langle s\rangle^S = x]$. The set S is said b-recognizable if there exists a b-bounded Turing machine M, such that M recognizes S.*

2.5. Labeled Transition System (LTS)

Here, we quickly survey the literature on I/O automata that led to PSIOA. We first present the very well-known Labeled Transition Systems (LTS). Then we briefly discuss the new features brought by I/O Automata, probabilistic I/O Automata, and signature I/O Automata.

2.5.1. Simple Labeled Transition System

Roberto Segala describes LTS as follows ([Seg95b], section 3.2, p. 37): "A Labeled Transition System is a state machine with labeled transitions. The labels, also called *actions*, are used to model communication between a system and its external environment." A possible definition of a LTS, using notation of [LV95], is a tuple $\mathcal{A} = (Q_{\mathcal{A}}, \bar{q}_{\mathcal{A}}, \check{sig}(\mathcal{A}), steps(\mathcal{A}))$ where $Q_{\mathcal{A}}$ represents the states of $\mathcal{A}$, $\bar{q}_{\mathcal{A}}$ represents the start state of $\mathcal{A}$, $\check{sig}(\mathcal{A}) = (\check{ext}(\mathcal{A}), \check{int}(\mathcal{A}))$ represents the *signature* of $\mathcal{A}$, i.e. the set of actions that can be triggered, that are partitioned into external and internal actions, and $steps(\mathcal{A}) \subseteq Q_{\mathcal{A}} \times acts(\mathcal{A}) \times Q_{\mathcal{A}}$ represent the possible transition of the transition with $acts(\mathcal{A}) = \check{ext}(\mathcal{A}) \cup \check{int}(\mathcal{A})$. We can note $enabled(\mathcal{A}) : q \in Q_{\mathcal{A}} \mapsto \{a \in acts(\mathcal{A}) | \exists (q, a, q') \in Steps(\mathcal{A})\}$ to model the actions enabled at a certain state. "The external actions model communication with the external environment; the internal actions model internal communication, not visible from the external environment." It is possible to make several LTS communicate with each other through shared external actions in CSP [Hoa85] style. Typically, if $\mathcal{A}$ and $\mathcal{B}$ are two LTS s.t. the compatibility condition $acts(\mathcal{A}) \cap \check{int}(\mathcal{B}) = acts(\mathcal{B}) \cap \check{int}(\mathcal{A}) = \emptyset$ is verified, we can define their composition, $\mathcal{A}||\mathcal{B}$ with

- $Q_{\mathcal{A}||\mathcal{B}} = Q_{\mathcal{A}} \times Q_{\mathcal{B}}$,
- $\bar{q}_{\mathcal{A}||\mathcal{B}} = (\bar{q}_{\mathcal{A}}, \bar{q}_{\mathcal{B}})$,
- $\check{sig}(\mathcal{A}||\mathcal{B}) = (\check{ext}(\mathcal{A}) \cup \check{ext}(\mathcal{B}), \check{int}(\mathcal{A}) \cup \check{int}(\mathcal{B}))$,
- $Steps(\mathcal{A}||\mathcal{B}) = \{((q_{\mathcal{A}}, q_{\mathcal{B}}), a, (q'_{\mathcal{A}}, q'_{\mathcal{B}})) \in Q_{\mathcal{A}||\mathcal{B}} \times acts(\mathcal{A}||\mathcal{B}) \times Q_{\mathcal{A}||\mathcal{B}} | a \in enabled(\mathcal{A}) \cup enabled(\mathcal{B}) \wedge \forall \mathcal{K} \in \{\mathcal{A}, \mathcal{B}\}, (q_{\mathcal{K}}, a, q'_{\mathcal{K}}) \notin Steps(\mathcal{K}) \implies (a \notin enabled(\mathcal{K}) \wedge q'_{\mathcal{K}} = q_{\mathcal{K}})\})$.

An *execution* of an LTS $\mathcal{A}$ is an alternating sequence of states and actions $q^0 a^1 q^1 a^2 ...$ such that each $(q^{i-1}, a^i, q^i) \in Steps(\mathcal{A})$. A *trace* is the restriction to external actions of an execution. A LTS $\mathcal{A}$ implements another LTS $\mathcal{B}$ if $Traces(\mathcal{A}) \subseteq Traces(\mathcal{B})$, where $Traces(\mathcal{K})$ represents the set of traces of $\mathcal{K}$.

2.5.2. I/O Automata

The input output Automata (IOA) [LT87] are LTS with the following additional points for some pair $(\mathcal{A}, \mathcal{B})$ of IOA :

- (I/O partitioning) There is a partition $(\check{in}(\mathcal{A}), \check{out}(\mathcal{A}))$ of $\check{ext}(\mathcal{A})$ where $\check{in}(\mathcal{A})$ denotes the *input* actions and $\check{out}(\mathcal{A})$ denotes the *output* actions. Moreover, $\check{loc}(\mathcal{A})$ denotes the *local* actions.
- (Output compatibility) The compatibility condition requires $out(\mathcal{A}) \cap out(\mathcal{B}) = \emptyset$ in addition.
- (I/O composition) After composition, we have in addition $out(\mathcal{A}||\mathcal{B}) = out(\mathcal{A}) \cup out(\mathcal{B})$ and $in(\mathcal{A}||\mathcal{B}) = in(\mathcal{A}) \cup in(\mathcal{B}) \setminus out(\mathcal{A}||\mathcal{B})$
- (Input enabling) $\forall q \in Q_{\mathcal{A}}, in(\mathcal{A}) \subseteq enabled(\mathcal{A})(q)$

The interests of these additional restrictions for formal verification are subtle (e.g. input enabling can avoid trivial liveness property implementation, locality allows simple definitions of fairness and oblivious scheduler, I/O partitioning allows the intuitive definition of forwarding, ...). However, they do not add complexity to the analysis of this thesis. Typically, they are never required in the key results of this thesis. Adapting this thesis to LTS is straightforward. We have kept I/O automata to be as close as possible from [AL16] and [CCK+18].

2.5.3. PIOA

The probabilistic input/output automata (PIOA) [WSS97] are kind of I/O automata where transitions are randomized, i.e. triggering an action leads to a probability measure on states instead of a particular state. The transitions of a PIOA $\mathcal{A}$ are then elements of $D_{\mathcal{A}} \subseteq Q_{\mathcal{A}} \times acts(\mathcal{A}) \times Disc(Q_{\mathcal{A}})$. Now, the set of steps is $Steps(\mathcal{A}) = \{(q, a, q') | \exists (q, a, \eta) \in D_{\mathcal{A}} \wedge q' \in supp(\eta)\}$. To define a measure of probability on the set of executions, it is convenient to call on a scheduler σ that will resolve the non-determinism and enable the construction of a measure of probability ϵ_σ on executions. The notion of implementation has to be adapted to a probabilistic setting to be relevant.

2.5.4. DIOA

The signature I/O automata (SIOA) [AL16] are a kind of I/O automata where the signature is evolving during the time. This feature is particularly convenient to model dynamicity. The signature of the automaton $\mathcal{A}$ becomes a function mapping each state q to a signature $sig(\mathcal{A})(q)$.

A configuration automaton (CA) is a a kind of SIOA, linked to a configuration (which is a set of SIOA associated with current states) whose membership can evolve along an execution of the CA. Some constraints have to be respected tu ensure the consistency of the whole.

A dynamic I/O automaton (DIOA) is either a PSIOA or a CA.

2.5.5. TIOA

The timed I/O automata (TIOA) [KLSV06] are a kind of I/O automata, extended with trajectories in their state space which capture the evolution of the state with real-time. The notion of trajectory is presented in section 2.2. This notion of timed automata, uniquely used in Chapter 5, is developped in the same chapter. Instead of being an alternate sequence of states and actions, an execution of a timed automaton is an alternate sequence of trajectories and actions. Hence, an execution of a timed automaton represents a succession of continuous and discrete transitions.

2.5.6. DPIOA

A PSIOA (resp. PCA, resp. DPIOA) is the result of the generalization of SIOA (resp. CA, resp. DIOA) and PIOA. The framework of DPIOA is introduced in Chapter 3. The figure 1.4 gives a first intuition of what is a DPIOA.

2.5.7. DPTIOA

A PTSIOA (resp. PTCA, resp. DPTIOA) is the result of the generalization of SIOA (resp. CA, resp. DIOA) and PTIOA, where a PTIOA is the generalization of both PIOA and TIOA. The framework of DPIOA is introduced in Chapter 3. We must underline that PTIOA have been introduced in the Mitra's thesis [Mit07].

2.5.8. Overwiev of the I/O Automata framework

We summarize this section about I/O automata framework with the figure 2.1. In this thesis, we do not cover the Hybrid I/O automata (HIOA) [LSVW95] that extend TIOA and allow modelling hybrid

systems that can interact with shared external variables in addition of shared external actions. We are not aware of a probabilistic (PHIOA) or a dynamic (DHIOA) extension of HIOA framework. The work that would most closely resemble a DHIOA would be the quantified differential dynamic logic for distributed hybrid systems from André Platzer [Pla10].

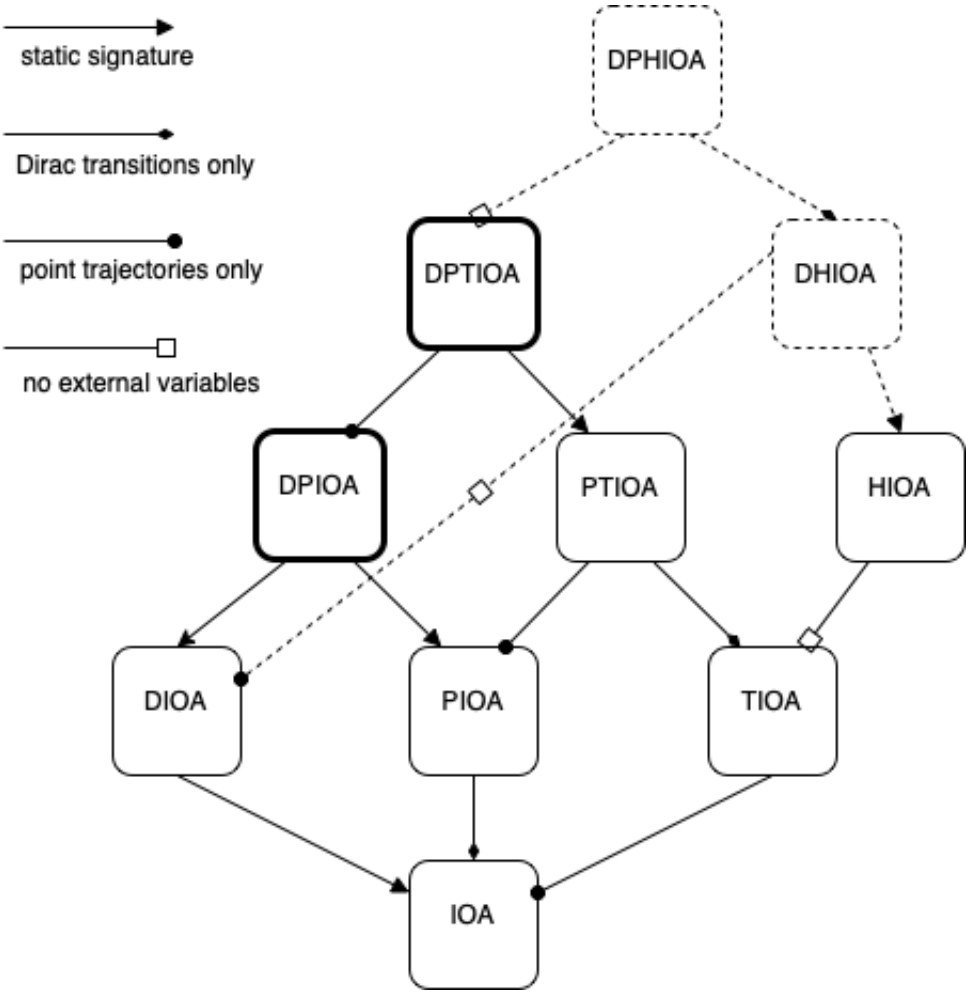


Figure 2.1. – the variations of the I/O Automata framework

Chapter 3

Dynamic Probablistic I/O Automata

This chapter introduces Dynamic Probabilistic Automata: a framework to reason about dynamic probabilistic distributed systems.

Overview

3.1. Probabilistic Signature Input/Output Automata (PSIOA) . . . 40
3.1.1. Action Signature . . . 40
3.1.2. PSIOA . . . 41
3.1.3. Local composition . . . 41
3.1.4. Hiding operator . . . 42
3.1.5. Action renaming . . . 43
3.2. Probabilistic Configuration Automata . . . 45
3.2.1. configuration . . . 45
3.2.2. probabilistic configuration automata (PCA) . . . 47
3.3. A Toy Example: Specification of a Dynamic Reliable Broadcast . . . 49
3.3.1. Modelisation of Byzantine Reliable Broadcast . . . 49
3.3.2. Static Consensus . . . 50
3.3.3. Dynamic Byzantine Reliable Broadcast . . . 52
3.4. Executions, reachable states, partially-compatible automata . . . 65
3.4.1. Executions, reachable states, traces . . . 65
3.4.2. PSIOA and PCA composition . . . 66
3.5. Toolkit for configurations & PCA closure under composition . . . 68
3.6. Scheduler, measure on executions, implementation . . . 74
3.6.1. General definition and probabilistic space $(Frags(\mathcal{A}), \mathcal{F}_{Frags(\mathcal{A})}, \epsilon_{\sigma,\mu})$ . . . 74
3.6.2. Implementation . . . 77
3.7. Introduction on PCA corresponding w.r.t. PSIOA $\mathcal{A}$, $\mathcal{B}$ to introduce monotonicity . . . 82
3.7.1. An informal statement of the theorem . . . 82
3.7.2. Naive correspondence between two PCA . . . 83
3.7.3. Conservatism: the additional assumption for relevant definition of correspondence w.r.t. $\mathcal{A}, \mathcal{B}$ . . . 84
3.7.4. Corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$ . . . 86
3.7.5. Creation-oblivious scheduler . . . 86
3.8. Summary . . . 88

This chapter introduces Dynamic Probabilistic Automata: a framework to reason about dynamic probabilistic distributed systems. We strongly encourage reading Subsection 1.3.2 page 8 to get a first intuition about the content of this chapter 3.

In section 3.1 we define probabilistic signature I/O automata (PSIOA) as a kind of labeled transition system (LTS) with (i) a signature that can evolve from one state to another, (ii) actions that lead to a probabilistic distribution on states instead of directly to a state, (iii) external actions are separate into input and output actions. Section 3.2 introduces probabilistic configuration automata (PCA), an automaton (equivalent to a PSIOA) representing a dynamic distributed system that can create/destroy dynamically PCA/PSIOA of a lower layer. Section 3.4 presents executions of a set of automata as alternating sequences of states and actions. It allows defining reachable states of a set of automata, that are said compatible if they are compatible at each reachable state (in the sense their signature are compatible). The closure of the set of PCA under parallel compositions is proved in section 3.5. Pure non-determinism is resolved in section 3.6, before defining the implementation relationship that is shown to be composable. In section 3.7, we give a definition of what are corresponding automata $X_{\mathcal{A}}$,$X_{\mathcal{B}}$ w.r.t. PSIOA $\mathcal{A}$,$\mathcal{B}$, that differ only by the fact that $X_{\mathcal{A}}$ dynamically creates/destroys $\mathcal{A}$ instead of $\mathcal{B}$ as $X_{\mathcal{B}}$ does, noted $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$. Finally, we give technical necessary conditions (partially captured in the definition of $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$) to obtain monotonicity of dynamic creation/destruction of PSIOA with implementation relationship.

3.1. Probabilistic Signature Input/Output Automata (PSIOA)

This section aims to introduce the first brick of our formalism: the probabilistic signature input/output automata (PSIOA).

3.1.1. Action Signature

We use the signature approach from [AL16]. We assume the existence of a countable set $Autids$ of unique probabilistic signature input/output automata (PSIOA) identifiers, an underlying universal set $Auts$ of PSIOA, and a mapping $aut : Autids \to Auts$. $aut(\mathcal{A})$ is the PSIOA with identifier $\mathcal{A}$. We use "the automaton $\mathcal{A}$" to mean "the PSIOA with identifier $\mathcal{A}$". We use the letters $\mathcal{A}, \mathcal{B}$, possibly subscripted or primed, for PSIOA identifiers. The executable actions of a PSIOA $\mathcal{A}$ are drawn from a signature $sig(\mathcal{A})(q) = (in(\mathcal{A})(q), out(\mathcal{A})(q), int(\mathcal{A})(q))$, called the state signature, which is a function of the current state q of $\mathcal{A}$.

$in(\mathcal{A})(q), out(\mathcal{A})(q), int(\mathcal{A})(q)$ are pairwise disjoint sets of input, output, and internal actions, respectively. We define $ext(\mathcal{A})(q)$, the external signature of $\mathcal{A}$ in state q, to be $ext(\mathcal{A})(q) = (in(\mathcal{A})(q), out(\mathcal{A})(q))$.

We define $loc(\mathcal{A})(q)$, the local signature of $\mathcal{A}$ in state q, to be $loc(\mathcal{A})(q) = (out(\mathcal{A})(q), int(\mathcal{A})(q))$. For any signature component, generally, the $\widehat{\cdot}$ operator yields the union of sets of actions within the signature, e.g., $\widehat{sig(\mathcal{A})} : q \in Q \mapsto \widehat{sig}(\mathcal{A})(q) = in(\mathcal{A})(q) \cup out(\mathcal{A})(q) \cup int(\mathcal{A})(q)$. Also we define $acts(\mathcal{A}) = \bigcup_{q \in Q} \widehat{sig}(\mathcal{A})(q)$, that is $acts(\mathcal{A})$ is the "universal" set of all actions that A could possibly trigger, in any state. In the same way $UI(\mathcal{A}) = \bigcup_{q \in Q} in(\mathcal{A})(q)$, $UO(\mathcal{A}) = \bigcup_{q \in Q} out(\mathcal{A})(q)$, $UH(\mathcal{A}) = \bigcup_{q \in Q} int(\mathcal{A})(q)$, $UL(\mathcal{A}) = \bigcup_{q \in Q} \widehat{loc}(\mathcal{A})(q)$, $UE(\mathcal{A}) = \bigcup_{q \in Q} \widehat{ext}(\mathcal{A})(q)$.

3.1.2. PSIOA

We combine the SIOA of [AL16] with the PIOA of [WSS97, CCK+06a]:

Definition 57 (PSIOA). *A PSIOA* $\mathcal{A} = (Q_{\mathcal{A}}, \bar{q}_{\mathcal{A}}, sig(\mathcal{A}), D_{\mathcal{A}})$*, where:*

- $Q_{\mathcal{A}}$ *is a countable set of* states*,* $(Q_{\mathcal{A}}, 2^{Q_{\mathcal{A}}})$ *is the* state space*,*
- $\bar{q}_{\mathcal{A}}$ *is the unique* start state*.*
- $sig(\mathcal{A}) : q \in Q_{\mathcal{A}} \mapsto sig(\mathcal{A})(q) = (in(\mathcal{A})(q), out(\mathcal{A})(q), int(\mathcal{A})(q))$ *is the signature function that maps each state to a triplet of mutually disjoint countable set of* actions*, respectively called* input*,* output *and* internal *actions.*
- $D_{\mathcal{A}} \subset Q_{\mathcal{A}} \times acts(\mathcal{A}) \times Disc(Q_{\mathcal{A}})$ *is the set of* probabilistic discrete transitions *where* $\forall (q, a, \eta) \in D_{\mathcal{A}} : a \in \widehat{sig}(\mathcal{A})(q)$*. If* (q, a, η) *is an element of* $D_{\mathcal{A}}$*, we write* $q \xrightarrow{a} \eta$ *and action* a *is said to be* enabled *at* q*. We note* $enabled(\mathcal{A}) : q \in Q_{\mathcal{A}} \mapsto enabled(\mathcal{A})(q)$ *where* $enabled(\mathcal{A})(q)$ *denotes the set of enabled actions at state* q*. We also note* $Steps(\mathcal{A}) \triangleq \{(q, a, q') \in Q_{\mathcal{A}} \times acts(\mathcal{A}) \times Q_{\mathcal{A}} | \exists (q, a, \eta) \in D_{\mathcal{A}}, q' \in supp(\eta)\}$*.*

In addition $\mathcal{A}$ must satisfy the following conditions

- **T** (Transition determinism): For every $q \in Q_{\mathcal{A}}$ and $a \in \widehat{sig}(\mathcal{A})(q)$ there is at most one $\eta_{(\mathcal{A},q,a)} \in Disc(Q_{\mathcal{A}})$, such that $(q, a, \eta_{(\mathcal{A},q,a)}) \in D_{\mathcal{A}}$.
- **E** (Enabled actions tracking): $\widehat{sig}(\mathcal{A}) = enabled(\mathcal{A})$.

The axiom **E** has been added for sake of simplicity. Indeed, for every state $q \in Q_{\mathcal{A}}$, $\widehat{sig}(\mathcal{A})(q) \subseteq enabled(\mathcal{A})(q)$, while an adaptation of classic input enabling axiom would be $q \in Q_{\mathcal{A}}$, $in(\mathcal{A})(q) \subseteq enabled(\mathcal{A})(q)$. Since the signature is dynamic, for any PSIOA that does not verify **E**, it is always possible to remove the non-enabled actions of its signature to obtain an equivalent PSIOA verifying **E**.

Later, we will define *execution fragments* as alternating sequences of states and actions with classic and natural consistency rules. But a subtlety will appear with the composability of a set of automata at reachable states. Hence, we will define *execution fragments* after "local composability" and "probabilistic configuration automata".

3.1.3. Local composition

The main aim of a formalism of concurrent systems is to compose several automata $\mathbf{A} = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$ and provide guarantees by composing the guarantees of the different elements of the system. Some syntactical rules have to be satisfied before defining the composition operation.

Definition 58 (Compatible signatures). *Let* $S = \{sig_i\}_{i \in \mathcal{I}}$ *be a set of signatures. Then* S *is compatible iff,* $\forall i, j \in \mathcal{I}$*,* $i \neq j$*, where* $sig_i = (in_i, out_i, int_i), sig_j = (in_j, out_j, int_j)$*, we have: 1.* $(in_i \cup out_i \cup int_i) \cap int_j = \emptyset$*, and 2.* $out_i \cap out_j = \emptyset$*.*

In the framework, automata communicate by shared external actions that cannot be locally triggered at more than one place. Moreover, internal actions are not shared.

Definition 59 (Composition of Signatures). *Let* $\Sigma = (in, out, int)$ *and* $\Sigma' = (in', out', int')$ *be compatible signatures. Then we define their composition* $\Sigma \times \Sigma = (in \cup in' - (out \cup out'), out \cup out', int \cup int')$ [1]*.*

1. not to be confused with Cartesian product. We keep this notation to stay as close as possible to the literature.

Signature composition is clearly commutative and associative.

Remark 1. *Without a distinction between inputs and outputs, two signatures sig and sig' would be compatible (only) under the condition that internal actions are not shared. The resulting external signature would be the same, i.e. the union of respective sets of external signatures. There is no additional difference. In this thesis, we kept this notation to be as close as possible from [AL16], but the entire framework and all the associated results can be translated to Dynamic Probabilistic Labeled Transition Systems (without explicit distinction between input and output actions) in a straightforward manner.*

Now we can define the compatibility of several automata at a state with the compatibility of their attached signatures. First, we define compatibility at a state and discrete transition for a set of automata for a particular compatible state.

Definition 60 (compatibility at a state)**.** *Let $\mathbf{A} = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$ be a set of PSIOA. A* state *of* $\mathbf{A}$ *is an element $q = (q_1, ..., q_n) \in Q_{\mathbf{A}} \triangleq Q_{\mathcal{A}_1} \times ... \times Q_{\mathcal{A}_n}$. We note $q \upharpoonright \mathcal{A}_i \triangleq q_i$. We say $\mathcal{A}_1, ..., \mathcal{A}_n$ are (or* $\mathbf{A}$ *is)* compatible at state q *if $\{sig(\mathcal{A}_1)(q_1), ..., sig(\mathcal{A}_n)(q_n)\}$ is a set of compatible signatures. In this case we note $sig(\mathbf{A})(q) \triangleq sig(\mathcal{A}_1)(q_1) \times ... \times sig(\mathcal{A}_n)(q_n)$ as per definition 59 and we note $\eta_{(\mathbf{A},q,a)} \in Disc(Q_{\mathbf{A}})$, s.t. $\forall a \in \widehat{sig}(\mathbf{A})(q)$, $\eta_{(\mathbf{A},q,a)} = \eta_1 \otimes ... \otimes \eta_n$ where $\forall j \in [1, n]$, $\eta_j = \eta_{(\mathcal{A}_j, q_j, a)}$ if $a \in sig(\mathcal{A}_j)(q_j)$ and $\eta_j = \delta_{q_j}$ otherwise. Moreover, we note $steps(\mathbf{A}) = \{(q, a, q') | q, q' \in Q_{\mathbf{A}}, a \in sig(\mathbf{A})(q), q' \in supp(\eta_{(\mathbf{A},q,a)})\}$. Finally, we note $\bar{q}_{\mathbf{A}} = (\bar{q}_{\mathcal{A}_1}, ..., \bar{q}_{\mathcal{A}_n})$.*

The probabilistic distribution $\eta_{(\mathbf{A},q,a)}$ is the product of "independent" probabilistic distributions η_j over respective sets of states $Q_{\mathcal{A}_j}$ of the component $\mathcal{A}_j$ of $\mathbf{A}$, triggered by action a at their respective state $q \upharpoonright \mathcal{A}_j$ (with the convention of triggering a Dirac if a is not enabled). If $\mathbf{A}$ is compatible at state q, it does not necessarily mean that $\mathbf{A}$ is compatible at every state in $supp(\eta_{(\mathbf{A},q,a)})$. Such a condition will characterize partial-compatibility of $\mathbf{A}$ introduced in section 3.4.

Let us note that an action a shared by two automata becomes an output action and not an internal action after composition. First, it permits the possibility of further communication using a. Second, it allows associativity. If this property is counter-intuitive, it is always possible to use the classic hiding operator that "hides" the output actions transforming them into internal actions.

3.1.4. Hiding operator

Definition 61 (hiding operator)**.** *Let $sig = (in, out, int)$ be a signature and H a set of actions. We note $hide(sig, H) \triangleq (in, out \setminus H, int \cup (out \cap H))$.*

Let $\mathcal{A} = (Q_{\mathcal{A}}, \bar{q}_{\mathcal{A}}, sig(\mathcal{A}), D_{\mathcal{A}})$ be a PSIOA. Let $h : q \in Q_{\mathcal{A}} \mapsto h(q) \subseteq out(\mathcal{A})(q)$. We note $hide(\mathcal{A}, h) \triangleq (Q_{\mathcal{A}}, \bar{q}_{\mathcal{A}}, sig'(\mathcal{A}), D_{\mathcal{A}})$, where $sig'(\mathcal{A}) : q \in Q_{\mathcal{A}} \mapsto hide(sig(\mathcal{A})(q), h(q))$. Clearly, $hide(\mathcal{A}, h)$ is a PSIOA.

Lemma 1 (hiding and composition are commutative)**.** *Let $sig_a = (in_a, out_a, int_a)$, $sig_b = (in_b, out_b, int_b)$ be compatible signature and H_a, H_b some set of actions, s.t.*

- $(H_a \cap out_a) \cap \widehat{sig}_b = \emptyset$ *and*
- $(H_b \cap out_b) \cap \widehat{sig}_b = \emptyset$,

then $sig'_a \triangleq hide(sig, H_a) \triangleq (in'_a, out'_a, int'_a)$ and $sig'_b \triangleq hide(sig_b, H_b) \triangleq (in'_b, out'_b, int'_b)$ are compatible. Furthermore, if

- $out_b \cap H_a = \emptyset$ *,and*
- $out_a \cap H_b = \emptyset$

then $sig'_a \times sig'_b = hide(sig_a \times sig_b, H_a \cup H_b)$.

Proof. – compatibility: After the hiding operation, we have:

- $in'_a = in_a$, $in'_b = in_b$
- $out'_a = out_a \setminus H_a$, $out'_b = out_b \setminus H_b$
- $int'_a = int_a \cup (out_a \cap H_a)$, $int'_b = int_b \cup (out_b \cap H_b)$

Since $out_a \cap out_b = \emptyset$, a fortiori $out'_a \cap out'_b = \emptyset$. $int_a \cap \widehat{sig_b} = \emptyset$, thus if $(out_a \cap H_a) \cap \widehat{sig_b} = \emptyset$, then $int'_a \cap \widehat{sig_b} = \emptyset$ and with the symetric argument, $int'_b \cap \widehat{sig_a} = \emptyset$. Hence, sig'_a and sig'_b are compatible.

– commutativity:

After composition of $sig'_c = sig'_a \times sig'_b$ operation, we have:

- $out'_c = out'_a \cup out'_b = (out_a \setminus H_a) \cup (out_b \setminus H_b)$. If $out_b \cap H_a = \emptyset$ and $out_a \cap H_b = \emptyset$, then $out'_c = (out_a \cup out_b) \setminus (H_a \cup H_b)$.
- $in'_c = in'_a \cup in'_b \setminus out'_c = in_a \cup in_b \setminus out'_c$
- $int'_c = int'_a \cup int'_b = int_a \cup (out_a \cap H_a) int_b \cup (out_b \cap H_b) = int_a \cup int_b \cup (out_a \cap H_a) \cup (out_b \cap H_b)$. If $out_b \cap H_a = \emptyset$ and $out_a \cap H_b = \emptyset$, then $int'_c = int_a \cup int_b \cup ((out_a \cup out_b) \cap (H_a \cup H_b)$.

and after composition of $sig_d = sig_a \times sig_b$

- $out_d = out_a \cup out_b$
- $in_d = in_a \cup in_b \setminus out_d$
- $int_d = int_a \cup int_b$

Finally, after hiding operation $sig'_d = hide(sig_d, H_a \cup H_b)$ we have :

- $in'_d = in_d$
- $out'_d = out_d \setminus H_a \cup H_b = (out_a \cup out_b) \setminus (H_a \cup H_b)$
- $int'_d = int_d \cup (out_d \cap (H_a \cup H_b)) = (int_a \cup int_b) \cup (out_d \cap (H_a \cup H_b))$

Thus, if $out_b \cap H_a = \emptyset$ and $out_a \cap H_b = \emptyset$

- $in'_d = in'_c$
- $out'_d = out'_c$
- $int'_d = int'_c$

□

Remark 2. *We can restrict the hiding operation to a set of actions included in the set of output actions of the signature ($H \subseteq out$). In this case, since we already have $out_a \cap out_b = \emptyset$ by compatibility, we immediately have $out_a \cap H_b = \emptyset$ and $out_b \cap H_a = \emptyset$. Thus to obtain compatibility, we only need $in_b \cap H_a = \emptyset$ and $in_a \cap H_b = \emptyset$. Later, the compatibility of PCA will implicitly assume this predicate (otherwise the PCA could not be compatible).*

In our momentum, we introduce the classic "action renaming" operator.

3.1.5. Action renaming

Action renaming is useful to make automata compatible. This operator is used in the proof of theorem 4 of transitivity of implementation relationship.

Definition 62 (Action renaming for PSIOA)**.** *Let $\mathcal{A}$ be a PSIOA and let r be a partial function on $Q_{\mathcal{A}} \times acts(\mathcal{A})$, s.t. $\forall q \in Q_{\mathcal{A}}$, $r(q)$ is an injective mapping with $\widehat{sig}(\mathcal{A})(q)$ as domain. Then $r(\mathcal{A})$ is the automata given by:*

1. $\bar{q}_{r(\mathcal{A})} = \bar{q}_{\mathcal{A}}$.
2. $Q_{r(\mathcal{A})} = Q_{\mathcal{A}}$.
3. $\forall q \in Q_{\mathcal{A}}$, $sig(r(\mathcal{A}))(q) = (in(r(\mathcal{A}))(q)\ , out(r(\mathcal{A}))\ (q)\ , int(r(\mathcal{A}))(q))$ *with*
   - $out(r(\mathcal{A}))(q) = r(out(\mathcal{A})(q))$,
   - $in(r(\mathcal{A}))(q) = r(in(\mathcal{A})(q))$,
   - $int(r(\mathcal{A}))(q) = r(int\ (\mathcal{A})(q))$.
4. $D_{r(\mathcal{A})} = \{(q, r(a), \eta) | (q, a, \eta) \in D_{\mathcal{A}}\}$ *(we note $\eta_{(r(\mathcal{A}),q,r(a))}$ the element of $Disc(Q_{r(\mathcal{A})})$ which is equal to $\eta_{(\mathcal{A},q,a)}$.*

Intuitively, $r(\mathcal{A})$ is just the automaton where each action a has been replaced by action $r(a)$.

Lemma 2 (PSIOA closure under action-renaming)**.** *Let $\mathcal{A}$ be a PSIOA and let r be a partial function on $Q_{\mathcal{A}} \times acts(\mathcal{A})$, s.t. $\forall q \in Q_{\mathcal{A}}$, $r(q)$ is an injective mapping with $\widehat{sig}(\mathcal{A})(q)$ as domain. Then $r(\mathcal{A})$ is a PSIOA.*

Proof. We need to show (1) $\forall (q, a, \eta), (q, a, \eta') \in D_{\mathcal{A}}$, $\eta = \eta'$ and $a \in \widehat{sig}(\mathcal{A})(q)$, (2) $\forall q \in Q_{\mathcal{A}}, \forall a \in \widehat{sig}(\mathcal{A})(q)$, $\exists \eta \in Disc(Q_{\mathcal{A}})$, $(q, a, \eta) \in D_{\mathcal{A}}$ and (3) $\forall q \in Q_{\mathcal{A}} : in(\mathcal{A})(q) \cap out(\mathcal{A})(s) = in(\mathcal{A})(q) \cap int(\mathcal{A})(q) = out(\mathcal{A})(q) \cap int(\mathcal{A})(q) = \emptyset$.

- Constraint 1: From definition 62, we have, for any $q \in Q_{r(\mathcal{A})}$: $\widehat{sig}(r(\mathcal{A}))(q) = out(r(\mathcal{A}))(q) \cup in(r(\mathcal{A}))\ (q) \cup int(r(\mathcal{A}))(q) = r(out(\mathcal{A})(q)) \cup r(in(\mathcal{A})(q)) \cup r(int(\mathcal{A})(q)) = r(\widehat{sig}(\mathcal{A})(q))$. Since $\mathcal{A}$ is a PSIOA, we have $\forall (q, a, \eta), (q, a, \eta') \in D_{\mathcal{A}} : a \in \widehat{sig}(\mathcal{A})(q)$ and $\eta = \eta'$. From definition 62, $D_{r(\mathcal{A})} = \{(q, r(a), \eta) |\ (q, a, \eta) \in D_{\mathcal{A}}\}$ Hence, if $(q, r(a), \eta)$, $(q, r(a), \eta')$ are arbitrary element of $D_{r(\mathcal{A})}$, then $(q, a, \eta), (q, a, \eta') \in D_{\mathcal{A}}$, and so $\eta = \eta'$ and $a \in \widehat{sig}(\mathcal{A})(q)$. Hence $r(a) \in r(\widehat{sig}(\mathcal{A})(q))$. Since $r(\widehat{sig}(\mathcal{A})(q)) = \widehat{sig}(r(\mathcal{A}))(q)$, we conclude $r(a) \in \widehat{sig}(r(\mathcal{A}))(q)$. Hence, $\forall (q, r(a), \eta)$, $(q, r(a), \eta') \in D_{r(\mathcal{A})} : r(a) \in \widehat{sig}(r(\mathcal{A}))(q)$ and $\eta = \eta'$. Thus, Constraint 1 holds for $r(\mathcal{A})$.
- Constraint 2: From definition 62, $D_{r(\mathcal{A})} = \{(q, r(a), \eta) | (q, a, \eta) \in D_{\mathcal{A}}\}$, $Q_{r(\mathcal{A})} = Q_{\mathcal{A}}$, and for all $q \in Q_{r(\mathcal{A})}, in(r(\mathcal{A}))(q) = r(in(\mathcal{A})(q))$. Let q be any state of $r(\mathcal{A})$, and let $q \in \widehat{sig}(r(\mathcal{A}))(q)$. Then $b = r(a)$ for some $a \in \widehat{sig}(\mathcal{A})(q)$. We have $(q, a, \eta) \in D_{\mathcal{A}}$ for some η, by Constraint 2 of action enabling for $\mathcal{A}$. Hence $(q, a, \eta) \in D_{r(\mathcal{A})}$. Hence $(q, b, \eta) \in D_{r(\mathcal{A})}$. Hence Constraint 2 holds for $r(\mathcal{A})$.
- Constraint 3: $\mathcal{A}$ is a PSIOA and so satisfies Constraint 3. From this and definition 62 and the requirement that r be injective, it is easy to see that $r(\mathcal{A})$ also satisfies Constraint 3.

□

3.2. Probabilistic Configuration Automata

In previous subsection, we gave a syntactical description of PSIOA, that are automata that probabilistically move from one state to another through triggered actions that belong to a dynamic signature. Before adressing the semantics of their interactions as we will do in section 3.4, we introduce some syntactical tools to describe dynamic behaviors. We combine the notion of configuration of [AL16] with the probabilistic setting of [WSS97, CCK$^+$06a]. A configuration is a set of automata associated with their current states. This will be a very useful tool to define dynamicity by mapping the state of an automaton of a certain "layer" to a configuration of automata of the lower layer, where the set of automata in the configuration can dynamically change from on state of the automaton of the upper level to another one.

3.2.1. configuration

Definition 63 (Configuration)**.** *A configuration is a pair* $(\mathbf{A}, \mathbf{S})$ *where*

- $\mathbf{A} = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$ *is a finite set of PSIOA identifiers and*
- $\mathbf{S}$ *maps each* $\mathcal{A}_k \in \mathbf{A}$ *to a state of* $\mathcal{A}_k$*.*

We note Q_{conf} *the set of configurations.*

In distributed computing, configuration usually refers to the union of states of **all** the automata of the "system". Here, there is a subtlety, since it captures a set of some automata ($\mathbf{A}$) in their current state ($\mathbf{S}$), but the set of automata of the systems will not be fixed in the time.

Proposition 7. *The set* Q_{conf} *of configurations is countable.*

Proof. (1) $\{\mathbf{A} \in \mathcal{P}(Autids) | \mathbf{A} \text{ is finite}\}$ is countable since $Autids$ is countable, (2) $\forall \mathcal{A} \in Autids, Q_{\mathcal{A}}$ is countable by definition 57 of PSIOA and (3) the cartesian product of countable sets is a countable set. □

We can define compatibility of configuration in the obvious manner, in a consistent manner with definition 60 of automata compatible at a state.

Definition 64 (Compatible configuration)**.** *A configuration* $(\mathbf{A}, \mathbf{S})$*, with* $\mathbf{A} = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$*, is compatible iff the set* $\mathbf{A}$ *is compatible at state* $(\mathbf{S}(\mathcal{A}_1), ..., \mathbf{S}(\mathcal{A}_n))$ *as per definition 60*

For sake of convenience, we equip configurations with some useful methods.

Definition 65 (Intrinsic attributes of a configuration)**.** *Let* $C = (\mathbf{A}, \mathbf{S})$ *be a compatible configuration. Then we define*

- $auts(C) = \mathbf{A}$ *represents the automata of the configuration,*
- $map(C) = \mathbf{S}$ *maps each automaton of the configuration with its current state,*
- $TS(C) = (\mathbf{S}(\mathcal{A}_1), ..., \mathbf{S}(\mathcal{A}_n))$ *yields the tuple of states of the automata of the configuration.*
- $sig(C) = (in(C), out(C), int(C)) = sig(auts(C))(TS(C))$ *in the sense of definition 60, is called the intrinsic signature of the configuration*

Here we define a reduced configuration as a configuration deprived of the automata that are in the very particular state where their current signatures are the empty set. This mechanism will be used later to capture the idea of the destruction of an automaton.

Definition 66 (Reduced configuration)**.** *$reduce(C) = (\mathbf{A}', \mathbf{S}')$, where $\mathbf{A}' = \{\mathcal{A} | \mathcal{A} \in \mathbf{A}$ and $sig(\mathcal{A})(\mathbf{S}(\mathcal{A})) \neq \emptyset\}$ and $\mathbf{S}'$ is the restriction of $\mathbf{S}$ to $\mathbf{A}'$, noted $\mathbf{S} \upharpoonright \mathbf{A}'$ in the remaining.*

A configuration C is a reduced configuration iff $C = reduce(C)$.

We will define some probabilistic transitions from configurations to others where some automata can be destroyed or created. To define it properly, we start by defining "preserving transition" where no automaton is neither created nor destroyed and then we define above this definition the notion of configuration transition.

Definition 67 (From preserving distribution to intrinsic transition)**.**

- *(preserving distribution) Let $\eta_p \in Disc(Q_{conf})$. We say η_p is a* preserving distribution *if it exists a finite set of automata $\mathbf{A}$, called* family support *of η_p, s.t. $\forall (\mathbf{A}', \mathbf{S}') \in supp(\eta_p), \mathbf{A} = \mathbf{A}'$.*
- *(preserving configuration transition $C \xrightarrow{a} \eta_p$) Let $C = (\mathbf{A}, \mathbf{S})$ be a compatible configuration, $a \in \widehat{sig}(C)$. Let η_p be the unique preserving distribution of $Disc(Q_{conf})$ such that (1) the family support of η_p is $\mathbf{A}$ and (2) $\eta_p \overset{TS}{\leftrightarrow} \eta_{(\mathbf{A}, TS(C), a)}$. We say that (C, a, η_p) is a* preserving configuration transition*, noted $C \xrightarrow{a} \eta_p$.*
- *($\eta_p \uparrow \varphi$) Let $\eta_p \in Disc(Q_{conf})$ be a preserving distribution with $\mathbf{A}$ as family support. Let φ be a finite set of of PSIOA identifiers with $\mathbf{A} \cap \varphi = \emptyset$. Let $C_\varphi = (\varphi, S_\varphi) \in Q_{conf}$ with $\forall \mathcal{A}_j \in \varphi, S_\varphi(\mathcal{A}_j) = \bar{q}_{\mathcal{A}_j}$. We note $\eta_p \uparrow \varphi$ the unique element of $Disc(Q_{conf})$ verifying $\eta_p \overset{u}{\leftrightarrow} (\eta_p \uparrow \varphi)$ with $u : C \in supp(\eta_p) \mapsto (C \cup C_\varphi)$.*
- *(distribution reduction) Let $\eta \in Disc(Q_{conf})$. We note $reduce(\eta)$ the element of $Disc(Q_{conf})$ verifying $\forall c \in Q_{conf}$, $(reduce(\eta))(c) = \Sigma_{(c' \in supp(\eta), c = reduce(c'))} \eta(c')$*
- *(intrinsic transition $C \xRightarrow{a}_\varphi \eta$) Let $C = (\mathbf{A}, \mathbf{S})$ be a compatible configuration, let $a \in \widehat{sig}(C)$, let φ be a finite set of of PSIOA identifiers with $\mathbf{A} \cap \varphi = \emptyset$. We note $C \xRightarrow{a}_\varphi \eta$, if $\eta = reduce(\eta_p \uparrow \varphi)$ with $C \xrightarrow{a} \eta_p$. In this case, we say that η is* generated by η_p and φ*.*

Preserving configuration transition (C, a, η_p) is the intuitive transition for configurations, corresponding to the transition $(TS(C), a, \eta_{(auts(C), TS(C), a)})$ (see figure 3.1). The operator $\uparrow \varphi$ describes the deterministic creation of automata in φ, who will appear at their respective start states. The *reduce* operator enables to remove "destroyed" automata from the possibly returned configurations (see figure 3.2).

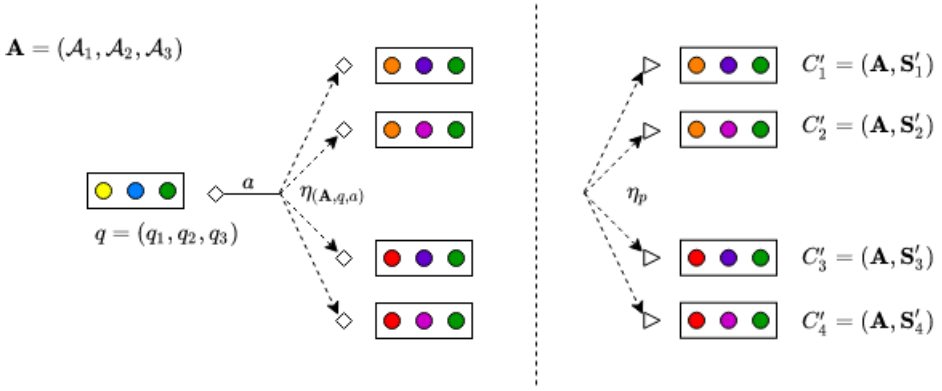


Figure 3.1. – Trivial homomorphism between preserving distribution and distribution on states of automata of its family support.

The homomorphism holds between η_p and $\eta_{(\mathbf{A}, TS(C), a)}$ with $C = (\mathbf{A}, \mathbf{S}) \xrightarrow{a} \eta_p$.

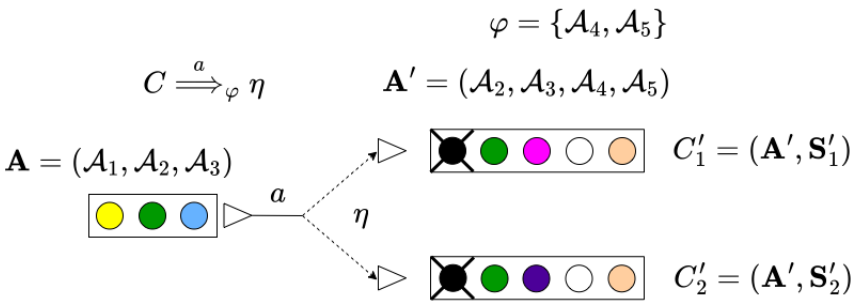


Figure 3.2. – An intrinsic transition

Automaton $\mathcal{A}_1$ is destroyed deterministically and automata in $\varphi = \{\mathcal{A}_4, \mathcal{A}_5\}$ are created deterministically. First, we have the preserving distribution η_p s.t. $C \xrightarrow{a} \eta_p$ with $\eta_p \overset{TS}{\leftrightarrow} \eta_{(\mathbf{A},TS(C),a)}$. Second, we take into account the created automata in φ, captured by the distribution $\eta_p \uparrow \varphi$. Third, we remove the automata in a particular state with an associated empty signature ($\mathcal{A}_1$ in our example). This is captured by distribution $reduce(\eta_p \uparrow \varphi)$.

3.2.2. probabilistic configuration automata (PCA)

Now we are ready to define our probabilistic configuration automata (see figure 3.3). Such an automaton is a kind of PSIOA that maintains a strong link with a dynamic configuration.

Definition 68 (Probabilistic Configuration Automaton)**.** *A probabilistic configuration automaton (PCA) X consists of the following components:*

- *1. A probabilistic signature I/O automaton $psioa(X)$. For brevity, we define $Q_X = Q_{psioa(X)}, \bar{q}_X = \bar{q}_{psioa(X)}$, $sig(X) = sig(psioa(X)), Steps(X) = Steps(psioa(X))$, and likewise for all attributes of $psioa(X)$.*
- *2. A configuration mapping $config(X)$ with domain Q_X and such that, for all $q \in Q_X$, $config(X)(q)$ is a reduced compatible configuration.*
- *3. For each $q \in Q_X$, a mapping $created(X)(q)$ with domain $sig(X)(q)$ and such that $\forall a \in sig(X)(q)$, $created(X)(q)(a) \subseteq Autids$ with $created(X)(q)(a)$ finite.*
- *4. A hidden-actions mapping $hidden\text{-}actions(X)$ with domain Q_X s.t. $\forall q \in Q_X$ $hidden\text{-}actions(X)(q) \subseteq out(config(X)(q))$.*

and satisfies the following constraints, for every $q \in Q_X$, $C = config(X)(q)$, $H = hidden\text{-}actions(X)(q)$.

- *1. (start states preservation) If $config(X)(\bar{q}_X) = (\mathbf{A}, \mathbf{S})$, then $\forall \mathcal{A}_i \in \mathbf{A}, \mathbf{S}(\mathcal{A}_i) = \bar{q}_{\mathcal{A}_i}$.*
- *2. (top/down transition preservation) If $(q, a, \eta_{(X,q,a)}) \in D_X$, then $\exists \eta' \in Disc(Q_{conf})$ s.t. $\eta_{(X,q,a)} \overset{c}{\leftrightarrow} \eta'$ with $C \xrightarrow{a}_{\varphi} \eta'$, where $\varphi = created(X)(q)(a)$ and $c = config(X)$.*
- *3. (bottom/up transition preservation) If $q \in Q_X$ and $C \xrightarrow{a}_{\varphi} \eta'$ for some action a, $\varphi = created(X)(q)(a)$, and reduced compatible probabilistic measure $\eta' \in Disc(Q_{conf})$, then $(q, a, \eta_{(X,q,a)}) \in D_X$, and $\eta_{(X,q,a)} \overset{c}{\leftrightarrow} \eta'$ where $c = config(X)$.*
- *4. (signature preservation modulo hiding) $\forall q \in Q_X$, $sig(X)(q) = hide(sig(C), H)$.*

This definition, proposed in a deterministic fashion in [AL16], captures the dynamicity of the system. Each state is linked with a configuration. The set of automata of the configuration can change during execution. If $\mathcal{A} \in created(X)(q)(a)$, the sub-automaton $\mathcal{A}$ is created from state q by the triggering

of action a. A sub-automaton $\mathcal{A}$ is destroyed if the non-reduced attached configuration distribution leads to a configuration where $\mathcal{A}$ is in a state $q_{\mathcal{A}}^{\phi}$ s.t. $\widehat{sig}(\mathcal{A})(q_{\mathcal{A}}^{\phi}) = \emptyset$. Then the corresponding reduced configuration will not hold $\mathcal{A}$. The last constraint states that the signature of a state q of X must be the same as the signature of its corresponding configuration $config(X)(q)$, except for the possible effects of hiding operators, so that some outputs of $config(X)(q)$ may be internal actions of X in state q.

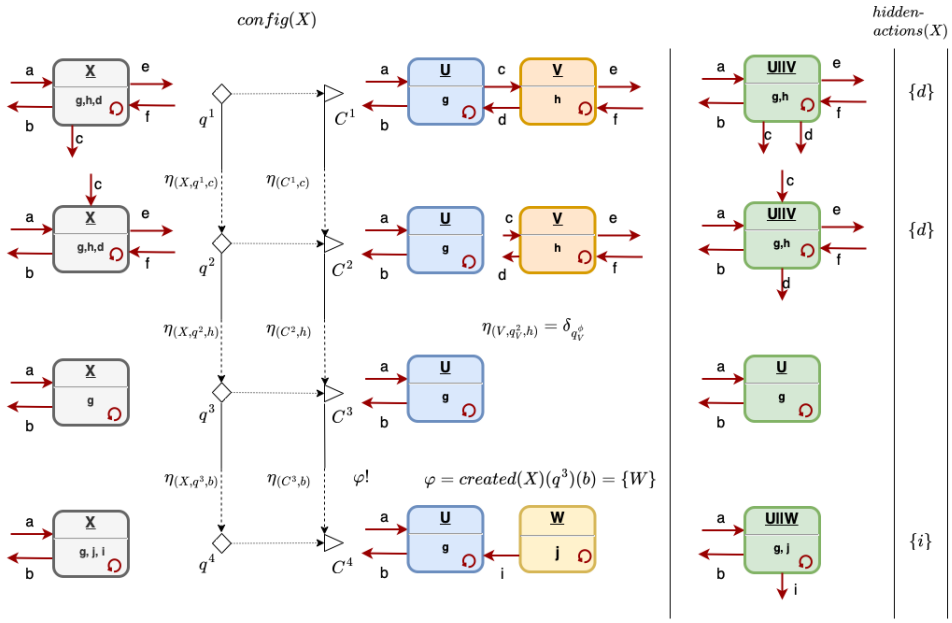


Figure 3.3. – A PCA life cycle.

As for PSIOA, we can define the hiding operator applied to PCA.

Definition 69 (hiding on PCA)**.** *Let X be a PCA. Let $h : q \in Q_X \mapsto h(q) \subseteq out(X)(q)$. We note $hide(X, h)$ the PCA X' that differs from X only on*

- $psioa(X') = hide(psioa(X), h)$
- $sig(X') = hide(sig(X), h)$ *and*
- $\forall q \in Q_X = Q_{X'}$, $hidden\text{-}actions(X')(q) = hidden\text{-}actions(X)(q) \cup h(q)$.

The notion of local compatibility can be naturally extended to the set of PCA.

Definition 70 (PCA compatible at a state)**.** *Let $\mathbf{X} = \{X_1, ..., X_n\}$ be a set of PCA. Let $q = (q_1, ..., q_n) \in Q_{X_1} \times ... \times Q_{X_n}$. Let us note $C_i = (\mathbf{A}_i, \mathbf{S}_i) = config(X_i)(q_i)$, $\forall i \in [1, n]$. The PCA in $\mathbf{X}$ are compatible at state q iff*[2]*:*

1. *PSIOA compatibility: $psioa(X_1), ..., psioa(X_n)$ are compatible at $q_{\mathbf{X}}$.*
2. *Sub-automaton exclusivity: $\forall i, j \in [1 : n], i \neq j : \mathbf{A}_i \cap \mathbf{A}_j = \emptyset$.*
3. *Creation exclusivity: $\forall i, j \in [1 : n], i \neq j, \forall a \in \widehat{sig}(X_i)(q_i) \cap \widehat{sig}(X_j)(q_j)$: $created(X_i)(q_i)(a) \cap created(X_j)(q_j)(a) = \emptyset$.*

If $\mathbf{X}$ is compatible at state q, for every action $a \in \widehat{sig}(psioa(\mathbf{X}))(q)$, we note $\eta_{(\mathbf{X},q,a)} = \eta_{(psioa(\mathbf{X}),q,a)}$ and we extend this notation with $\eta_{(\mathbf{X},q,a)} = \delta_q$ if $a \notin \widehat{sig}(psioa(\mathbf{X}))(q)$.

2. We can remark that the conjunction of PSIOA compatibility and sub-automata exclusivity implies the compatibility of respective configurations as defined later in definition 76

3.3. A Toy Example: Specification of a Dynamic Reliable Broadcast

We illustrate DPIOA with the example of a distributed task: the Dynamic Byzantine Reliable Broadcast [GKK+20]. Intuitively, implementing this task would allow processes to reliably broadcast messages despite both (1) the presence of a certain fraction of malicious (Byzantine) processes that can arbitraly deviate from the prescribed protocol and (2) a non-static set of participants that can dynamically evolve along the execution. The presentation is progressive. In subsection 3.3.1, we give the specification of Static Byzantine Reliable Broadcast, where only one (potentially malicious) process known by everyone can broadcast one unique message in a reliable manner among a known static set of n participants. We also model an implementation of this task. Then, we very briefly discuss in subsection 3.3.2 how randomization can appear in the model, with a randomized implementation of the static Byzantine consensus problem, that guarantees that some processes can agree on a common value despite the presence of a certain fraction of Byzantine processes. Thereafter, we describe the Dynamic Byzantine Reliable Broadcast Problem [GKK+20] and gives a specification in subsection 3.3.3. Finally, we discuss how randomization can appear in an implementation of a relaxed specification of the Dynamic Byzantine Reliable Broadcast Problem [GKM+19].

First of all, we explain the pseudo-language to describe an automaton. As in [KLSV06], we assume a universal set V of variables. A variable represents a location within the state of a system. For each variable v, we assume a (static) type, which gives the set of values it may take on. A valuation $\mathbf{v}$ for $V' \subset \mathsf{V}$ is a function that associates with each variable $v \in V'$ a value in $type(v)$. We write $val(V')$ the set of evaluations of V'. Each automaton $\mathcal{A}$ presented in this section is associated with a set of variables $V_{\mathcal{A}}$ and its set of states $Q_{\mathcal{A}}$ is a set of valuations for this set of variables, i.e. $Q_{\mathcal{A}} \subseteq val(V_{\mathcal{A}})$. Each action of an automaton $\mathcal{A}$ is associated with conditions on its set of variables $V_{\mathcal{A}}$. A precondition Pre means that the action is enabled in the state $q \in val(V_{\mathcal{A}})$ if and only if it corresponds to an evaluation of the variables that verifies Pre. A postcondition Post means that if the action is triggered from a state $q \in val(V_{\mathcal{A}})$, then the next state q' will be the evaluation of variables that corresponds to q to which we have applied the updates of Post. A post condition can be randomized. For example if there is a variable v with $type(v) = \{0, 1\}$, a postcondition can be written Post: $v := rand(0 : 1)$ to says that variable v will be updated according to a uniform distribution on $\{0, 1\}$, while the other variales stay unchanged.

3.3.1. Modelisation of Byzantine Reliable Broadcast

Let us assume a static set of n processes $\mathbf{P} = \{P_{i_1}, ..., P_{i_n}\}$ with known ids in a set $Ids = \{i_1, ..., i_n\}$, connected by a reliable point-to-point network, i.e. they can directly communicate by message-passing with the certitude that any message sent will be eventually received. Among processes in $\mathbf{P}$, a clearly identified source P_{src} for some $src \in Ids$ want to broadcast an important message. Ideally, we would like that processes agree on this message. Of course, if there is no doubt that the source is honest, it is enough for him to simply send a copy of this message to each participant (we say he broadcasts its message). However, the processes suspect the presence of some traitors among them, that might even be the source. How to proceed to ensure that (1) if the source is correct, every correct process eventually deliver its message and (2) if one correct process delivers a message m, then every correct process eventually delivers m and never deliver another different message, even if the source and some of its friends are malicious?

This problem [LSP82], often called Byzantine Reliable Broadcast has been widely studied in the litterature. Imbs and Raynal have proposed a very simple asynchronous solution [IR15] that tolerates up to $t < n/5$ Byzantine failures, without randomization nor authentication. Imbs and Raynal present the specification of the problem as follows:

The reliable broadcast (denoted RB-broadcast) communication abstraction provides the processes with two operations denoted RB-Broadcast() and RB-Deliver(). When a process invokes RB-Broadcast(), we say that it "RB-Broadcasts a message". Similarly, when a process executes RB-Deliver(), we say that it "RB-delivers a message". RB-broadcast is defined by the following properties:

- RB-Validity. If a correct process RB-Delivers the message m from a correct process P_i, then P_i RB-Broadcast m.
- RB-Integrity. A correct process RB-delivers at most one message from any process P_i.
- RB-Agreement. Given a process P_i, no two correct processes RB-Deliver distinct messages from P_i
- RB-Termination-1. If a correct process RB-Broadcast a message, all correct processes eventually RB-Deliver this message.
- RB-Termination-2. If a correct process RB-Delivers a message from P_i (possibly faulty) then all correct processes eventually RB-Deliver a message from P_i.

A description of this abstraction is given with a pseudo-language in figure 3.5. An intuition is given with the figures 3.4 and 3.6.

The pseudo-code of the $n/5$-resilient solution of [IR15] is given in figure 3.7, while its automata-based description is given in figure 3.8 . The automata-based description of the reliable point-to-point network is given in figure 3.9.

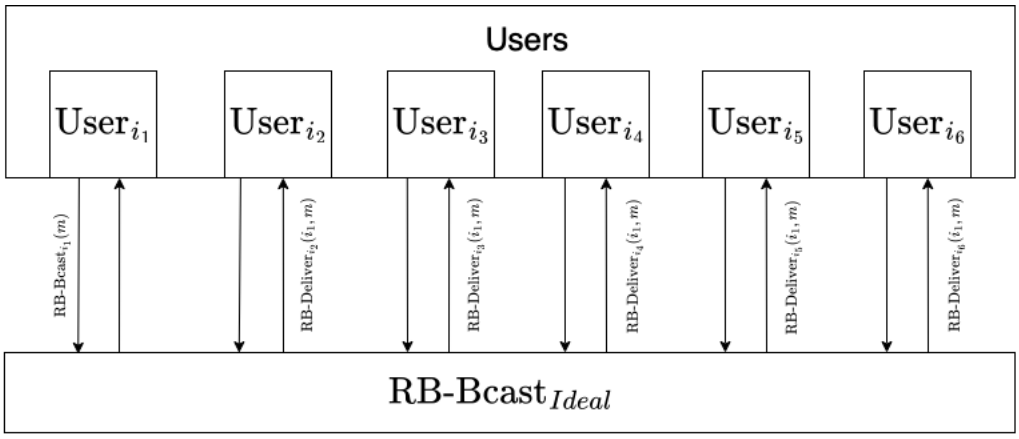
In figure 3.6, we can see that the process P_{i_1} is the adversary for the real-world implementation. In real-ideal paradigm [Lin17], we often compare this situation to an idealized-world, where the users interact with an ideal specification that also interact with an idealized adversary called the simulator. This simulator has a very limited power and is often invoked for sake of fair comparison. Its presence becomes very useful to specify subtle security properties such as privacy.

The natural question, "does the protocol solve the problem?" can be reformulated as follows: "is it possible for an environment (here the users) to distinguish the ideal world from the real world?", where the real world is composed of processes exchanging messages through a network to coordinate with each other and reach a decision.

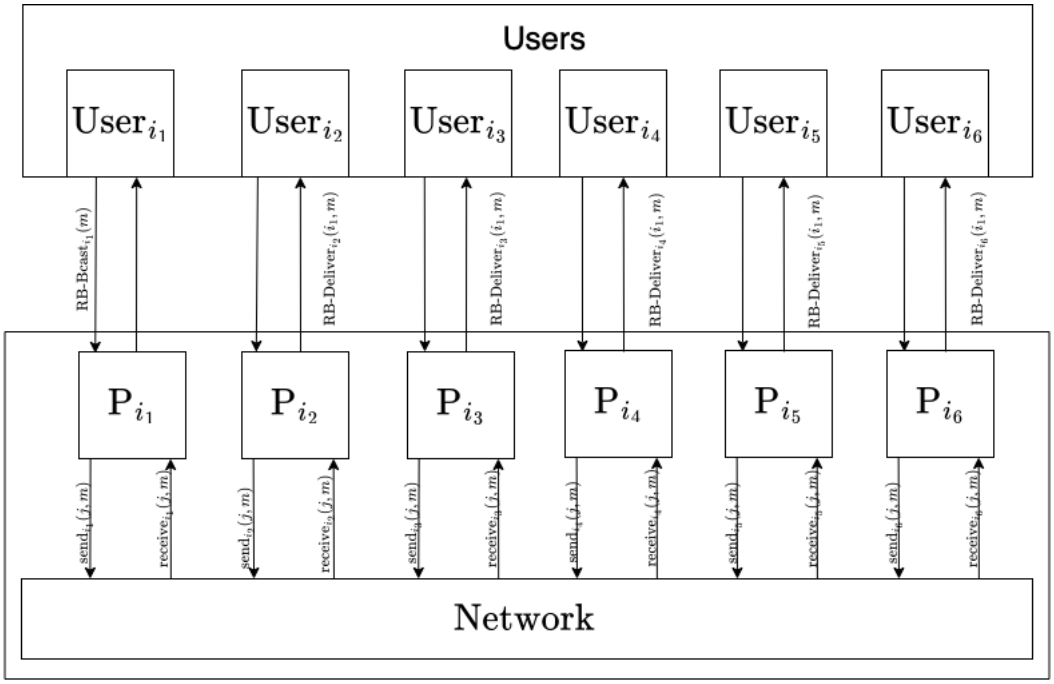
3.3.2. Static Consensus

The Byzantine Consensus Problem [PSL80] is another agreement problem where the different parties (1) propose their inputs and (2) have to agree on a common value. Moreover, (3) if all correct parties propose the same value, then this value has to be decided. It is well-known that a fully asynchronous solution of this problem is necessarily randomized if it is resilient to at least one crash [FLP85]. The first fully asynchronous solution of the binary version of the problem has been proposed by Ben-Or [BO83]. The solution is resilient to up to $t < n/5$. The figures 3.10 and 3.11 underline the similarities between distributed tasks (see figures 3.4 and 3.6) .

The solution of Ben-Or is given in figure 3.12. We do not provide the automata-based description of this solution. We just give the part of the modelisation that concern the potentially probabilistic transition of step (3)(c).



(a) BRB ideal without failure



(b) BRB real without failure

Figure 3.4. – Static Non-probabilistic Byzantine Reliable Broadcast

Static Signature
Input:
$\{\text{RBcast}_{src}(m) | src \in Ids, m \in \mathcal{M}\}$
Output:
$\{\text{Deliver}_i(src, m) | i, src \in Ids, m \in \mathcal{M}\}$
Steps
Output $\text{Deliver}_i(src, m)$
Pre: $m \in$ bcastReq
$\wedge (m, i) \notin$ delivered
$\wedge (m' \neq m) \notin$ deliveredOnce
Post: delivered.add(m, i)
$\wedge$deliveredOnce.add(m)

Variables
corrupted $\subset Ids$, init: $\emptyset$
bcastReqCorr $\subset \mathcal{M}$, init: $\emptyset$
bcastReqByz $\subset \mathcal{M}$, init: $\emptyset$
bcastReq $\subset \mathcal{M}$, init: $\emptyset$
deliveredOnce $\subset \mathcal{M}$, init: $\emptyset$
delivered $\subset \mathcal{M} \times Ids$, init: $\emptyset$
always:
bcastReq = bcastReqByz $\cup$ bcastReqCorr

Input $\text{D-RBcast}_{src}(m)$
Post: if $src \notin$ corruptedOnce
bcastReqCorr.add(m)
elif $src \in$ corrupted
bcastReqByz.add(m)

Figure 3.5. – Specification of Byzantine Reliable Broadcast

3.3.3. Dynamic Byzantine Reliable Broadcast

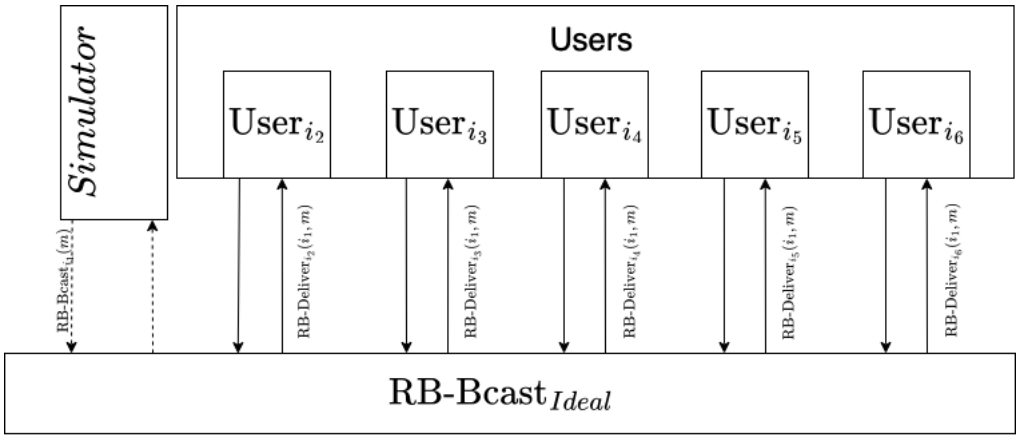
The Dynamic Byzantine Reliable Broadcast [GKK+20] is a primitive that extends the (static) Byzantine Reliable Broadcast to the dynamic setting, where the set of participants is dynamically evolving during the time. Hence, the specification is enriched with the following operations and callbacks:

- D-Join$_i$ used outside the system by user with id i to join the system. The operation eventually returns the associated acknowledgement D-Join-Ack$_i$.
- D-Leave$_i$ used by a participating user with id i to leave the system. The operation eventually returns the associated acknowledgement D-Leave-Ack$_i$.

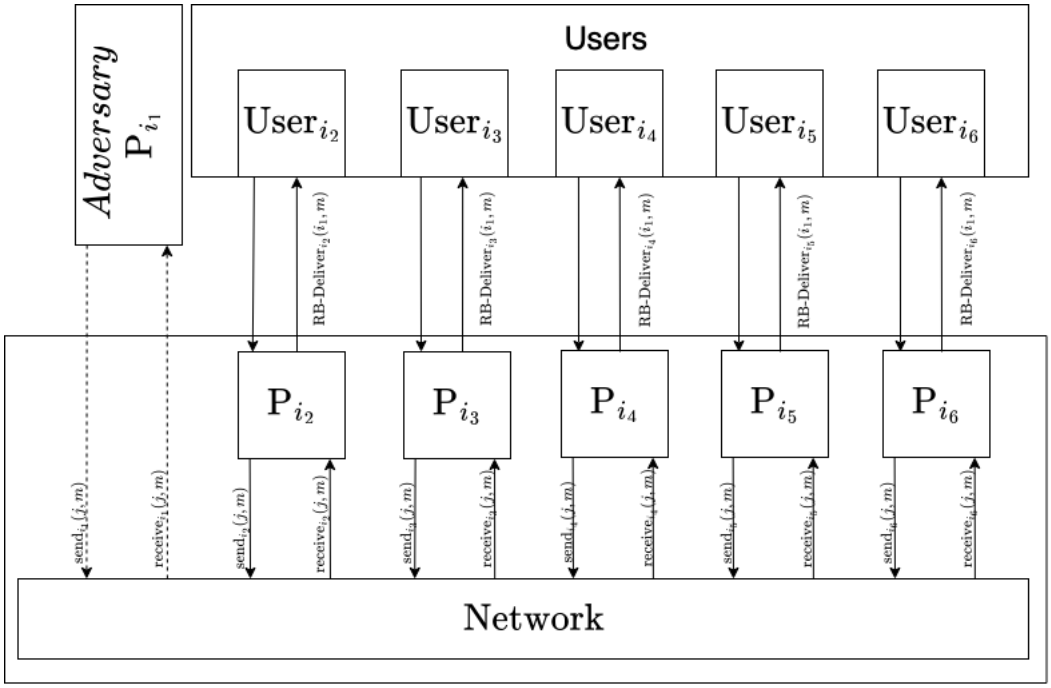
We also allow consider the case where it is possible to D-RBBcast and D-RBDeliver several messages. To do so, a sequence number is associated to every message, and the protocol should guarantee that two conflicting messages for the same source and the same sequence number cannot be D-BRDelivered by two corret processes.

The figure 3.13 gives a first intuition of the structure of a potential implementation without any failure. This implementation consists of the composition of the composition of two DIOA. The first DIOA is the network (see specification in figure 3.18), while the second DIOA is the dynamic set of processes. The specification (see figure 3.17) of the dynamic set of processes simply gives (1) the initial configuration corresponding to the start state and (2) the mechanism of creation. Here, the creation is always allowed by a passive automaton, called the manager, with one unique state and all the inputs of the form D-join$_i$() enabling the creation of new processes (see figure 3.15) within the DIOA representing a dynamic set of processes. A possible automata-based specification of the failure-free version of the primitive is given in figure 3.14.

Of course, the problem is more interesting with Byzantine failures. An intuition of a possible modelisation is given in figure 3.19. In this representation, an adversary can corrupt in the flight the processes. It can also repair the corrupted processes and corrupt new ones. In our modelisation, the corruption is not direct, but is performed by a kind of handshake. The manager has been replaced by the relay (see figure 3.22) to handle this handshake. First, the adversary sends a request corrupt-ack$_i$ to the relay. Then, the relay corrupts the process P_i that is destroyed and finally the relay acknowledge the adversary that can create a new process P_i' that will send messages on behalf of the process P_i. Such an handshake allows the environment to be aware of the different corruptions and so to partition its



(a) BRB ideal with Byzantine failures



(b) BRB real with Byzantine failures

Figure 3.6. – Static Non-probabilistic Byzantine Reliable Broadcast with Byzantine failures

operation RB_broadcast MSG(v_i) **is**
(1) broadcast INIT(i, v_i).

when INIT(j, v) **is received from** p_j **do**
(2) **if** (first reception of INIT$(j, -)$ and WITNESS$(j, -)$ not yet broadcast) **then** broadcast WITNESS(j, v) **end if**.

when WITNESS(j, v) **is received do**
(3) **if** (WITNESS(j, v) received from $(n - 2t)$ different processes and WITNESS(j, v) not yet broadcast)
(4) **then** broadcast WITNESS(j, v)
(5) **end if**;
(6) **if** (WITNESS(j, v) received from $(n - t)$ different processes and MSG$(j, -)$ not yet RB_delivered)
(7) **then** RB_deliver MSG(j, v)
(8) **end if**.

Figure 3.7. – A simple solution from Imbs and Raynal to Byzantine Reliable Broadcast

Static Signature
Input:
$\{\text{BRBcast}_i(m) | m \in \mathcal{M}\}$ if $i = src$
$\{\text{rcv}_i(j, m') | j \in Ids, m' \in \mathcal{M} \times \text{TAG}\}$
Output:
$\{\text{BRDeliver}_i(m) | m \in \mathcal{M}\}$
$\{\text{send}_i(j, m') | j \in Ids, m' \in \mathcal{M} \times \text{TAG}\}$

Steps
Input $\text{BRBcast}_i(m)$
Post: if $(\text{INIT}, m' \neq m) \notin$ toBcast:
    toBcast.add(INIT, m)

Input $\text{rcv}_i(j, (\text{TAG}, m))$
Post:
if TAG = INIT:
    if $(\text{WITNESS}, m' \neq m) \notin$ toBcast:
        toBcast.add$(\text{WITNESS}, m)$
if TAG = WITNESS:
    witness$[m]$.add(j)
    if $|\text{witness}[m]| = n - 2t + 1$:
        toBcast.add$(\text{WITNESS}, m)$
    if $|\text{witness}[m]| = n - t + 1$:
        toDeliver.add(m)

Variables
toBcast $\subset \text{TAG} \times \mathcal{M}$
toDeliver $\in \mathcal{M}$
sent $\subset Ids \times \text{TAG} \times \mathcal{M}$
delivered $\in \{0, 1\}$
dict witness
witness.keys() $\subset \mathcal{M}$
witness.values() $\subset Ids$

Constant
$n = |Ids|$
$t < n/5$
$src \in Ids$
$\text{TAG} = \{INIT, WITNESS\}$

Output $\text{send}_i(j, (tag, m))$
Pre: $(tag, m) \in$ toBcast
    $\wedge (j, (tag, m)) \notin$ sent
Post: sent.add$(j, (tag, m))$

Output $\text{BRDeliver}_i(m)$
Pre: toDeliver $= m \wedge$ delivered $= 0$
Post: delivered $= 1$

Figure 3.8. – Local Protocol: Process P_i following the solution proposed by Imbs and Raynal.

Static Signature
Input:
$\{\text{send}_i(j, m) | i, j \in Ids, m \in \mathcal{M}'\}$
Output:
$\{\text{rcv}_i(j, m) | i, j \in Ids, m \in \mathcal{M}'\}$

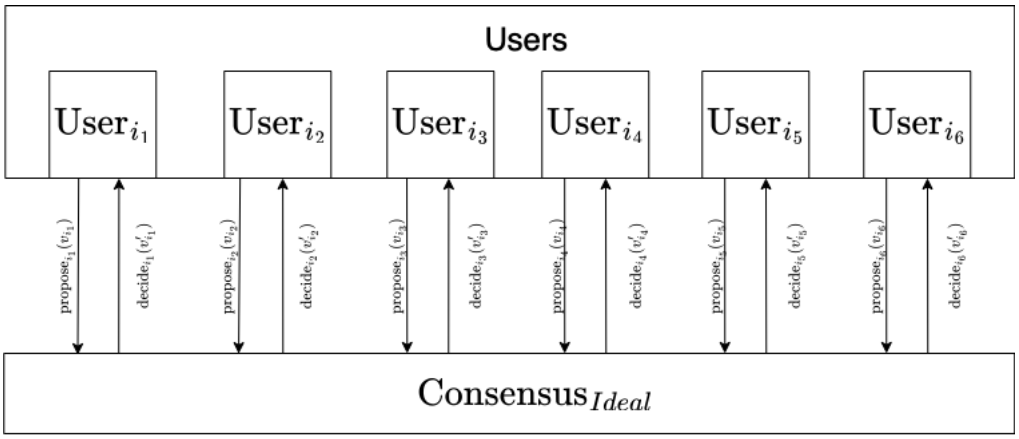
Variables
multiset toSend $\subset Ids^2 \times \mathcal{M}$

Steps
Input $\text{send}_i(j, m)$
Post: toSend.addInstance(i, j, m)

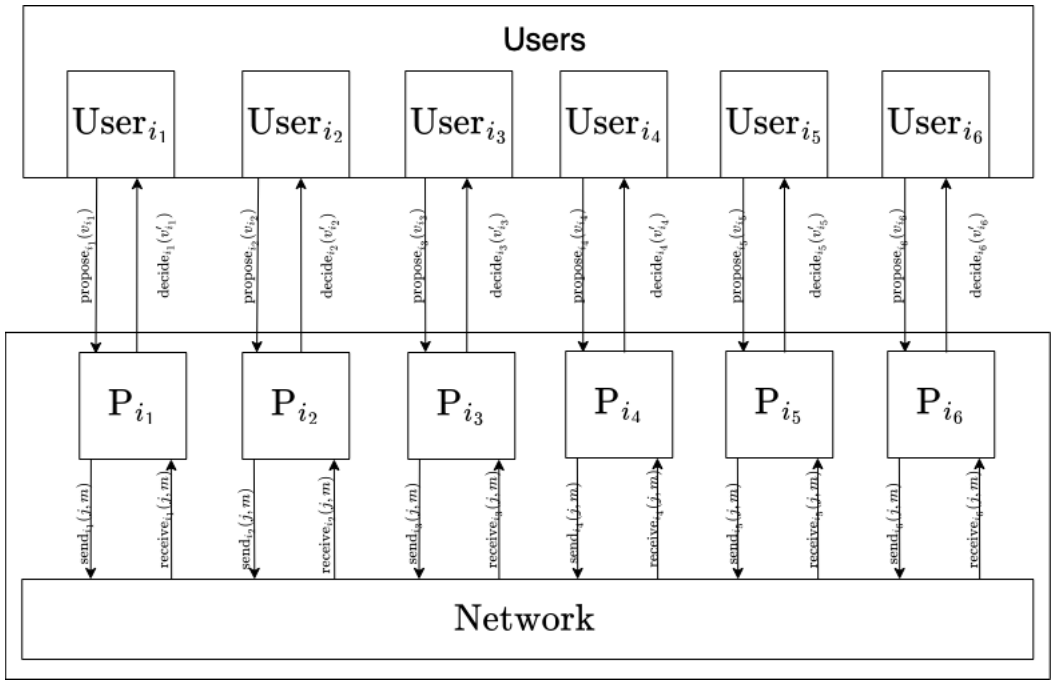
Output $\text{rcv}_i(j, m)$
Pre: $(j, i, m) \in$ toSend
Post: toSend.removeInstance$((j, i, m))$

Figure 3.9. – Static Network

observations into the 2 cases with or without an overcorrupted system. This modelisation can help to show that the real system implements the ideal specification in the sense that an environment cannot distinguish the two situations (see figure 3.20). We propose a specification of the problem (see figure 3.21) with a naive condition for the overcorruption. The solution proposed in [GKK+20] does not implement this exact specification, but something similar. This solution has a certain cost in term of communication complexity. To improve its complexity, we could be tempted to relax the specification with some probabilistic guarantees. Typically, it is possible to improve significantly the complexity of (static) Byzantine Reliable Broadcast with randomization [GKM+19]. It is an open problem to precisely specify and implement a combination of [GKK+20] and [GKM+19] that would be a scalable Dynamic Probabilistic Byzantine Reliable Broadcast.

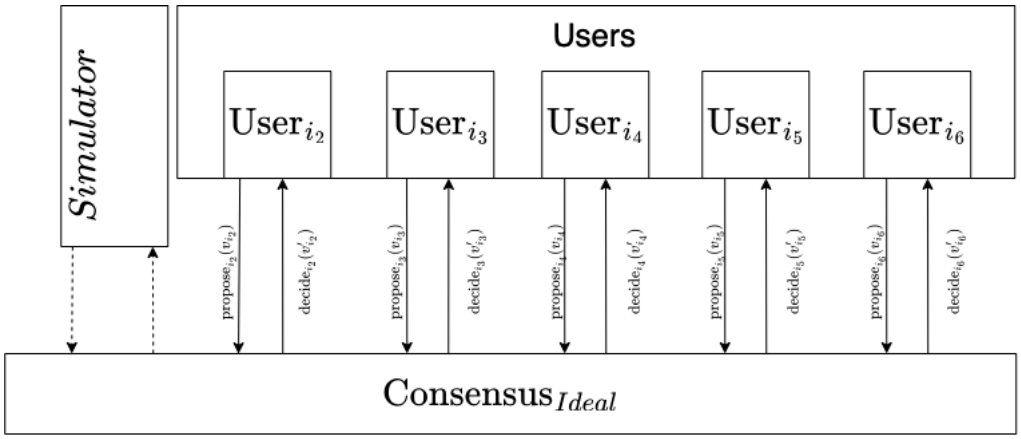


(a) Consensus ideal without failure

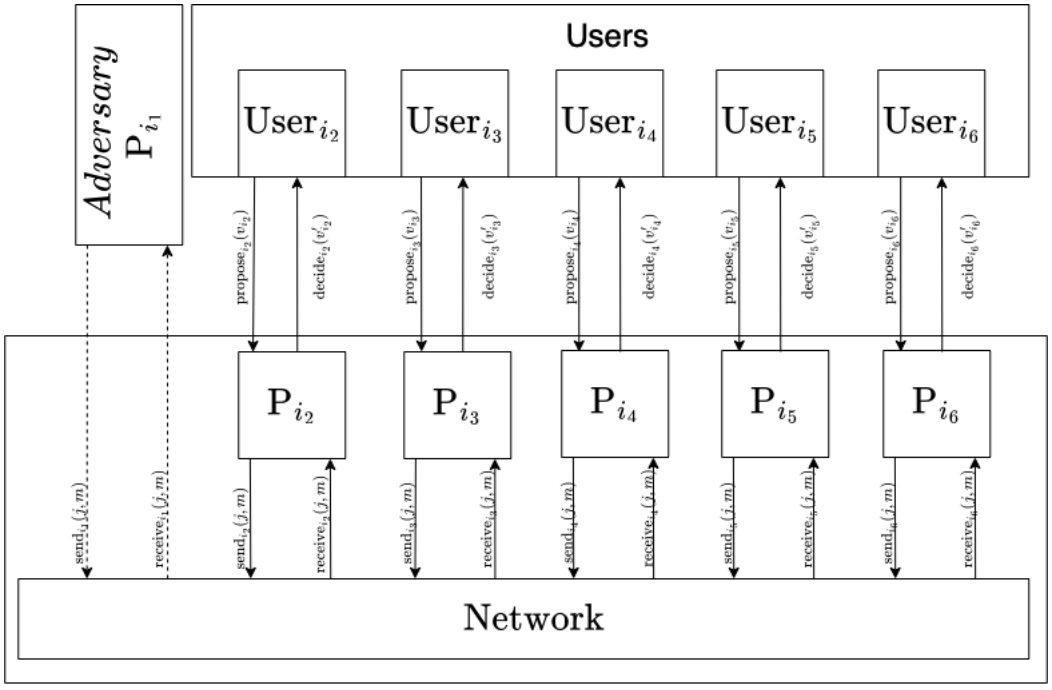


(b) Consensus real without failure

Figure 3.10. – Static Probabilistic Consensus without failure



(a) Consensus ideal with failures



(b) Consensus real with Byzantine failures

Figure 3.11. – Static Probabilistic Consensus with Byzantine failures

B — Byzantine Protocol

Process P**:** Initial value x_P.

step 0: set $r := 1$.

step 1: Send the message $(1, r, x_P)$ to all the processes.

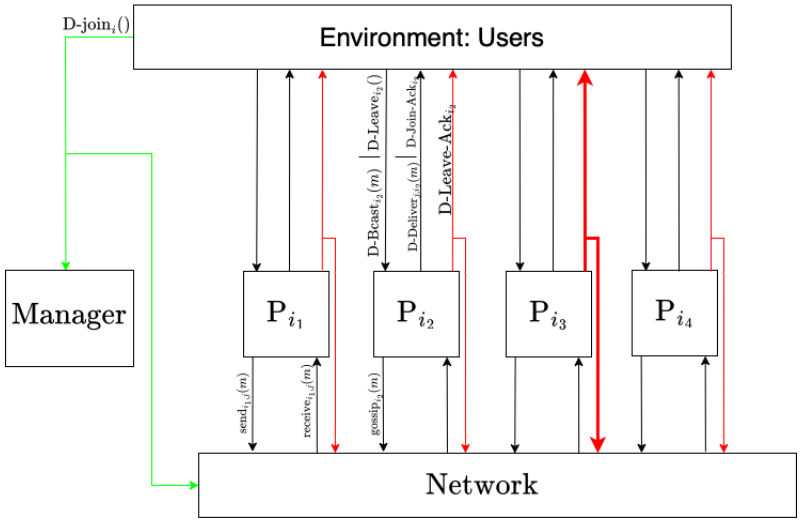
step 2: Wait till messages of type $(1, r, *)$ are received from $N - t$ processes. If more than $(N + t)/2$ messages have the same value v, then send the message $(2, r, v, D)$ to all processes. Else send the message $(2, r, ?)$ to all processes.

step 3: Wait till messages of type $(2, r, *)$ arrive from $N - t$ processes.

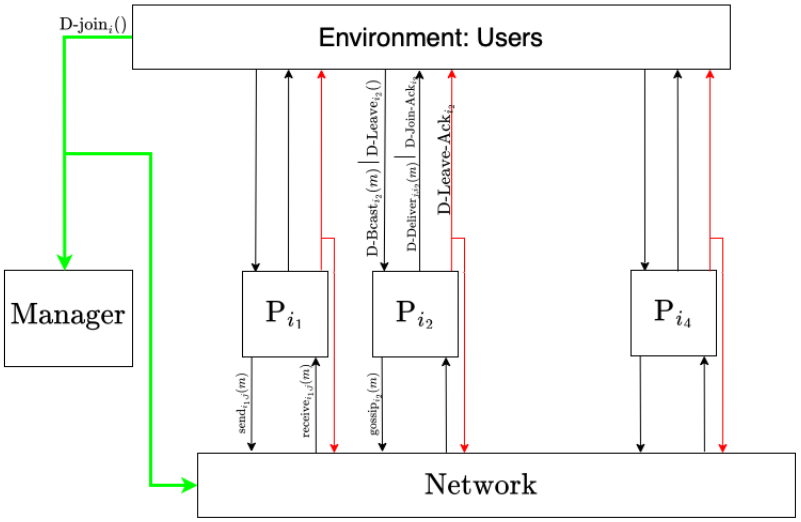
(a) If there are at least $t+1$ D-messages $(2, r, v, D)$, then set $x_P := v$.
(b) If there are more than $(N + t)/2$ D-messages then **decide** v.
(c) Else set $x_P = 1$ or 0 each with probability $\frac{1}{2}$.
step 4: Set $r := r + 1$ and go to step 1.

Constant
S = $\{(0, D), (1, D), ?\}$
N
$t < N/5$
Variables
round $\in \mathbb{N}$
$\mathrm{x}_P \in \{0, 1\}$
2dimension-table delivered2[][]
delivered2$[r \in \mathbb{N}][s \in S] \subset Ids$
...
Steps
...
Input $\mathrm{rcv}_i(j, (2, r, s))$
Post: delivered2$[r][s].add(j)$
if $|\text{delivered2}[r]| = N - t$:
if (a):
...
if (b):
...
else $\mathrm{x}_P := rand(0 : 1)$
round ++

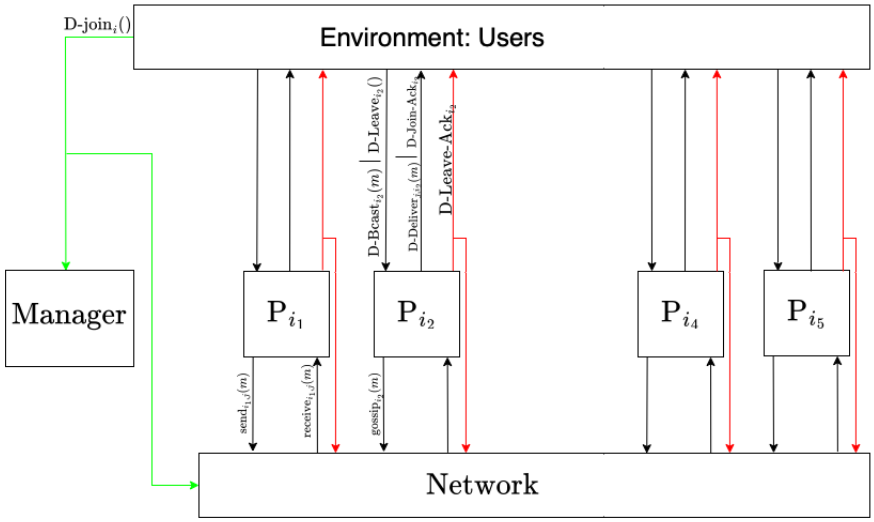
Figure 3.12. – BenOr Consensus



(a) DBRB: Initially 4 processes, but process P_3 is about to leave



(b) DBRB: process P_3 left, but process P_5 is about to join



(c) DBRB: process P_5 joined the system

Figure 3.13. – Dynamic Byzantine Reliable Broadcast Real without-failure.

The Manager is just a passive automaton with one unique state and all the inputs of the form D-join$_i$() enabling the creation of new processes within the DIOA representing a dynamic set of processes.

Universal Signature
Input:
$\{\text{D-Join}_i(), \text{D-Leave}_i() | i \in Ids\}$
$\{\text{D-RBcast}_i(s, m) | i \in Ids, s \in \mathbb{N}, m \in \mathcal{M}\}$
Output:
$\{\text{D-Join-Ack}_i, \text{D-Leave-Ack}_i | i \in Ids\}$,
$\{\text{D-RDeliver}_i(j, s, m) | i, j \in Ids, s \in \mathbb{N}, m \in \mathcal{M}\}$

Steps
Output D-Join-Ack$_i$
Pre: $i \in$ entering
Post: installed.add(i)
$\wedge$entering.remove(i)

Output D-Leave-Ack$_i$
Pre: $i \in$ leaving
Post: installed.remove(i)
$\wedge$leaving.remove(i)

Output D-RDeliver$_j(i, s, m)$
Pre: $j \in$ inside
$\wedge(i, s, m) \in$ bcastReq
$\wedge(i, s, m, j) \notin$ delivered
$\wedge(i, s' \neq s, m) \notin$ deliveredOnce
Post: delivered.add(i, s, m, j)
$\wedge$deliveredOnce.add(i, s, m)

Variables
installed $\subset Ids$, init: $\mathbf{I}^0$
entering $\subset Ids$, init: $\emptyset$
leaving $\subset Ids$, init: $\emptyset$
inside $\subset Ids$, init: $\mathbf{I}^0$
bcastReq $\subset Ids \times \mathbb{N} \times \mathcal{M}$, init: $\emptyset$
deliveredOnce $\subset Ids \times \mathbb{N} \times \mathcal{M}$, init: $\emptyset$
delivered $\subset Ids \times \mathbb{N} \times \mathcal{M} \times Ids$, init: $\emptyset$
always:
inside = installed $\cup$ entering $\cup$ leaving

Input D-Join()$_i$
Post: if $i \notin$ inside,
entering.add(i)

Input D-Leave()$_i$
Post: if $i \in$ installed,
leaving.add(i)

Input D-RBcast$_i(s, m)$
Post: if $i \in$ intalled $\setminus$ crashed
bcastReq.add(i, s, m)

Figure 3.14. – Specification of Dynamic Reliable Broadcast without failure: DRB$_{ideal}$

Universal Signature
Input:
D-Leave$_i$()
$\{\text{D-RBcast}_i(m) | m \in \mathcal{M}\}$
$\{\text{rcv}_i(j, m) | j \in Ids, m \in \mathcal{M}\}$
Output:
D-Join-Ack$_i$, D-Leave-Ack$_i$
$\{\text{D-RDeliver}_i(m) | m \in \mathcal{M}\}$
$\{\text{send}_i(j, m) | j \in Ids, m \in \mathcal{M}\}$
Internal:

Variables
destroyed $\in \{0, 1\}$, initially 0

Destruction
q.destroyed $= 1 \iff \widehat{sig}(\text{Alg}_i)(q) = \emptyset$

Steps
Output D-Leave-Ack$_i$
Pre: **UNSPECIFIED**
Post: destroyed := 1

Figure 3.15. – Local protocol without failure: Process P$_i$

The steps are unspecified

Universal Signature
Input:
D-Join$_i$()

Steps
Input D-Join$_i$()

Figure 3.16. – Manager

Configurations
$\{C \in Q_{conf} | auts(C) \subset \mathcal{P}(\{\text{Manager}\} \cup \{\text{P}_i | i \in Ids\})$
initially, $auts(Config(\text{Processes})(\bar{q}_{\text{Processes}})) = \{\text{Manager}\} \cup \mathbf{P}^0$
with $\mathbf{P}^0 = \{\text{P}_i | i \in \mathbf{I}^0\}$

Creations
$created(\text{Processes})(q)(\text{D-Join}_i()) = \{\text{P}_i\}$

Figure 3.17. – Dynamic Set of Processes (DPIOA)

Universal Signature
Input:
$\{\text{D-Join}_i, \text{D-Leave-Ack}_i | i \in Ids\}$
$\{\text{bcast}_i(m) | i \in Ids, m \in \mathcal{M}\}$
$\{\text{send}_i(j, m) | i, j \in Ids, m \in \mathcal{M}\}$
Output:
$\{\text{rcv}_i(j, m) | i, j \in Ids, m \in \mathcal{M}\}$

Steps
Output $\text{rcv}_i(j, m)$
Pre: $(j, i, m) \in$ toSend
Post: toSend.removeInstance$((j, i, m))$

Variables
inside $\subset Ids$
multiset toSend $\subset Ids^2 \times \mathcal{M}$

Steps
Input D-Join_i
Post: inside.add(i)

Input D-Leave-Ack_i
Post: inside.remove(i)

Input $\text{send}_i(j, m)$
Post: toSend.addInstance(i, j, m)

Input $\text{bcast}_i(m)$
Post: for each $j \in inside$:
toSend.addInstance(i, j, m)

Figure 3.18. – Network

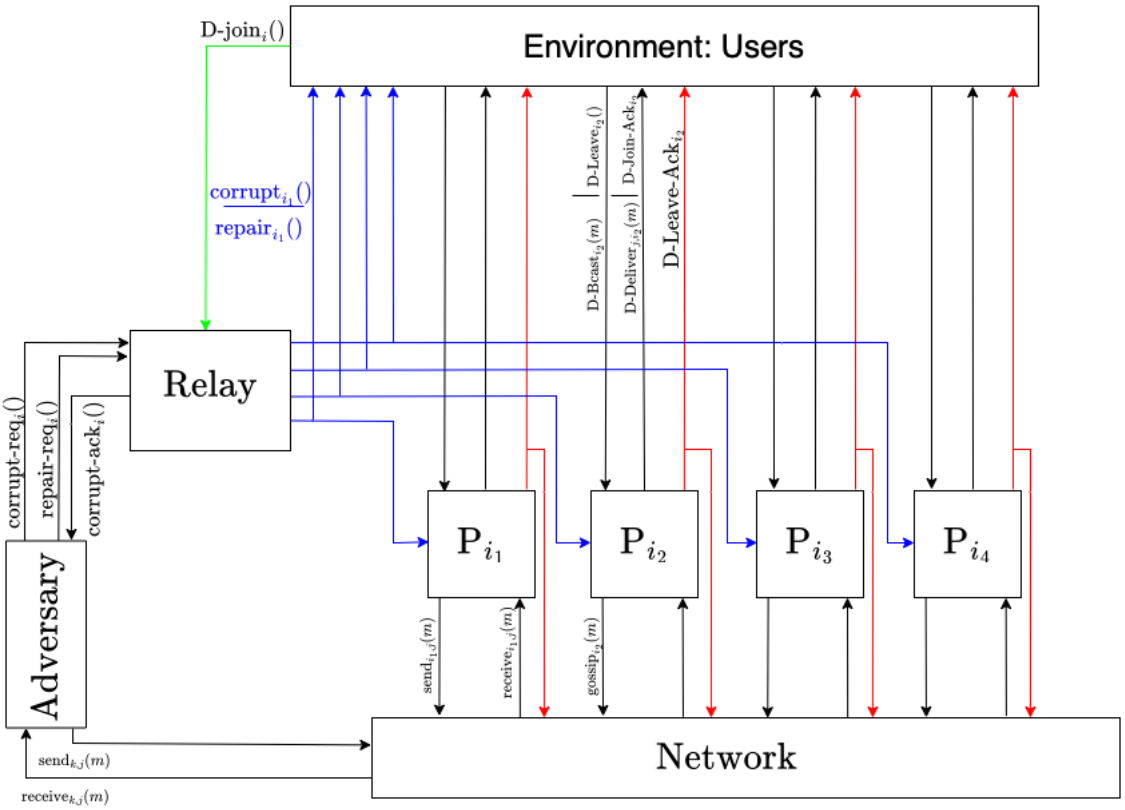
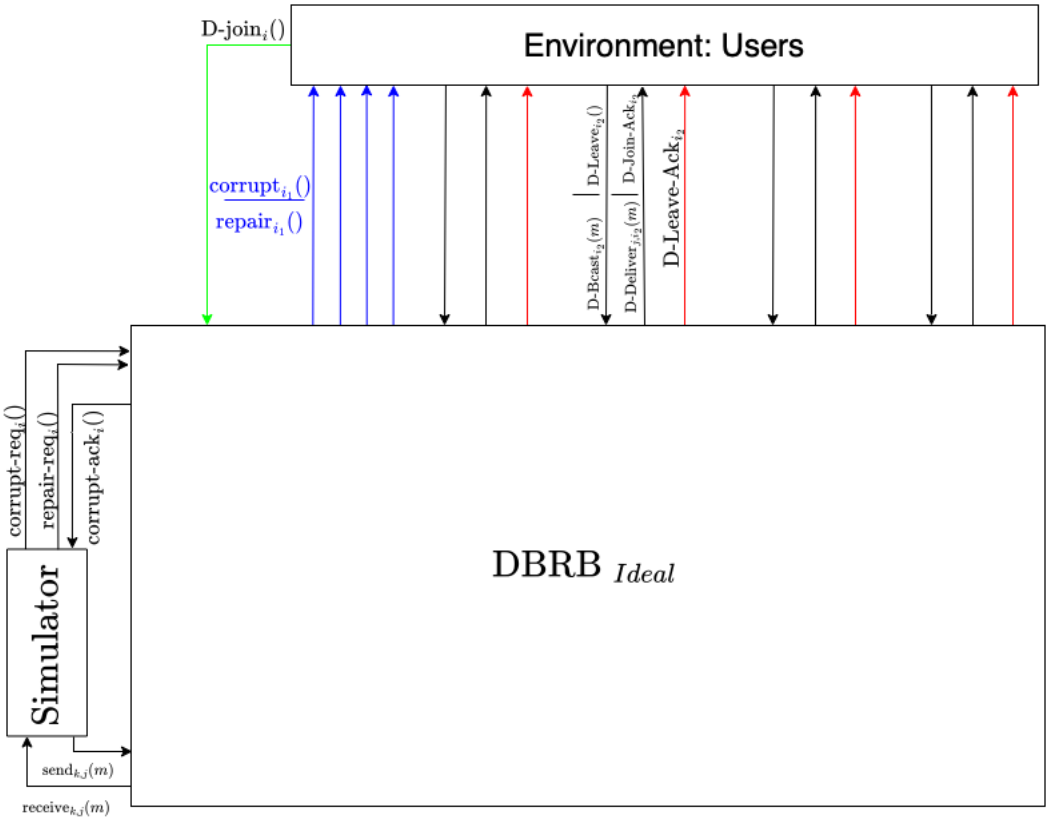
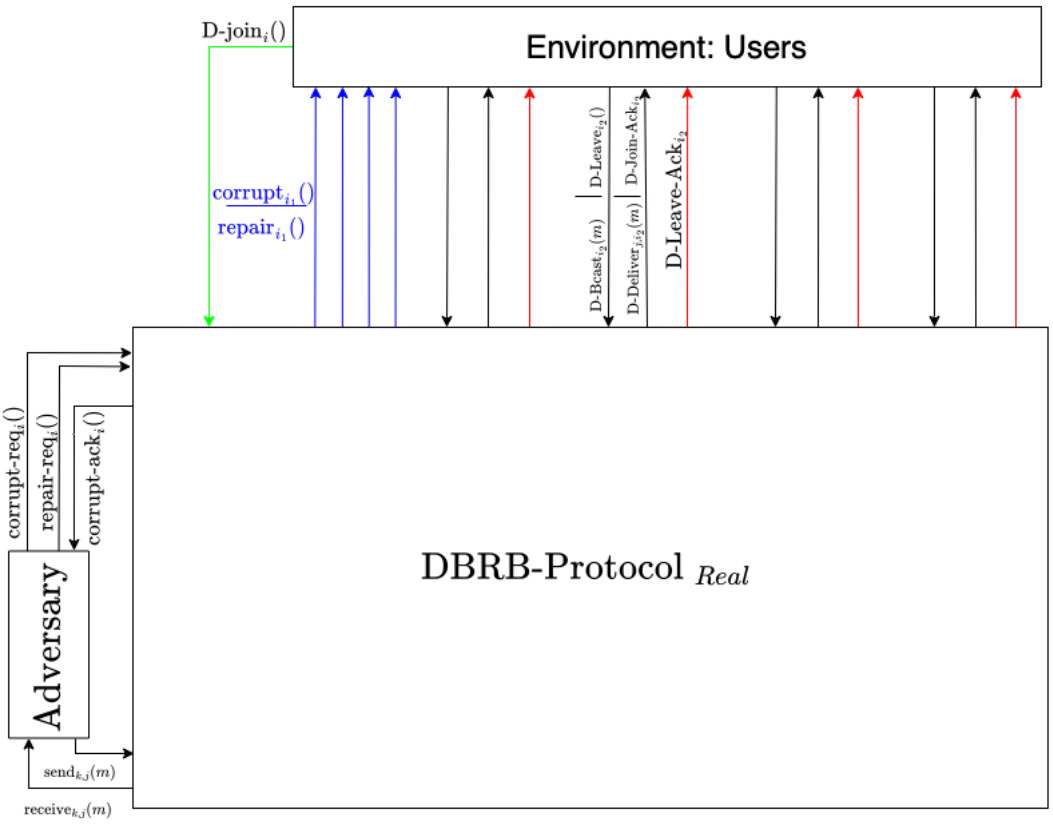


Figure 3.19. – Dynamic Byzantine Reliable Broadcast with Byzantine failures



(a) DBRB ideal with failures



(b) DBRB real-box with Byzantine failures

Figure 3.20. – Dynamic Byzantine Reliable Broadcast with Byzantine failures

Universal Signature
Input:
$\{\text{D-Join}_i(), \text{D-Leave}_i() | i \in Ids\}$
$\{\text{D-RBcast}_i(s, m) | i \in Ids, s \in \mathbb{N}, m \in \mathcal{M}\}$
$\{\text{Corrupt-Req}_i, \text{Repair-Req}_i | i \in Ids\}$
Output:
$\{\text{D-Join-Ack}_i, \text{D-Leave-Ack}_i | i \in Ids\}$,
$\{\text{D-RDeliver}_i(j, s, m) | i, j \in Ids, s \in \mathbb{N}, m \in \mathcal{M}\}$
$\{\text{Corrupt}_i, \text{Corrupt-Ack}_i, \text{Repair}_i | i \in Ids\}$

Steps
Output Corrupt_i
Pre: $i \in (\text{inside} \cap \text{corruptReq}) \setminus \text{corrupted}$
Post: corruptedOnce.add(i)
$\wedge$corrupted.add(i)
$\wedge$corruptReq.remove(i)
$\wedge$corruptToAck.add(i)
$\wedge$M := $\{(i, s, m) \in \text{bcastReqCorr}\}$
$\wedge$bcastReqByz.add(M)
$\wedge$bcastReqCorr.remove(M)
$\wedge[|\text{corruptedOnce} \cap \text{inside}| \geq |\text{installed}|/3]$
$\implies$ overcorrupted := 1

Output CorruptAck_i
Pre: $i \in$ corruptToAck
Post: corruptToAck.remove(i)

Output Repair_i
Pre: $i \in \text{inside} \cap \text{corrupted} \cap \text{repairReq}$
Post: corrupted.remove(i)
$\wedge$ repairReq.remove(i)

Output D-Join-Ack_i
Pre: $i \in$ entering
Post: installed.add(i)
$\wedge$entering.remove(i)

Output D-Leave-Ack_i
Pre: $i \in$ leaving
Post: installed.remove(i)
$\wedge$leaving.remove(i)
$\wedge[|\text{corruptedOnce} \cap \text{inside}| \geq |\text{installed}|/3]$
$\implies$ overcorrupted := 1

Output $\text{D-RDeliver}_j(i, s, m)$
Pre: $j \in$ inside
$\wedge(i, s, m) \in$ bcastReq
$\wedge(i, s, m, j) \notin$ delivered
$\wedge[(i, s' \neq s, m) \notin$ deliveredOnce
$\vee$overcorrupted == 1]
Post: delivered.add(i, s, m, j)
$\wedge$deliveredOnce.add(i, s, m)

Variables
installed $\subset Ids$, init: $\mathbf{I}^0$
entering $\subset Ids$, init: $\emptyset$
leaving $\subset Ids$, init: $\emptyset$
inside $\subset Ids$, init: $\mathbf{I}^0$
corruptReq $\subset Ids$, init: $\emptyset$
corruptedOnce $\subset Ids$, init: $\emptyset$, init: $\emptyset$
corrupted $\subset Ids$, init: $\emptyset$
corruptToAck $\subset Ids$, init: $\emptyset$
bcastReqCorr $\subset Ids \times \mathbb{N} \times \mathcal{M}$, init: $\emptyset$
bcastReqByz $\subset Ids \times \mathbb{N} \times \mathcal{M}$, init: $\emptyset$
bcastReq $\subset Ids \times \mathbb{N} \times \mathcal{M}$, init: $\emptyset$
deliveredOnce $\subset Ids \times \mathbb{N} \times \mathcal{M}$, init: $\emptyset$
delivered $\subset Ids \times \mathbb{N} \times \mathcal{M} \times Ids$, init: $\emptyset$
overcorrupted $\in \{0, 1\}$, init: 0
always:
bcastReq = bcastReqByz $\cup$ bcastReqCorr
inside = installed $\cup$ entering $\cup$ leaving

Input CorruptReq_i
Post: corruptReq.add(i)

Input RepairReq_i
Post: repairReq.add(i)

Input $\text{D-Join}()_i$
Post: if $i \notin$ inside,
entering.add(i)

Input $\text{D-Leave}()_i$
Post: if $i \in$ installed,
leaving.add(i)

Input $\text{D-RBcast}_i(s, m)$
Post: if $i \in$ intalled $\setminus$ corruptedOnce
bcastReqCorr.add(i, s, m)
elif $i \in \text{inside} \cap \text{corruptedOnce}$
bcastReqByz.add(i, s, m)

Figure 3.21. – Dynamic Byzantine Reliable Broadcast Specification: DBRB_{ideal}

Universal Signature
Input:
$\{\text{D-Join}_i()|i \in Ids\}$
$\{\text{Corrupt-Req}_i, \text{Repair-Req}_i|i \in Ids\}$
Output:
$\{\text{Corrupt}_i, \text{Corrupt-Ack}_i, \text{Repair}_i|i \in Ids\}$

Steps
Output Corrupt_i
Pre: $i \in$ corruptReq
Post: corruptReq.remove(i)
$\wedge$corruptToAck.add(i)

Output CorruptAck_i
Pre: $i \in$ corruptToAck
Post: corruptToAck.remove(i)

Output Repair_i
Pre: $i \in$ repairReq
Post: repairReq.remove(i)

Variables
corruptReq $\subset Ids$, init: $\emptyset$
corrupted $\subset Ids$, init: $\emptyset$
corruptToAck $\subset Ids$, init: $\emptyset$

Steps
Input CorruptReq_i
Post: corruptReq.add(i)

Input RepairReq_i
Post: repairReq.add(i)

Input $\text{D-Join}()_i$

Figure 3.22. – Relay

3.4. Executions, reachable states, partially-compatible automata

3.4.1. Executions, reachable states, traces

In previous sections, we have described how to model probabilistic transitions that might lead to the creation and destruction of some components of the system. In this section, we will define pseudo execution fragments of a set of automata to model the run of a set $\mathbf{A}$ of several dynamic systems interacting with each other. With such a definition, we will kill two birds with one stone, since it will allow defining *reachable states* of $\mathbf{A}$ and then compatibility of $\mathbf{A}$ as compatibility of $\mathbf{A}$ at each reachable state.

Definition 71 (pseudo execution, reachable states, partial-compatibility)**.** *Let $\mathbf{A} = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$ be a finite set of PSIOA (resp. PCA). A* pseudo execution fragment *of $\mathbf{A}$ is a finite or infinite sequence $\alpha = q^0 a^1 q^1 a^2 ...$ of alternating states and actions, such that:*

1. *If α is finite, it ends with a state. In that case, we note $lstate(\alpha)$ the last state of α.*
2. *$\mathbf{A}$ is compatible at each state of α, with the potential exception of $lstate(\alpha)$ if α is finite.*
3. *for ever action a^i, $(q^{i-1}, a^i, q^i) \in steps(\mathbf{A})$.*

The first state of a pseudo execution fragment α is noted $fstate(\alpha)$. A pseudo execution fragment α of $\mathbf{A}$ is a pseudo execution of $\mathbf{A}$ if $fstate(\alpha) = \bar{q}_{\mathbf{A}}$. The length $|\alpha|$ of a finite pseudo execution fragment α is the number of actions in α. A state q of $\mathbf{A}$ is said reachable *if there is a pseudo execution α s.t. $lstate(\alpha) = q$. We note $Reachable(\mathbf{A})$ the set of* reachable *states of $\mathbf{A}$. If $\mathbf{A}$ is compatible at every reachable state q, $\mathbf{A}$ is said* partially-compatible*.*

In [AL16], a compatible set of PCA is compatible at every (potentially non-reachable) state of the associated Cartesian product. In addition to being slightly less restrictive, our definition has other advantages. First, it could be more intuitive to model the movement of an agent $\mathcal{A}$ (a PSIOA) from location X (a PCA) to another location Y (another PCA). Indeed, if both X and Y can create the same PSIOA $\mathcal{A}$, then X and Y could not be compatible according to the corresponding definition in [AL16], while they can be partially-compatible according to definition 71. Second, this definition allows the construction of the next chapter 4 which is the main contribution of this manuscript.

Definition 72 (executions, concatenations)**.** *Let $\mathcal{A}$ be an automaton. An* execution fragment *(resp.* execution*) of $\mathcal{A}$ is a pseudo execution fragment (resp. pseudo execution) of $\{\mathcal{A}\}$. We use $Frags(\mathcal{A})$ (resp., $Frags^*(\mathcal{A})$) to denote the set of all (resp., all finite) execution fragments of $\mathcal{A}$. $Execs(\mathcal{A})$ (resp. $Execs^*(\mathcal{A})$) denotes the set of all (resp., all finite) executions of $\mathcal{A}$.*

We define a concatenation operator $\frown$ for execution fragments as follows:
If $\alpha = q^0 a^1 q^1 ... a^n q^n \in Frags^*(\mathcal{A})$ and $\alpha' = q^{0\prime} a^{1\prime} q^{1\prime} ... \in Frags^*(\mathcal{A})$, we define $\alpha \frown \alpha' \triangleq q^0 a^1 q^1 ... a^n q^n a^{1\prime} q^{1\prime} ...$ only if $s^0 = q^n$, otherwise $\alpha \frown \alpha'$ is undefined. Hence the notation $\alpha \frown \alpha'$ implicitly means $fstate(\alpha') = lstate(\alpha)$.

Let $\alpha, \alpha' \in Frags(\mathcal{A})$, then α is a proper prefix of α' iff $\exists \alpha'' \in Frags(\mathcal{A})$ such that $\alpha' = \alpha \frown \alpha''$ with $\alpha \neq \alpha'$. In that case, we note $\alpha < \alpha'$. We note $\alpha \leq \alpha'$ if $\alpha < \alpha'$ or $\alpha = \alpha'$ and say that α is a prefix of α'. Finally, α, α' are said comparable if either $\alpha \leq \alpha'$ or $\alpha' \leq \alpha$.

Definition 73 (traces)**.** *The* trace *of an execution α represents its externally visible part, i.e. the external actions. Let $\mathcal{A}$ be a PSIOA (resp. PCA). Let $q^0 \in Q_{\mathcal{A}}$, $(q, a, q') \in Steps(\mathcal{A})$, $\alpha, \alpha' \in Execs^*(\mathcal{A}) \times Execs(\mathcal{A})$ with $fstate(\alpha') = lstate(\alpha)$.*

$trace_{\mathcal{A}}(q^0)$ is the empty sequence, noted λ,

$$trace_{\mathcal{A}}(qaq') = \begin{cases} a \text{ if } a \in \widehat{ext}(\mathcal{A})(q) \\ \lambda \text{ otherwise.} \end{cases},$$

$trace_{\mathcal{A}}(\alpha^\frown\alpha') = trace_{\mathcal{A}}(\alpha)^\frown trace_{\mathcal{A}}(\alpha')$

We say that β is a trace of $\mathcal{A}$ if $\exists\alpha \in Execs(\mathcal{A})$ with $\beta = trace_{\mathcal{A}}(\alpha)$. We note $Traces(\mathcal{A})$ (resp. $Traces^*(\mathcal{A})$, resp. $Traces^\omega(\mathcal{A})$) the set of traces (resp. finite traces, resp. infinite traces) of $\mathcal{A}$. When the automaton $\mathcal{A}$ is understood from context, we write simply $trace_\alpha$.

The projection of a pseudo-execution α on an automaton $\mathcal{A}_i$, noted $\alpha \upharpoonright \mathcal{A}_i$, represents the contribution of $\mathcal{A}_i$ to this execution.

Definition 74 (projection)**.** *Let $\mathbf{A}$ be a set of PSIOA (resp. PCA), let $\mathcal{A}_i \in \mathbf{A}$. We define projection operator $\upharpoonright$ recursively as follows: For every $(q, a, q') \in Steps(\mathbf{A})$, for every α, α' being two pseudo executions of $\mathbf{A}$ with $fstate(\alpha') = lstate(\alpha)$.*

$$(q, a, q') \upharpoonright \mathcal{A}_i = \begin{cases} (q \upharpoonright \mathcal{A}_i), a, (q' \upharpoonright \mathcal{A}_i) \text{ if } a \in \widehat{sig}(\mathcal{A}_i)(q \upharpoonright \mathcal{A}_i) \\ (q \upharpoonright \mathcal{A}_i) = (q' \upharpoonright \mathcal{A}_i) \text{ otherwise.} \end{cases},$$

$(\alpha^\frown\alpha') \upharpoonright \mathcal{A}_i = (\alpha \upharpoonright \mathcal{A}_i)^\frown(\alpha' \upharpoonright \mathcal{A}_i)$

3.4.2. PSIOA and PCA composition

We are ready to define the composition operator, the most important operator for concurrent systems. Such a definition is normally simpler but a detour was necessary to integrate PCAs compatible at each reachable state, but not necessarily at all states of the Cartesian product.

Definition 75 (PSIOA partial-composition)**.** *If $\mathbf{A} = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$ is a partially-compatible set of PSIOA, with $\mathcal{A}_i = (Q_{\mathcal{A}_i}, \bar{q}_{\mathcal{A}_i}, sig(\mathcal{A}_i), D_{\mathcal{A}_i})$, then their partial-composition $\mathcal{A}_1||...||\mathcal{A}_n$, is defined to be $\mathcal{A} = (Q_{\mathcal{A}}, \bar{q}_{\mathcal{A}}, sig(\mathcal{A}), D_{\mathcal{A}})$, where:*

- $Q_{\mathcal{A}} = Reachable(\mathbf{A})$
- $\bar{q}_{\mathcal{A}} = (\bar{q}_{\mathcal{A}_1}, ..., \bar{q}_{\mathcal{A}_n})$
- $sig(\mathcal{A}) : q \in Q_{\mathcal{A}} \mapsto sig(\mathcal{A})(q) = sig(\mathbf{A})(q)$ *as per definition 60.*
- $D_{\mathcal{A}} = \{(q, a, \eta_{(\mathbf{A},q,a)}) | q \in Q_{\mathcal{A}}, a \in \widehat{sig}(\mathbf{A})(q)\}$ *as per definition 60.*

The next definition describes the union of several configurations in the obvious way.

Definition 76 (Union of configurations)**.** *Let $C_1 = (\mathbf{A}_1, \mathbf{S}_1)$ and $C_2 = (\mathbf{A}_2, \mathbf{S}_2)$ be configurations such that $\mathbf{A}_1 \cap \mathbf{A}_2 = \emptyset$. Then, the union of C_1 and C_2, denoted $C_1 \cup C_2$, is the configuration $C = (\mathbf{A}, \mathbf{S})$ where $\mathbf{A} = \mathbf{A}_1 \cup \mathbf{A}_2$ and $\mathbf{S}$ agrees with $\mathbf{S}_1$ on $\mathbf{A}_1$, and with $\mathbf{S}_2$ on $\mathbf{A}_2$. Moreover, if $C_1 \cup C_2$ is a compatible configuration, we say that C_1 and C_2 are* compatible configurations*. It is clear that configuration union is commutative and associative. Hence, we will freely use the n-ary notation $C_1 \cup ... \cup C_n$, whenever $\forall i, j \in [1 : n], i \neq j, auts(C_i) \cap auts(C_j) = \emptyset$.*

Lemma 3. *Let $C_1 = (\mathbf{A}_1, \mathbf{S}_1)$ and $C_2 = (\mathbf{A}_2, \mathbf{S}_2)$ be configurations such that $\mathbf{A}_1 \cap \mathbf{A}_2 = \emptyset$. Let $C = (\mathbf{A}, \mathbf{S}) = C_1 \cup C_2$ be a compatible configuration. Then $sig(C) = sig(C_1) \times sig(C_2)$ (in the sense of definition 59).*

Proof.

$$\begin{aligned} out(C) &= \bigcup_{\mathcal{A}_k \in \mathbf{A}} out(\mathcal{A}_k)(\mathbf{S}(\mathcal{A}_k)) \\ &= (\bigcup_{\mathcal{A}_i \in \mathbf{A}_1} out(\mathcal{A}_i)(\mathbf{S}(\mathcal{A}_i))) \cup (\bigcup_{\mathcal{A}_j \in \mathbf{A}_2} out(\mathcal{A}_j)(\mathbf{S}(\mathcal{A}_j))) \\ &= (\bigcup_{\mathcal{A}_i \in \mathbf{A}_1} out(\mathcal{A}_i)(\mathbf{S}_1(\mathcal{A}_i))) \cup (\bigcup_{\mathcal{A}_j \in \mathbf{A}_2} out(\mathcal{A}_j)(\mathbf{S}_2(\mathcal{A}_j))) \\ &= out(C_1) \cup out(C_2) \end{aligned}$$

$$\begin{aligned} in(C) &= \bigcup_{\mathcal{A}_k \in \mathbf{A}} in(\mathcal{A}_k)(\mathbf{S}(\mathcal{A}_k)) \setminus out(C) \\ &= (\bigcup_{\mathcal{A}_i \in \mathbf{A}_1} in(\mathcal{A}_i)(\mathbf{S}(\mathcal{A}_i))) \cup (\bigcup_{\mathcal{A}_j \in \mathbf{A}_2} in(\mathcal{A}_j)(\mathbf{S}(\mathcal{A}_j))) \setminus out(C) \\ &= (\bigcup_{\mathcal{A}_i \in \mathbf{A}_1} in(\mathcal{A}_i)(\mathbf{S}_1(\mathcal{A}_i))) \cup (\bigcup_{\mathcal{A}_j \in \mathbf{A}_2} in(\mathcal{A}_j)(\mathbf{S}_2(\mathcal{A}_j))) \setminus out(C) \\ &= in(C_1) \cup in(C_2) \setminus (out(C_1) \cup out(C_2)) \end{aligned}$$

$$\begin{aligned} int(C) &= \bigcup_{\mathcal{A}_k \in \mathbf{A}} int(\mathcal{A}_k)(\mathbf{S}(\mathcal{A}_k)) \\ &= (\bigcup_{\mathcal{A}_i \in \mathbf{A}_1} int(\mathcal{A}_i)(\mathbf{S}(\mathcal{A}_i))) \cup (\bigcup_{\mathcal{A}_j \in \mathbf{A}_2} int(\mathcal{A}_j)(\mathbf{S}(\mathcal{A}_j))) \\ &= (\bigcup_{\mathcal{A}_i \in \mathbf{A}_1} int(\mathcal{A}_i)(\mathbf{S}_1(\mathcal{A}_i))) \cup (\bigcup_{\mathcal{A}_j \in \mathbf{A}_2} int(\mathcal{A}_j)(\mathbf{S}_2(\mathcal{A}_j))) \\ &= int(C_1) \cup int(C_2) \end{aligned}$$

□

Partial-composition can be extended to PCA in the obvious manner.

Definition 77 (PCA partial-composition)**.** *If $\mathbf{X} = \{X_1, ..., X_n\}$ is a partially-compatible set of PCA, then their partial-composition $X_1||...||X_n$, is defined to be the PCA X (proved in theorem 2 in section 3.5) s.t. $psioa(X) = psioa(X_1)||...||psioa(X_n)$ and $\forall q \in Q_X$:*

- $config(X)(q) = \bigcup_{i \in [1,n]} config(X_i)(q \upharpoonright X_i)$
- $\forall a \in \widehat{sig}(X)(q)$, $created(X)(q)(a) = \bigcup_{i \in [1,n]} created(X_i)(q \upharpoonright X_i)(a)$,
  with the convention $created(X_i)(q_i)(a) = \emptyset$ if $a \notin \widehat{sig}(X_i)(q_i)$
- $hidden\text{-}actions(q) = \bigcup_{i \in [1,n]} hidden\text{-}actions(X_i)(q \upharpoonright X_i)$

3.5. Toolkit for configurations & PCA closure under composition

In this section, we define some tools to manipulate measure preserving bijections between probability distributions (relations of the form $\eta \overset{f}{\leftrightarrow} \eta'$). These tools will be used to prove (1) the closure of PCA under parallel composition (theorem 2) and (2) some intermediate results in the proof of monotonicity of dynamic creation/destruction of PSIOA with implementation relationship.

Merge, join, split Next definition introduces join function that returns the product of of several probabilistic distributions.

Definition 78 (join)**.** *Let $\tilde{\eta} = (\eta_1, ..., \eta_n) \in Disc(Q_1) \times ... \times Disc(Q_n)$ with each Q_i being a set. We define, $join(\tilde{\eta})$:* $\begin{cases} Q_1 \times ... \times Q_n & \rightarrow & [0,1] \\ \tilde{q} & \mapsto & (\eta_1 \otimes ... \otimes \eta_n)(\tilde{q}) \end{cases}$

The next lemma simply reduces the product measure of several measures over configurations into the associated join measure over the associated union of configurations.

Lemma 4 (Joint preserving probability distribution for union of configuration)**.** *Let $n \in \mathbb{N}$, let $\{C_k\}_{k\in[1:n]}$ be a set of compatible configurations and $C_0 = \bigcup_{k\in[1:n]} C_k$. Let $(\eta_p^0, ..., \eta_p^n) \in Disc(Q_{conf})^{n+1}$ s.t. $\forall k \in [0:n]$, $C_k \overset{a}{\rightharpoonup} \eta_p^k$ if $a \in \widehat{sig}(C_k)$ and $\eta_p^k = \delta_{C_k}$ otherwise.*
Then, $\forall (C_1', ..., C_n') \in Q_{conf}^n$, s.t. $\forall k \in [1:n], aut(C_k') = aut(C_k)$,

$$\eta_p^0(\bigcup_{k\in[1:n]} C_k') = (\eta_p^1 \otimes ... \otimes \eta_p^n)(C_1', ..., C_n') \ .$$

Proof. We note $\{C_k = (\mathbf{A}_k, \mathbf{S}_k)\}_{k\in[1:n]}$, $C_0 = (\mathbf{A}_0, \mathbf{S}_0)$, $q_k = TS(C_k)$ for every $k \in [0:n]$. We note $(\mathcal{I}, \mathcal{J})$ the partition of $[1:n]$ s.t. $\forall i \in \mathcal{I}, a \in \widehat{sig}(C_i)$ and $\forall j \in \mathcal{J}, a \notin \widehat{sig}(C_j)$. Since $\mathbf{A}_0 = \bigcup_{k\in[1:n]} \mathbf{A}_k$ and $\mathbf{S}_0$ agrees with $\mathbf{S}_k$ on $\mathcal{A} \in \mathbf{A}_k$ for every $k \in [1:n]$, we have $\eta_{\mathbf{A}_0,q_0,a} = \eta_{(\mathbf{A}_1,q_1,a)} \otimes ... \otimes \eta_{(\mathbf{A}_n,q_n,a)}$ with the convention $\eta_{(\mathbf{A}_j,q_j,a)} = \delta_{q_j}$, $\forall j \in \mathcal{J}$. Furthermore, for every $k \in [1,n], \eta_p^k \overset{TS}{\leftrightarrow} \eta_{(\mathbf{A}_k,q_k,a)}$, that is for every $(C_k', q_k') \in Q_{conf} \times Q_{\mathbf{A}_k}$ with $q_k' = TS(C_k')$, $\eta_p^k(C_k') = \eta_{(\mathbf{A}_k,q_k,a)}(q_k')$. Hence for every $((C_1', ..., C_n'), (q_1', ..., q_n')) \in Q_{conf}^n \times Q_{\mathbf{A}_0}$ with $q_1' = TS(C_1'), ..., q_n' = TS(C_n')$, $\eta_{(\mathbf{A}_0,q_0,a)}((q_1', ..., q_n')) = (\eta_{(\mathbf{A}_1,q_1,a)} \otimes ... \otimes \eta_{(\mathbf{A}_n,q_n,a)}))((q_1', ..., q_n')) = (\eta_p^1 \otimes ... \otimes \eta_p^n((C_1', ..., C_n'))$ (*).
By definition of η_p^0, $\forall (C_0', q_0') \in Q_{conf} \times Q_{\mathbf{A}_0}$, with $q_0' = TS(C_0')$, $\eta_{(\mathbf{A}_0,q_0,a)}(q_0') = \eta_p^0(C_0')$. Since we deal with preserving distribution and $\mathbf{A}_0 = \bigcup_{k\in[1:n]} \mathbf{A}_k$, q_0' is of the form $(q_1', ..., q_n')$ with $q_k' \in Q_{\mathbf{A}_k}$ and verifies $C_0' = C_1' \cup ... \cup C_n'$ with $auts(C_k') = \mathbf{A}_k$ and $TS(C_k') = q_k'$ (**).
Hence we compose (*) and (**) to obtain for every configuration $C_0' = (\mathbf{A}_0, \mathbf{S}_0')$, for every finite set of configurations $\{C_k' = (\mathbf{A}_k, \mathbf{S}_k')\}_{k\in[1:n]}$, s.t. $C_0' = \bigcup_{k\in[1:n]} C_k'$, then $\eta_p^0(C_0') = (\eta_p^1 \otimes ... \otimes \eta_p^n)((C_1', ..., C_n'))$.

□

The merged measure introduced in next definition represents the probability to obtain a certain configuration from the product of a tuple of several measures. Let us remark that a configuration can be obtained by different union operations (e.g. $(C_1 \cup C_2) \cup C_3 = C_1 \cup (C_2 \cup C_3)$ or $C_1 \cup C_2 = C_2 \cup C_1$).

Definition 79 (merge)**.** *Let $\tilde{\eta} = (\eta_1, ...\eta_n) \in Disc(Q_{conf})^n$. We define*

$$merge(\tilde{\eta})\colon \begin{cases} Q_{conf} & \to & [0,1] \\ C & \mapsto & \sum\limits_{(C'_1,...,C'_n)\in Q^n_{conf}} join(\tilde{\eta})((C'_1,...,C'_n)) \cdot \mathbb{1}_{(C'_1\cup...\cup C'_n)=C} \end{cases}$$

Here, we give a few results for merging operation applied to preserving transitions triggered from compatible configurations. In that case, every configuration C' reacheable by the merge measure $merge(\tilde{\eta}_p)$ with $\tilde{\eta}_p = (\eta_p^1, ..., \eta_p^n)$ can be obtained with a unique decomposition $(C'_1, ..., C'_n)$ where each element C'_i is reacheable from the preserving distribution η_p^i. This unique decomposition is noted $split_{\tilde{\eta}_p}(C')$ and is a measure-preserving homomorphism between $merge(\tilde{\eta}_p)$ and $join(\tilde{\eta}_p)$.

Lemma 5 (Preserving-merging)**.** *Let $n \in \mathbb{N}$, let $\{C_k\}_{k\in[1:n]}$ be a set of compatible configurations. Let $\tilde{\eta}_p = (\eta_p^1, ..., \eta_p^n) \in Disc(Q_{conf})^n$. Assume $\forall k \in [1:n]$, if $a \in \widehat{sig}(C_k)$, then $C_k \xrightarrow{a} \eta_p^k$ and otherwise, $\eta_p^k = \delta_{C_k}$.*

Then, $\forall C'_0 \in supp(merge(\tilde{\eta}_p))$, there exists a unique $(C'_1, ..., C'_n)$, noted $split_{\tilde{\eta}}(C'_0)$, s.t.

(a) $C'_0 = \bigcup\limits_{k\in[1:n]} C'_k$ and (b) $\forall k \in [1,n], C'_k \in supp(\eta_p^k)$.

$$\text{We note } split_{\tilde{\eta}} : \begin{cases} supp(merge(\tilde{\eta}_p)) & \to & supp(\eta_p^1) \times ... \times supp(\eta_p^n) \\ C'_0 & \mapsto & split_{\tilde{\eta}_p}(C'_0) \end{cases}$$

Moreover, $merge(\tilde{\eta}_p) \overset{s}{\leftrightarrow} join(\tilde{\eta}_p)$ with $s = split_{\tilde{\eta}_p}$

Proof. (Uniqueness) Let us imagine two candidates $(C'_1, ..., C'_n)$ and $(C''_1, ..., C''_n)$ verifying both (a) and (b). Let $k, \ell \in [1:n], k \neq \ell$. First, by compatibility of C_0, $\varphi_k \cap \varphi_\ell = \emptyset$. Hence $auts(C'_k) \cap auts(C''_\ell) = auts(C_k) \cap auts(C_\ell) = \emptyset$. Since $auts(\bigcup\limits_{k\in[1:n]} C'_k) = auts(\bigcup\limits_{k\in[1:n]} C'_k)$, $\forall k \in [1:n]$, $auts(C'_k) = auts(C''_k)$. By equality, $\forall k \in [1:n]$, $map(C'_k) = map(C''_k)$ and so $\forall k \in [1:n]$, $C'_k = C''_k$. (Existence) By construction of *merge*. By uniqueness and existence properties, $s = split_{\tilde{\eta}_p}$ is then a bijection from $supp(merge(\tilde{\eta}_p))$ and $supp(\eta_p^1) \times ... \times supp(\eta_p^n)$. Let $C'_0 \in supp(merge(\tilde{\eta}_p))$. By definition $merge(\tilde{\eta}_p)(C'_0) = \sum\limits_{(C'_1,...,C'_n)\in Q^n_{conf}} join(\tilde{\eta}_p)((C'_1, ..., C'_n)) \cdot \mathbb{1}_{(C'_1\cup...\cup C'_n)=C'_0}$. By bijectivity, $merge(\tilde{\eta}_p)(C'_0) = join(\tilde{\eta}_p)(split_{\tilde{\eta}_p}(C'_0))$. □

Here, we define *deter-dest*(C, a) that represents the set of automata that will be deterministically destroyed from configuration C if the action a is triggered.

Definition 80 (deter-dest)**.** *Let $C = (\mathbf{A}, \mathbf{S})$ be a configuration. For every $\mathcal{A} \in \mathbf{A}$, we note $q = \mathbf{S}(\mathcal{A})$. Let $\varphi \in \mathcal{P}(Autids)$. We define deter-dest$(C, a) = \{\mathcal{A} \in \mathbf{A} | \eta_{\mathcal{A},q_\mathcal{A},a} = \delta_{q^\phi_\mathcal{A}}\}$ if $a \in \widehat{sig}(\mathcal{A})(q)$ and $\emptyset$ otherwise.*

This last definition allows a generalization of intrisinc transitions where the triggering of an action a shared by two compatible configurations C_1 and C_2 can deterministically destroys automaton $\mathcal{A}$ from C_1 and creates $\mathcal{A}$ in C_2 at the same time. Such a mechanism is useful to describe the motion of an agent $\mathcal{A}$ from one place to another one in one atomic step. The results of this subsection tolerates this extension, but it can be ignored in a first reading.

The next lemma gives the main results used in the proof of theorem 2 of closure of PCA under composition. It proves the soundness of several intuitive merging operations.

Lemma 6 (Merging)**.** *Let $n \in \mathbb{N}$, Let $(\varphi_1, ..., \varphi_n) \in \mathcal{P}(Autids)^n$ with $\forall k, \ell \in [1:n]$, $\varphi_k \cap \varphi_\ell = \emptyset$. Let $\{C_k\}_{k\in[1:n]}$ be a set of compatible configurations. Let $\tilde{\eta} = (\eta_1, ..., \eta_n) \in Disc(Q_{conf})^n$. Assume $\forall k \in [1:n]$, if $a \in \widehat{sig}(C_k)$, then $C_k \overset{a}{\Longrightarrow}_{\varphi_k} \eta^k$ and otherwise, $\eta^k = \delta_{C_k}$ and $\varphi_k = \emptyset$. We note $\varphi_0 = \bigcup_{k\in[1:n]} \varphi_k$ and $C_0 = \bigcup_{k\in[1:n]} C_k$.*

1. *Assume, $\forall k, \ell \in [1:n], k \neq \ell, \varphi_k \cap auts(C_\ell) \subseteq deter\text{-}dest(C_\ell, a)$.*
   a) *$\forall C'_0 \in supp(merge(\tilde{\eta}))$, there exists a unique $(C'_1, ..., C'_n)$, noted $split_{\tilde{\eta}}(C'_0)$, s.t. (a) $C'_0 = \bigcup_{k\in[1:n]} C'_k$ and (b) $\forall k \in [1,n], C'_k \in supp(\eta_k)$.*

   We note $split_{\tilde{\eta}} : \left\{ \begin{array}{lll} supp(merge(\tilde{\eta})) & \rightarrow & supp(\eta_1) \times ... \times supp(\eta_n) \\ C'_0 & \mapsto & split_{\tilde{\eta}}(C'_0) \end{array} \right.$

   b) *$merge(\tilde{\eta}) \overset{s}{\leftrightarrow} join(\tilde{\eta})$ with $s = split_{\tilde{\eta}}$*

   c) *$merge(\tilde{\eta}) = reduce(merge(\tilde{\eta}_p) \uparrow \varphi_0)$.*

   d) *$C_0 \overset{a}{\Longrightarrow}_{\varphi_0} merge(\tilde{\eta})$ if $a \in \widehat{sig}(C_0)$ and $merge(\tilde{\eta}) = \delta_{C_0}$ otherwise.*
2. *Assume $\forall C'_0 \in supp(merge(\tilde{\eta}))$, C'_0 is compatible. Then, $\forall k, \ell \in [1:n], k \neq \ell, \varphi_k \cap auts(C_\ell) \subseteq deter\text{-}dest(C_\ell, a)$.*

Proof. 1.

a) Indeed, let us imagine two candidates $(C'_1, ..., C'_n)$ and $(C''_1, ..., C''_n)$ verifying both (a) and (b). Let $k, \ell \in [1:n], k \neq \ell$. By contradiction, let $\mathcal{A} \in auts(C'_k) \cap auts(C''_\ell)$. By compatibility, $\mathcal{A} \notin auts(C_k) \cap auts(C_\ell)$. W.l.o.g., $\mathcal{A} \in \varphi_k \cap auts(C_\ell)$. By assumption $\mathcal{A} \in deter\text{-}dest(C_\ell, a)$ and so $mathcal A \notin auts(C''_\ell)$ which leads to a contradiction. Hence, $\forall k \in [1:n]$, $auts(C'_k) = auts(C''_k)$. Since $auts(\bigcup_{k\in[1:n]} C'_k) = auts(\bigcup_{k\in[1:n]} C'_k)$, $\forall k \in [1:n]$, $auts(C'_k) = auts(C''_k)$. By equality, $\forall k \in [1:n]$, $map(C'_k) = map(C''_k)$ and so $\forall k \in [1:n]$, $C'_k = C''_k$. The existence comes from the construction of $join$.

b) The fact that $s = split_{\tilde{\eta}}$ is a bijection from $supp(merge(\tilde{\eta}))$ and $supp(\eta_1) \times ... \times supp(\eta_1)$ comes from the existence and the uniqueness of pre-image proved in item 1a. Let $C'_0 \in supp(merge(\tilde{\eta}))$.
By definition $merge(\tilde{\eta})(C'_0) = \sum_{(C'_1,...,C'_n)\in Q^n_{conf}} join(\tilde{\eta})((C'_1, ..., C'_n)) \cdot \mathbb{1}_{(C'_1\cup...\cup C'_n)=C'_0}$. By bijectivity, $merge(\tilde{\eta})(C'_0) = join(\tilde{\eta})(split_{\tilde{\eta}}(C'_0))$.

c) We want to show that $merge(\tilde{\eta}) \triangleq merge((reduce(\eta^1_p \uparrow \varphi_1), ..., (reduce(\eta^n_p \uparrow \varphi_n)) = reduce(merge(\tilde{\eta}_p) \uparrow \bigcup_{k\in[1:n]} \varphi_k) \triangleq reduce(merge(\tilde{\eta}_p) \uparrow \varphi_0)$. Intuitively, it comes from 1b that gives $merge(\tilde{\eta}) \overset{s}{\leftrightarrow} join(\tilde{\eta})$ with $s = split_{\tilde{\eta}}$ and $\forall k \in [1:n]$, $\eta^k = reduce(\eta^k_p \uparrow \varphi_k)$, with $\forall k, \ell \in [1:n], k \neq \ell$, $\varphi_k \cap \varphi_\ell = \emptyset$. Let us elaborate.
Let $C'_0 \in supp(merge(\tilde{\eta}))$. $merge(\tilde{\eta})(C'_0) = join(\tilde{\eta})(split_{\tilde{\eta}}(C'_0))$ by 1b.
Hence, $merge(\tilde{\eta})(C'_0) = \Pi_{k\in[1:n]}(reduce(\eta^k_p \uparrow \varphi_k)(C'_k)$ with $split_{\tilde{\eta}}(C'_0) = (C'_1, ..., C'_n)$. Thus, for every $k \in [1,n]$, $C'_k = (\mathbf{A}'_k, \mathbf{S}'_k)$ with (i) $\mathbf{A}'_k = \mathbf{A}''_k \cup \varphi_k$, (ii) $\forall \mathcal{A} \in \varphi_k, \mathbf{S}'_k(\mathcal{A}) = \bar{q}_\mathcal{A}$ (iii) $\forall \mathcal{A} \in \mathbf{A}'_k, \mathbf{S}'_k(\mathcal{A}) \neq q^\phi_\mathcal{A}$ (*). This leads to $merge(\tilde{\eta})(C'_0) = \Pi_{k\in[1:n]}(reduce(\eta^k_p))(C''_k)$ with $C''_k = (\mathbf{A}''_k, \mathbf{S}''_k)$ where $\mathbf{S}''_k = \mathbf{S}'_k \upharpoonright \mathbf{A}''_k$.
Hence, $merge(\tilde{\eta})(C'_0) = \Pi_{k\in[1:n]}(\sum_{C''_{k,\ell}, reduce(C''_{k,\ell})=C''_k} \eta^k_p(C''_{k,\ell}))$ where every $C''_{k,\ell} = (\mathbf{A}''_{k,\ell}, \mathbf{S}''_{k,\ell}) \in supp(\eta^k_p)$ with $reduce(C''_{k,\ell}) = C''_k$ verifies $\mathbf{A}''_{k,\ell} = \mathbf{A}_k$ and $\mathbf{S}''_{k,\ell} \upharpoonright \mathbf{A}''_k = \mathbf{S}''_k$ (**).

Second, for every $k \in [1:n]$, we note $\mathbf{A}_k^d = \textit{deter-dest}(C_k, a)$, $\eta_{p,d}^k$ the unique preserving distribution such that $\eta_p^k \overset{dest^k}{\leftrightarrow} \eta_{p,d}^k$ with $dest^k : (\mathbf{A}'_k, \mathbf{S}'_k) \mapsto (\mathbf{A}'_k \setminus \mathbf{A}_k^d, \mathbf{S}'_k \upharpoonright (\mathbf{A}'_k \setminus \mathbf{A}_k^d))$ and we note $\eta_{p,d,\uparrow}^k = \eta_{p,d}^k \uparrow \varphi_k$. We note $\tilde{\eta}_{p,d,\uparrow} = (\eta_{p,d,\uparrow}^1, ..., \eta_{p,d,\uparrow}^n)$. Clearly, $(\textit{reduce}(\textit{merge}(\tilde{\eta}_p) \uparrow \varphi_0)) = (\textit{reduce}(\textit{merge}(\tilde{\eta}_{p,d,\uparrow}))$.
$(\textit{reduce}(\textit{merge}(\tilde{\eta}_{p,d,\uparrow}))(C'_0) = \sum_{C'_{0,d,\ell}, \textit{reduce}(C'_{0,d,\ell})=C'_0} (\textit{merge}(\tilde{\eta}_{p,d,\uparrow}))(C'_{0,d,\ell})$, where every $C'_{0,d,\ell} = (\mathbf{A}'_{0,d,\ell}, \mathbf{S}'_{0,d,\ell}) \in \textit{supp}((\textit{merge}(\tilde{\eta}_{p,d,\uparrow}))$ with $\textit{reduce}(C'_{0,d,\ell}) = C'_0$ verifies $\mathbf{A}'_{0,\ell} = \mathbf{A}_0 \setminus \bigcup_{k[1:n]} \mathbf{A}_k^d$ and $\mathbf{S}'_{0,d,\ell} \upharpoonright \mathbf{A}'_0 = \mathbf{S}'_0$.
By lemma 5, for each ℓ, $(\textit{merge}(\tilde{\eta}_{p,d,\uparrow}))(C'_{0,d,\ell}) = \textit{split}_{\tilde{\eta}_{p,d,\uparrow}}(C'_{0,d,\ell}) = \Pi_{k\in[1:n]}\eta_{p,d,\uparrow}^k(C''_{k,d,\ell})$, with $\textit{split}_{\tilde{\eta}_{p,d,\uparrow}}(C'_{0,d,\ell}) \triangleq (C'_{1,d,\ell}, ..., C'_{n,d,\ell})$.
Moreover, every $C'_{k,d,\ell} \triangleq (\mathbf{A}'_{k,d,\ell}, \mathbf{S}'_{k,d,\ell}) \in \textit{supp}(\eta_{p,d}^k \uparrow \varphi_k))$ with $\textit{reduce}(C'_{k,d,\ell}) = C'_{k,d}$, $\mathbf{A}'_{k,d,\ell} = (\mathbf{A}_k \setminus \mathbf{A}_k^d) \cup \varphi_k$, $\mathbf{S}'_{k,d,\ell} \upharpoonright \mathbf{A}'_k = \mathbf{S}'_k$. We obtain $(\textit{reduce}(\textit{merge}(\tilde{\eta}_{p,d,\uparrow}))(C'_0) = \sum_{C'_{0,d,\ell}, \textit{reduce}(C'_{0,d,\ell})=C'_0} (\textit{join}(\tilde{\eta}_{p,d,\uparrow})(\textit{split}_{\tilde{\eta}_{p,d,\uparrow}}(C'_{0,d,\ell})))$ and so
$(\textit{reduce}(\textit{merge}(\tilde{\eta}_{p,d,\uparrow}))(C'_0) = \sum_{C'_{0,d,\ell}, \textit{reduce}(C'_{0,d,\ell})=C'_0} (\Pi_{k\in[1:n]}(\eta_{p,d,\uparrow}^k)(C'_{k,d,\ell}))$ (***).
Clearly, for every $k \in [1:n]$, $(\eta_p^k \uparrow \varphi_k) \overset{dest^k}{\leftrightarrow} \eta_{p,d,\uparrow}^k$.
Combined with (**) and (***), we find $\textit{merge}(\tilde{\eta})(C'_0) = (\textit{reduce}(\textit{merge}(\tilde{\eta}_p) \uparrow \varphi))(C'_0)$ for every $C'_0 \in \textit{supp}(\textit{merge}(\tilde{\eta}))$, which ends the proof.

d) If $a \notin \widehat{sig}(C_0)$, the result is trivial. Assume $a \in \widehat{sig}(C_0)$ Let $\tilde{\eta}_p = (\eta_p^1, ..., \eta_p^n) \in \textit{Disc}(Q_{conf})^n$ s.t. $\forall k \in [1:n]$, $C_k \xrightarrow{a} \eta_p^k$ if $a \in \widehat{sig}(C_k)$ and $\eta_p^k = \delta_{C_k}$ otherwise. For every $k \in [1:n]$, $\eta^k = \textit{reduce}(\eta_p^1 \uparrow \varphi_k)$. By compatibility of C_0, for every $k, \ell \in [1,n], k \neq \ell$, $\mathbf{A}_k^p \cap \mathbf{A}_\ell^p = \emptyset$. Hence, we can apply lemma 4 and we have $C_0 \xrightarrow{a} \textit{merge}(\tilde{\eta}_p)$. Thus, $C_0 \overset{a}{\Longrightarrow}_{\varphi_0} \textit{reduce}(\textit{merge}(\tilde{\eta}_p) \uparrow \varphi_0)$. Finally, $\textit{merge}(\tilde{\eta}) = \textit{reduce}(\textit{merge}(\tilde{\eta}_p) \uparrow \varphi_0)$ by 1c.

2. By contradiction. W.l.o.g., let us assume $\mathcal{A} \in \varphi_k \cap \textit{auts}(C_\ell) \setminus \textit{deter-dest}(C_\ell, a)$. Since C is compatible, $\mathcal{A} \notin \mathbf{A}_k \cap \mathbf{A}_\ell$. By definition of *deter-dest* it exists $(C'_k, C'_\ell) \in \textit{supp}(\eta_k) \times \textit{supp}(\eta_\ell)$, $\mathcal{A} \in \textit{auts}(C'_k) \cap \textit{auts}(C'_\ell)$ and $C'_k \cup C'_\ell$ is not compatible. So it exists $(C'_1, ..., C'_n) \in \textit{supp}(\eta_1 \otimes ... \otimes \eta_n)$ s.t. $(C'_1 \cup ... \cup C'_n)$ is not compatible.

□

trivial results about homomorphisms between probability measures Before tackling the theorem 2 of closure of PCA under composition, we gives a few trivial results about measure-preserving bijections between two measures.

First, the relationship of measure-preserving bijection between two measures is shown to be transitive and symmetric.

Lemma 7. *Let $(\eta_1, \eta_2, \eta_3) \in \textit{Disc}(Q_1) \times \textit{Disc}(Q_2) \times \textit{Disc}(Q_3)$, with Q_i being a set for each $i \in \{1,2,3\}$. Let $f : Q_1 \rightharpoonup Q_2$ and $g : Q_1 \rightharpoonup Q_2$ defined on $\textit{supp}(\eta_1)$ and $\textit{supp}(\eta_2)$ respectively. Let $\tilde{f}$ (resp. $\tilde{g}$) denotes the restriction of f (resp. g) on $\textit{supp}(\eta_1)$ (resp. $\textit{supp}(\eta_2)$).*

If $\eta_1 \overset{f}{\leftrightarrow} \eta_2$ and $\eta_2 \overset{g}{\leftrightarrow} \eta_3$, then

1. *$\eta_1 \overset{h}{\leftrightarrow} \eta_3$ where the restriction $\tilde{h}$ of h on $\textit{supp}(\eta_1)$ verifies $\tilde{h} = \tilde{g} \circ \tilde{f}$ and*
2. *$\eta_2 \overset{k}{\leftrightarrow} \eta_1$ where the restriction $\tilde{k}$ of k to $\textit{supp}(\eta_2)$ verifies $\tilde{k} = \tilde{f}^{-1}$.*

Proof.

- (bijectivity) The composition of two bijections is a bijection and the reverse function of a bijection is a bijection.
- (measure preservation) In the first case, $\forall q \in supp(\eta_1), \eta_1(q) = \eta_2(f(q))$ with $f(q) \in supp(\eta_2)$ which means $\eta_2(f(q)) = \eta_3(g(f(q)))$. In the second case $\forall q' \in supp(\eta_2), \exists! q \in supp(\eta_1), \eta_1(q) = \eta_2(q' = \tilde{f}(q))$ and hence $\forall q' \in supp(\eta_2), \eta_2(q') = \eta_1(q = \tilde{f}^{-1}(q'))$.

□

Second, the relationship of measure-preserving bijection between two measures is shown to be preserved by join operation.

Lemma 8 (correspondence preservation for joint probability)**.** *Let $\tilde{\eta} = (\eta_1, ..., \eta_n) \in Disc(Q_1) \times ... \times Disc(Q_n)$, $\tilde{\eta}' = (\eta'_1, ..., \eta'_n) \in Disc(Q'_1) \times ... \times Disc(Q'_n)$ with each Q_i (resp. Q'_i) being a set. For each $i \in [1:n]$, let $f_i : Q_i \rightharpoonup Q'_i$, where $dom(f_i) \subseteq supp(\eta_i)$, with $\eta_i \overset{f_i}{\leftrightarrow} \eta'_i$.*

Then $join(\tilde{\eta}) \overset{f}{\leftrightarrow} join(\tilde{\eta}')$ with $f : \left\{ \begin{array}{ccc} Q_1 \times ... \times Q_n & \rightharpoonup & range(f_1) \times ... \times range(f_n) \\ (x_1, ..., x_n) & \mapsto & (f_1(x_1), ..., f_n(x_n)) \end{array} \right.$.

Proof. The restriction $\tilde{f}$ of f on $supp(join(\tilde{\eta})) = supp(\eta_1) \times ... \times supp(\eta_n)$ is still a bijection and $\forall x = (x_1, ..., x_n) \in dom(f_1) \times ... \times dom(f_n)$, $join(\tilde{\eta})(x) = \eta_1(x_1) \cdot ... \cdot \eta_n(x_n) = \eta'_1(f_1(x_1)) \cdot ... \cdot \eta'_n(f_n(x_n)) = join(\tilde{\eta}')(f(x_1, ..., x_n))$. □

PCA closure under composition Now we are ready to prove the theorem that claims that a composition of PCA is a PCA.

Theorem 2 (PCA closure under composition)**.** *Let $X_1, ..., X_n$, be partially-compatible PCA. Then $X = X_1||...||X_n$ is a PCA.*

Proof. We need to show that X verifies all the constraints of definition 68.

- (Constraint) 1: The demonstration is the same as the one in [AL16], section 5.1, proposition 21, p 32-33. Let $\bar{q}_X$ and $(\mathbf{A}, \mathbf{S}) = config(X)(\bar{q}_X)$. By the composition of psioa, then $\bar{q}_X = (\bar{q}_{X_1}, ..., \bar{q}_{X_n})$. By definition, $config(X)(\bar{q}_X) = config(X_1)(\bar{q}_{X_1}) \cup ... \cup config(X_n)(\bar{q}_{X_n})$. Since for every $j \in [1:n]$, X_j is a configuration automaton, we apply constraint 1 to X_j to conclude $\mathbf{S}(\mathcal{A}_\ell) = \bar{q}_{\mathcal{A}_\ell}$ for every $\mathcal{A}_\ell \in auts(config(X_j)(\bar{q}_{X_j})$.
  Since $(auts(config(X_1)(\bar{q}_{X_1}), ..., auts(config(X_n)(\bar{q}_{X_n}))$ is a partition of $\mathbf{A}$ by definition of composition, $\mathbf{S}(\mathcal{A}_\ell) = \bar{q}_{\mathcal{A}_\ell}$ for every $\mathcal{A}_\ell \in \mathbf{A}$ which ensures X verifies constraint 1.
- (Constraint 2)
  Let $(q, a, \eta_{(X,q,a)}) \in D_X$. We will establish $\exists \eta' \in Disc(Q_{conf})$ s.t. $\eta_{(X,q,a)} \overset{c}{\leftrightarrow} \eta'$ where $c = config(X)$ and $config(X)(q) \overset{a}{\Longrightarrow}_\varphi \eta'$ with $\varphi = created(X)(q)(a)$.
  For brevity, let $P_i = psioa(X_i)$ for every $i \in [1:n]$. By definition 77 of PCA compositon, $psioa(X) = psioa(X_1)||...||psioa(X_n) = P_1||...||P_n$. By definition 196 of PSIOA composition, $q = (q_1, ..., q_n) \in Q_{P_1} \times ... \times Q_{P_n}$, while $a \in \bigcup_{i \in [1:n]} \widehat{sig}(P_i)(q_i)$ and $\eta_{X,q,a} = \eta_{P_1,q_1,a} \otimes ... \otimes \eta_{P_n,q_n,a}$ with the convention $\eta_{P_i,q_i,a} = \delta_{q_i}$ if $a \notin \widehat{sig}(P_i)(q_i)$.
  Let $(\mathcal{I}, \mathcal{J})$ be a partition of $[1:n]$ s.t. $\forall i \in \mathcal{I}$, $a \in \widehat{sig}(P_i)(q_i)$ and $\forall j \in \mathcal{J}$, $a \notin \widehat{sig}(P_j)(q_j)$. Then by PCA top/down transition preservation, it exists $\eta'_i \in Disc(Q_{conf})$ s. t. $\eta_{X_i,q_i,a} = \eta_{P_i,q_i,a} \overset{c_i}{\leftrightarrow} \eta'_i$

with $c_i = config(X_i)$ and $config(X_i)(q_i) \stackrel{a}{\Longrightarrow}_{\varphi_i} \eta'_i$ with $\varphi_i = created(X_i)(q_i)(a)$. For every $j \in \mathcal{J}$, we note $\varphi_j = \emptyset$ and $\eta'_j = \delta_{config(X_j)(q_j)}$ that verifies $\delta_{q_j} \stackrel{c_j}{\leftrightarrow} \eta'_j$ with $c_j = config(X_j)$.
We note $\tilde{\eta}' = (\eta'_1, ..., \eta'_n)$ and $\varphi = \bigcup_{i \in [1:n]} \varphi_i$. By definition 77 of PCA composition, $\varphi = created(X)(q)(a)$.
We have $\eta_{X,q,a} \stackrel{c'}{\leftrightarrow} \eta'$ with $c' : q = (q_1, ..., q_n) \mapsto (c_1(q_1), ..., c_n(q_n))$ by lemma 8.
Moreover $merge(\tilde{\eta}') \stackrel{s}{\leftrightarrow} join(\tilde{\eta}')$ with $s = split_{\tilde{\eta}}$ by lemma 6, item 1b.
So $\eta_{X,q,a} \stackrel{c}{\leftrightarrow} merge(\tilde{\eta}')$ with $c = s^{-1} \circ c' = config(X)$.
Moreover we have $config(X)(q) \stackrel{a}{\Longrightarrow}_{\varphi} merge(\tilde{\eta}')$ by lemma 6, item 1d.

– (Constraint 3)
Let $q \in Q_X$, $C = config(X)(q)$, $a \in \widehat{sig}(X)(q)$, $\varphi = created(X)(q)(a)$ that verify $C \stackrel{a}{\Longrightarrow}_{\varphi} \eta'$.
We need to show that it exists $(q, a, \eta_{(X,q,a)}) \in D_X$ s.t. $\eta_{(X,q,a)} \stackrel{c}{\leftrightarrow} \eta'$ with $c = config(X)$.
For brevity, let $P_i = psioa(X_i)$ for every $i \in [1 : n]$. By definition 77 of PCA composition $psioa(X) = psioa(X_1)||...||psioa(X_n) = P_1||...||P_n$. By definition 196 of PSIOA composition, $q = (q_1, ..., q_n) \in Q_{P_1} \times ... \times Q_{P_n}$, while $a \in \bigcup_{i \in [1:n]} \widehat{sig}(P_i)(q_i)$.
Let $(\mathcal{I}, \mathcal{J})$ be a partition $[1 : n]$ s.t. $\forall i \in \mathcal{I}$, $a \in \widehat{sig}(P_i)(q_i)$ and $\forall j \in \mathcal{J}$, $a \notin \widehat{sig}(P_j)(q_j)$. For every $i \in \mathcal{I}$, we note $\varphi_i = created(X_i)(q_i)(a)$,while for every $j \in \mathcal{J}$, we note $\varphi_j = \emptyset$ and $\eta'_j = \delta_{config(X_j)(q_j)}$ that verifies $\delta_{q_j} \stackrel{c_j}{\leftrightarrow} \eta'_j$ with $c_j = config(X_j)$.
We note $\varphi = created(X)(q)(a)$. By pca-composition definition, $\varphi = \bigcup_{k \in [1:n]} \varphi_k$. For every $k \in [1 : n]$, we note $C_k = config(X_k)(q_k)$ and for every $i \in \mathcal{I}$, $\eta'_i \in Disc(Q_{conf})$ s.t. $C_i \stackrel{a}{\Longrightarrow}_{\varphi_i} \eta'_i$. We note $\tilde{\eta}' = (\eta'_1, ..., \eta'_n)$
By constraint 3 (bottom/up transition preservation), $\forall i \in \mathcal{I}$, $\exists (q_i, a, \eta_{X_i,q_i,a}) \in D_{X_i}$ s.t. $\eta_{X_i,q_i,a} \stackrel{c_i}{\leftrightarrow} \eta'_i$ with $c_i = config(X_i)$. by lemma 8, $\eta_{X,q,a} = \eta_{X_1,q_1,a} \otimes ... \otimes \eta_{X_n,q_n,a} \stackrel{c'}{\leftrightarrow} \eta'_1 \otimes ... \otimes \eta'_n = join(\tilde{\eta}')$ with the convention $\eta_{X_j,q_j,a} = \delta_{q_j}$ for $j \in \mathcal{J}$ and $c' : q = (q_1, ..., q_n) \in states(X) \mapsto (c_1(q_1), ..., c_n(q_n))$.
By partial-compatibility, for every $C' \in supp(merge(\tilde{\eta}'))$, C' is compatible. Hence we can apply lemma 6, item 1b, which gives $merge(\tilde{\eta}') \stackrel{s}{\leftrightarrow} join(\tilde{\eta}')$ with $s = split_{\tilde{\eta}'}$. Hence $\eta_{X,q,a} \stackrel{c''}{\leftrightarrow} merge(\tilde{\eta}')$ with $c'' = s^{-1} \circ c'$, that is $\eta_{X,q,a} \stackrel{c}{\leftrightarrow} \eta'$ with $c = config(X)$ and the restriction of c'' on $supp(\eta_{X,q,a})$ is c. We can apply lemma 6 again, but for item 1d, which gives $C \stackrel{a}{\Longrightarrow}_{\varphi} merge(\tilde{\eta}')$.

– (Constraint 4).
Let $q = (q_1, ..., q_n) \in Q_X$. For every $i \in [1, n]$, we note $h_i = hidden\text{-}actions(X_i)(q_i)$, $C_i = config(X_i)(q_i)$, $h = \bigcup_{i \in [1,n]} h_i$ and $C = config(X)(q)$. Since $X_1, ..., X_n$ are compatible at state q, we have both $\{C_i | i \in [1, n]\}$ compatible and $\forall i, j \in [1, n], in(C_i) \cap h_j = \emptyset$. By compatibility, $\forall i, j \in [1, n], i \neq j, out(C_i) \cap out(C_j) = int(C_i) \cap \widehat{sig}(C_j) = \emptyset$, which finally gives $\forall i, j \in [1, n], i \neq j$, $\widehat{sig}(C_i) \cap h_j = \emptyset$.
Hence, we can apply lemma 1 of commutativity between hiding and composition to obtain $hide(sig(C_1) \times .... \times sig(C_n), h_1 \cup ... \cup h_n) = hide(sig(C_1), h_1) \times ... \times hide(sig(C_n), h_n)$ where $\times$ has to be understood in the sense of definition 59 of signature composition.
That is $sig(psioa(X))(q) = sig(psioa(X_1))(q_1)) \times .... \times sig(psioa(X_n))(q_n))$, as per definition 59, with $sig(psioa(X))(q) = hide(sig(config(X)(x)), h)$. Furthermore $h \subseteq out(config(X)(q))$, since $\forall i \in [1, n]$, $h_i \subseteq out(C_i)$. This terminates the proof.

□

3.6. Scheduler, measure on executions, implementation

In previous sections we have describe a semantics for the behaviours of interacting dynamic probabilistic systems. Namely, we have shown theorem 2 that the composition of two PCA is a PCA and we gave some tools to handle executions of a PCA. However, we are still unable to express the probability of the occurrence of an event for a particular PCA. As a matter of fact, an inherent pure non-determinism appears in concurrent systems. Indeed, after composition (or even before), it is natural to obtain a state with several enabled actions. The most common case is the reception of two concurrent messages in flight from two different processes. This pure non-determinism is desirable to capture our lack of knowledge about the real worl, but it must be solved if we want to define a probability measure on the automata executions and be able to say whether a situation is likely to occur or not. To solve the pure non-determinism, we invoke the well-known *scheduler*, an abstract entity that chooses the next enabled action to trigger after a certain finite execution. For example, if Alice broadcasts the same message to Bob and Carol and the two possible orders of reception are possible, then this is the role of the scheduler to specify who will first receive the message among Bob and Carol.

3.6.1. General definition and probabilistic space $(Frags(\mathcal{A}), \mathcal{F}_{Frags(\mathcal{A})}, \epsilon_{\sigma,\mu})$

A scheduler is hence a function that takes an execution fragment as input and outputs the probability distribution on the set of transitions that will be triggered. We reuse the formalism from [Seg95b] with the syntax from [CCK+18].

Definition 81 (scheduler)**.** *A* scheduler *of a PSIOA (resp. PCA) $\mathcal{A}$ is a function*

$\sigma : Frags^*(\mathcal{A}) \to SubDisc(D_{\mathcal{A}})$ *such that* $(q, a, \eta) \in supp(\sigma(\alpha))$ *implies* $q = lstate(\alpha)$*. We note* $schedulers(\mathcal{A})$ *the set of schedulers of* $\mathcal{A}$*.*

Here $SubDisc(D_{\mathcal{A}})$ is the set of discrete sub-probability distributions on $D_{\mathcal{A}}$. Loosely speaking, σ decides (probabilistically) which transition to take after each finite execution fragment α. Since this decision is a discrete sub-probability measure, it may be the case that σ chooses to halt after α with non-zero probability: $1 - \sigma(\alpha)(D_{\mathcal{A}}) > 0$. If we want a deterministic scheduler, it suffices to impose that for each finite execution α the support of $\sigma(\alpha)$ is a singleton.

Since the scheduler resolves the pure non-determinism, we are able to define a measure of probability over the set of executions of an automaton.

Definition 82 (measure $\epsilon_{\sigma,\alpha}$ generated by a scheduler and a fragment)**.** *A scheduler σ and a finite execution fragment α generate a measure $\epsilon_{\sigma,\alpha}$ on the sigma-algebra $\mathcal{F}_{Frags(\mathcal{A})}$ generated by cones of execution fragments, where each cone $C_{\alpha'}$ is the set of execution fragments that have α' as a prefix, i.e. $C_{\alpha'} = \{\alpha \in Frags(\mathcal{A}) | \alpha' \leq \alpha\}$. The measure of a cone $C_{\alpha'}$ is defined recursively as follows:*

$$\epsilon_{\sigma,\alpha}(C_{\alpha'}) =: \begin{cases} 0 & \text{if both } \alpha' \not\leq \alpha \text{ and } \alpha \not\leq \alpha' \\ 1 & \text{if } \alpha' \leq \alpha \\ \epsilon_{\sigma,\alpha}(C_{\alpha''}) \cdot \sigma(\alpha'')(\eta_{(\mathcal{A},q',a)}) \cdot \eta_{(\mathcal{A},q',a)}(q) & \text{if } \alpha \leq \alpha'' \text{ and } \alpha' = \alpha''^\frown q' a q \end{cases}$$

Intuitvely, we have fixed α as a known event, i.e. we assume that α occurs and we want to measure some cones of execution. The first case says that a cone with α' as prefix, where α' is not comparable with α, has a zero measure. It comes as no surprise since an execution in this cone would be incompatible with α. The second case says that a cone with α' as prefix, where α' is a prefix of α, has a measure of 1. Here again, it comes as no surprise since assuming α occurs implies that α' occurs. The third case describes a simple conditional probability for $\alpha' = \alpha''^\frown q' a q$: the probability of having an execution in

cone $C_{\alpha'}$ is the probability of the conjunction of (1) having an execution in cone $C_{\alpha''}$, (2) the scheduler triggers action a after α'' and (3) we reache the state q from distribution $\eta_{(\mathcal{A},q',a)}$.

For the time being, we only gave a measure over cones of executions, but clearly, the set of cones of executions is not a sigma-algebra.

Extend $\epsilon_{\sigma,\alpha}$ to $\mathcal{F}_{Frags(\mathcal{A})}$ Standard measure theoretic arguments [Seg95b] ensure that $\epsilon_{\sigma,\alpha}$ is well-defined. The proof of [Seg95b] (terminating with theorem 4.2.10, section 4.2) is very general and might appear discouraging for a brief reading. For sake of completeness, we adapt the construction of [Seg95b] to the formalism of [CCK+18] [3].

First, for every set $\mathscr{C}$ of subset of a set Ω, we define $F_1(\mathscr{C})$, $F_2(\mathscr{C})$, $F_3(\mathscr{C}) = \mathit{field}(\mathscr{C})$ as in the construction of proposition 1 and $\mathcal{F}_\Omega \triangleq \mathit{sigma}(F_3(\mathscr{C})) = \mathit{sigma}(\mathscr{C})$.

If μ is a measure on $F_3(\mathcal{C}) = \mathit{field}(\mathscr{C})$, by famous Carathéodory's extension theorem [Dud04], there exists a unique extension μ' of μ to the sigma-algebra $\mathcal{F}_\Omega$ (defining $\mu'(\biguplus_{k\in\mathbb{N}} E_k) \triangleq \sum_{k\in\mathbb{N}} \mu(E_k)$).

Hence, we just need to make explicit the extension of $\epsilon_{\sigma,\alpha}$ to $F_3(\mathcal{C}) = \mathit{field}(\mathscr{C})$ with $\mathscr{C} = \{C_{\alpha'} | \alpha' \in \mathit{Frags}(\mathcal{A})\}$.

Let $\mathscr{C} = \{C_{\alpha'} | \alpha' \in \mathit{Frags}(\mathcal{A})\}$ be the set of cones. Clearly, $\mathscr{C}$ is a set of subsets of $\mathit{Frags}(\mathcal{A})$. As mentioned earlier, we define $\mathcal{F}_{\mathit{Frags}(\mathcal{A})}$ as the sigma-algebra on $\mathit{Frags}(\mathcal{A})$ generated by $\mathscr{C}$.

Also, for every pair of execution fragments α_1 and α_2, if α_1 and α_2 are non-comparable, then $C_{\alpha_1} \cup C_{\alpha_2}$ is not a cone, while if α_1 and α_2 are comparable, C_{α_1} and C_{α_2} are not disjoint. Hence, sigma-additivity is trivially ensured by $\epsilon_{\sigma,\alpha}$ on $\mathscr{C}$. Now, let us generate the appropriate sigma-algebra $\mathcal{F}_{\mathit{Frags}(\mathcal{A})}$ on $\mathit{Frags}(\mathcal{A})$ and let us extend $\epsilon_{\sigma,\alpha}$ to $\mathcal{F}_{\mathit{Frags}(\mathcal{A})}$.

- Let $F_1(\mathscr{C})$ be the be the family containing $\emptyset$, $\mathit{Frags}(\mathcal{A})$, and all $C \subseteq \mathit{Frags}(\mathcal{A})$ such that either $C \in \mathscr{C}$ or $\mathit{Frags}(\mathcal{A}) \setminus C \in \mathscr{C}$.

  There exists a unique extension $\epsilon^i_{\sigma,\alpha}$ of $\epsilon_{\sigma,\alpha}$ to $F_1(\mathscr{C})$. Indeed, there is a unique way to extend the measure of the cones to their complements since for each α', $\epsilon^i_{\sigma,\alpha}(C_{\alpha'}) + \epsilon^i_{\sigma,\alpha}(\mathit{Frags}(\mathcal{A}) \setminus C_{\alpha'}) = 1$. Therefore $\epsilon^i_{\sigma,\alpha}$ coincides with $\epsilon_{\sigma,\alpha}$ on the cones and $\epsilon^i_{\sigma,\alpha}$ is defined to be $1 - \epsilon^i_{\sigma,\alpha}(C_\alpha)$ for the complement of any cone C_α . By countably branching structure of $\mathit{Frags}(\mathcal{A})$ ($Q_\mathcal{A}$ and $\mathit{acts}(\mathcal{A})$ are both countable), the complement of a cone is a countable union of cones. Indeed, let $\alpha' \in \mathit{Frags}^*(\mathcal{A}), C_{\alpha'} \in \mathscr{C}$, then $\mathit{Frags}(\mathcal{A}) \setminus C_{\alpha'} = \bigcup_{\alpha'' \in \mathit{Frags}^*(\mathcal{A}), \alpha'' \not\leq \alpha' \wedge \alpha' \not\leq \alpha''} C_{\alpha''}$. Hence, σ-additivity is preserved.
- Let $F_2(\mathscr{C})$ be the family containing all finite intersections of elements of $F_1(\mathscr{C})$. There exists a unique extension $\epsilon^{ii}_{\sigma,\alpha}$ of $\epsilon^i_{\sigma,\alpha}$ to $F_2(\mathscr{C})$. Indeed, let us fix a pair of execution fragments α_1 and α_2, if α_1 and α_2 are non-comparable, then $C_{\alpha_1} \cap C_{\alpha_2} = \emptyset$ is not a cone, while if α_1 and α_2 are comparable, let say $\alpha_1 \leq \alpha_2$, then $C_{\alpha_1} \cap C_{\alpha_2} = C_{\alpha_2}$. Thus, the intersection of finitely many sets of $F_1(\mathscr{C})$ is a countable union of cones. Therefore σ-additivity enforces a unique measure on the new sets of $F_1(\mathscr{C})$.
- Let $F_3(\mathscr{C})$ be the family containing all finite unions of disjoint elements of $F_2(\mathscr{C})$.

  There exists a unique extension $\epsilon^{iii}_{\sigma,\alpha}$ of $\epsilon^{ii}_{\sigma,\alpha}$ to $F_2(\mathscr{C})$. Indeed, there is a unique way of assigning a measure to the finite union of disjoint sets whose measure is known, i.e., adding up their measures. Since all the sets of $F_3(\mathscr{C})$ are countable unions of cones, σ-additivity is preserved.

3. We are not aware of such an adaptation in the literature. This concise presentation might have its own pedagogical interest

– Clearly, $F_3(\mathscr{C})$ is a field on $Frags(\mathcal{A})$, i.e. it is a family of subsets of $Frags(\mathcal{A})$ that contains $\emptyset, Frags(\mathcal{A})$, and that is closed under complementation and finite union. $\mathcal{F}_{Frags(\mathcal{A})}$ is defined as the smallest sigma-algebra containing $F_3(\mathscr{C})$. (This is also the smallest σ-algebra containing $\mathscr{C}$). By famous Carathéodory's extension theorem [Dud04], there exists a unique extension $\epsilon^{iv}_{\sigma,\alpha}$ of $\epsilon^{iii}_{\sigma,\alpha}$ to the sigma-algebra $\mathcal{F}_{Frags(\mathcal{A})}$ (defining $\epsilon^{iv}_{\sigma,\alpha}(\biguplus_{k\in\mathbb{N}} E_k) = \sum_{k\in\mathbb{N}} \epsilon^{iii}_{\sigma,\alpha}(E_k)$).

In the remaining, we abuse the notation and use $\epsilon_{\sigma,\alpha}$ to denote its extension $\epsilon^{iv}_{\sigma,\alpha}$ on $\mathcal{F}_{Frags(\mathcal{A})}$.

Limit We can remark that

– $\forall \alpha' \in Frags^*(\mathcal{A}), \{\alpha'\} = C_{\alpha'} \setminus (\bigcup_{\alpha''\in Frags^*(\mathcal{A}),\alpha'<\alpha''} C_{\alpha''})$.

– $\forall \alpha' \in Frags^\omega(\mathcal{A}), \{\alpha'\} = Frags(\mathcal{A}) \setminus (\bigcup_{i\in\mathbb{N}} \bigcup_{\alpha''\in Frags^*(\mathcal{A}),\alpha'|_i<\alpha'',\alpha'|_{i+1}\neq\alpha''|_{i+1}} C_{\alpha''})$.

Hence $\forall \alpha' \in Frags(\mathcal{A}), \{\alpha'\} \in \mathcal{F}_{Frags(\mathcal{A})}$.

Necessarily, we have $\forall \alpha' \in Frags^\omega(\mathcal{A}), \epsilon_{\sigma,\alpha}(\alpha') = lim_{i\to\infty}\epsilon_{\sigma,\alpha}(\alpha'|_i)$. Let us note that the limit is well-defined, since $\forall i \in \mathbb{N}$, (1) $\epsilon_{\sigma,\alpha}(\alpha'|_{i+1}) \leq \epsilon_{\sigma,\alpha}(\alpha'|_i)$ and (2) $\epsilon_{\sigma,\alpha}(\alpha'|_i) \geq 0$.

Notations We call the state $fstate(\alpha)$ the first state of $\epsilon_{\sigma,\alpha}$ and denote it by $fstate(\epsilon_{\sigma,\alpha})$. If α consists of the start state $\bar{q}_{\mathcal{A}}$ only, we call $\epsilon_{\sigma,\alpha}$ a probabilistic execution of $\mathcal{A}$. Let μ be a discrete probability measure over $Frags^*(\mathcal{A})$. We denote by $\epsilon_{\sigma,\mu}$ the measure $\sum_{\alpha\in supp(\mu)} \mu(\alpha)\cdot\epsilon_{\sigma,\alpha}$ and we say that $\epsilon_{\sigma,\mu}$ is generated by σ and μ. We call the measure $\epsilon_{\sigma,\mu}$ a generalized probabilistic execution fragment of $\mathcal{A}$. If every execution fragment in $supp(\mu)$ consists of a single state, then we call $\epsilon_{\sigma,\mu}$ a probabilistic execution fragment of $\mathcal{A}$.

The collection $F(\mathscr{C}_{Execs(\mathcal{A})})$ of sets obtained by taking the intersection of each element in $F_3(\mathscr{C})$ with $Execs(\mathcal{A})$ is a field in $Execs(\mathcal{A})$. We note $\mathcal{F}_{Execs(\mathcal{A})}$ the smallest sigma-algebra containing $F(\mathscr{C}_{Execs(\mathcal{A})})$. In the remaining part of the thesis, we will mainly focus on probabilistic executions of $\mathcal{A}$ of the form $\epsilon_\sigma \triangleq \epsilon_{\sigma,\delta_{\bar{q}_{\mathcal{A}}}} = \epsilon_{\sigma,\bar{q}_{\mathcal{A}}}$. Hence, we will deal with probablistic space of the form $(Execs(\mathcal{A}), \mathcal{F}_{Execs(\mathcal{A})}, \epsilon_\sigma)$.

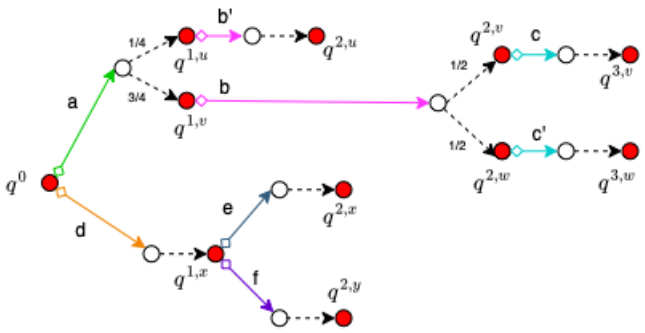


Figure 3.23. – Non-deterministic execution requires a scheduler

The scheduler allows us to solve the pure non-determinism, by triggering an action among the enabled ones. Typically after execution $\alpha = q^0\ d\ q^{1,x}$, the actions e and f are enabled and the probability to take one transition is given by the scheduler σ that computes $\sigma(\alpha)$.

Scheduler Schema Without restriction, a scheduler could become a too powerful deamon for practical applications. Hence, it is common to only consider a subset of schedulers, called a *scheduler schema*. Typically, a classic limitation is often described by a scheduler with "partial online information". Some formalism has already been proposed in [Seg95b] (section 5.6) to impose the scheduler that its choices are correlated for executions fragments in the same equivalence class where both the equivalence relation and the correlation must to be defined. This idea has been reused and simplified in [CCK+06b] that defines equivalence classes on actions, called *tasks*. Then, a task-scheduler (a.k.a. "off-line" scheduler) selects a sequence of tasks $T_1, T_2, ...$ in advance that it cannot modify during the execution of the automaton. After each transition, the next task T_i triggers an enabled action if there is no ambiguity and is ignored otherwise. One of our main contributions, the theorem of monotonicity of dynamic creation/destrution of PSIOA with implementation relationship is ensured only for a certain scheduler schema, so-called *creation-oblivious*. However, we will see that the practical set of task schedulers is not composed of creation-oblivious schedulers only.

Definition 83 (scheduler schema)**.** *A* scheduler schema *is a function that maps every PSIOA (resp. PCA) $\mathcal{A}$ to a subset of schedulers($\mathcal{A}$).*

3.6.2. Implementation

In last subsection, we have defined a measure of probability on executions with the help of a scheduler to resolve the pure non-determinism. Now we can define the notion of implementation. The intuition behind this notion is the fact that any environment $\mathcal{E}$ that would interact with both $\mathcal{A}$ and $\mathcal{B}$, would not be able to distinguish $\mathcal{A}$ from $\mathcal{B}$. The classic use-case is to formally show that a (potentially very sophisticated) algorithm implements a specification.

For us, an environment is simply a partially-compatible automaton, but in practice, he will play the role of a "distinguisher".

Definition 84 (Environment)**.** *A probabilistic environment for PSIOA $\mathcal{A}$ is a PSIOA $\mathcal{E}$ such that $\mathcal{A}$ and $\mathcal{E}$ are partially-compatible. We note env($\mathcal{A}$) the set of environments of $\mathcal{A}$.*

Now we define *insight function* which is a function that captures the insights that could be obtained by an external observer to attempt a distinction.

Definition 85 (insight function)**.** *An* insight-function *is a function $f_{(.,.)}$ parametrized by a pair $(\mathcal{E}, \mathcal{A})$ of PSIOA where $\mathcal{E} \in env(\mathcal{A})$ s.t. $f_{(\mathcal{E},\mathcal{A})}$ is a measurable function from $(Execs(\mathcal{E}||\mathcal{A}), \mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})})$ to some measurable space $(G_{(\mathcal{E},\mathcal{A})}, \mathcal{F}_{G_{(\mathcal{E},\mathcal{A})}})$.*

Some examples of insight-functions are the trace function and the environment projection function. In cryptography, the insight function can be the occurence or the absence of an explicit "accept" action triggered by the environment depending on whether he thinks he is interacting with the (real) automaton $\mathcal{A}$ or the (ideal) automaton $\mathcal{B}$.

Since an insight-function $f_{(.,.)}$ is measurable, we can define the image measure of $\epsilon_{\sigma,\mu}$ under $f_{(\mathcal{E},\mathcal{A})}$, i.e. the probability to obtain a certain external perception under a certain scheduler σ and a certain probability distribution μ on the starting executions.

Definition 86 (f-dist)**.** *Let $f_{(.,.)}$ be an insight-function. Let $(\mathcal{E}, \mathcal{A})$ be a pair of PSIOA where $\mathcal{E} \in env(\mathcal{A})$. Let μ be a probability measure on $(Execs(\mathcal{E}||\mathcal{A}), \mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})})$, and $\sigma \in schedulers(\mathcal{E}||\mathcal{A})$. We define $f\text{-}dist_{(\mathcal{E},\mathcal{A})}(\sigma, \mu)$, to be the image measure of $\epsilon_{\sigma,\mu}$ under $f_{(\mathcal{E},\mathcal{A})}$ (i.e. the function that maps any $C \in \mathcal{F}_{G_{(\mathcal{E},\mathcal{A})}}$ to $\epsilon_{\sigma,\mu}(f^{-1}_{(\mathcal{E},\mathcal{A})}(C))$) . We note $f\text{-}dist_{(\mathcal{E},\mathcal{A})}(\sigma)$ for $f\text{-}dist_{(\mathcal{E},\mathcal{A})}(\sigma, \delta_{\bar{q}_{(\mathcal{E}||\mathcal{A})}})$. We slightly abuse the notation by defining $f\text{-}dist_{(\mathcal{E},\mathcal{A})}(\sigma)(C) = 0$ if $C \notin \mathcal{F}_{G_{(\mathcal{E},\mathcal{A})}}$.*

Now we define the notion of *balanced schedulers*, which will help to capture the incapacity of an environment to distinguish two situations under two so-called balanced schedulers.

Definition 87 (balanced schedulers)**.** *Let f be an insight function. Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA (resp. PCA), let $\mathcal{E} \in env(\mathcal{A}) \cap env(\mathcal{B})$, let σ, σ' be schedulers of $\mathcal{E}||\mathcal{A}$ and $\mathcal{E}||\mathcal{B}$ respectively. Let $\varepsilon \in \mathbb{R}^{\geq 0}$. We note $\sigma B^{\leq \varepsilon}_{(\mathcal{E},\mathcal{A},\mathcal{B}),f} \sigma'$, if for every "external perception" $\zeta \in range(f_{(\mathcal{E},\mathcal{A})}) \cup range(f_{(\mathcal{E},\mathcal{B})})$, $|(f\text{-}dist_{(\mathcal{E},\mathcal{B})}(\sigma')(\zeta) - f\text{-}dist_{(\mathcal{E},\mathcal{A})}(\sigma)(\zeta))| \leq \varepsilon$.*

We can remark schedulers are "perfectly balanced", noted $\sigma B^{\leq 0}_{(\mathcal{E},\mathcal{A},\mathcal{B}),f} \sigma'$, if $\forall \zeta \in range(f_{(\mathcal{E},\mathcal{A})}) \cup range(f_{(\mathcal{E},\mathcal{B})})$, $f\text{-}dist_{(\mathcal{E},\mathcal{B})}(\sigma')(\zeta) = f\text{-}dist_{(\mathcal{E},\mathcal{A})}(\sigma)(\zeta)$.

Also, we could consider an alternative definition where $\sigma B^{\leq \varepsilon}_{(\mathcal{E},\mathcal{A},\mathcal{B}),f} \sigma'$, if for every set of "external perceptions" $\underline{\zeta} \subseteq range(f_{(\mathcal{E},\mathcal{A})}) \cup range(f_{(\mathcal{E},\mathcal{B})})$, $|(f\text{-}dist_{(\mathcal{E},\mathcal{B})}(\sigma')(\underline{\zeta}) - f\text{-}dist_{(\mathcal{E},\mathcal{A})}(\sigma)(\underline{\zeta}))| \leq \varepsilon$. Such a definition might be more desirable for monotonicity of dynamic creation/destruction of PSIOA with approximate implementation relationship.

Next lemma simply states that the relationship $B^{\leq \varepsilon}_{(\mathcal{E},.,.),f}$ between balanced schedulers is transitive.

Lemma 9 ($B^{\leq \varepsilon}_{(\mathcal{E},.,.),f}$ transitivity)**.** *Let f be an insight function. Let $\mathcal{A}$,$\mathcal{B}$,$\mathcal{C}$ be three PSIOA (resp. PCA), let $\mathcal{E} \in env(\mathcal{A}) \cap env(\mathcal{B}) \cap env(\mathcal{C})$, let $\sigma, \sigma', \sigma''$ be schedulers of $\mathcal{E}||\mathcal{A}$, $\mathcal{E}||\mathcal{B}$, and $\mathcal{E}||\mathcal{C}$ respectively. Let $\varepsilon_{12}, \varepsilon_{23}, \varepsilon_{13} \in \mathbb{R}^{\geq 0}$ with $\varepsilon_{13} = \varepsilon_{12} + \varepsilon_{23}$. If $\sigma B^{\leq \varepsilon_{12}}_{(\mathcal{E},\mathcal{A},\mathcal{B}),f} \sigma'$ and $\sigma' B^{\leq \varepsilon_{23}}_{(\mathcal{E},\mathcal{B},\mathcal{C}),f} \sigma''$, then $\sigma B^{\leq \varepsilon_{13}}_{(\mathcal{E},\mathcal{A},\mathcal{C}),f} \sigma''$*

Proof. Let $\underline{\zeta} \subseteq range(f_{(\mathcal{E},\mathcal{A})}) \cup range(f_{(\mathcal{E},\mathcal{C})})$, then $\underline{\zeta} \subseteq range(f_{(\mathcal{E},\mathcal{A})}) \cup range(f_{(\mathcal{E},\mathcal{B})})$ and $\underline{\zeta} \subseteq range(f_{(\mathcal{E},\mathcal{B})}) \cup range(f_{(\mathcal{E},\mathcal{C})})$. By the triangle inequality, $|(f\text{-}dist_{(\mathcal{E},\mathcal{C})}(\sigma'')(\underline{\zeta}) - f\text{-}dist_{(\mathcal{E},\mathcal{A})}(\sigma)(\underline{\zeta}))| \leq |(f\text{-}dist_{(\mathcal{E},\mathcal{C})}(\sigma'')(\underline{\zeta}) - f\text{-}dist_{(\mathcal{E},\mathcal{B})}(\sigma')(\underline{\zeta}))| + |(f\text{-}dist_{(\mathcal{E},\mathcal{B})}(\sigma')(\underline{\zeta}) - f\text{-}dist_{(\mathcal{E},\mathcal{A})}(\sigma)(\underline{\zeta}))| \leq \varepsilon_{12} + \varepsilon_{23}$. □

We can see the next definition of f-implementation as the incapacity of an environment to distinguish two automata if it uses only information filtered by the insight function f.

Definition 88 (f-implementation)**.** *Let $f_{(.,.)}$ be an insight-function. Let S be a scheduler schema. Let $\epsilon \in \mathbb{R}^{\geq 0}$. We say that $\mathcal{A}$ f-*implements* $\mathcal{B}$ according to S with approximation ε, noted $\mathcal{A} \leq^{S,f}_{\varepsilon} \mathcal{B}$, if $\forall \mathcal{E} \in env(\mathcal{A}) \cap env(\mathcal{B})$, $\forall \sigma \in S(\mathcal{E}||\mathcal{A})$, $\exists \sigma' \in S(\mathcal{E}||\mathcal{B})$, $\sigma B^{\leq \varepsilon}_{(\mathcal{E},\mathcal{A},\mathcal{B}),f} \sigma'$.*

This definition is a formal generalisation of the informal definition of implementation that says that a (real) system $\mathcal{A}$ implements a (ideal) system $\mathcal{B}$ iff for every way of resolving the pure non-determinism in the $\mathcal{A}$-world, there exists a way of resolving the pure non-determinism in the $\mathcal{B}$-world, such that the two worlds are undistinguishable from an external point of view.

We would like to state a natural sufficient condition to obtain composability of f-implementation.

Definition 89 (Possible definitions of stability by composition)**.** *Let $f_{(.,.)}$ be an insight-function. Here are several possible definitions of "stability by composition".*

1. *for every triplet of PSIOA (resp. PCA) $(\mathcal{A}, \mathcal{B}, \mathcal{E})$, s.t. $\mathcal{B} \in env(\mathcal{A})$ and $\mathcal{E} \in env(\mathcal{B}||\mathcal{A})$, $f_{(\mathcal{E},\mathcal{B}||\mathcal{A})} = f_{(\mathcal{E},\mathcal{B})} \circ f_{(\mathcal{E}||\mathcal{B},\mathcal{A})}$.*
2. *for every quadruplet of PSIOA (resp. PCA) $(\mathcal{A}_1, \mathcal{A}_2, \mathcal{B}, \mathcal{E})$, s.t. $\mathcal{B} \in env(\mathcal{A}_1) \cap env(\mathcal{A}_2)$ and $\mathcal{E} \in env(\mathcal{B}||\mathcal{A}_1) \cap env(\mathcal{B}||\mathcal{A}_2)$, for every $(C_1, C_2) \in \mathcal{F}_{Execs(\mathcal{E}||\mathcal{B}||\mathcal{A}_1)} \times \mathcal{F}_{Execs(\mathcal{E}||\mathcal{B}||\mathcal{A}_2)}$, $f_{(\mathcal{E}||\mathcal{B},\mathcal{A}_1)}(C_1) = f_{(\mathcal{E}||\mathcal{B},\mathcal{A}_2)}(C_2) \Longrightarrow f_{(\mathcal{E},\mathcal{B}||\mathcal{A}_1)}(C_1) = f_{(\mathcal{E},\mathcal{B}||\mathcal{A}_2)}(C_2)$.*
3. *for every triplet of PSIOA (resp. PCA) $(\mathcal{A}, \mathcal{B}, \mathcal{E})$, s.t. $\mathcal{B} \in env(\mathcal{A})$ and $\mathcal{E} \in env(\mathcal{B}||\mathcal{A})$, for every $C, C' \in \mathcal{F}_{Execs(\mathcal{E}||\mathcal{B}||\mathcal{A})}$, $f_{(\mathcal{E}||\mathcal{B},\mathcal{A})}(C) = f_{(\mathcal{E}||\mathcal{B},\mathcal{A})}(C') \Longrightarrow f_{(\mathcal{E},\mathcal{B}||\mathcal{A})}(C) = f_{(\mathcal{E},\mathcal{B}||\mathcal{A})}(C')$.*

4. *for every quadruplet of PSIOA (resp. PCA)* $(\mathcal{A}_1, \mathcal{A}_2, \mathcal{B}, \mathcal{E})$*, s.t.* $\mathcal{B} \in env(\mathcal{A}_1) \cap env(\mathcal{A}_2)$ *and* $\mathcal{E} \in env(\mathcal{B}||\mathcal{A}_1) \cap env(\mathcal{B}||\mathcal{A}_2)$*, for every* σ, σ' *scheduler of* $\mathcal{E}||\mathcal{B}||\mathcal{A}_1$ *and* $\mathcal{E}||\mathcal{B}||\mathcal{A}_2$ *respectively,* $\forall \varepsilon \in \mathbb{R}^{\geq 0}$, $\sigma B^{\leq \varepsilon}_{(\mathcal{E}||\mathcal{B},\mathcal{A}_1,\mathcal{A}_2),f} \sigma' \Longrightarrow \sigma B^{\leq \varepsilon}_{(\mathcal{E},\mathcal{B}||\mathcal{A}_1,\mathcal{B}||\mathcal{A}_2),f} \sigma'$.

Lemma 10. – *(1) implies (2)*
– *(2) implies (3)*
– *(1) implies (4)*

Proof. – (1) implies (2) Let $C_1, C_2 \in \mathcal{F}_{Execs(\mathcal{E}||\mathcal{B}||\mathcal{A}_1)} \times \mathcal{F}_{Execs(\mathcal{E}||\mathcal{B}||\mathcal{A}_2)}$, we note $D \triangleq f_{(\mathcal{E}||\mathcal{B},\mathcal{A}_1)}(C_1) = f_{(\mathcal{E}||\mathcal{B},\mathcal{A}_2)}(C_2)$ by assumption. Hence, $f_{(\mathcal{E},\mathcal{B}||\mathcal{A}_1)}(C_1) = (f_{(\mathcal{E},\mathcal{B})} \circ f_{(\mathcal{E}||\mathcal{B},\mathcal{A}_1)})(C_1) = f_{(\mathcal{E},\mathcal{B})}(D)$, while $f_{(\mathcal{E},\mathcal{B}||\mathcal{A}_2)}(C_2) = (f_{(\mathcal{E},\mathcal{B})} \circ f_{(\mathcal{E}||\mathcal{B},\mathcal{A}_2)})(C_2) = f_{(\mathcal{E},\mathcal{B})}(D)$, which gives the desired result.
– (3) implies (2) Take $\mathcal{A}_1 = \mathcal{A}_2$
– (1) implies (4) Let $\zeta \subset \bigcup_{i \in \{1,2\}} range(f_{(\mathcal{E},\mathcal{B}||\mathcal{A}_i)})$. Let $\zeta' = f^{-1}_{(\mathcal{E},\mathcal{B})}(\zeta)$. $|f\text{-}dist_{(\mathcal{E},\mathcal{B}||\mathcal{A}_1)}(\sigma)(\zeta) - f\text{-}dist_{(\mathcal{E},\mathcal{B}||\mathcal{A}_2)}(\sigma')(\zeta)| = |f\text{-}dist_{(\mathcal{E}||\mathcal{B},\mathcal{A}_1)}(\sigma)(\zeta') - f\text{-}dist_{(\mathcal{E}||\mathcal{B},\mathcal{A}_2)}(\sigma')(\zeta')| \leq \varepsilon$

□

The expression 4 is what we want for composability of implementation, but property 3 is used in lemma 46. for the monotonicity theorem. Moreover, the expression 1 is simple to understand. The perception of $\mathcal{E}$ of the system $\mathcal{W} \triangleq \mathcal{E}||\mathcal{B}||\mathcal{A}$ is the perception of $\mathcal{E}$ of the perception of $\mathcal{E}||\mathcal{B}$ of the system $\mathcal{W}$. This why we keep this first expression.

If an insight function is stable by composition, it is said to be a *perception-function.*

Definition 90 (Perception function)**.** *A* perception-function *is an insight function* $f_{(.,.)}$*, such that for every triplet of PSIOA (resp. PCA)* $(\mathcal{A}, \mathcal{B}, \mathcal{E})$*, s.t.* $\mathcal{B} \in env(\mathcal{A})$ *and* $\mathcal{E} \in env(\mathcal{B}||\mathcal{A})$*,* $f_{(\mathcal{E},\mathcal{B}||\mathcal{A})} = f_{(\mathcal{E},\mathcal{B})} \circ f_{(\mathcal{E}||\mathcal{B},\mathcal{A})}$.

This property captures the fact that an environment $\mathcal{E}$ does not have a greater power of distinction than $\mathcal{E}$ composed with another system $\mathcal{B}$, which is quite intuitive. Any reasonable function that captures the perception of an automaton $\mathcal{A}$ by an environment $\mathcal{E} \in env(\mathcal{A})$ should be a perception function.

Substitutability We can restate the classic theorem of composability of implementation in a quite general form.

Theorem 3 (Implementation composability)**.** *Let* $f_{(.,.)}$ *be a perception-function. Let* $\varepsilon \in \mathbb{R}^{\geq 0}$ *Let* S *be a scheduler schema. Let* $\mathcal{A}_1$*,* $\mathcal{A}_2$*,* $\mathcal{B}$ *be PSIOA, s.t.* $\mathcal{A}_1 \leq^{S,f}_{\varepsilon} \mathcal{A}_2$*. If* $\mathcal{B} \in env(\mathcal{A}_1) \cap env(\mathcal{A}_2)$*, then* $\mathcal{B}||\mathcal{A}_1 \leq^{S,f}_{\varepsilon} \mathcal{B}||\mathcal{A}_2$.

Proof. If $\mathcal{E}$ is an environment for both $\mathcal{B}||\mathcal{A}_1$ and $\mathcal{B}||\mathcal{A}_2$, then $\mathcal{E}' = \mathcal{E}||\mathcal{B}$ is an environment for both $\mathcal{A}_1$ and $\mathcal{A}_2$. By associativity of parallel composition, we have for every $i \in \{1, 2\}$, $(\mathcal{E}||\mathcal{B})||\mathcal{A}_i = \mathcal{E}||(\mathcal{B}||\mathcal{A}_i)$. Since $\mathcal{A}_1 \leq^{S,f}_{\varepsilon} \mathcal{A}_2$, for any scheduler $\sigma \in S((\mathcal{E}||\mathcal{B})||\mathcal{A}_1)$, there exists a corresponding scheduler $\sigma' \in S((\mathcal{E}||\mathcal{B})||\mathcal{A}_2)$, s.t. $\sigma B^{\leq \varepsilon}_{(\mathcal{E}||\mathcal{B},\mathcal{A}_1,\mathcal{A}_2),f} \sigma'$. Thus, by stability by composition, for any scheduler $\sigma \in S(\mathcal{E}||(\mathcal{B}||\mathcal{A}_1))$, there exists a corresponding schedule $\sigma' \in S(\mathcal{E}||(\mathcal{B}||\mathcal{A}_2))$, s.t. $\sigma B^{\leq \varepsilon}_{(\mathcal{E},\mathcal{B}||\mathcal{A}_1,\mathcal{B}||\mathcal{A}_2),f} \sigma'$, which ends the proof. □

We also want to restate the classic theorem of f-implementation transitivity in the same form.

Theorem 4 (Implementation transitivity)**.** *Let S be a scheduler schema. Let $f_{(.,.)}$ be an insight-function. Let $\varepsilon_{12}, \varepsilon_{23}, \varepsilon_{13} \in \mathbb{R}^{\geq 0}$ with $\varepsilon_{13} = \varepsilon_{12} + \varepsilon_{23}$. Let $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ be PSIOA, s.t. $\mathcal{A}_1 \leq^{S,f}_{\epsilon_{12}} \mathcal{A}_2$ and $\mathcal{A}_2 \leq^{S,f}_{\epsilon_{23}} \mathcal{A}_3$, then $\mathcal{A}_1 \leq^{S,f}_{\epsilon_{13}} \mathcal{A}_3$.*

Proof. Let $\mathcal{E} \in env(\mathcal{A}_1) \cap env(\mathcal{A}_3)$.

Case 1: $\mathcal{E} \in env(\mathcal{A}_2)$. Let $\sigma_1 \in S(\mathcal{E}||\mathcal{A}_1)$ then, since $\mathcal{A}_1 \leq^{S,f}_{\varepsilon_{12}} \mathcal{A}_2$ there exists $\sigma_2 \in S(\mathcal{E}||\mathcal{A}_2)$, s.t. $\sigma_1 B^{\leq \varepsilon_{12}}_{(\mathcal{E},\mathcal{A}_1,\mathcal{A}_2),f} \sigma_2$and since $\mathcal{A}_2 \leq^{S,f}_{\varepsilon_{13}} \mathcal{A}_3$, there exists $\sigma_3 \in S(\mathcal{E}||\mathcal{A}_3)$ s.t. $\sigma_2 B^{\leq \varepsilon_{23}}_{(\mathcal{E},\mathcal{A}_2,\mathcal{A}_3),f} \sigma_3$and so for every $\sigma_1 \in S(\mathcal{E}||\mathcal{A}_1)$,there exists $\sigma_3 \in S(\mathcal{E}||\mathcal{A}_3)$ s.t. $\sigma_1 B^{\leq \varepsilon_{13}}_{(\mathcal{E},\mathcal{A}_1,\mathcal{A}_3),f} \sigma_3$ by lemma 9, i.e. $\mathcal{A}_1 \leq^{S,f}_{\varepsilon_{13}} \mathcal{A}_3$.

Case 2: $\mathcal{E} \notin env(\mathcal{A}_2)$. A renaming procedure has to be performed before applying Case 1.

Let $\mathbf{A} = \{\mathcal{E}, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}$. We note $acts(\mathbf{A}) = \bigcup_{\mathcal{B} \in \mathbf{A}} acts(\mathcal{B})$. We use the special character ® for our renaming which is assumed to not be present in any syntactical representation of any action in $acts(\mathbf{A})$.

We note r_{int} the action renaming fonction s.t. $\forall q \in Q_{\mathcal{E}}$, $\forall a \in \widehat{sig}(\mathcal{E})(q)$, if $a \in int(\mathcal{E})(q)$, then $r_{int}(q)(a) = a_{®int}$ and $r_{int}(q)(a) = a$ otherwise. Then we note $\mathcal{E}' = r_{int}(\mathcal{E})$.

If $\mathcal{E}'$ and $\mathcal{A}_2$ are not partially-compatible, it is only because of some reachable state $(q_{\mathcal{E}}, q_{\mathcal{A}_2}) \in Q'_{\mathcal{E}} \times Q_{\mathcal{A}_2}$ s.t. $out(\mathcal{A}_2)(q_{\mathcal{A}_2}) \cap out(\mathcal{E}')(q_{\mathcal{E}}) \neq \emptyset$. Thus, we rename the actions for each state to avoid this conflict.

We note r_{out} the renaming function for $\mathcal{E}'$, s.t. $\forall q_{\mathcal{E}} \in Q_{\mathcal{E}}$, $\forall a \in \widehat{sig}(\mathcal{E})(q_{\mathcal{E}})$, $r_{out}(q_{\mathcal{E}})(a) = a_{®out}$ if $a \in out(\mathcal{E})(q_{\mathcal{E}})$ and a otherwise. In the same way, We note, for every $i \in \{1,2,3\}$ r^i_{in} the renaming function for $\mathcal{A}_i$, s.t. $\forall q_{\mathcal{A}_i} \in Q_{\mathcal{A}_i}$, $\forall a \in \widehat{sig}(\mathcal{A}_i)(q_{\mathcal{A}_i})$ $r_{in}(q_{\mathcal{A}_i})(a) = a_{®out}$ if $a \in in(\mathcal{A}_i)(q_{\mathcal{A}_i})$ and a otherwise. By lemma 2, $\mathcal{E}'' \triangleq r_{out}(\mathcal{E}')$ is a PSIOA. Finally, $\mathcal{E}''$ and $\mathcal{A}''_i = r^i_{in}(\mathcal{A}_i)$ are obviously partially-compatible (and even compatible) for each $i \in \{1,2,3\}$.

There is an obvious isomorphism between $\mathcal{E}''||\mathcal{A}''_1$ and $\mathcal{E}||\mathcal{A}_1$ and between $\mathcal{E}''||\mathcal{A}''_3$ and $\mathcal{E}||\mathcal{A}_3$ that allows us to apply case 1, which ends the proof.

□

The two last theorems allow stating the classical theorem of substitutability.

Theorem 5 (Implementation substitutability)**.** *Let $f_{(.,.)}$ be a perception-function. Let S be a scheduler schema. Let $\varepsilon_a, \varepsilon_b, \varepsilon_c \in \mathbb{R}^{\geq 0}$ with $\varepsilon_c = \varepsilon_a + \varepsilon_b$. Let $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{B}_1$, $\mathcal{B}_2$ be PSIOA (resp. PCA), s.t. $\mathcal{A}_1 \leq^{S,f}_{\varepsilon_a} \mathcal{A}_2$ and $\mathcal{B}_1 \leq^{S,f}_{\varepsilon_b} \mathcal{B}_2$. If both $\mathcal{B}_1$ and $\mathcal{B}_2$ are partially compatible with both $\mathcal{A}_1$ and $\mathcal{A}_2$ then $\mathcal{A}_1||\mathcal{B}_1 \leq^{S,f}_{\varepsilon_c} \mathcal{A}_2||\mathcal{B}_2$.*

Proof. By theorem 3 of implementation composability, $\mathcal{A}_1||\mathcal{B}_1 \leq^{S,f}_{\varepsilon_a} \mathcal{A}_2||\mathcal{B}_1$ and $\mathcal{A}_2||\mathcal{B}_1 \leq^{S,f}_{\varepsilon_b} \mathcal{A}_2||\mathcal{B}_2$. By theorem 4 of implementation transitivity $\mathcal{A}_1||\mathcal{B}_1 \leq^{S,f}_{\varepsilon_c} \mathcal{A}_2||\mathcal{B}_2$. □

Trace and projection on the environment are perception-functions

Proposition 8 (trace is measurable)**.** *Let $\mathcal{A}$ be a PSIOA (resp. PCA).*

$trace_{\mathcal{A}} : (Execs(\mathcal{A}), \mathcal{F}_{Execs(\mathcal{A})}) \to (Traces(\mathcal{A}), \mathcal{F}_{Traces(\mathcal{A})})$ is measurable.

Proof. This is enough to show that $\forall \beta \in Traces^*(\mathcal{A})$, $trace^{-1}_{\mathcal{A}}(C_\beta) \in \mathcal{F}_{Execs(\mathcal{A})}$. Yet, $trace^{-1}_{\mathcal{A}}(C_\beta) = \bigcup_{\alpha \in Execs^*(\mathcal{A}), trace_{\mathcal{A}}(\alpha) = \beta} C_\alpha$. Hence, this is a countable union of cones of executions of $\mathcal{A}$, i.e. an element of $\mathcal{F}_{Execs(\mathcal{A})}$. □

Proposition 9 (projection is measurable)**.** *Let $\mathcal{A}$ be a PSIOA (resp. PCA) and $\mathcal{E} \in env(\mathcal{A})$.*

$$proj_{(\mathcal{E},\mathcal{A})} : \begin{cases} (Execs(\mathcal{E}||\mathcal{A}), \mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}) & \rightarrow & (Execs(\mathcal{E}), \mathcal{F}_{Execs(\mathcal{E})}) \\ \alpha & \mapsto & \alpha \upharpoonright \mathcal{E} \end{cases} \text{ is measurable.}$$

Proof. This is enough to show that $\forall \alpha' \in Execs^*(\mathcal{E}), proj^{-1}_{(\mathcal{E},\mathcal{A})}(C_{\alpha'}) \in \mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}$. Yet, $proj^{-1}_{(\mathcal{E},\mathcal{A})}(C_{\alpha'}) = \bigcup_{\alpha \in Execs^*(\mathcal{A}), \alpha\upharpoonright\mathcal{E}=\alpha'} C_\alpha$. Hence, this is a countable union of cones of executions of $\mathcal{E}||\mathcal{A}$, i.e. an element of $\mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}$. □

Lemma 11 (projection is a perception function)**.** *The function $proj_{(.,.)}$ parametrized with PSIOA $\mathcal{E}, \mathcal{A}$ where $\mathcal{E} \in env(\mathcal{A})$ is a perception function.*

Proof.
- (measurability) Immediate by proposition 9.
- (stability by composition) in the sense 1 of definition 89 Let $\mathcal{A}$ be a PSIOA (resp. PCA). Let $\mathcal{B} \in env(\mathcal{A})$, $\mathcal{E} \in env(\mathcal{B}||\mathcal{A})$. Let $\alpha \in Execs(\mathcal{E}||\mathcal{B}||\mathcal{A})$, clearly $\alpha \upharpoonright \mathcal{E} = (\alpha \upharpoonright (\mathcal{E}||\mathcal{B})) \upharpoonright \mathcal{E}$.

□

We have the same kind of result for the trace function, which is more often used for the perception function in practice.

Lemma 12 (trace is a perception function)**.** *The function $trace_{(.,.)}$ parametrized with PSIOA $\mathcal{E}, \mathcal{A}$ where $\mathcal{E} \in env(\mathcal{A})$ (with $trace_{(\mathcal{E},\mathcal{A})} = trace_{(\mathcal{E}||\mathcal{A})}$) is a perception function.*

Proof.
- Immediate by proposition 8.
- (stability by composition) in the sense 3 and 4 of definition 89. The result is even more trivial since for every triplet of PSIOA $(\mathcal{A}, \mathcal{B}, \mathcal{E})$, s.t. $\mathcal{B} \in env(\mathcal{A})$, $\mathcal{E} \in env(\mathcal{B}||\mathcal{A})$, $trace_{(\mathcal{E},\mathcal{B}||\mathcal{A})} = trace_{(\mathcal{E}||\mathcal{B},\mathcal{A})}$

□

Thus, given an environment $\mathcal{E}$ of $\mathcal{A}$ probability measure μ on $\mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}$, and a scheduler σ of $(\mathcal{E}||\mathcal{A})$ we define $\mathtt{p}dist_{(\mathcal{E},\mathcal{A})}(\sigma, \mu) \triangleq proj\text{-}dist_{(\mathcal{E},\mathcal{A})}(\sigma, \mu)$, to be the image measure of $\epsilon_{\sigma,\mu}$ under $proj_{(\mathcal{E},\mathcal{A})}$. We note $\mathtt{p}dist_{(\mathcal{E},\mathcal{A})}(\sigma)$ for $\mathtt{p}dist_{(\mathcal{E},\mathcal{A})}(\sigma, \delta_{\bar{q}_{\mathcal{E}||\mathcal{A}}})$.

This choice that slightly differs from $tdist_{(\mathcal{E},\mathcal{A})}(\sigma, \mu) = trace\text{-}dist_{(\mathcal{E},\mathcal{A})}(\sigma, \mu)$ used in [CCK+07], is motivated by the achievement of monotonicity of p-implementation w.r.t. PSIOA creation.

The combination of theorems 2 and 5 allow to study dynamic probabilistic distributed systems in a modular manner. For "horizontal interactions" in the "same layer", we can substitute a specification with a concrete protocol without losing hyper-properties. However, can we perform the same kind of susbtitution for "vertical interactions" at "different layers"? This question is discussed in the next section.

3.7. Introduction on PCA corresponding w.r.t. PSIOA $\mathcal{A}$, $\mathcal{B}$ to introduce monotonicity

Here, we take an interest in PCA $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ that differ only on the fact that $\mathcal{B}$ supplants $\mathcal{A}$ in $X_{\mathcal{B}}$.

3.7.1. An informal statement of the theorem

Definition 91 ((Informal) corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$)**.** *Intuitively, $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ are corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$, noted $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$ if they differ only in that $X_{\mathcal{A}}$ dynamically creates and destroys automaton $\mathcal{A}$ instead of creating and destroying automaton $\mathcal{B}$ as $X_{\mathcal{B}}$ does. Some technical minor assumptions have to be verified:*

1. *$config(X_{\mathcal{A}})(\bar{q}_{X_{\mathcal{A}}}) \lhd_{AB} config(X_{\mathcal{B}})(\bar{q}_{X_{\mathcal{B}}})$: The associated configuration of respective start states are identical except that the automaton $\mathcal{B}$ supplants $\mathcal{A}$ but with the same external signature.*
2. *$X_{\mathcal{A}}, X_{\mathcal{B}}$ are creation&hiding-corresponding w.r.t. $\mathcal{A}, \mathcal{B}$: the two PCA hide some output actions and create some PSIOA in the same manner, excepting for the creation of $\mathcal{B}$ that supplants the creation of $\mathcal{A}$.*
3. *$\forall \mathcal{K} \in \{\mathcal{A}, \mathcal{B}\}$, $\forall q \in Q_{X_{\mathcal{K}}}$, for every $\mathcal{K}$-exclusive action a at state q, $created(X_{\mathcal{K}})(q)(a) = \emptyset$, where a $\mathcal{K}$-exclusive action is an action which is in the signature of sub-automaton $\mathcal{K}$ only.*
4. *(Technical)*
   a) *$\forall \mathcal{K} \in \{\mathcal{A}, \mathcal{B}\}$, $X_{\mathcal{K}}$ is $\mathcal{K}$-conservative: Each state of $X_{\mathcal{K}}$ is perfectly defined by its configuration deprived of sub-automaton $\mathcal{K}$ and external actions of $\mathcal{K}$ are not hidden.*
   b) *$\forall \mathcal{K} \in \{\mathcal{A}, \mathcal{B}\}$, $X_{\mathcal{K}}$ is $\mathcal{K}$-creation explicit: the creation of $\mathcal{K}$ is equivalent to the triggering of an action in a dedicated set.*

We would like to state the monotonicity of PSIOA creation with p-implementation. However, it holds only for a specific class of schedulers, so-called *creation-oblivious* that do not take into account the visited states and the triggered exclusive actions of a sub-automaton before its last destruction to output the next action to trigger.

Definition 92 ((Informal) creation-oblivious scheduler)**.** *Let $\tilde{\mathcal{A}}$ be a PSIOA, $\tilde{W}$ be a PCA, $\tilde{\sigma} \in schedulers(\tilde{W})$. We say that $\tilde{\sigma}$ is $\mathcal{A}$-*creation oblivious *if for every triplet $(\tilde{\alpha}_1, \tilde{\alpha}_2, \tilde{\alpha}_3)$ s.t. (1) $lstate(\tilde{\alpha}_1) = lstate(\tilde{\alpha}_2) = fstate(\tilde{\alpha}_3)$ and (2) $\tilde{\alpha}_1$ and $\tilde{\alpha}_2$ differ only on $\mathcal{A}$-exclusive actions and visited states of sub-automaton $\mathcal{A}$, then (3) $\tilde{\sigma}(\tilde{\alpha}_1^\frown \tilde{\alpha}_3) = \tilde{\sigma}(\tilde{\alpha}_2^\frown \tilde{\alpha}_3)$. A* creation oblivious *scheduler is a $\mathcal{A}$-*creation oblivious *for every PSIOA $\mathcal{A}$. The scheduler schema that outputs only creation oblivious oblivious scheduler is denoted S_o.*

The formal definitions of the two last concepts are defined in the remaining part of this section. It is crucial to limit the power of the scheduler to decompose the measure of a class of comportment as a function of measures of classes of shorter comportment where no creation of $\mathcal{A}$ or $\mathcal{B}$ occurs excepting potentially at very last action. This reduction is more or less necessary to obtain monotonicity of dynamic creation/destruction:

Theorem 6 (monotonicity with p-implementation $\leq_0^{S_o,\mathtt{p}}$)**.** *Let $\mathcal{A}, \mathcal{B}$ be PSIOA, let $X_{\mathcal{A}}$, $X_{\mathcal{B}}$ be PCA.Let S_o be the schema of creation-oblivious schedulers and $\mathtt{p} = proj_{(.,.)}$.*
If (1) $\mathcal{A} \leq_0^{S_o,\mathtt{p}} \mathcal{B}$ and (2) $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$, then (3) $X_{\mathcal{A}} \leq_0^{S_o,\mathtt{p}} X_{\mathcal{B}}$

3.7.2. Naive correspondence between two PCA

We formalize the idea that two configurations are identical except that the automaton $\mathcal{B}$ supplants $\mathcal{A}$ but with the same external signature. The following definition comes from [AL16].

Definition 93 ($\lhd_{\mathcal{AB}}$-corresponding configurations)**.** *(see figure 4.11) Let $\Phi \subseteq Autids$, and $\mathcal{A}, \mathcal{B}$ be PSIOA identifiers. Then we define $\Phi[\mathcal{B}/\mathcal{A}] = (\Phi \setminus \mathcal{A}) \cup \{\mathcal{B}\}$ if $\mathcal{A} \in \Phi$, and $\Phi[\mathcal{B}/\mathcal{A}] = \Phi$ if $\mathcal{A} \notin \Phi$. Let C, D be configurations. We define $C \lhd_{\mathcal{AB}} D$ iff (1) $auts(D) = auts(C)[\mathcal{B}/\mathcal{A}]$, (2) for every $\mathcal{A}' \notin auts(C) \setminus \{\mathcal{A}\}$: $map(D)(\mathcal{A}') = map(C)(\mathcal{A}')$, and (3) $ext(\mathcal{A})(s) = ext(\mathcal{B})(t)$ where $s = map(C)(\mathcal{A}), t = map(D)(\mathcal{B})$. That is, in $\lhd_{\mathcal{AB}}$-corresponding configurations, the SIOA other than $\mathcal{A}, \mathcal{B}$ must be the same and must be in the same state. $\mathcal{A}$ and $\mathcal{B}$ must have the same external signature. In the sequel, when we write $\Psi = \Phi[\mathcal{B}/\mathcal{A}]$, we always assume that $\mathcal{B} \notin \Phi$ and $\mathcal{A} \notin \Psi$.*

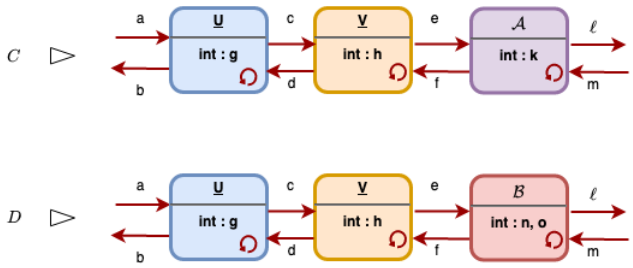


Figure 3.24. – $\lhd_{\mathcal{AB}}$ corresponding-configuration

Remark 3. *It is possible to have two configurations C, D s.t. $C \lhd_{\mathcal{AA}} D$. That would mean that C and D only differ on the state of $\mathcal{A}$ (s or t) that has even the same external signature in both cases $ext(\mathcal{A})(s) = ext(\mathcal{A})(t)$, while we would have $int(\mathcal{A})(s) \neq int(\mathcal{A})(t)$.*

Now, we formalize the fact that two PCA create some PSIOA in the same manner, except for $\mathcal{B}$ that supplants $\mathcal{A}$. Here again, this definition comes from [AL16].

Definition 94 (Creation corresponding configuration automata)**.** *Let X, Y be PCA and $\mathcal{A}, \mathcal{B}$ be PSIOA. We say that X, Y are creation-corresponding w.r.t. $\mathcal{A}, \mathcal{B}$ iff*

1. *X never creates $\mathcal{B}$ and Y never creates $\mathcal{A}$.*
2. *Let $(\alpha, \pi) \in \underline{Execs}^*(X) \times \underline{Execs}^*(Y)$ s.t. $trace_{\mathcal{A}}(\alpha) = trace_{\mathcal{B}}(\pi)$. Let $q = lstate(\alpha), q' = lstate(\pi)$. Then $\forall a \in \widehat{sig}(X)(q) \cap \widehat{sig}(Y)(q') : created(Y)(q')(a) = created(X)(q)(a)[\mathcal{B}/\mathcal{A}]$.*

In the same way, as in definition 94, we formalize the fact that two PCA hide some output actions in the same manner. Here again, this definition is inspired by [AL16].

Definition 95 (Hiding corresponding configuration automata)**.** *Let X, Y be PCA and $\mathcal{A}, \mathcal{B}$ be PSIOA. We say that X, Y are hiding-corresponding w.r.t. $\mathcal{A}, \mathcal{B}$ iff*

1. *X never creates $\mathcal{B}$ and Y never creates $\mathcal{A}$.*
2. *Let $(\alpha, \pi) \in Execs^*(X) \times Execs^*(Y)$ s.t. $trace_{\mathcal{A}}(\alpha) = trace_{\mathcal{B}}(\pi)$. Let $q = lstate(\alpha), q' = lstate(\pi)$. Then $hidden\text{-}actions(Y)(q') = hidden\text{-}actions(X)(q)$.*

Next definition is the immediate conjunction of the two previous ones.

Definition 96 (creation&hiding-corresponding)**.** *Let X, Y be PCA and $\mathcal{A}, \mathcal{B}$ be PSIOA. We say that X, Y are creation&hiding-corresponding w.r.t. $\mathcal{A}, \mathcal{B}$, if they are both creation-corresponding and hiding-corresponding w.r.t. $\mathcal{A}, \mathcal{B}$*

Now we define the notion of $\mathcal{A}$-exclusive action that corresponds to an action which is in the signature of $\mathcal{A}$ only. This definition is motivated by the fact that monotonicity induces that $\mathcal{A}$-exclusive (resp. $\mathcal{B}$-exclusive) actions do not create automata. Indeed, otherwise two internal actions a and a' of $\mathcal{A}$ and $\mathcal{B}$ respectively could create different automata $\mathcal{C}$ and $\mathcal{D}$ and break the correspondence.

Definition 97 ($\mathcal{A}$-exclusive action)**.** *Let $\mathcal{A} \in$ Autids, X be a PCA. Let $q \in Q_X$, $(\mathbf{A}, \mathbf{S}) = config(X)(q)$, $act \in \widehat{sig}(X)(q)$. We say that act is $\mathcal{A}$-exclusive at state q if for every $\mathcal{A}' \in \mathbf{A} \setminus \{\mathcal{A}\}$, $act \notin \widehat{sig}(\mathcal{A}')(\mathbf{S}(\mathcal{A}'))$ (and so $act \in \widehat{sig}(\mathcal{A})(\mathbf{S}(\mathcal{A}))$ only).*

The previous definitions 93, 94, 95 and 97 allow us to define a first (naive) definition of PCA corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$.

Definition 98 (naively corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$)**.** *Let $\mathcal{A}, \mathcal{B} \in$ Autids, $X_\mathcal{A}$ and $X_\mathcal{B}$ be PCA we say that $X_\mathcal{A}$ and $X_\mathcal{B}$ are naively corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$, if they verify:*

1. *$config(X_\mathcal{A})(\bar{q}_{X_\mathcal{A}}) \lhd_{AB} config(X_\mathcal{B})(\bar{q}_{X_\mathcal{B}})$.*
2. *$X_\mathcal{A}, X_\mathcal{B}$ are creation&hiding-corresponding w.r.t. $\mathcal{A}, \mathcal{B}$*
3. *(No exclusive creation from $\mathcal{A}$ and $\mathcal{B}$) for each $\mathcal{K} \in \{\mathcal{A}, \mathcal{B}\}$, $\forall q \in Q_{X_\mathcal{K}}$, for every $\mathcal{K}$-exclusive action a, $created(X_\mathcal{K})(q)(a) = \emptyset$*

However, the PSIOA creation is not proved monotonic with the p-implementation relationship (introduced in subsection 3.6.2) without some additional technical assumptions informally introduced in defintion 91 and presented in next subsection 3.7.3. Roughly speaking, it allows to 1) define a PCA $Y = X \setminus \{\mathcal{A}\}$ that corresponds to X "deprived" of $\mathcal{A}$ and 2) define the composition between Y and $\mathcal{A}$, 3) avoiding some ambiguities during the construction. In the first instance, the reader should skip the next subsection 3.7.3 on conservatism and keep in mind the intuition only. This sub-section 3.7.3 can be used to know the assumptions of the theorems of monotonicity and use them as black boxes. The assumptions will be recalled during the proof.

3.7.3. Conservatism: the additional assumption for relevant definition of correspondence w.r.t. $\mathcal{A}, \mathcal{B}$

This subsection aims to define the notion of $\mathcal{A}$-conservative PCA, informally introduced in definition 91, item 4a.

Some definitions relative to configurations In the remaining, it will often be useful to reason on the configurations. This is why we introduce some definitions that will be used again and again in the demonstrations.

The next definition captures the idea that two states of a certain layer represent the same situation for the bottom layer.

Definition 99 (configuration-equivalence between two states)**.** *Let K, K' be PCA and $(q, q') \in Q_K \times Q_{K'}$. We say that q and q' are* config-equivalent*, noted $qR_{conf}q'$, if $config(K)(q) = config(K')(q')$. Furthermore, if*

- *$config(K)(q) = config(K')(q')$,*
- *$hidden\text{-}actions(K)(q) = hidden\text{-}actions(K')(q')$ and*
- *$\forall a \in \widehat{sig}(K)(q) = \widehat{sig}(K')(q')$, $created(K)(q)(a) = created(K')(q')(a)$,*

we say that q and q' are strictly-equivalent*, noted $qR_{strict}q'$.*

Now, we define a special subset of PCA that do not tolerate different configuration-equivalent states.

Definition 100 (Configuration-conflict-free PCA)**.** *Let K be a PCA. We say K is configuration-conflict-free, if for every $q, q' \in Q_K$ s.t. $qR_{conf}q'$, then $q = q'$. The current state of a configuration-conflict-free PCA can be defined by its current attached configuration.*

For some elaborate definitions, we found it useful to introduce the set of potential output actions of $\mathcal{A}$ in a configuration $config(X)(q)$ coming from a state q of a PCA X:

Definition 101 (potential ouput)**.** *Let $\mathcal{A} \in autids$. Let X be a PCA. Let $q \in Q_X$. We note pot-out$(X)(q)(\mathcal{A})$ the set of potential output actions of $\mathcal{A}$ in $config(X)(q)$ that is*

- *pot-out$(X)(q)(\mathcal{A}) = \emptyset$ if $\mathcal{A} \notin auts(config(X)(q))$*
- *pot-out$(X)(q)(\mathcal{A}) = out(\mathcal{A})(map(config(X)(q))(\mathcal{A}))$ if $\mathcal{A} \in auts(config(X)(q))$*

Here, we define a configuration C deprived of an automaton $\mathcal{A}$ in the most natural way.

Definition 102 ($C \setminus \{\mathcal{A}\}$ Configuration deprived of an automaton)**.** *$C = (\mathbf{A}, \mathbf{S})$. $C \setminus \{\mathcal{A}\} = (\mathbf{A}', \mathbf{S}')$ with $\mathbf{A}' = \mathbf{A} \setminus \{\mathcal{A}\}$ and $\mathbf{S}'$ the restriction of $\mathbf{S}$ on $\mathbf{A}'$*

The two last definitions 101 and 102 allows us to define in compact way a new relation between states that captures the idea that two states $q \in Q_X$ and $q' \in Q_Y$ are equivalent modulo a difference uniquely due to the presence of automaton $\mathcal{A}$ in $config(X)(q)$ and $config(Y)(q')$.

Definition 103 ($R^{\setminus\{\mathcal{A}\}}$ relationship (equivalent if we forget $\mathcal{A}$))**.** *Let $S = \{Q_X | X \text{ is a PCA }\}$ be the set of states of any PCA. Let $\mathcal{A} \in Autids$. We defined the equivalence relation $R_{conf}^{\setminus\{\mathcal{A}\}}$ and $R_{conf}^{\setminus\{\mathcal{A}\}}$ on S defined by $\forall X, Y$ PCA, $\forall (q_X, q_Y) \in Q_X \times Q_Y$:*

- *$q_X R_{conf}^{\setminus\{\mathcal{A}\}} q_Y \iff config(X)(q_X) \setminus \{A\} = config(Y)(q_Y) \setminus \{A\}$*
- *$q_X R_{strict}^{\setminus\{\mathcal{A}\}} q_Y \iff$ the conjonction of the 3 following properties:*
  - *$q_X R_{conf}^{\setminus\{\mathcal{A}\}} q_Y$*
  - *$\forall a \in \widehat{sig}(X)(q_X) \cap \widehat{sig}(Y)(q_Y)$, $created(Y)(q_Y)(a) \setminus \{\mathcal{A}\} = created(X)(q_X)(a) \setminus \{\mathcal{A}\}$*
  - *hidden-actions$(X)(q_X) \setminus$pot-out$(X)(q_X)(\mathcal{A}) =$ hidden-actions$(Y)(q_Y) \setminus$pot-out$(Y)(q_Y)(\mathcal{A})$*

$\mathcal{A}$-fair and $\mathcal{A}$-conservative: necessary assumptions to authorize the construction used in the proof
Now, we are ready to define $\mathcal{A}$-fairness and then $\mathcal{A}$-conservatism.

A $\mathcal{A}$-fair PCA is a PCA s.t. we can deduce its current properties from its current configuration deprived of $\mathcal{A}$. This assumption will allow us to define $Y = X \setminus \{\mathcal{A}\}$ in the proof of monotonicity.

Definition 104 ($\mathcal{A}$-fair PCA)**.** *Let $\mathcal{A} \in Autids$. Let X be a PCA. We say that X is $\mathcal{A}$-fair if*

- *(configuration-conflict-free) X is configuration-conflict-free.*
- *(no conflict for projection) $\forall q_X, q'_X \in Q_X$, s.t. $q_X R_{conf}^{\setminus\{\mathcal{A}\}} q'_X$ then $q_X R_{strict}^{\setminus\{\mathcal{A}\}} q'_X$.*
- *(no exclusive creation by $\mathcal{A}$) $\forall q_X \in Q_X$, $\forall a \in \widehat{sig}(X)(q_X)$ $\mathcal{A}$-exclusive in q_X, $created(X)(q_X)(a) = \emptyset$*

This definition 104 allows the next definition 105 to be well-defined. A $\mathcal{A}$-conservative PCA is a $\mathcal{A}$-fair PCA that does not hide any output action that could be an external action of $\mathcal{A}$. This assumption will allow us to define the composition between $\mathcal{A}$ and $Y = X \setminus \{\mathcal{A}\}$ in the proof of monotonicity.

Definition 105 ($\mathcal{A}$-conservative PCA)**.** *Let X be a PCA, $\mathcal{A}$ be a PSIOA. We say that X is $\mathcal{A}$-conservative if it is $\mathcal{A}$-fair and $\forall q \in Q_X$, $C = config(X)(q)$ s.t. $\mathcal{A} \in auts(C)$ and $map(C)(\mathcal{A}) \triangleq q_{\mathcal{A}}$, hidden-actions$(X)(q) \cap \widehat{ext}(\mathcal{A})(q_{\mathcal{A}}) = \emptyset$.*

3.7.4. Corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$

We are closed to state all the technical assumptions to achieve monotonicity of PSIOA creation with p-implementation. We introduce one last assumption so-called *creation-explicitness*, used in section 4.5 to reduce implementation of $X_{\mathcal{B}}$ by $X_{\mathcal{A}}$ to implementation of $\mathcal{B}$ by $\mathcal{A}$.

Intuitively, a PCA is $\mathcal{A}$-*creation-explicit* if the creation of a sub-automaton $\mathcal{A}$ is equivalent to the triggering of action in a dedicated set. This property will allow obtaining the reduction of lemma 56.

Definition 106 (creation-explicit PCA)**.** *Let $\mathcal{A}$ be a PSIOA and X be a PCA. We say that X is* $\mathcal{A}$-creation-explicit *iff: it exists a set of actions, noted $creation\text{-}actions(X)(\mathcal{A})$, s.t. $\forall q_X \in Q_X$, $\forall a \in \widehat{sig}(X)(q_X)$, if we note $\mathbf{A}_X = auts(config(X)(q_X))$ and $\varphi_X = created(X)(q_X)(a)$, then $\mathcal{A} \notin \mathbf{A}_X \wedge \mathcal{A} \in \varphi_X \iff a \in creation\text{-}actions(X)(\mathcal{A})$.*

Now we can define new (non naive) correspondence w.r.t. PSIOA $\mathcal{A}$, $\mathcal{B}$ to define (non naively) monotonic relationship.

Definition 107 ($X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$)**.** *Let $\mathcal{A}, \mathcal{B} \in Autids$, $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ be PCA we say that $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ are corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$, noted $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$, if 1) they are naively corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$, 2) they are $\mathcal{A}$-conservative and $\mathcal{B}$-conservative respectively and 3) they are $\mathcal{A}$-creation explicit and $\mathcal{B}$-creation explicit respectively, i.e. they verify:*

1. *$config(X_{\mathcal{A}})(\bar{q}_{X_{\mathcal{A}}}) \lhd_{AB} config(X_{\mathcal{B}})(\bar{q}_{X_{\mathcal{B}}})$ in the sense of definition 93.*
2. *$X_{\mathcal{A}}, X_{\mathcal{B}}$ are creation&hiding-corresponding w.r.t. $\mathcal{A}, \mathcal{B}$, in the sense of definition 96.*
3. *$\forall \mathcal{K} \in \{\mathcal{A}, \mathcal{B}\}$, $\forall q \in Q_{X_{\mathcal{K}}}$, for every $\mathcal{K}$-exclusive action a at state q (in the sense of definition 97), $created(X_{\mathcal{K}})(q)(a) = \emptyset$.*
4. *(Technical)*
   a) *$\forall \mathcal{K} \in \{\mathcal{A}, \mathcal{B}\}$, $X_{\mathcal{K}}$ is $\mathcal{K}$-conservative in the sense of definition 105.*
   b) *$\forall \mathcal{K} \in \{\mathcal{A}, \mathcal{B}\}$, $X_{\mathcal{K}}$ is $\mathcal{K}$-creation explicit in the sense of definition 106.*

Definition 108 (Preorder allowing monotonicity of PSIOA creation)**.** *Let P be a preorder on PSIOA and PCA. We say that P allows monotonicity of PSIOA creation if for every pair of PSIOA $(\mathcal{A}, \mathcal{B})$ and every pair of PCA $(X_{\mathcal{A}}, X_{\mathcal{B}})$, we have: (1) $(\mathcal{A}, \mathcal{B}) \in P$ and (2) $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$ implies (3) $(X_{\mathcal{A}}, X_{\mathcal{B}}) \in P$.*

We would like to claim that p-implementation allows monotonicity of PSIOA creation, but it holds only for a certain class of schedulers, so-called *creation-oblivious* introduced in next subsection 3.7.5.

3.7.5. Creation-oblivious scheduler

Here we present a particular scheduler schema, that does not take into account previous exclusive actions of a particular sub-automaton to output its probability over transitions to trigger.

We start by defining *strict oblivious-schedulers* that output the same transition with the same probability for pair of execution fragments that differ only by prefixes in the same class of equivalence. This definition is inspired by the one provided in the Segala's thesis but is more restrictive since we require strict equality instead of a correlation (section 5.6.2 in [Seg95b]).

Definition 109 (oblivious scheduler)**.** *Let $\tilde{W}$ be a PCA or a PSIOA, let $\tilde{\sigma} \in schedulers(\tilde{W})$ and let $\equiv$ be an equivalence relation on $Frags^*(\tilde{W})$ verifying $\forall \tilde{\alpha}_1, \tilde{\alpha}_2 \in Frags^*(\tilde{W})$ s.t. $\tilde{\alpha}_1 \equiv \tilde{\alpha}_2$, $lstate(\alpha_1) = lstate(\alpha_2)$. We say that $\tilde{\sigma}$ is $(\equiv)$-strictly oblivious if $\forall \tilde{\alpha}_1, \tilde{\alpha}_2, \tilde{\alpha}_3 \in Frags^*(\tilde{W})$ s.t. 1) $\alpha_1 \equiv \alpha_2$ and 2) $fstate(\tilde{\alpha}_3) = lstate(\tilde{\alpha}_2) = lstate(\tilde{\alpha}_1)$, then $\tilde{\sigma}(\tilde{\alpha}_1 \frown \tilde{\alpha}_3) = \tilde{\sigma}(\tilde{\alpha}_2 \frown \tilde{\alpha}_3)$.*

Now we define the relation of equivalence that defines our subset of creation-oblivious schedulers. Intuitively, two executions fragments ending on $\mathcal{A}$ creation are in the same equivalence class if they differ only in terms of exclusive actions of $\mathcal{A}$.

Definition 110 ($\tilde{\alpha} \equiv^{cr}_{\mathcal{A}} \tilde{\alpha}'$)**.** *Let $\mathcal{A}$ be a PSIOA, and $\tilde{W}$ be a PCA. For every $\tilde{\alpha}, \tilde{\alpha}' \in Frags^*(\tilde{W})$, we say $\tilde{\alpha} \equiv^{cr}_{\mathcal{A}} \tilde{\alpha}'$ iff:*

1. *$\tilde{\alpha}, \tilde{\alpha}'$ both ends on $\mathcal{A}$-creation.*
2. *$\tilde{\alpha}$ and $\tilde{\alpha}'$ differ only in the $\mathcal{A}$-exclusive actions and the states of $\mathcal{A}$, i.e. $\mu(\tilde{\alpha}) = \mu(\tilde{\alpha}')$ where $\mu(\tilde{\alpha} = \tilde{q}^0 a^1 \tilde{q}^1 ... a^n \tilde{q}^n) \in Frags^*(\tilde{W})$ is defined as follows:*
   - *remove the $\mathcal{A}$-exclusive actions*
   - *replace each state $\tilde{q}^i$ by its configuration $Config(\tilde{W})(\tilde{q}) = (\mathbf{A}^i, \mathbf{S}^i)$*
   - *replace each configuration $(\mathbf{A}^i, \mathbf{S}^i)$ by $(\mathbf{A}^i, \mathbf{S}^i) \setminus \{\mathcal{A}\}$*
   - *replace the (non-alternating) sequences of identical configurations (due to $\mathcal{A}$-exclusiveness of removed actions) by one unique configuration.*

   More formally, μ can be recursively defined as follows:
   - $\mu(\tilde{\alpha}) = Config(\tilde{W})(\tilde{q}) \setminus \{\mathcal{A}\}$ *if* $\tilde{\alpha} = \tilde{q} \in Q_{\tilde{W}}$
   - $\mu(\tilde{\alpha} a \tilde{q}) = \begin{cases} \mu(\tilde{\alpha}) a (Config(\tilde{W})(\tilde{q}) \setminus \{\mathcal{A}\}) & \text{if } a \text{ is not } \mathcal{A}\text{-exclusive} \\ \mu(\tilde{\alpha}) & \text{otherwise.} \end{cases}$
3. *$lstate(\tilde{\alpha}) = lstate(\tilde{\alpha}')$*

Remark 4. *We can remark that the items 3 can be deduced from 1 and 2 if X is configuration-conflict-free.*

Definition 111 (creation-oblivious scheduler)**.** *Let $\tilde{\mathcal{A}}$ be a PSIOA, $\tilde{W}$ be a PCA, $\tilde{\sigma} \in schedulers(\tilde{W})$. We say that $\tilde{\sigma}$ is* $\mathcal{A}$-creation oblivious *if it is $(\equiv^{cr}_{\mathcal{A}})$-strictly oblivious.*

We say that $\tilde{\sigma}$ is creation-oblivious *if it is $\mathcal{A}$-creation oblivious for every sub-automaton $\mathcal{A}$ of $\tilde{W}$ ($\mathcal{A} \in \bigcup_{q \in Q_{\tilde{W}}} auts(config(\tilde{W})(q))$). We note S_o the function that maps any PCA $\tilde{W}$ to the set of creation-oblivious schedulers of $\tilde{W}$.*

We have formally defined our notion of creation-oblivious scheduler. This will be a key property to ensure lemma 56 that allows reducing the measure of a class of comportment into a function of measures of classes of shorter comportment where no creation of $\mathcal{A}$ or $\mathcal{B}$ occurs excepting potentially at very last action. This reduction is more or less necessary to obtain monotonicity of implementation relation:

Theorem 7 ($\leq_0^{S_o, \mathtt{p}}$ allows monotonicity)**.** *Let $\mathcal{A}, \mathcal{B}$ be PSIOA, let $X_{\mathcal{A}}$, $X_{\mathcal{B}}$ be PCA. Let S_o be the scheduler schema of creation-oblivious schedulers and $\mathtt{p} = proj_{(.,.)}$.*
If $\mathcal{A} \leq_0^{S_o, \mathtt{p}} \mathcal{B}$ and $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\nabla} X_{\mathcal{B}}$, then $X_{\mathcal{A}} \leq_0^{S_o, p} X_{\mathcal{B}}$.

The next chapter is dedicated to the proof of this theorem 7. We start by defining in section 4.1 a morphism between executions of automata, so-called *executions-matching*, that preserves structure and measure of probability under *alter ego schedulers*. Next, we define in section 4.2 the notion of an automaton $X_{\mathcal{A}}$ deprived of a PSIOA $\mathcal{A}$, noted $X_{\mathcal{A}} \backslash \{\mathcal{A}\}$. Furthermore, we show in section 4.3 that there is an executions-matching from a PCA $X_{\mathcal{A}}$ to $(X_{\mathcal{A}} \setminus \{\mathcal{A}\}) || \tilde{\mathcal{A}}^{sw}$ where $\tilde{\mathcal{A}}^{sw}$ is the *simpleton wrapper* of $\mathcal{A}$, i.e. a PCA that only handle $\mathcal{A}$. The section 4.5 uses the morphism of section 4.3 to reduce the implementation of $X_{\mathcal{B}}$ by $X_{\mathcal{A}}$ to the implementation of $\mathcal{B}$ by $\mathcal{A}$ and finally obtain the monotonicity of implementation w.r.t. PSIOA creation. Finally section 4.6 explains why the task-scheduler introduced in [CCK+07] is not creation-oblivious.

3.8. Summary

In this chapter, we introduced Dynamic Probabilistic I/O Automata. The model naturally merges Dynamic I/O Automata of Attie et Lynch [AL16] and Probabilistic I/O Automata of Segala et al. [Seg95b, CCK$^+$06b, CCK$^+$18] with each other. The first main result is theorem 2 that ensures that the composition of two (partially-compatible) dynamic automata is itself a dynamic automaton. This comes with the associated constraints that link a dynamic automaton with a configuration, keeping track of the evolving set of elements of a lower level with their associated current states. Then, benefiting from well established probabilistic observational semantic [Seg95a, LSV03], it was easy to reformulate composability results for dynamic automata, namely in theorem 5 of substitutability of implementation relationship $\leq_\varepsilon^{S,f}$ for any "perception function" $f_{(.,.)}$, which is a function, characterizing the observational semantic, that ensures the very natural property that an environment $\mathcal{E}$ does not have a power of perception greater than an environment $\mathcal{E}||\mathcal{B}$ for an arbitrary automaton $\mathcal{B}$ ($\leq_\varepsilon^{S,f}$ is then a precongruence for parallel composition $||$). We exhibit two classic perception functions, the trace function and the projection on the environment itself. Finally, we have defined (see definition 107) what it means for two dynamic systems $X_\mathcal{A}$ and $X_\mathcal{B}$ to differ only on the fact that $X_\mathcal{A}$ dynamically creates and destroys $\mathcal{A}$ instead of $\mathcal{B}$ as $X_\mathcal{B}$ does. In that case, we say that $X_\mathcal{A}$ and $X_\mathcal{B}$ are corresponding w.r.t. $\mathcal{A}$,$\mathcal{B}$, noted $X_\mathcal{A} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_\mathcal{B}$. We stated following theorem (see theorem 7): (i) $X_\mathcal{A} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_\mathcal{B}$ and (ii) $\mathcal{A} \leq_0^{S_o,\mathtt{p}} \mathcal{B}$ implies $X_\mathcal{A} \leq_0^{S_o,\mathtt{p}} X_\mathcal{B}$, where $\mathtt{p}$ is the function that maps each execution to its projection on the environment and S_o represents the set of schedulers that do not take into account the exclusive past lives of sub-automata before their last creation to trigger the next action. Hence, if $\mathcal{A}$ implements $\mathcal{B}$ and $X_\mathcal{A}$ and $X_\mathcal{B}$ only differ on that $X_\mathcal{A}$ dynamically creates and destroys $\mathcal{A}$ instead of $\mathcal{B}$ as $X_\mathcal{B}$ does, then $X_\mathcal{A}$ implements $X_\mathcal{B}$. These results guarantee the soundness of modular design based on an observational semantic.

In next chapter, we prove this last mentioned theorem 7.

Chapter 4

Monotonicty of dynamic creation/destruction of PSIOA with implementation

This chapter contains the proof of the monotonicity of dynamic creation/destruction with implementation relationship.

Overview

4.1. Executions-matching . . . 92
4.1.1. PSIOA executions-matching and semantic equivalence . . . 92
4.1.2. PCA-matching execution . . . 104
4.1.3. Digest . . . 108
4.2. Projection . . . 109
4.2.1. Projection on Configurations . . . 109
4.2.2. $\mathcal{A}$-fairness assumption, motivated by our definition of PCA deprived of an internal PSIOA: $X \setminus \{\mathcal{A}\}$ . . . 113
4.2.3. $Y = X \setminus \{\mathcal{A}\}$ is a PCA if X is $\mathcal{A}$-fair . . . 115
4.3. Reconstruction . . . 119
4.3.1. Simpleton wrapper : $\tilde{A}^{sw}$ . . . 119
4.3.2. Partial-compatibility of $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})$ and $\tilde{\mathcal{A}}^{sw}$ . . . 119
4.3.3. Executions-matching from X to $(X \setminus \{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw}$ . . . 125
4.3.4. Composition and projection are commutative . . . 128
4.4. PCA corresponding w.r.t. PSIOA $\mathcal{A}$, $\mathcal{B}$ . . . 131
4.5. Top/Down corresponding classes . . . 138
4.5.1. Creation-oblivious scheduler . . . 138
4.5.2. Tools: proxy function, creation-explicitness, classes . . . 140
4.5.3. Homomorphism between simple classes . . . 141
4.5.4. Decomposition, pasting-friendly functions . . . 144
4.5.5. Creation oblivious scheduler applied to decomposition . . . 148
4.5.6. Monotonicity of implementation . . . 150
4.6. Task schedule . . . 154
4.6.1. Task-schedulers for PIOA . . . 154
4.6.2. Discussion on adaptation of task-structure in dynamic setting . . . 154
4.6.3. task-schedule for dynamic setting . . . 158
4.6.4. Why a task-scheduler is not creation-oblivious ? . . . 159
4.7. Monotonicity of Tenacious Task-Implementation . . . 160
4.7.1. schedule notations . . . 161
4.7.2. Tenacious Implementation . . . 162
4.7.3. sub-classes according to a task schedule . . . 162
4.7.4. Preparation before monotonicity . . . 170

4.7.5. Tenacious Implementation Monotonicity 171
4.8. Summary 176

In previous chapter 3 we have introduced a framework to reason about dynamic probabilistic systems, with two important theorems of stability and composability (theorem 2 page 72 and theorem 5 page 80). Moreover, we have stated the theorem 7 of monotonicity of dynamic creation/destruction of PSIOA with $\leq_0^{S_o,\mathfrak{p}} X_{\mathcal{B}}$, i.e. (i) $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$ and (ii) $\mathcal{A} \leq_0^{S_o,\mathfrak{p}} \mathcal{B}$ implies $X_{\mathcal{A}} \leq_0^{S_o,\mathfrak{p}} X_{\mathcal{B}}$. Informally, if $\mathcal{A}$ implements $\mathcal{B}$ and $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ only differ on that $X_{\mathcal{A}}$ dynamically creates and destroys $\mathcal{A}$ instead of $\mathcal{B}$ as $X_{\mathcal{B}}$ does, then $X_{\mathcal{A}}$ implements $X_{\mathcal{B}}$.

In this chapter, we give a detailed modular proof of theorem 7 of monotonicty of dynamic creation/destruction with implementation relationship. In order to get a first intuition about the content of this chapter 4, we encourage taking a look at the figures of the Subsection 1.3.3 of thesis overview that we recall here:

First, we define in section 4.1, the notion of executions-matching (see figure 1.5) to capture the idea that two automata have the same "comportment" [1] along with some corresponding executions. Basically an execution-matching from a PSIOA $\mathcal{A}$ to a PSIOA $\mathcal{B}$ is a morphism $f^{ex} : Execs'_{\mathcal{A}} \rightarrow Execs(\mathcal{B})$ where $Execs'_{\mathcal{A}} \subseteq Execs(\mathcal{A})$. This morphism preserves some properties along the pair of matched executions: signature, transition, ... in such a way that for every pair $(\alpha, \alpha') \in Execs(\mathcal{A}) \times Execs(\mathcal{B})$ s.t. $\alpha' = f^{ex}(\alpha)$, $\epsilon_{\sigma}(\alpha) = \epsilon_{\sigma'}(\alpha')$ for every pair of scheduler (σ, σ') (so-called *alter ego*) that are "very similar" in the sense they take into account only the "structure" of the argument to return a sub-probability distribution, i.e. $\alpha' = f^{ex}(\alpha)$ implies $\sigma(\alpha) = \sigma'(\alpha')$. When the executions-matching is a bijection function from $Execs(\mathcal{A})$ to $Execs(\mathcal{B})$, we say $\mathcal{A}$ and $\mathcal{B}$ are semantically-equivalent (they differ only syntactically).

Second, we define in section 4.2 the notion of a PCA $X_{\mathcal{A}}$ deprived of a PSIOA $\mathcal{A}$ noted $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})$. Such an automaton corresponds to the intuition of a similar automaton where $\mathcal{A}$ is systematically removed from the configuration of the original PCA (see figure 1.6a and 1.6b).

Thereafter we show in section 4.3 that under technical minor assumptions $X_{\mathcal{A}} \setminus \{\mathcal{A}\}$ and $\tilde{\mathcal{A}}^{sw}$ are composable where $\tilde{\mathcal{A}}^{sw}$ and $\mathcal{A}$ are semantically equivalent in the sense loosely introduced in the section 1.3.3. In fact $\tilde{\mathcal{A}}^{sw}$ is the simpleton wrapper of $\mathcal{A}$, which is a PCA that only owns $\mathcal{A}$ in its attached configuration (see figure 1.7). Let us note that if $\mathcal{A}$ implements $\mathcal{B}$, then $\tilde{\mathcal{A}}^{sw}$ implements $\tilde{\mathcal{B}}^{sw}$.

Then we show that there is an (incomplete) execution-matching from $X_{\mathcal{A}}$ to $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw}$ (see figure 1.8). The domain of this executions-matching is the set of executions where $\mathcal{A}$ is not (re-)created.

After this, we always try to reduce any reasoning on $X_{\mathcal{A}}$ (resp. $X_{\mathcal{B}}$) on a reasoning on $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw}$ (resp. $(X_{\mathcal{B}} \setminus \{\mathcal{B}\})||\tilde{\mathcal{B}}^{sw}$).

We show in section 4.4 that, under certain reasonable technical assumptions (captured in the definition of corresponding PCA w.r.t. $\mathcal{A}$, $\mathcal{B}$), $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})$ and $(X_{\mathcal{B}} \setminus \{\mathcal{B}\})$ are semantically-equivalent. We can note Y an arbitrary PCA semantically-equivalent to $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})$ and $(X_{\mathcal{B}} \setminus \{\mathcal{B}\})$.

Finally, a reasoning on $\mathcal{E}||X_{\mathcal{A}}$ (resp. $\mathcal{E}||X_{\mathcal{B}}$) can be reduced to a reasoning on $\mathcal{E}'||\tilde{\mathcal{A}}^{sw}$ (resp. $\mathcal{E}'||\tilde{\mathcal{B}}^{sw}$) with $\mathcal{E}' = \mathcal{E}||Y$. Since $\tilde{\mathcal{A}}^{sw}$ implements $\tilde{\mathcal{B}}^{sw}$, we have already some results on $\mathcal{E}'||\tilde{\mathcal{A}}^{sw}$ and $\mathcal{E}'||\tilde{\mathcal{B}}^{sw}$ and so on $\mathcal{E}||X_{\mathcal{A}}$ and $\mathcal{E}||X_{\mathcal{B}}$.

However, these results are a priori valid only for the subset of executions without the creation of neither $\mathcal{A}$ nor $\mathcal{B}$ before the very last action). This reduction is represented in figures 1.9a and 1.9b.

The reduction, which is roughly described in figures 1.9a and 1.9b, holds only for executions fragments that do not create the automata $\mathcal{A}$ and $\mathcal{B}$ after their destruction (or at the very last action). Some technical precautions have to be taken to be allowed to paste these fragments together to finally say that $\mathcal{A}$ implements $\mathcal{B}$ implies $X_{\mathcal{A}}$ implements $X_{\mathcal{B}}$. In fact, such a pasting is generally not possible for a perfect-

1. The term is deliberately chosen to be vague.

information online scheduler, that can trigger an action taking into account the whole history. This observation motivated us to introduce the class S_o of schedulers that outputs (randomly) a transition without taking into account the triggered internal actions and the visited states of a sub-automaton $\mathcal{A}$ preceding its last destruction. We prove the monotonicity of dynamic creation/destruction with $\leq_0^{S_o,\mathtt{P}}$ in Section 4.5. This result is encapsulated in Theorem 23 page 150: if (1) $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$, and (2) $\mathcal{A} \leq_0^{S_o,\mathtt{P}} \mathcal{B}$, then (3) $X_{\mathcal{A}} \leq_0^{S_o,\mathtt{P}} X_{\mathcal{B}}$. The figure 1.10 represents the issue with schedulers absent from S_o. We also discuss a special case of task-schedulers [CCK+18], a very user-friendly class of fully-offline schedulers, that allows straightforward oblivious fair scheduling. Surprisingly, a naive adaptation of task-schedulers, to the dynamic paradigm is not a subset of S_o. We show how a more sophisticated adaptation of task-based implementation relationship, called tenacious implementation, noted $\leq_0^{ten}$, can allow obtaining monotonicity of dynamic creation/destruction of automata. This result is stated in theorem 24 page 174: if (1) $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$, and (2) $\mathcal{A} \leq_0^{ten} \mathcal{B}$, then (3) $X_{\mathcal{A}} \leq_0^{ten} X_{\mathcal{B}}$.

4.1. Executions-matching

In this section, we introduce some tools to formalize the fact that two automata have the same comportment for the same scheduler. This section is composed of two sub-sections on PSIOA executions-matching and PCA executions-matching. Basically, an executions-matching execution from an automaton $\mathcal{A}$ to another automaton $\mathcal{B}$ is a morphism f^{ex} from $Execs(\mathcal{A})$ to $Execs(\mathcal{B})$ that is structure-preserving. In the remaining, we will often use an executions-matching to show that a pair of executions $(\alpha, \pi = f^{ex}(\alpha)) \in Execs(\mathcal{A}) \times Execs(\mathcal{B})$ have the same probability $\epsilon_\sigma(\alpha) = \epsilon_{\sigma'}(\pi)$ under a pair of so-called *alter-ego* schedulers $(\sigma, \sigma') \in schedulers(\mathcal{A}) \times schedulers(\mathcal{B})$ that have corresponding comportment after corresponding executions fragment $(\alpha', \pi' = f^{ex}(\alpha')) \in Frags^*(\mathcal{A}) \times Frags^*(\mathcal{B})$.

4.1.1. PSIOA executions-matching and semantic equivalence

This first subsection is about PSIOA executions-matching.

matching execution An executions-matching need a states-matching (see definition 112) and a transitions-matching (see definition 114) to be defined itself.

First, we define states-matching, which is a mapping between states of 2 automata that preserves starting states and signatures modulo a possible hiding operation.

Definition 112 (states-matching)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA, let $Q'_{\mathcal{A}} \subset Q_{\mathcal{A}}$ and let $f : Q'_{\mathcal{A}} \to Q_{\mathcal{B}}$ be a mapping that verifies:*

- *Starting state preservation: If $\bar{q}_{\mathcal{A}} \in Q'_{\mathcal{A}}$ then $f(\bar{q}_{\mathcal{A}}) = \bar{q}_{\mathcal{B}}$*
- *Signature preservation (modulo an hiding operation): $\forall (q, q') \in Q'_{\mathcal{A}} \times Q_{\mathcal{B}}$, s.t. $q' = f(q)$, $sig(\mathcal{A})(q) = hide(sig(\mathcal{B})(q'), h(q'))$ with $h(q') \subseteq out(\mathcal{B})(q')$ (resp. with $h(q') = \emptyset$, that is $sig(\mathcal{A})(q) = sig(\mathcal{B})(q')$).*

then we say that f is a weak *(resp.* strong*)* states-matching *from $\mathcal{A}$ to $\mathcal{B}$. If $Q'_{\mathcal{A}} = Q_{\mathcal{A}}$, then we say that f is a* complete *(weak or strong) states-matching from $\mathcal{A}$ to $\mathcal{B}$.*

Before being able to define transitions-matching, some requirements have to be ensured. A set of transition that would ensure these requirements would be called *eligible to transitions-matching.*

Definition 113 (transitions set eligible to transitions matching)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA,let $Q'_{\mathcal{A}} \subset Q_{\mathcal{A}}$ and let $f : Q'_{\mathcal{A}} \to Q_{\mathcal{B}}$ be a states-matching from $\mathcal{A}$ to $\mathcal{B}$. Let $D'_{\mathcal{A}} \subseteq D_{\mathcal{A}}$ be a subset of transition. If $D'_{\mathcal{A}}$ verifies that $\forall (q, a, \eta_{(\mathcal{A},q,a)}) \in D'_{\mathcal{A}}$:*

- *Matched states preservation: $q \in Q'_{\mathcal{A}}$ and*
- *Equitable corresponding distribution: $\forall q'' \in supp(\eta_{(\mathcal{A},q,a)}), q'' \in Q'_{\mathcal{A}}$ and $\eta_{(\mathcal{A},q,a)} \xleftrightarrow{f} \eta_{(\mathcal{B},f(q),a)}$*

then we say that $D'_{\mathcal{A}}$ is eligible to transitions-matching domain from f*. We omit to mention the states-matching f when this is clear in the context.*

Now, we are able to define a transitions-matching, which is a property-preserving mapping from a set of transitions $D'_{\mathcal{A}} \subseteq D_{\mathcal{A}}$ to another set of transitions $D'_{\mathcal{B}} \subseteq D_{\mathcal{B}}$.

Definition 114 (transitions-matching)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA, let $Q'_{\mathcal{A}} \subset Q_{\mathcal{A}}$ and let $f : Q'_{\mathcal{A}} \to Q_{\mathcal{B}}$ be a states-matching from $\mathcal{A}$ to $\mathcal{B}$. Let $D'_{\mathcal{A}} \subseteq D_{\mathcal{A}}$ be a subset of transition eligible to transitions-matching domain from f.*

We define the transitions-matching (f, f^{tr}) from $\mathcal{A}$ to $\mathcal{B}$ induced by the states-matching f and the subset of transition $D'_{\mathcal{A}}$ *s.t.* $f^{tr} : D'_{\mathcal{A}} \to D_{\mathcal{B}}$ *is defined by* $f^{tr}((q, a, \eta_{(\mathcal{A},q,a)})) = (f(q), a, \eta_{(\mathcal{B},f(q),a)})$ *. If f is complete and $D'_{\mathcal{A}} = D_{\mathcal{A}}$, (f, f^{tr}) is said to be a complete transitions-matching. If f is weak (resp. strong) (f, f^{tr}) is said to be a weak (resp. strong) transitions-matching. If f is clear in the context, with a slight abuse of notation, we say that f^{tr} is a transitions-matching.*

The function f^{tr} needs to verify some constraints imposed by f, but if the set $D'_{\mathcal{A}}$ of concerned transitions is correctly-chosen to ensure the 2 properties of definition 113, then such a transitions-matching is unique.

Now, we can easily define an executions-matching with a transitions-matching, which is a property-preserving mapping from a set of execution fragments $F'_{\mathcal{A}} \subseteq Frags(\mathcal{A})$ to another set of execution fragments $F'_{\mathcal{B}} \subseteq Frags(\mathcal{B})$.

Definition 115 (executions-matching)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA. Let (f, f^{tr}) be a transitions-matching from $\mathcal{A}$ to $\mathcal{B}$. Let $F'_{\mathcal{A}} = \{\alpha \triangleq q^0 a^1 q^1 ... a^n q^n ... \in Frags(\mathcal{A}) | \forall i \in [0 : |\alpha|-1], (q^i, a^{i+1}, \eta_{(\mathcal{A},q^i,a^{i+1})}) \in dom(f^{tr})\}$. Let $f^{ex} : F'_{\mathcal{A}} \to Frags(\mathcal{B})$, built from (f, f^{tr}) s.t. $\forall \alpha = q^0_{\mathcal{A}} a^1 q^1_{\mathcal{A}} ... a^n q^n_{\mathcal{A}} ... \in F'_{\mathcal{A}}$, $f^{ex}(\alpha) = f(q^0_{\mathcal{A}}) a^1 f(q^1_{\mathcal{A}}) ... a^n f(q^n_{\mathcal{A}}) ...$*

We say that (f, f^{tr}, f^{ex}) is an executions-matching *from $\mathcal{A}$ to $\mathcal{B}$. Furthermore, if (f, f^{tr}) is complete and $F'_{\mathcal{A}} = Frags(\mathcal{A})$, (f, f^{tr}, f^{ex}) is said to be a complete executions-matching. If (f, f^{tr}) is weak (resp. strong) (f, f^{tr}, f^{ex}) is said to be a weak (resp. strong) executions-matching. When (f, f^{tr}) is clear in the context, with a slight abuse of notation, we say that f^{ex} is an executions-matching.*

The function f^{ex} is completely defined by (f, f^{tr}), hence we call (f, f^{tr}, f^{ex}) the executions-matching induced by the transition matching (f, f^{tr}) or the executions-matching induced by the states-matching f and the subset of transitions $dom(f^{tr})$.

The construction of f^{ex} allows us to see two executions mapped by an executions-mapping as a sequence of pairs of transitions mapped by the attached transitions-matching. This result is formalised in next lemma 13.

Lemma 13 (executions-matching seen as a sequence of transitions-matchings)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA. Let (f, f^{tr}, f^{ex}) be an executions-matching from $\mathcal{A}$ to $\mathcal{B}$. Let $\alpha = q^0_{\mathcal{A}} a^1 q^1_{\mathcal{A}} ... a^n q^n_{\mathcal{A}} ... \in dom(f^{ex})$ and $\pi = f^{ex}(\alpha) = q^0_{\mathcal{B}} a^1 q^1_{\mathcal{B}} ... a^n q^n_{\mathcal{B}} ... = f(q^0_{\mathcal{A}}) a^1 f(q^1_{\mathcal{A}}) ... a^n f(q^n_{\mathcal{A}}) ....$ Then for every $i \in [0 : |\alpha| - 1]$, $(q^i_{\mathcal{B}}, a^{i+1}, \eta_{(\mathcal{B},q^i_{\mathcal{B}},a^{i+1})}) = f^{tr}((q^i_{\mathcal{A}}, a^{i+1}, \eta_{(\mathcal{A},q^i_{\mathcal{A}},a^i)}))$*

Proof. First, matched states preservation and action preservation are ensured by construction. By definition, for every $i \in [0 : |\alpha|-1]$, $(q^i_{\mathcal{A}}, a^{i+1}, \eta_{(\mathcal{A},q^i_{\mathcal{A}},a^{i+1})}) \in dom(f^{tr})$. We note $tr^i_{\mathcal{B}} \triangleq f^{tr}((q^i_{\mathcal{A}}, a^{i+1}, \eta_{(\mathcal{A},q^i_{\mathcal{A}},a^{i+1})}))$. By definition, $tr^i_{\mathcal{B}}$ is of the form $(f(q^i_{\mathcal{A}}), a^{i+1}, \eta)$. But a transition of this form is unique, which means $tr^i_{\mathcal{B}} = (f(q^i_{\mathcal{A}}), a^{i+1}, \eta_{(\mathcal{B},f(q^i_{\mathcal{A}}),a^{i+1})})$ which ends the proof. □

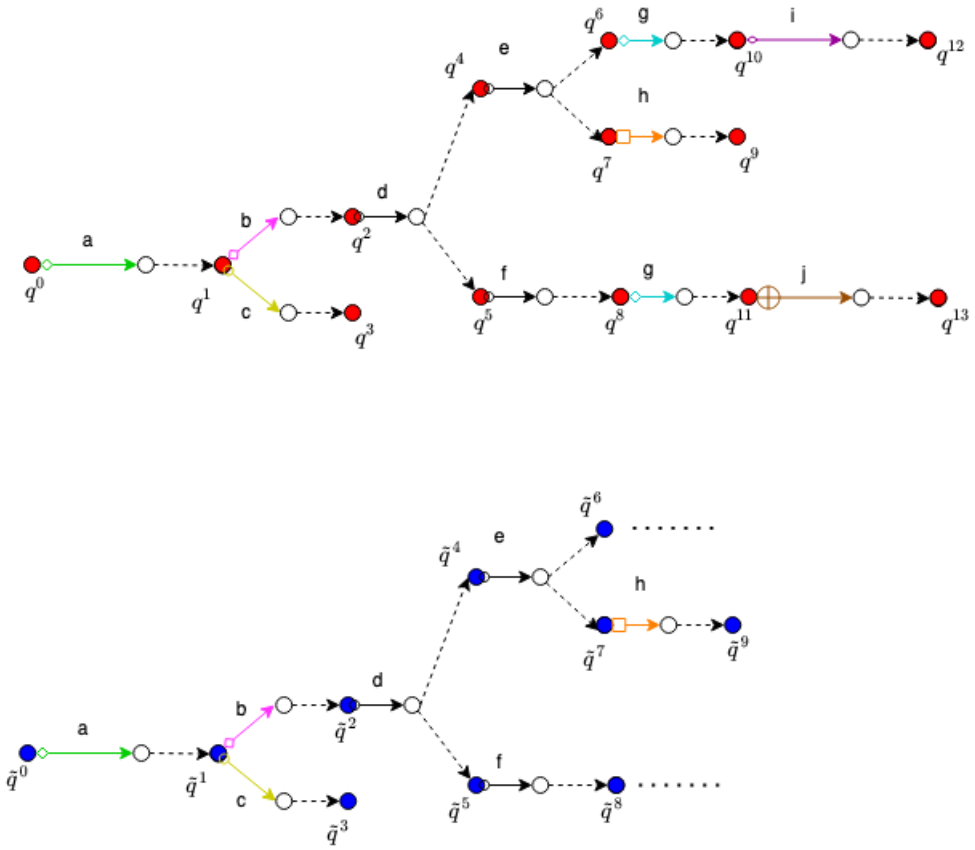


Figure 4.1. – Executions-matching

Here we define the states-matching $f : Q'_{\mathcal{A}} \to Q_{\mathcal{B}}$ with $Q'_{\mathcal{A}} = \{q^0, q^1, ..., q^9\} \subsetneq Q_{\mathcal{A}}$, s.t. $\forall k \in [1, 9], f(q^k) = \tilde{q}^k$, and $D'_{\mathcal{A}} = \{(q^0, a, \eta_{(\mathcal{A},q^0,a)}), (q^1, b, \eta_{(\mathcal{A},q^1,b)}), (q^1, c, \eta_{(\mathcal{A},q^1,c)}), (q^2, d, \eta_{(\mathcal{A},q^2,d)}), (q^4, e, \eta_{(\mathcal{A},q^4,e)}), (q^5, f, \eta_{(\mathcal{A},q^5,f)}), (q^7, h, \eta_{(\mathcal{A},q^7,h)})\}$. We can define the executions-matching (f, f^{tr}, f^{ex}) induced by f and $D'_{\mathcal{A}}$.

Now we overload the definition of executions-matching to be able to state the main result of this paragraph i.e. theorem 8.

Definition 116 (executions-matching overload: pre-execution-distribution)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA. Let (f, f^{tr}, f^{ex}) be an executions-matching from $\mathcal{A}$ to $\mathcal{B}$. Let $(\mu, \mu') \in Disc(Frags(\mathcal{A})) \times Disc(Frags(\mathcal{B}))$ s.t. $\mu \overset{f^{ex}}{\leftrightarrow} \mu'$. Then we say that (f, f^{tr}, f^{ex}) is an* executions-matching *from $(\mathcal{A}, \mu)$ to $(\mathcal{B}, \mu')$.*

In practice, we will often use executions-matching from $(\mathcal{A}, \delta_{\bar{q}_{\mathcal{A}}})$ to $(\mathcal{B}, \delta_{\bar{q}_{\mathcal{B}}})$.

Continued executions-matching Motivated by PSIOA creation that would break the states-matching from a PCA $X_{\mathcal{A}}$ to the PCA $Z_{\mathcal{A}} \triangleq (X \setminus \{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw}$ defined in section 4.3, we introduce the notion of continuation of executions-matching. This continuation will allow us to keep a matching for one

additional step (after a new creation in practice), which will be convenient for copy-pasting operations over execution fragments.

Definition 117 (Continued executions-matching)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA. Let (f, f^{tr}, f^{ex}) be an executions-matching from $\mathcal{A}$ to $\mathcal{B}$ with $dom(f) \triangleq Q'_{\mathcal{A}} \subset Q_{\mathcal{A}}$ and $dom(f^{tr}) \triangleq D'_{\mathcal{A}} \subset D_{\mathcal{A}}$.*

Let $f^+ : Q''_{\mathcal{A}} \to Q_{\mathcal{B}}$ with $Q''_{\mathcal{A}} \subset Q_{\mathcal{A}}$. Let $D''_{\mathcal{A}} \subset D_{\mathcal{A}}$ be a subset of transitions verifying for every $(q, a, \eta_{(\mathcal{A},q,a)}) \in D''_{\mathcal{A}} \setminus D'_{\mathcal{A}}$:

- *Matched states preservation: $q \in Q'_{\mathcal{A}}$*
- *Extension of equitable corresponding distribution: $\forall q'' \in supp(\eta_{(\mathcal{A},q,a)}), q'' \in Q''_{\mathcal{A}}$ and $\eta_{(\mathcal{A},q,a)} \overset{f^+}{\longleftrightarrow} \eta_{(\mathcal{B},f(q),a)}$.*

We define the $(f^+, D''_{\mathcal{A}})$-continuation of f^{tr} *as the function $f^{tr,+} : D'_{\mathcal{A}} \cup D''_{\mathcal{A}} \to D_{\mathcal{B}}$ s.t. $\forall (q, a, \eta_{(\mathcal{A},q,a)}) \in D'_{\mathcal{A}} \cup D''_{\mathcal{A}}$, $f^{tr,+}((q, a, \eta_{(\mathcal{A},q,a)})) = (f(q), a, \eta_{(\mathcal{B},f(q),a)})$.*

Let $F''_{\mathcal{A}} = dom(f^{ex}) \cup \{\alpha^\frown qaq' \in Execs^*(\mathcal{A}) | \alpha \in dom(f^{ex}) \wedge (q, a, \eta_{(\mathcal{A},q,a))} \in D''_{\mathcal{A}}\}$. We define the $(f^{tr,+})$-continuation of f^{ex} *as the function $f^{ex,+} : F''_{\mathcal{A}} \to Frags(\mathcal{B})$ s.t. $\forall \alpha \in dom(f^{ex}), f^{ex,+}(\alpha) = f^{ex}(\alpha)$ and $\forall \alpha' = \alpha^\frown q, a, q' \in F''_{\mathcal{A}} \setminus dom(f^{ex})$, $f^{ex,+}(\alpha') = f^{ex}(\alpha)^\frown f(q), a, f^+(q')$.*

Then, we say that $((f, f^+), f^{tr,+}, f^{ex,+})$ is the $(f^+, D''_{\mathcal{A}})$-continuation *of (f, f^{tr}, f^{ex}) which is a* continuation *of (f, f^{tr}, f^{ex}) and a* continued executions-matching from $\mathcal{A}$ to. $\mathcal{B}$.

Moreover, if $(\mu, \mu') \in Disc(Frags(\mathcal{A})) \times Disc(Frags(\mathcal{B}))$ s.t. $\mu \overset{f^{ex,+}}{\longleftrightarrow} \mu'$, then we say that $((f, f^+), f^{tr,+}, f^{ex,+})$ is a continued executions-matching *from $(\mathcal{A}, \mu)$ to $(\mathcal{B}, \mu')$.*

From executions-matching to probabilistic distribution preservation We want to states that a (potentially-continued) executions-matching preserves measure of probability of the corresponding executions.

To do so, we define alter egos schedulers to a certain executions-matching. Such pair of schedulers are very similar in the sense that their outputs depends only on the semantic structure of the input, preserved by the executions-matching.

Definition 118 ((f, f^{tr}, f^{ex})-alter egos schedulers)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA. Let (f, f^{tr}, f^{ex}) be an executions-matching from $\mathcal{A}$ to $\mathcal{B}$. Let $(\tilde{\sigma}, \sigma) \in schedulers(\mathcal{A}) \times schedulers(\mathcal{B})$. We say that $(\tilde{\sigma}, \sigma)$ are (f, f^{tr}, f^{ex})*-alter egos *(or f^{ex}*-alter egos*) if, and only if, for every $(\tilde{\alpha}, \alpha) \in Frags^*(\mathcal{A}) \times Frags^*(\mathcal{B})$ s.t. $\alpha = f^{ex}(\tilde{\alpha})$ (which means $\widehat{sig}(\mathcal{A})(\tilde{q}) = \widehat{sig}(\mathcal{B})(q) \triangleq sig$ with $\tilde{q} = lstate(\tilde{\alpha})$ and $q = lstate(\alpha)$ by signature preservation property of the associated states-matching), $\forall a \in sig, \tilde{\sigma}(\tilde{\alpha})((\tilde{q}, a, \eta_{(\mathcal{A},\tilde{q},a)})) = \sigma(\alpha)((q, a, \eta_{(\mathcal{B},q,a)}))$.*

Let us remark that the previous definition implies that the probability of halting after corresponding executions fragments $(\tilde{\alpha}, \alpha)$ is also the same.

Now we are ready to states an intuitive result that will be often used in the remaining.

Theorem 8 (Executions-matching preserves general probabilistic distribution)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA. Let $(\tilde{\mu}, \mu) \in Disc(Frags(\mathcal{A})) \times Disc(Frags(\mathcal{B}))$. Let (f, f^{tr}, f^{ex}) be an executions-matching from $(\mathcal{A}, \tilde{\mu})$ to $(\mathcal{B}, \mu)$. Let $(\tilde{\sigma}, \sigma) \in schedulers(\mathcal{A}) \times schedulers(\mathcal{B})$, s.t. $(\tilde{\sigma}, \sigma)$ are (f, f^{tr}, f^{ex})-alter egos. Let $(\tilde{\alpha}, \alpha) \in Frags^*(\mathcal{A}) \times Frags^*(\mathcal{B})$ s.t. $\alpha = f^{ex}(\tilde{\alpha})$.*

Then $\epsilon_{\tilde{\sigma},\tilde{\mu}}(C_{\tilde{\alpha}}) = \epsilon_{\sigma,\mu}(C_\alpha)$ and $\epsilon_{\tilde{\sigma},\tilde{\mu}}(\tilde{\alpha}) = \epsilon_{\sigma,\mu}(\alpha)$.

Proof. First, by definition 116 of executions-matching, f^{ex} is a bijection from $supp(\tilde{\mu})$ to $supp(\mu)$ where $\forall \tilde{\alpha}_o \in supp(\tilde{\mu}), \mu(f^{ex}(\tilde{\alpha}_o)) = \tilde{\mu}(\tilde{\alpha}_o)$ (*). Second, by definition 82 of measure generated by a

scheduler, $\epsilon_{\sigma,\mu}(C_{\alpha'}) = \Sigma_{\alpha_o \in supp(\mu)} \mu(\alpha_o) \cdot \epsilon_{\sigma,\alpha_o}(C_{\alpha'})$ and $\epsilon_{\tilde{\sigma},\tilde{\mu}}(C_{\tilde{\alpha}'}) = \Sigma_{\tilde{\alpha}_o \in supp(\tilde{\mu})} \tilde{\mu}(\tilde{\alpha}_o) \cdot \epsilon_{\tilde{\sigma},\tilde{\alpha}_o}(C_{\tilde{\alpha}'})$ (**). Hence, by combining (*) and (**), we only need to show that for every $(\tilde{\alpha}_o, \alpha_o) \in supp(\tilde{\mu}) \times supp(\mu)$ with $f^{ex}(\tilde{\alpha}_o) = \alpha_o$, for every $(\tilde{\alpha}', \alpha') \in Frags^*(\mathcal{A}) \times Frags^*(\mathcal{B})$ with $f^{ex}(\tilde{\alpha}') = \alpha'$, we have $\epsilon_{\sigma,\alpha_o}(C_{\alpha'}) = \epsilon_{\tilde{\sigma},\tilde{\alpha}_o}(C_{\tilde{\alpha}'})$ that we show by induction on the size $s = |\tilde{\alpha}| = |\alpha|$. We fix $(\tilde{\alpha}_o, \alpha_o) \in supp(\tilde{\mu}) \times supp(\mu)$ with $f^{ex}(\tilde{\alpha}_o) = \alpha_o$.

Basis: $s = 0$

Let $\tilde{\alpha}' = \tilde{q}' \in Frags^*(\mathcal{A})$, $\alpha' = q' \in Frags^*(\mathcal{B})$ with $\alpha' = f^{ex}(\tilde{\alpha}')$. We have $|\tilde{\alpha}'| = |\alpha'| = 0$. By definition 82 of measure generated by a scheduler,

$$\epsilon_{\tilde{\sigma},\tilde{\alpha}_o}(C_{\tilde{\alpha}'}) =: \begin{cases} 0 & \text{if both } \tilde{\alpha}' \not\leq \tilde{\alpha}_o \text{ and } \tilde{\alpha}_o \not\leq \tilde{\alpha}' \\ 1 & \text{if } \tilde{\alpha}' \leq \tilde{\alpha}_o \\ \epsilon_{\tilde{\sigma},\tilde{\alpha}_o}(C_{\tilde{\alpha}}) \cdot \tilde{\sigma}(\tilde{\alpha})(\eta_{(\mathcal{A},\tilde{q},a)}) \cdot \eta_{(\mathcal{A},\tilde{q},a)}(\tilde{q}') & \text{if } \tilde{\alpha}_o \leq \tilde{\alpha} \text{ and } \tilde{\alpha}' = \tilde{\alpha}^\frown \tilde{q} a \tilde{q}' \end{cases} \text{ and}$$

$$\epsilon_{\sigma,\alpha_o}(C_{\alpha'}) =: \begin{cases} 0 & \text{if both } \alpha' \not\leq \alpha_o \text{ and } \alpha_o \not\leq \alpha' \\ 1 & \text{if } \alpha' \leq \alpha_o \\ \epsilon_{\sigma,\alpha_o}(C_{\alpha}) \cdot \sigma(\alpha)(\eta_{(\mathcal{B},q,a)}) \cdot \eta_{(\mathcal{B},q,a)}(q') & \text{if } \alpha_o \leq \alpha \text{ and } \alpha' = \alpha^\frown q a q' \end{cases}$$

Since $|\tilde{\alpha}'| = |\alpha'| = 0$ the third case is never met. The second case can be written: $\tilde{\alpha}' \leq \tilde{\alpha}_o$ (resp. $\alpha' \leq \alpha_o$) iff $fstate(\tilde{\alpha}_o) = \tilde{q}'$ (resp. $fstate(\alpha_o) = q'$). Hence, for every $(\tilde{\alpha}_o, \alpha_o)$ s.t. $f^{ex}(\tilde{\alpha}_o) = \alpha_o$, $\epsilon_{\tilde{\sigma},\tilde{\alpha}_o}(C_{\tilde{\alpha}'}) = \epsilon_{\sigma,\alpha_o}(C'_\alpha)$ which ends the basis.

Induction: We assume the result to be true up to size s and we show it implies the result is true for size $s+1$. Let $(\tilde{\alpha}', \tilde{\alpha}, \alpha', \alpha) \in Frags^*(\mathcal{A})^2 \times Frags^*(\mathcal{B})^2$ with $\tilde{\alpha}' = \tilde{\alpha}^\frown \tilde{q} a \tilde{q}'$ and $\alpha' = \alpha^\frown q a q'$ s.t. $\alpha' = f^{ex}(\tilde{\alpha}')$ with $|\tilde{\alpha}'| = |\alpha'| = s+1$. We want to show that $\epsilon_{\tilde{\sigma},\tilde{\mu}}(C_{\tilde{\alpha}'}) = \epsilon_{\sigma,\mu}(C_{\alpha'})$. By definition 82 of measure generated by a scheduler,

$$\epsilon_{\tilde{\sigma},\tilde{\alpha}_o}(C_{\tilde{\alpha}'}) =: \begin{cases} 0 & \text{if both } \tilde{\alpha}' \not\leq \tilde{\alpha}_o \text{ and } \tilde{\alpha}_o \not\leq \tilde{\alpha}' \\ 1 & \text{if } \tilde{\alpha}' \leq \tilde{\alpha}_o \\ \epsilon_{\tilde{\sigma},\tilde{\alpha}_o}(C_{\tilde{\alpha}}) \cdot \tilde{\sigma}(\tilde{\alpha})(\eta_{(\mathcal{A},\tilde{q},a)}) \cdot \eta_{(\mathcal{A},\tilde{q},a)}(\tilde{q}') & \text{if } \tilde{\alpha}_o \leq \tilde{\alpha} \text{ and } \tilde{\alpha}' = \tilde{\alpha}^\frown \tilde{q} a \tilde{q}' \end{cases} \text{ and}$$

$$\epsilon_{\sigma,\alpha_o}(C_{\alpha'}) =: \begin{cases} 0 & \text{if both } \alpha' \not\leq \alpha_o \text{ and } \alpha_o \not\leq \alpha' \\ 1 & \text{if } \alpha' \leq \alpha_o \\ \epsilon_{\sigma,\alpha_o}(C_{\alpha}) \cdot \sigma(\alpha)(\eta_{(\mathcal{B},q,a)}) \cdot \eta_{(\mathcal{B},q,a)}(q') & \text{if } \alpha_o \leq \alpha \text{ and } \alpha' = \alpha^\frown q a q' \end{cases}$$

Again, the executions-matching implies that i) both $\tilde{\alpha}' \not\leq \tilde{\alpha}_o$ and $\tilde{\alpha}_o \not\leq \tilde{\alpha}' \iff$ both $\alpha' \not\leq \alpha_o$ and $\alpha_o \not\leq \alpha'$, ii) $\tilde{\alpha} \leq \tilde{\alpha}_o \iff \alpha \leq \alpha_o$ and iii) $\tilde{\alpha}_o \leq \tilde{\alpha} \iff \alpha_o \leq \alpha$. Moreover, by induction assumption $\epsilon_{\tilde{\sigma},\tilde{\alpha}_o}(C_{\tilde{\alpha}}) = \epsilon_{\sigma,\alpha_o}(C_\alpha)$. Hence we only need to show that $\tilde{\sigma}(\tilde{\alpha})(\eta_{(\mathcal{A},\tilde{q},a)}) \cdot \eta_{(\mathcal{A},\tilde{q},a)}(\tilde{q}') = \sigma(\alpha)(\eta_{(\mathcal{B},q,a)}) \cdot \eta_{(\mathcal{B},q,a)}(q')$ (***). By definition of alter-ego schedulers, $\tilde{\sigma}(\tilde{\alpha})(\eta_{(\mathcal{A},\tilde{q},a)}) = \sigma(\alpha)(\eta_{(\mathcal{B},q,a)})$ (j). By definition of executions-matching, $\eta_{(\mathcal{A},\tilde{q},a)}(\tilde{q}') = \eta_{(\mathcal{B},q,a)}(q')$ (jj). Thus (j) and (jj) implies (***) which allows us to terminate the induction to obtain $\epsilon_{\tilde{\sigma},\tilde{\alpha}_o}(C_{\tilde{\alpha}'}) = \epsilon_{\sigma,\alpha_o}(C_{\alpha'})$.

Finally, let $sig = \widehat{sig}(\mathcal{A})(lstate(\tilde{\alpha}')) = \widehat{sig}(\mathcal{A})(lstate(\alpha'))$, then $\epsilon_{\tilde{\sigma},\tilde{\alpha}_o}(\tilde{\alpha}') = \epsilon_{\tilde{\sigma},\tilde{\alpha}_o}(C_{\tilde{\alpha}'}) \cdot (1 - \Sigma_{a \in sig} \tilde{\sigma}(\tilde{\alpha}')(a)) = \epsilon_{\sigma,\alpha_o}(C_{\alpha'}) \cdot (1 - \Sigma_{a \in sig} \sigma(\alpha')(a)) = \epsilon_{\sigma,\alpha_o}(\alpha')$, which ends the proof.

□

We restate the previous theorem with continued executions-matching.

Theorem 9 (Continued executions-matching preserves general probabilistic distribution)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA. Let $(\tilde{\mu}, \mu) \in Disc(Frags(\mathcal{A})) \times Disc(Frags(\mathcal{B}))$. Let (f, f^{tr}, f^{ex}) be an executions-matching from $(\mathcal{A}, \tilde{\mu})$ to $(\mathcal{B}, \mu)$. Let $((f, f^+), f^{tr,+}, f^{ex,+})$ be a continuation of (f, f^{tr}, f^{ex}). Let $(\tilde{\sigma}, \sigma) \in schedulers(\mathcal{A}) \times schedulers(\mathcal{B})$, s.t. $(\tilde{\sigma}, \sigma)$ are (f, f^{tr}, f^{ex})*-alter egos. *Let $(\tilde{\alpha}, \alpha) \in Frags^*(\mathcal{A}) \times Frags^*(\mathcal{B})$ s.t. $\alpha = f^{ex,+}(\tilde{\alpha})$. Then $\epsilon_{\tilde{\sigma},\tilde{\mu}}(C_{\tilde{\alpha}}) = \epsilon_{\sigma,\mu}(C_\alpha)$.*

Proof. The proof is exactly the same than the one for theorem 8 □

Before dealing with composability of executions-matching, we prove two results about injectivity and surjectivity of executions-matching in next lemma 14 and 15.

Lemma 14 (Injectivity of executions-matching)**.** *Let (f, f^{tr}, f^{ex}) be an executions-matching from $\mathcal{A}$ to $\mathcal{B}$ and $((f, f^{+}), f^{tr,+}, f^{ex,+})$ a continuation of (f, f^{tr}, f^{ex}).*

Let $\tilde{f}^{ex,+} : F''_{\mathcal{A}} \subseteq dom(f^{ex,+}) \to \tilde{F}_{\mathcal{B}} \subseteq range(f^{ex,+})$. Let $\tilde{f} : Q''_{\mathcal{A}} \subseteq dom(f) \to Q_{\mathcal{B}}$ be the restriction of f on a set $Q''_{\mathcal{A}} \subseteq dom(f)$.

1. *If i) $\forall \alpha \in F''_{\mathcal{A}}$, $fstate(\alpha) \in Q''_{\mathcal{A}}$ and ii) $\tilde{f}$ is injective, then $\tilde{f}^{ex,+}$ is injective.*
2. *(Corollary) if $F''_{\mathcal{A}} \subseteq Execs(\mathcal{A})$, $f^{ex,+}$ is injective.*

Proof. 1. By induction on the size k of the prefix: Basis: By i) $fstate(\alpha), fstate(\alpha') \in Q''_{\mathcal{A}}$, by construction of $f^{ex,+}$, $f(fstate(\alpha)) = f(fstate(\alpha')) = fstate(\pi)$ and by ii) $fstate(\alpha) = fstate(\alpha')$ Induction. We assume the injectivity of $\tilde{f}^{ex,+}$ to be true for execution on size k and we show this is also true for size $k+1$. Let $\pi = s^0 b^1 s^1 ... s^k b^{k+1} s^{k+1} \in F''_{\mathcal{B}}$ Let $\alpha = q^0 a^1 q^1 ... q^k a^{k+1} q^{k+1}, \alpha' = q'^0 a'^1 q'^1 ... q'^k a'^{k+1} q'^{k+1} \in F''_{\mathcal{A}}$ s.t. $f(\alpha) = f(\alpha') = \pi$. By construction of $f^{ex,+}$, $\forall i \in [1, n]$, $b^i = a^i = a'^i$. By construction of $f^{ex,+}$, $f^{ex,+}(q'^0 a'^1 q'^1 ... q'^k) = f^{ex,+}(q^0 a^1 q^1 ... q^k) = s^0 a^1 s^1 ... s^k$. By induction assumption $q'^0 a'^1 q'^1 ... q'^k) = q^0 a^1 q^1 ... q^k$. By definition of execution, $s^{k+1} \in supp(\eta_{(\mathcal{B}, s^k, a^{k+1})})$. By equitable corresponding distribution, If $\eta_{(\mathcal{A}, q^k, a^{k+1})} \in dom(f^{tr})$, the restriction of f, $\tilde{f} : supp(\eta_{(\mathcal{A}, q^k, a^{k+1})}) \to supp(\eta_{(\mathcal{B}, s^k, a^{k+1})})$ is bijective and $\eta_{(\mathcal{A}, q^k, a^{k+1})} \in dom(f^{tr,+}) \setminus dom(f^{tr})$, the restriction of f^+, $\tilde{f}^+ : supp(\eta_{(\mathcal{A}, q^k, a^{k+1})}) \to supp(\eta_{(\mathcal{B}, s^k, a^{k+1})})$ is bijective so $q^{k+1} = q'^{k+1}$ which ends the proof.

2. We have $|start(\mathcal{A})| = 1$. Hence the restriction of f on $start(\mathcal{A})$ is necessarily injective (ii). Let $\alpha \in Execs(\mathcal{A})$. By definition of execution, $fstate(\alpha) \in start(\mathcal{A})$ (i). All the requirements of lemma 14, first item are met, which ends the proof.

□

Lemma 15 (Surjectivity property preserved by continuation)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA. Let (f, f^{tr}, f^{ex}) be an executions-matching from $\mathcal{A}$ to $\mathcal{B}$. Let $((f, f^{+}), f^{tr,+}, f^{ex,+})$ be the $(f^{+}, D''_{\mathcal{A}})$-continuation of (f, f^{tr}, f^{ex}) (where by definition $D''_{\mathcal{A}} \setminus dom(f^{tr})$ respect the properties of matched states preservation and extension of equitable corresponding distribution from definition 117). If the restriction $\tilde{f}^{ex} : E'_{\mathcal{A}} \subseteq Execs(\mathcal{A}) \to \tilde{E}_{\mathcal{B}} \subseteq Execs(\mathcal{B})$ is surjective, then $\tilde{f}^{ex,+} : E'^{,+}_{\mathcal{A}} = \{\alpha' = \alpha^\frown q_{\mathcal{A}}, a, q'_{\mathcal{A}} \in Execs(\mathcal{A}) | \alpha \in E_{\mathcal{A}}, (q_{\mathcal{A}}, a, \eta_{\mathcal{A}, q_{\mathcal{A}}, a}) \in dom(f^{tr,+})\} \to \tilde{E}^{+}_{\mathcal{B}} = \{\pi' = \pi^\frown q_{\mathcal{B}}, a, q'_{\mathcal{B}} \in Execs(\mathcal{B}) | \pi \in \tilde{E}_{\mathcal{B}}, \exists \alpha \in (f^{ex})^{-1}(\pi) \cap E'_{\mathcal{A}}, q_{\mathcal{A}} = lstate(\alpha), (q_{\mathcal{A}}, a, \eta_{\mathcal{A}, q_{\mathcal{A}}, a}) \in dom(f^{tr,+})\}$ is surjective.*

Proof. Let $\pi' \in \tilde{E}_{\mathcal{B}}$. We have $\pi' = \pi^\frown q_{\mathcal{B}}, a, q'_{\mathcal{B}} \in Execs(\mathcal{B})$ s.t. $\pi \in \tilde{E}_{\mathcal{B}}$ and $\exists \alpha \in (f^{ex})^{-1}(\pi) \cap E'_{\mathcal{A}}, q_{\mathcal{A}} = lstate(\alpha)$ and $(q_{\mathcal{A}}, a, \eta_{(\mathcal{A}, q_{\mathcal{A}}, a)}) \in dom(f^{tr,+})$. By $(q_{\mathcal{A}}, a, \eta_{\mathcal{A}, q_{\mathcal{A}}, a}) \in dom(f^{tr,+})$, if i) $(q_{\mathcal{A}}, a, \eta_{\mathcal{A}, q_{\mathcal{A}}, a}) \in dom(f^{tr,+}) \setminus dom(f^{tr})$ $\eta_{\mathcal{A}, q_{\mathcal{A}}, a} \overset{f^+}{\longleftrightarrow} \eta_{\mathcal{B}, q_{\mathcal{B}}, a}$ and if ii) $(q_{\mathcal{A}}, a, \eta_{\mathcal{A}, q_{\mathcal{A}}, a}) \in dom(f^{tr})$ $\eta_{\mathcal{A}, q_{\mathcal{A}}, a} \overset{f}{\longleftrightarrow} \eta_{\mathcal{B}, q_{\mathcal{B}}, a}$. In both cases, it exists $q'_{\mathcal{A}} \in supp(\eta_{\mathcal{A}, q_{\mathcal{A}}, a})$ s.t. $f^{ex,+}(\alpha' = \alpha^\frown q_{\mathcal{A}}, a, q'_{\mathcal{A}}) = \pi'$ with $\alpha' \in E'^{,+}_{\mathcal{A}}$.

□

We finish this paragraph with the concept of semantic equivalence that describes a pair of PSIOA that differ only syntactically.

Definition 119 (semantic equivalence)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA. We say that $\mathcal{A}$ and $\mathcal{B}$ are semantically-equivalent if it exists $f : Execs(\mathcal{A}) \to Execs(\mathcal{B})$ which is a complete bijective executions-matching from $\mathcal{A}$ to $\mathcal{B}$.*

Composability of executions-matching relationship Now we are looking for composability of executions-matching. First we define natural extension of notions presented in previous paragraph for the automaton obtained after composition with another automaton $\mathcal{E}$.

Definition 120 ($\mathcal{E}$-extension)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA. Let $\mathcal{E}$ be partially-compatible with both $\mathcal{A}$ and $\mathcal{B}$.*

1. *Let $Q'_{\mathcal{A}} \subset Q_{\mathcal{A}}$. We call* $\mathcal{E}$-extension *of $Q'_{\mathcal{A}}$ the set of states $Q'_{\mathcal{A}||\mathcal{E}} = \{q \in Q_{\mathcal{A}||\mathcal{E}} | q \restriction \mathcal{A} \in Q'_{\mathcal{A}}\}$*
2. *Let $f : Q'_{\mathcal{A}} \subset Q_{\mathcal{A}} \to Q_{\mathcal{B}}$. We call $\mathcal{E}$-extension of f the function $g : Q'_{\mathcal{A}||\mathcal{E}} \to Q_{\mathcal{B}} \times Q_{\mathcal{E}}$ s.t. $\forall (q_{\mathcal{A}}, q_{\mathcal{E}}) \in Q'_{\mathcal{A}||\mathcal{E}}$, $g((q_{\mathcal{A}}, q_{\mathcal{E}})) = (f(q_{\mathcal{A}}), q_{\mathcal{E}}))$*
3. *Let $D'_{\mathcal{A}} \subset D_{\mathcal{A}}$ a subset of transitions. We call $\mathcal{E}$-extension of $D'_{\mathcal{A}}$ the set $D'_{\mathcal{A}||\mathcal{E}} = \{((q_{\mathcal{A}}, q_{\mathcal{E}}), a, \eta_{((\mathcal{A},\mathcal{E}),(q_{\mathcal{A}},q_{\mathcal{E}}),a)}) \in D_{\mathcal{A}||\mathcal{E}} | q_{\mathcal{A}} \in Q'_{\mathcal{A}} \wedge [(q_{\mathcal{A}}, a, \eta_{(\mathcal{A},q_{\mathcal{A}},a)}) \in D'_{\mathcal{A}} \vee\ a \notin enabled(\mathcal{A})(q_{\mathcal{A}})]\}$.*

Now, we can start with the composability of states-matching.

Lemma 16 (Composability of states-matching)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA. Let $\mathcal{E}$ be partially-compatible with $\mathcal{A}$ and $\mathcal{B}$. Let $f : Q'_{\mathcal{A}} \subset Q_{\mathcal{A}} \to Q_{\mathcal{B}}$ be a states-matching. Let g be the $\mathcal{E}$-extension of f.*

If $range(g) \subset Q_{\mathcal{B}||\mathcal{E}}$, then g is a states-matching from $\mathcal{A}||\mathcal{E}$ to $\mathcal{B}||\mathcal{E}$.

Proof.
- Starting state preservation: if $(\bar{q}_{\mathcal{A}}, \bar{q}_{\mathcal{E}}) \in Q_{\mathcal{A}||\mathcal{E}}$ then $\bar{q}_{\mathcal{A}} \in Q'_{\mathcal{A}}$ which means $f(\bar{q}_{\mathcal{A}}) = \bar{q}_{\mathcal{B}}$, thus $g((\bar{q}_{\mathcal{A}}, \bar{q}_{\mathcal{E}})) = (\bar{q}_{\mathcal{B}}, \bar{q}_{\mathcal{E}})$.
- Signature preservation (modulo an hiding operation): $\forall ((q_{\mathcal{A}}, q_{\mathcal{E}}), (q_{\mathcal{B}}, q_{\mathcal{E}})) \in Q'_{\mathcal{A}||\mathcal{E}} \times Q_{\mathcal{B}||\mathcal{E}}$ with $(q_{\mathcal{B}}, q_{\mathcal{E}}) = g((q_{\mathcal{A}}, q_{\mathcal{E}}))$, we have $sig(\mathcal{A})(q_{\mathcal{A}}) = sig(\mathcal{B})(f(q_{\mathcal{A}})) = hide(sig(\mathcal{B})(q_{\mathcal{B}}), h(q_{\mathcal{B}}))$ with $h(q_{\mathcal{B}}) \subseteq out(\mathcal{B})(q_{\mathcal{B}})$. Since $\mathcal{A}$ and $\mathcal{E}$ are partially-compatible, $sig(\mathcal{A})(q_{\mathcal{A}}) = hide(sig(\mathcal{B})(q_{\mathcal{B}}), h(q_{\mathcal{B}}))$ is compatible with $sig(\mathcal{E})(q_{\mathcal{E}})$ which means a fortiori $sig(\mathcal{B})(q_{\mathcal{B}})$ is compatible with $sig(\mathcal{E})(q_{\mathcal{E}})$. Namely $\forall act \in h(q_{\mathcal{B}}), act \notin in(\mathcal{E})(q_{\mathcal{E}})$.

  Hence $sig((\mathcal{A}, \mathcal{E}))((q_{\mathcal{A}}), q_{\mathcal{E}})) = hide(sig((\mathcal{B}, \mathcal{E}))((q_{\mathcal{B}}, q_{\mathcal{E}})), h'((q_{\mathcal{B}}, q_{\mathcal{E}}))$ with $h'((q_{\mathcal{B}}, q_{\mathcal{E}})) = h(q_{\mathcal{B}}) \subseteq out(\mathcal{B})(q_{\mathcal{B}}) \subseteq out(\mathcal{B}||\mathcal{E})((q_{\mathcal{B}}, q_{\mathcal{E}}))$ which ends the proof.

□

The composability of states-matching is ensured under the condition $range(g) \subset Q_{\mathcal{B}||\mathcal{E}}$ where g is the $\mathcal{E}$-extension of the original states-matching $f : Q'_{\mathcal{A}} \subseteq Q_{\mathcal{A}} \to Q_{\mathcal{B}}$. In next lemma, we give a sufficient condition to ensure $range(g) \subset Q_{\mathcal{B}||\mathcal{E}}$. This is the one that we will use in practice.

Definition 121 (reachable-by and states of execution (recall))**.** *Let $\mathcal{A}$ be a PSIOA or a PCA. Let $E'_{\mathcal{A}} \subseteq Execs(\mathcal{A})$. We note reachable-by$(E'_{\mathcal{A}}) = \{q \in Q_{\mathcal{A}} | \exists \alpha \in E'_{\mathcal{A}}, lstate(\alpha) = q\}$. Let $\alpha = q^0, a^1, q^1, ...a^n, q^n, ...$ We note $states(\alpha) = \bigcup_{i \in |\alpha|} q^i$.*

Lemma 17 (A sufficient condition to obtain $range(g) \subset Q_{\mathcal{B}||\mathcal{E}}$)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA. Let $\mathcal{E}$ be partially-compatible with both $\mathcal{A}$ and $\mathcal{B}$. Let $f : Q'_{\mathcal{A}} \subset Q_{\mathcal{A}} \to Q_{\mathcal{B}}$ be a states-matching. Let $Q'_{\mathcal{A}||\mathcal{E}}$ be the $\mathcal{E}$-extension of $Q'_{\mathcal{A}}$.*

Let $Q''_{\mathcal{A}||\mathcal{E}} \subset Q'_{\mathcal{A}||\mathcal{E}}$ the set of states reachable by an execution that counts only states in $Q'_{\mathcal{A}||\mathcal{E}}$, i.e.

- $E''_{\mathcal{A}||\mathcal{E}} = \{\alpha \in Execs(\mathcal{A}||\mathcal{E}) | states(\alpha) \subseteq Q'_{\mathcal{A}||\mathcal{E}}\}$
- $Q''_{\mathcal{A}||\mathcal{E}} = reachable\text{-}by(E''_{\mathcal{A}||\mathcal{E}})$

Let f'' the restriction of f to set $Q''_{\mathcal{A}} = \{q_{\mathcal{A}} = ((q_{\mathcal{A}}, q_{\mathcal{E}}) \upharpoonright \mathcal{A}) | (q_{\mathcal{A}}, q_{\mathcal{E}}) \in Q''_{\mathcal{A}||\mathcal{E}}\}$.

Then the $\mathcal{E}$-extension of f'', noted g'' verifies $range(g'') \subset Q_{\mathcal{B}||\mathcal{E}}$.

Proof. By induction on the minimum size of an execution $\tilde{\alpha} = q^0 a^1 ... q^n$ with $q^* = q^n, \forall i \in [0, n], q^i \in Q'_{\mathcal{A}||\mathcal{E}}$. Basis ($|\alpha| = 0 \implies \alpha = \bar{q}_{\mathcal{A}}$): we consider $q^* = \bar{q}_{\mathcal{A}}$. We have $g((\bar{q}_{\mathcal{A}}, \bar{q}_{\mathcal{E}})) = (f(\bar{q}_{\mathcal{A}}), \bar{q}_{\mathcal{E}}) = (\bar{q}_{\mathcal{B}}, \bar{q}_{\mathcal{E}}) \in Q_{\mathcal{B}||\mathcal{E}}$.

We assume this is true for $\tilde{\alpha}$ with $lstate(\tilde{\alpha}) = q$ and we show this is also true for $\tilde{\alpha}' = \tilde{\alpha}^\frown q a q'$. By induction hypothesis $q \in Q_{\mathcal{B}||\mathcal{E}}$. Since $q' \in Q_{\mathcal{A}||\mathcal{E}}$, $\mathcal{A}$ and $\mathcal{E}$ are compatible at state $(q'_{\mathcal{A}}, q'_{\mathcal{E}})$, that is $sig(\mathcal{A})(q'_{\mathcal{A}})$ and $sig(\mathcal{E})(q'_{\mathcal{E}})$ are compatible, which means that a fortiori, $(sig(\mathcal{B})(f''(q'_{\mathcal{A}}))$ and $sig(\mathcal{E})(q'_{\mathcal{E}})$ are compatible and so $\mathcal{B}$ and $\mathcal{E}$ are compatible at state $(f''(q'_{\mathcal{A}}), q'_{\mathcal{E}}) = g''(q')$. Hence $g''(q')$ is a reachable compatible state of $(\mathcal{B}, \mathcal{E})$ which means this is a state of $\mathcal{B}||\mathcal{E}$.

□

Now, we can continue with the composability of transitions-matching.

Lemma 18 (Composability of eligibility for transitions-matching)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA. Let $\mathcal{E}$ be partially-compatible with $\mathcal{A}$ and $\mathcal{B}$. Let $f : Q'_{\mathcal{A}} \subset Q_{\mathcal{A}} \to Q_{\mathcal{B}}$ be a states-matching and $D'_{\mathcal{A}}$ a subset of transitions eligible to transitions-matching domain from f. Let g be the $\mathcal{E}$-extension of f and $D'_{\mathcal{A}||\mathcal{E}}$ the $\mathcal{E}$-extension of $D_{\mathcal{A}}$.*

If $range(g) \subset Q_{\mathcal{B}||\mathcal{E}}$, then $D'_{\mathcal{A}||\mathcal{E}}$ is eligible to transitions-matching domain from g.

Proof. Let $((q_{\mathcal{A}}, q_{\mathcal{E}}), a, \eta_{((\mathcal{A},\mathcal{E}),(q_{\mathcal{A}},q_{\mathcal{E}}),a)}) \in D'_{\mathcal{A}||\mathcal{E}}$.

By definition, $q_{\mathcal{A}} \in Q'_{\mathcal{A}}$ which means $(q_{\mathcal{A}}, q_{\mathcal{E}}) \in Q'_{\mathcal{A}||\mathcal{E}}$, so the matched states preservation is ensured. We still need to ensure the equitable corresponding distribution.

- Let $(q''_{\mathcal{A}}, q''_{\mathcal{E}}) \in supp(\eta_{((\mathcal{A},\mathcal{E}),(q_{\mathcal{A}},q_{\mathcal{E}}),a)})$. If $a \in \widehat{sig}(\mathcal{A})(q_{\mathcal{A}})$, then $q''_{\mathcal{A}} \in supp(\eta_{(\mathcal{A},q_{\mathcal{A}},a)})$ which means $q''_{\mathcal{A}} \in Q'_{\mathcal{A}}$ and hence $(q''_{\mathcal{A}}, q''_{\mathcal{E}}) \in Q'_{\mathcal{A}||\mathcal{E}}$. If $a \notin \widehat{sig}(\mathcal{A})$, $\eta_{(\mathcal{A},q_{\mathcal{A}},a)} = \delta_{q_{\mathcal{A}}}$, which means $q''_{\mathcal{A}} = q_{\mathcal{A}} \in Q'_{\mathcal{A}}$ and hence $(q''_{\mathcal{A}}, q''_{\mathcal{E}}) \in Q'_{\mathcal{A}||\mathcal{E}}$. Thus for every $(q''_{\mathcal{A}}, q''_{\mathcal{E}}) \in supp(\eta_{((\mathcal{A},\mathcal{E}),(q_{\mathcal{A}},q_{\mathcal{E}}),a)})$, $(q''_{\mathcal{A}}, q''_{\mathcal{E}}) \in Q'_{\mathcal{A}||\mathcal{E}}$.
- $\eta_{((\mathcal{A},\mathcal{E}),(q_{\mathcal{A}},q_{\mathcal{E}}),a)}((q''_{\mathcal{A}}, q''_{\mathcal{E}})) = \eta_{(\mathcal{A},q_{\mathcal{A}},a)} \otimes \eta_{(\mathcal{E},q_{\mathcal{E}},a)}(q''_{\mathcal{A}}, q''_{\mathcal{E}}) = \eta_{(\mathcal{A},q_{\mathcal{A}},a)}(q''_{\mathcal{A}}) \cdot \eta_{(\mathcal{E},q_{\mathcal{E}},a)}(q''_{\mathcal{E}}) = \eta_{(\mathcal{B},f(q_{\mathcal{A}}),a)}(f(q''_{\mathcal{A}})) \cdot \eta_{(\mathcal{E},q_{\mathcal{E}},a)}(q''_{\mathcal{E}}) = \eta_{(\mathcal{B},f(q_{\mathcal{A}}),a)} \otimes \eta_{(\mathcal{E},q_{\mathcal{E}},a)}(f(q''_{\mathcal{A}}), q''_{\mathcal{E}}) = \eta_{((\mathcal{B},\mathcal{E}),g(q_{\mathcal{A}},q_{\mathcal{E}}),a)}(g(q''_{\mathcal{A}}, q''_{\mathcal{E}}))$ which ends the proof of equitable corresponding distribution.

□

Definition 122 ($\mathcal{E}$-extension of an executions-matching)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA. Let $\mathcal{E}$ be partially-compatible with both $\mathcal{A}$ and $\mathcal{B}$. Let (f, f^{tr}, f^{ex}) be an executions-matching from $\mathcal{A}$ to $\mathcal{B}$. Let g the $\mathcal{E}$-extension of f. If $range(g) \subset Q_{\mathcal{B}||\mathcal{E}}$, then*

1. *we call the $\mathcal{E}$-extension of f^{tr} the function $g^{tr} : (q, a, \eta_{(\mathcal{A}||\mathcal{E},q,a)}) \in D'_{\mathcal{A}||\mathcal{E}} \mapsto (g(q), a, \eta_{(\mathcal{B}||\mathcal{E},g(q),a)})$ where $D'_{\mathcal{A}||\mathcal{E}}$ is the $\mathcal{E}$-extension of the domain $dom(f^{tr})$ of f^{tr}.*
2. *we call the $\mathcal{E}$-extension of (f, f^{tr}, f^{ex}) the matching-execution (g, g^{tr}, g^{ex}) from $\mathcal{A}||\mathcal{E}$ to $\mathcal{B}||\mathcal{E}$ induced by g and $dom(g^{tr})$.*

Finally we can states the main result of this paragraph, i.e. theorem 10 of executions-matching composability.

Theorem 10 (Composability of executions-matching)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA. Let $\mathcal{E}$ be partially-compatible with both $\mathcal{A}$ and $\mathcal{B}$. Let (f, f^{tr}, f^{ex}) be an executions-matching from $\mathcal{A}$ to $\mathcal{B}$ where g represents the $\mathcal{E}$-extension of f. If $range(g) \subset Q_{\mathcal{B}||\mathcal{E}}$, then the $\mathcal{E}$-extension of (f, f^{tr}, f^{ex}) is a matching-execution (g, g^{tr}, g^{ex}) from $\mathcal{A}||\mathcal{E}$ to $\mathcal{B}||\mathcal{E}$ induced by g and $dom(g^{tr})$.*

Proof. We repeated the previous definition, while an executions-matching only need a states-matching g and a set $dom(g^{tr})$ of transitions eligible to transitions-matching domain from g which is provided by construction. □

Here we give some properties preserved by $\mathcal{E}$-extension of an executions-matching.

Lemma 19 (Some properties preserved by $\mathcal{E}$-extension of an executions-matching)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be PSIOA. Let (f, f^{tr}, f^{ex}) be an executions-matching from $\mathcal{A}$ to $\mathcal{B}$.*

1. *If f is bijective and f^{-1} is complete, then for every PSIOA $\mathcal{E}$ partially-compatible with $\mathcal{A}$, $\mathcal{E}$ is partially-compatible with $\mathcal{B}$.*
2. *Let $\mathcal{E}$ partially-compatible with both $\mathcal{A}$ and $\mathcal{B}$, let g be the $\mathcal{E}$-extension of f.*
   a) *If f is bijective and f^{-1} is complete, then $range(g) = Q_{\mathcal{B}||\mathcal{E}}$ and so we can talk about the $\mathcal{E}$-extension of (f, f^{tr}, f^{ex})*
   b) *If (f, f^{tr}) is a bijective complete transition-matching, (g, g^{tr}) is a bijective complete transition-matching. (And (f, f^{tr}, f^{ex}) and (g, g^{tr}, g^{ex}) are bijective complete executions-matching.)*
   c) *If f is strong, then g is strong*
3. *Let $\mathcal{E}$ partially-compatible with both $\mathcal{A}$ and $\mathcal{B}$, let g be the $\mathcal{E}$-extension of f. Let assume $range(g) \subseteq Q_{\mathcal{B}||\mathcal{E}}$. Let (g, g^{tr}, g^{ex}) be the $\mathcal{E}$-extension of (f, f^{tr}, f^{ex})*
   a) *If the restriction $\tilde{f}^{ex} : E'_{\mathcal{A}} \subseteq Execs(\mathcal{A}) \to \tilde{E}_{\mathcal{B}} \subseteq Execs(\mathcal{B})$ is surjective, then $\tilde{g}^{ex} : \{\alpha \in Execs(\mathcal{A}||\mathcal{E}) | \alpha \upharpoonright \mathcal{A} \in E'_{\mathcal{A}}\} \to \{\pi \in Execs(\mathcal{B}||\mathcal{E}) | \pi \upharpoonright \mathcal{B} \in \tilde{E}_{\mathcal{B}}\}$ is surjective*
   b) *If f is strong, g is strong.*

Proof. 1. We need to show that every pseudo-execution of $(\mathcal{B}, \mathcal{E})$ ends on a compatible state. Let $\pi = q^0 a^1 q^1 ... a^n q^n$ be a finite pseudo-execution of $(\mathcal{B}, \mathcal{E})$. We note $\alpha = (f^{-1}(q^0_{\mathcal{B}}), q^0_{\mathcal{E}}) a^1 (f^{-1}(q^1_{\mathcal{B}}), q^1_{\mathcal{E}}) ... a^n (f^{-1}(q^n_{\mathcal{B}}), q^n_{\mathcal{E}})$. The proof is in two steps.
First, we show by induction that $\alpha = (f^{-1}(q^0_{\mathcal{B}}), q^0_{\mathcal{E}}) a^1 (f^{-1}(q^1_{\mathcal{B}}), q^1_{\mathcal{E}}) ... a^n (f^{-1}(q^n_{\mathcal{B}}), q^n_{\mathcal{E}})$ is an execution of $\mathcal{A}||\mathcal{E}$.
Second, we deduce that it means $(f^{-1}(q^n_{\mathcal{B}}), q^n_{\mathcal{E}})$ is a compatible state of $(\mathcal{A}, \mathcal{E})$ which means that a fortiori, $(q^n_{\mathcal{B}}, q^n_{\mathcal{E}})$ is a compatible state of $(\mathcal{B}, \mathcal{E})$ which ends the proof.

- First, we show by induction that α is an execution of $\mathcal{A}||\mathcal{E}$. We have $(f^{-1}(\bar{q}_{\mathcal{B}}), \bar{q}_{\mathcal{E}}) = (\bar{q}_{\mathcal{A}}, \bar{q}_{\mathcal{E}})$ which ends the basis.
  Let assume $(f^{-1}(q^0_{\mathcal{B}}), q^0_{\mathcal{E}}) a^1 (f^{-1}(q^1_{\mathcal{B}}), q^1_{\mathcal{E}}) ... a^k (f^{-1}(q^k_{\mathcal{B}}), q^k_{\mathcal{E}})$ is an execution of $\mathcal{A}||\mathcal{E}$. Hence $(f^{-1}(q^k_{\mathcal{B}}), q^k_{\mathcal{E}})$ is a compatible state of $(\mathcal{A}, \mathcal{E})$ which means that a fortiori q^k is a compatible state of $(\mathcal{B}, \mathcal{E})$ because of signature preservation of f.
  For the same reason, $\widehat{sig}(\mathcal{B}||\mathcal{E})(q^k) = \widehat{sig}(\mathcal{A}, \mathcal{E})((f^{-1}(q^k_{\mathcal{B}}), q^k_{\mathcal{E}}))$, so $a^{k+1} \in \widehat{sig}(\mathcal{A}, \mathcal{E})((f^{-1}(q^k_{\mathcal{B}}), q^k_{\mathcal{E}}))$. Then we use the completeness of $(f^{-1}, (f^{tr})^{-1})$, to obtain the fact that either $\eta_{(\mathcal{B}, q^k_{\mathcal{B}}, a^{k+1})} \in dom((f^{tr})^{-1})$ or $a^{k+1} \notin \widehat{sig(\mathcal{B})}(q^k_{\mathcal{B}})$ (and we recall the convention that in this second case $\eta_{(\mathcal{B}, q^k_{\mathcal{B}}, a^{k+1})} = \delta_{q^k_{\mathcal{B}}}$). which means either $(f^{-1}(q^k_{\mathcal{B}}), a^{k+1}, \eta_{(\mathcal{A}, f^{-1}(q^k_{\mathcal{B}}), a^{k+1})})$ is a transition of $\mathcal{A}$ that ensures $\forall q'' \in supp(\eta_{(\mathcal{B}, q^k_{\mathcal{B}}, a^{k+1})}), f^{-1}(q'') \in supp(\eta_{(\mathcal{A}, f^{-1}(q^k_{\mathcal{B}}), a^{k+1})})$ or $a^{k+1} \notin \widehat{sig(\mathcal{A})}(f^{-1}(q^k_{\mathcal{B}}))$ (and we recall the convention that in this second case $\eta_{(\mathcal{A}, f^{-1}(q^k_{\mathcal{B}}), a^{k+1})} = \delta_{f^{-1}(q^k_{\mathcal{B}})}$). Thus for every $(q'', q''') \in supp(\eta_{(\mathcal{B}, \mathcal{E}), q^k, a^{k+1})})$, $(f^{-1}(q''), q''') = g^{-1}((q'', q''')) \in supp(\eta_{(\mathcal{A}, \mathcal{E}), g^{-1}(q^k), a^{k+1})})$ namely for $(q'', q''') = (q^{k+1}_{\mathcal{B}}, q^{k+1}_{\mathcal{E}})$. Hence, $(f^{-1}(q^{k+1}_{\mathcal{B}}), q^{k+1}_{\mathcal{E}})$ is reachable by $(\mathcal{A}, \mathcal{E})$ which means the alternating sequence $(f^{-1}(q^0_{\mathcal{B}}), q^0_{\mathcal{E}}) a^1 (f^{-1}(q^1_{\mathcal{B}}), q^1_{\mathcal{E}}) ... a^k (f^{-1}(q^k_{\mathcal{B}}), q^k_{\mathcal{E}}) a^k (f^{-1}(q^k_{\mathcal{B}}), q^k_{\mathcal{E}}) a^{k+1} (f^{-1}(q^{k+1}_{\mathcal{B}}), q^{k+1}_{\mathcal{E}})$ is an execution of $\mathcal{A}||\mathcal{E}$. Thus by induction α is an execution of $\mathcal{A}||\mathcal{E}$.

– Since $\mathcal{A}$ and $\mathcal{E}$ are partially-compatible $(f^{-1}(q^n_{\mathcal{B}}), q^n_{\mathcal{E}})$ is a state of $\mathcal{A}||\mathcal{E}$, so $(f^{-1}(q^n_{\mathcal{B}}), q^n_{\mathcal{E}})$ is a compatible state of $(\mathcal{A}, \mathcal{E})$ which means $(q^k_{\mathcal{B}}, q^k_{\mathcal{E}})$ is a fortiori a compatible state of $(\mathcal{B}, \mathcal{E})$. Hence every reachable state of $(\mathcal{B}, \mathcal{E})$ is compatible which means $\mathcal{B}$ and $\mathcal{E}$) are partially compatible which ends the proof.

2. a) – Let $(q^n_{\mathcal{B}}, q^n_{\mathcal{E}}) \in Q_{\mathcal{B}||\mathcal{E}}$. This state is reachable, so we note $\pi = (q^0_{\mathcal{B}}, q^0_{\mathcal{E}})a^1(q^1_{\mathcal{B}}, q^1_{\mathcal{E}})...a^n(q^n_{\mathcal{B}}, q^n_{\mathcal{E}})$ the execution of $\mathcal{B}||\mathcal{E}$. Thereafter, we note $\alpha = (f^{-1}(q^0_{\mathcal{B}}), q^0_{\mathcal{E}})a^1(f^{-1}(q^1_{\mathcal{B}}), q^1_{\mathcal{E}})...a^n(f^{-1}(q^n_{\mathcal{B}}), q^n_{\mathcal{E}})$. We can show by induction that α is an execution of $\mathcal{A}||\mathcal{E}$. The proof is exactly the same than in 1.

Hence α is an execution of $\mathcal{A}||\mathcal{E}$ which means $(f^{-1}(q^n_{\mathcal{B}}), q^n_{\mathcal{E}})$ is a state of $\mathcal{A}||\mathcal{E}$ and then $g((f^{-1}(q^n_{\mathcal{B}}), q^n_{\mathcal{E}})) = (q^n_{\mathcal{B}}, q^n_{\mathcal{E}})$ to finally prove that it exists q^* s.t. $g(q^*) = (q^n_{\mathcal{B}}, q^n_{\mathcal{E}})$ which means $states(\mathcal{B}||\mathcal{E}) \subseteq dom(g)$.

We can reuse the proof of 1. to show that if $q \in Q_{\mathcal{A}||\mathcal{E}}$, then $g(q) \in Q_{\mathcal{B}||\mathcal{E}}$ which means $dom(g) \subseteq Q_{\mathcal{B}||\mathcal{E}}$.

Hence $dom(g) = Q_{\mathcal{B}||\mathcal{E}}$.

– We can apply the previous lemma 18 to obtain the eligibility of $D_{\mathcal{A}||\mathcal{E}}$.

b) Let assume (f, f^{tr}) are bijective. The bijectivity of g is immediate $g(.,.) = (f(.), Id(.))$. The bijectivity of g^{tr} is also immediate since $g^{tr} : \eta_{(\mathcal{A},q_{\mathcal{A}},a)} \otimes \eta_{(\mathcal{E},q_{\mathcal{E}},a)} \to f^{tr}(\eta_{(\mathcal{A},q_{\mathcal{A}},a)}) \otimes \eta_{(\mathcal{E},q_{\mathcal{E}},a)}$ with f^{tr} bijective.

c) Immediate, since in this case $sig(\mathcal{A})(q_{\mathcal{A}}) = sig(\mathcal{B})(f(q_{\mathcal{A}}))$ implies $sig(\mathcal{A}||\mathcal{E})((q_{\mathcal{A}}, q_{\mathcal{E}})) = sig(\mathcal{B}||\mathcal{E})((f(q_{\mathcal{A}}), q_{\mathcal{E}}))$.

3. a) Let $\pi = ((q^0_{\mathcal{B}}, q^0_{\mathcal{E}}), a^1, (q^1_{\mathcal{B}}, q^1_{\mathcal{E}}), ..., a^n, (q^n_{\mathcal{B}}, q^n_{\mathcal{E}})) \in Execs(\mathcal{B}||\mathcal{E})$ with $\pi \upharpoonright \mathcal{B} = \hat{q}^0_{\mathcal{B}}, \hat{a}^1, \hat{q}^1_{\mathcal{B}}, ..., \hat{a}^m, \hat{q}^m_{\mathcal{B}} \in \tilde{E}_{\mathcal{B}}$, where the monotonic function $k : [0, n] \to [0, m]$, verifies $\forall i \in [0, n], k(i) \in [0, m], q^i_{\mathcal{B}} = \hat{q}^{k(i)}_{\mathcal{B}}$ By surjectivity of f^{ex} we have $\hat{\alpha} = \hat{q}^0_{\mathcal{A}}, \hat{a}^1, \hat{q}^1_{\mathcal{A}}, ..., \hat{a}^m, \hat{q}^m_{\mathcal{A}} \in E'_{\mathcal{A}}$ s.t. $f^{ex}(\hat{\alpha}) = \pi \upharpoonright \mathcal{B}$. We note $\alpha = (q^0_{\mathcal{A}}, q^0_{\mathcal{E}})a^1(q^1_{\mathcal{A}}, q^1_{\mathcal{E}})...a^n(q^n_{\mathcal{A}}, q^n_{\mathcal{E}})$ where $\forall i \in [0, n], q^i_{\mathcal{A}} = \hat{q}^{k(i)}_{\mathcal{A}}$. Hence, $\forall i \in [0, n], g((q^i_{\mathcal{A}}, q^i_{\mathcal{E}})) = (q^i_{\mathcal{B}}, q^i_{\mathcal{E}})$. Moreover, by signature preservation, $\forall i \in [0, n-1], a^{i+1} \in \widehat{sig}(\mathcal{A})(q^i_{\mathcal{A}}) \cup \widehat{sig}(\mathcal{E})(q^i_{\mathcal{E}})$. Furthermore, $\forall i \in [0, n-1]$. $(q^{i+1}_{\mathcal{A}}, q^{i+1}_{\mathcal{E}}) \in supp(\eta_{(\mathcal{A},q^i_{\mathcal{A}},a^i)} \otimes \eta_{(\mathcal{B},q^i_{\mathcal{B}},a^i)})$ since $(q^{i+1}_{\mathcal{B}}, q^{i+1}_{\mathcal{E}}) \in supp(\eta_{(\mathcal{B},q^i_{\mathcal{B}},a^i)} \otimes \eta_{(\mathcal{B},q^i_{\mathcal{B}},a^i)})$, $(q^i_{\mathcal{B}}, a^i, \eta_{(\mathcal{B},q^i_{\mathcal{B}},a^i)}) = f^{tr}(q^i_{\mathcal{A}}, a^i, \eta_{(\mathcal{A},q^i_{\mathcal{A}},a^i)})$ and $q^{i+1}_{\mathcal{B}} = f(q^{i+1}_{\mathcal{A}})$. Thus, $\alpha \in Execs(\mathcal{A}||\mathcal{E})$. Finally, by signature preservation of f, $\forall i \in [1, n] \widehat{sig}(\mathcal{A})(q_{\mathcal{A}}) = \widehat{sig}(\mathcal{B})(q_{\mathcal{B}})$, which lead us to $\alpha \upharpoonright \mathcal{A} = \hat{\alpha} \in E'_{\mathcal{A}}$. So for every $\pi \in Execs(\mathcal{B}||\mathcal{E})$ with $\pi \upharpoonright \mathcal{B} \in \tilde{E}_{\mathcal{B}}$, it exists $\alpha \in Execs(\mathcal{A}||\mathcal{E})$ with $\alpha \upharpoonright \mathcal{A} \in E'_{\mathcal{A}}$ which ends the proof.

b) Immediate by rules of composition of signature: $\forall (q_{\mathcal{A}}, q_{\mathcal{E}}) \in states(\mathcal{A}||\mathcal{E})$, $\forall (q_{\mathcal{B}}, q_{\mathcal{E}}) \in states(\mathcal{B}||\mathcal{E})$ if $sig(\mathcal{A})(q_{\mathcal{A}}) = sig(\mathcal{B})(q_{\mathcal{B}})$, then $sig(\mathcal{A}||\mathcal{E})(q_{\mathcal{A}}, q_{\mathcal{E}}) = sig(\mathcal{B}||\mathcal{E}))(q_{\mathcal{B}}, q_{\mathcal{E}})$.

□

We are ready to states the composability of semantic equivalence.

Theorem 11 (composability of semantic equivalence)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be PSIOA semantically-equivalent. Then for every PSIOA $\mathcal{E}$:*

– *$\mathcal{E}$ is partially-compatible with $\mathcal{A} \iff \mathcal{E}$ is partially-compatible with $\mathcal{B}$*

– *if $\mathcal{E}$ is partially-compatible with both $\mathcal{A}$ and $\mathcal{B}$, then $\mathcal{A}||\mathcal{E}$ and $\mathcal{B}||\mathcal{E}$ are semantically-equivalent PSIOA.*

Proof. – The first item ($\mathcal{E}$ is partially-compatible with $\mathcal{A} \iff \mathcal{E}$ is partially-compatible with $\mathcal{B}$) comes from lemma 19, first item.

– The second item (if $\mathcal{E}$ is partially-compatible with both $\mathcal{A}$ and $\mathcal{B}$, then $\mathcal{A}||\mathcal{E}$ and $\mathcal{B}||\mathcal{E}$ are semantically-equivalent PSIOA) comes from lemma 19, second item.

□

A weak complete bijective transition-matching implies a weak complete bijective executions-matching which means the two automata are completely sementically equivalent modulo some hiding operation that implies that some PSIOA are partially-compatible with one of the automaton and not with the other and that the traces are not necessarily the same ones.

composition of continuation of executions-matching Here we define $\mathcal{E}$-extension of continued executions-matching in the same way we defined $\mathcal{E}$-extension of executions-matching just before.

Definition 123 ($\mathcal{E}$-extension of continued executions-matching)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA. Let $\mathcal{E}$ be partially-compatible with both $\mathcal{A}$ and $\mathcal{B}$. Let (f, f^{tr}, f^{ex}) be an executions-matching from $\mathcal{A}$ to $\mathcal{B}$. Let $((f, f^+), f^{tr,+}, f^{ex,+})$ be the $(f^+, D''_{\mathcal{A}})$-continuation of (f, f^{tr}, f^{ex}) (where by definition $D''_{\mathcal{A}} \setminus dom(f^{tr})$ respect the properties of matched states preservation and extension of equitable corresponding distribution from definition 117). If the respective $\mathcal{E}$-extension of f and f^+, noted g and g^+, verifie $range(g) \cup range(g^+) \subseteq (\mathcal{B}||\mathcal{E})$, we define the $\mathcal{E}$-extension of $((f, f^+), f^{tr,+}, f^{ex,+})$ as $((g, g^+), g^{tr,+}, g^{ex,+})$, where*

– *(g, g^{tr}, g^{ex}) is the $\mathcal{E}$-extension of (f, f^{tr}, f^e)*
– *$g^{tr,+} : (q, a, \eta_{(\mathcal{A}||\mathcal{E}),q,a}) \in D''_{\mathcal{A}||\mathcal{E}} \mapsto (g(q), a, \eta_{(\mathcal{A}||\mathcal{E}),g(q),a})$ where $D''_{\mathcal{A}||\mathcal{E}}$ is the $\mathcal{E}$-extension of $dom(f^{tr,+})$*
– *$\forall \alpha' = \alpha^\frown q, a, q'$, with $\alpha' \in dom(g^{ex})$, if $(q, a, \eta_{(\mathcal{A}||\mathcal{E}),q,a}) \in dom(g^{tr})$ $g^{ex,+}(\alpha) = g^{ex}(\alpha)$ and if $(q, a, \eta_{(\mathcal{A}||\mathcal{E}),q,a}) \in dom(g^{tr,+}) \setminus dom(g^{tr})$ $g^{ex,+}(\alpha') = g^{ex}(\alpha)^\frown g(q), a, g^+(q)$*

Lemma 20 (Commutativity of continuation and extension)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA. Let $\mathcal{E}$ be partially-compatible with both $\mathcal{A}$ and $\mathcal{B}$. Let (f, f^{tr}, f^{ex}) be an executions-matching from $\mathcal{A}$ to $\mathcal{B}$. Let $((f, f^+), f^{tr,+}, f^{ex,+})$ be the $(f^+, D''_{\mathcal{A}})$-continuation of (f, f^{tr}, f^{ex}) (where by definition $D''_{\mathcal{A}}$ respect the properties of matched states preservation and extension of equitable corresponding distribution from definition 117). Let*

– *(g, g^{tr}, g^{ex}) be the $\mathcal{E}$-extension of (f, f^{tr}, f^e) verifying $range(g) \subseteq Q_{\mathcal{B}||\mathcal{E}}$,*
– *$D''^{,(c,e)}_{\mathcal{A}||\mathcal{E}}$ the $\mathcal{E}$-extension of $dom(f^{tr,+})$, i.e. $D''^{,(c,e)}_{\mathcal{A}||\mathcal{E}} = \{((q_{\mathcal{A}}, q_{\mathcal{E}}), a, \eta_{(\mathcal{A}||\mathcal{E},(q_{\mathcal{A}},q_{\mathcal{E}}),a)}) \in D_{\mathcal{A}||\mathcal{E}} | q_{\mathcal{A}} \in dom(f) \wedge [(q_{\mathcal{A}}, a, \eta_{(\mathcal{A},q_{\mathcal{A}},a)}) \in dom(f^{tr,+}) \vee a \notin \widehat{sig}(\mathcal{A})(q_{\mathcal{A}})]\}$.*
– *$g^+_{(c,e)}$ be the $\mathcal{E}$-extension of f^+*

Then

1. *$D''_{\mathcal{A}||\mathcal{E}} \setminus dom(g^{tr})$ verifies matched states preservation and extension of equitable corresponding distribution.*
2. *the $(g^+_{(c,e)}, (D''^{,(c,e)}_{\mathcal{A}||\mathcal{E}}))$-continuation of (g, g^{tr}, g^{ex}), noted $((g, g^+_{(c,e)}), g^{tr,+}_{(c,e)}, g^{ex,+}_{(c,e)})$ is equal to the $\mathcal{E}$-extension of $((f, f^+), f^{tr,+}, f^{ex,+})$, noted $((g, g^+_{(e,c)}), g^{tr,+}_{(e,c)}, g^{ex,+}_{(e,c)})$.*

We show that the operation of continuation and extension are in fact commutative.

Proof. We start by showing $D''^{,(c,e)}_{\mathcal{A}||\mathcal{E}} \setminus dom(g^{tr})$ verifies matched states preservation and extension of equitable corresponding distribution. By definition 117 of $\mathcal{E}$-extension, $D''^{,(c,e)}_{\mathcal{A}||\mathcal{E}} = \{((q_{\mathcal{A}}, q_{\mathcal{E}}), a, \eta_{(\mathcal{A}||\mathcal{E},(q_{\mathcal{A}},q_{\mathcal{E}}),a)}) \in D_{\mathcal{A}||\mathcal{E}} | q_{\mathcal{A}} \in dom(f) \wedge [(q_{\mathcal{A}}, a, \eta_{(\mathcal{A},q_{\mathcal{A}},a)}) \in dom(f^{tr,+}) \vee a \notin \widehat{sig}(\mathcal{A})(q_{\mathcal{A}})]\}$, while $dom(g^{tr}) =$

$\{((q_{\mathcal{A}}, q_{\mathcal{E}}), a, \eta_{(\mathcal{A}||\mathcal{E},(q_{\mathcal{A}},q_{\mathcal{E}}),a)}) \in D_{\mathcal{A}||\mathcal{E}} | q_{\mathcal{A}} \in dom(f) \wedge [(q_{\mathcal{A}}, a, \eta_{(\mathcal{A},q_{\mathcal{A}},a)}) \in dom(f^{tr}) \vee a \notin \widehat{sig}(\mathcal{A})(q_{\mathcal{A}})]\}$.

Thus $D''^{,(c,e)}_{\mathcal{A}||\mathcal{E}} \setminus dom(g^{tr}) = \{((q_{\mathcal{A}}, q_{\mathcal{E}}), a, \eta_{(\mathcal{A}||\mathcal{E},(q_{\mathcal{A}},q_{\mathcal{E}}),a)}) \in D_{\mathcal{A}||\mathcal{E}} | q_{\mathcal{A}} \in dom(f) \wedge [(q_{\mathcal{A}}, a, \eta_{(\mathcal{A},q_{\mathcal{A}},a)}) \in dom(f^{tr,+}) \setminus dom(f^{tr})]\}$ (*)

Let $tr = ((q_{\mathcal{A}}, q_{\mathcal{E}}), a, \eta_{(\mathcal{A}||\mathcal{E}),(q_{\mathcal{A}},q_{\mathcal{E}}),a)}) \in D''^{,(c,e)}_{\mathcal{A}||\mathcal{E}} \setminus dom(g^{tr})$, then

- Matched states preservation: By (*) $q_{\mathcal{A}} \in dom(f)$ which leads immediately to $(q_{\mathcal{A}}, q_{\mathcal{E}}) \in dom(g)$
- Extension of equitable corresponding distribution: $\forall (q''_{\mathcal{A}}, q''_{\mathcal{E}}) \in supp(\eta_{(\mathcal{A}||\mathcal{E},(q_{\mathcal{A}},q_{\mathcal{E}}),a)})$, $(q''_{\mathcal{A}}, q''_{\mathcal{E}}) \in supp(\eta_{(\mathcal{A}q_{\mathcal{A}},a)} \otimes \eta_{(\mathcal{E},q_{\mathcal{E}},a)})$ with $\eta_{(\mathcal{A}q_{\mathcal{A}},a)} \in dom(f^{tr,+}) \setminus dom(f^{tr})$ by (*) which means $q''_{\mathcal{A}} \in dom(f^{+})$ and $\eta_{(\mathcal{A}q_{\mathcal{A}},a)}(q''_{\mathcal{A}}) = \eta_{(\mathcal{B}f(q_{\mathcal{A}}),a)}(f^{+}(q''_{\mathcal{A}}))$ and so $(q''_{\mathcal{A}}, q''_{\mathcal{E}}) \in dom(g^{+})$ and $\eta_{(\mathcal{A},q_{\mathcal{A}},a)} \otimes \eta_{(\mathcal{E},q_{\mathcal{E}},a)}(q''_{\mathcal{A}}, q''_{\mathcal{E}}) = \eta_{(\mathcal{A},q_{\mathcal{A}},a)}(q''_{\mathcal{A}}) \cdot \eta_{(\mathcal{E},q_{\mathcal{E}},a)}(q''_{\mathcal{E}}) = \eta_{(\mathcal{B},f(q_{\mathcal{A}}),a)}(f^{+}(q''_{\mathcal{A}})) \cdot \eta_{(\mathcal{E},q_{\mathcal{E}},a)}(q''_{\mathcal{E}}) = \eta_{(\mathcal{B},f(q_{\mathcal{A}}),a)} \otimes \eta_{(\mathcal{E},q_{\mathcal{E}},a)}(f^{+}(q''_{\mathcal{A}}), q''_{\mathcal{E}}) = \eta_{(\mathcal{B}||\mathcal{E},g(q_{\mathcal{A}},q_{\mathcal{E}}),a)}(g^{+}(q''_{\mathcal{A}}, q''_{\mathcal{E}}))$

We have shown that $D''^{,(c,e)}_{\mathcal{A}||\mathcal{E}} \setminus dom(g^{tr})$ verifies matched states preservation and extension of equitable corresponding distribution.

Now, we show the second point.

- By definition 117 of continuation, $g^{+}_{(c,e)} = g^{+}_{(e,c)}$.
- We prove $dom(g^{tr,+}_{(c,e)}) = dom(g^{tr,+}_{(e,c)}) = D''^{,(c,e)}_{\mathcal{A}||\mathcal{E}}$. By definition 117 of continuation, $dom(g^{tr,+}_{(e,c)}) = dom(g^{tr}) \cup D''^{,(c,e)}_{\mathcal{A}||\mathcal{E}} = \{((q_{\mathcal{A}}, q_{\mathcal{E}}), a, \eta_{(\mathcal{A}||\mathcal{E},(q_{\mathcal{A}},q_{\mathcal{E}}),a)}) \in D_{\mathcal{A}||\mathcal{E}} | q_{\mathcal{A}} \in dom(f) \wedge [(q_{\mathcal{A}}, a, \eta_{(\mathcal{A},q_{\mathcal{A}},a)}) \in dom(f^{tr}) \vee a \notin \widehat{sig}(\mathcal{A})(q_{\mathcal{A}})]\} \cup \{((q_{\mathcal{A}}, q_{\mathcal{E}}), a, \eta_{(\mathcal{A}||\mathcal{E},(q_{\mathcal{A}},q_{\mathcal{E}}),a)}) \in D_{\mathcal{A}||\mathcal{E}} | q_{\mathcal{A}} \in dom(f) \wedge [(q_{\mathcal{A}}, a, \eta_{(\mathcal{A},q_{\mathcal{A}},a)}) \in dom(f^{tr,+}) \vee a \notin \widehat{sig}(\mathcal{A})(q_{\mathcal{A}})]\} = \{((q_{\mathcal{A}}, q_{\mathcal{E}}), a, \eta_{(\mathcal{A}||\mathcal{E},(q_{\mathcal{A}},q_{\mathcal{E}}),a)}) \in D_{\mathcal{A}||\mathcal{E}} | q_{\mathcal{A}} \in dom(f) \wedge [(q_{\mathcal{A}}, a, \eta_{(\mathcal{A},q_{\mathcal{A}},a)}) \in dom(f^{tr,+}) \vee a \notin \widehat{sig}(\mathcal{A})(q_{\mathcal{A}})]\} = D''^{,(c,e)}_{\mathcal{A}||\mathcal{E}}$.

  Parrallely, by definition 122 of $\mathcal{E}$-extension, $dom(g^{tr,+}_{(c,e)}) = \{((q_{\mathcal{A}}, q_{\mathcal{E}}), a, \eta_{(\mathcal{A}||\mathcal{E},(q_{\mathcal{A}},q_{\mathcal{E}}),a)}) \in D_{\mathcal{A}||\mathcal{E}} | q_{\mathcal{A}} \in dom(f) \wedge [(q_{\mathcal{A}}, a, \eta_{(\mathcal{A},q_{\mathcal{A}},a)}) \in dom(f^{tr,+}) \vee a \notin \widehat{sig}(\mathcal{A})(q_{\mathcal{A}})]\} = D''^{,(c,e)}_{\mathcal{A}||\mathcal{E}}$. Thus $dom(g^{tr,+}_{(c,e)}) = dom(g^{tr,+}_{(e,c)}) = D''^{,(c,e)}_{\mathcal{A}||\mathcal{E}}$.
- We prove $g^{tr,+}_{(c,e)} = g^{tr,+}_{(e,c)}$ Let $((q_{\mathcal{A}}, q_{\mathcal{E}}), a, \eta_{(\mathcal{A}||\mathcal{E},(q_{\mathcal{A}},q_{\mathcal{E}}),a)}) \in D''_{\mathcal{A}||\mathcal{E}}$.

  By definition 122 of $\mathcal{E}$-extension, $g^{tr,+}_{(c,e)}(((q_{\mathcal{A}}, q_{\mathcal{E}}), a, \eta_{(\mathcal{A}||\mathcal{E},(q_{\mathcal{A}},q_{\mathcal{E}}),a)})) = (g(q_{\mathcal{A}}, q_{\mathcal{E}}), a, \eta_{(\mathcal{A}||\mathcal{E},g(q_{\mathcal{A}},q_{\mathcal{E}}),a)}))$, while by definition 117 of continuation, $g^{tr,+}_{(e,c)}(((q_{\mathcal{A}}, q_{\mathcal{E}}), a, \eta_{(\mathcal{A}||\mathcal{E},(q_{\mathcal{A}},q_{\mathcal{E}}),a)})) = (g(q_{\mathcal{A}}, q_{\mathcal{E}}), a, \eta_{(\mathcal{A}||\mathcal{E},g(q_{\mathcal{A}},q_{\mathcal{E}}),a)}))$.
  We can remark that properties of equitable corresponding distribution are not conflicting since $dom(g^{tr,+}_{c,e}) \setminus dom(g^{tr}) = dom(g^{tr,+}_{e,c}) \setminus dom(g^{tr})$.
- $g^{e,+}_{(e,c)}$ and $g^{e,+}_{(c,e)}$ are entirely defined by $((g, g^{+}_{(e,c)}), (g^{tr}, g^{tr,+}_{(e,c)}))$ and $((g, g^{+}_{(c,e)}), (g^{tr}, g^{tr,+}_{(c,e)}))$ that are equal.

□

application for renaming Before dealing with PCA-executions-matching, we state an intuitive theorem of executions-matching after renaming.

Theorem 12. *(weak complete bijective executions-matching after hiding) Let $\mathcal{A}$ be a PSIOA. Let h defined on states($\mathcal{A}$), s.t. $\forall q \in Q_{\mathcal{A}}, h(q) \subseteq out(\mathcal{A})(q)$. Let $\mathcal{B} = hiding(\mathcal{A}, h)$. Let Id the identity function from states($\mathcal{A}$) to states($\mathcal{B}$) $= Q_{\mathcal{A}}$. Then (Id, Id^{tr}, Id^{ex}) is a weak complete bijective executions-matching from $\mathcal{A}$ to $\mathcal{B}$.*

Proof. By definition Id ensures starting state preservation and weak signature preservation. By definition Id is a complete bijection, which implies matched state preservation. The equitable corresponding distribution is also ensured by definition of *hiding*. Hence, all the properties are ensured □

4.1.2. PCA-matching execution

Here we extend the notion of executions-matching to PCA. In practice, we will build executions-matchings that preserve the sequence of configurations visited by concerned executions. Hence, the definition of PCA states-matching is slightly more restrictive to capture this notion of configuration equivalence (modulo action hiding operation), while the other definitions are exactly the same ones.

matching execution

Definition 124 (PCA states-matching)**.** *Let X and Y be two PCA and let $f : Q'_X \subset Q_X \to Q_Y$ be a mapping s.t. :*

- *Starting state preservation: If $\bar{q}_X \in Q'_X$, then $f(\bar{q}_X) = \bar{q}_Y$.*
- *Configuration preservation (modulo hiding): $\forall (q, q') \in Q'_X \times Q_Y$, s.t. $q' = f(q)$, if $auts(config(X)(q)) = (\mathcal{A}_1, ..., \mathcal{A}_n)$, then $auts(config(Y)(q')) = (\mathcal{A}'_1, ..., \mathcal{A}'_n)$ where $\forall i \in [1 : n], \mathcal{A}_i = hide(\mathcal{A}'_i, h_i)$ with h_i defined on $states(\mathcal{A}'_i)$, s. t. $h_i(q_{\mathcal{A}'_i}) \subseteq out(\mathcal{A}'_i)(q_{\mathcal{A}'_i})$ (resp. s.t. $h_i(q_{\mathcal{A}'_i}) = \emptyset$, that is $\mathcal{A}_i = \mathcal{A}'_i$)*
- *Hiding preservation (modulo hiding): $\forall (q, q') \in Q'_X \times Q_Y$, s.t. $q' = f(q)$, $hidden\text{-}actions(X)(q) = hidden\text{-}actions(Y)(q') \cup h^+(q')$ where h^+ defined on $states(Y)$, s. t. $h^+(q_Y) \subseteq out(Y)(q_Y)$ (resp. s.t. $h^+(q_Y) = \emptyset$, that is $hidden\text{-}actions(X)(q) = hidden\text{-}actions(Y)(q')$)*
- *Creation preservation $\forall (q, q') \in Q'_X \times Q_Y$, s.t. $q' = f(q)$, $\forall a \in \widehat{sig}(X)(q) = \widehat{sig}(Y)(q')$, $created(X)(q)(a) = created(Y)(q')(a)$.*

then we say that f is a weak *(resp.* strong*)* PCA states-matching *from X to Y. If $Q'_X = Q_X$, then we say that f is a* complete *(weak or strong) PCA states-matching from X to Y.*

We naturally obtain that a PCA states-matching is a PSIOA states-matching:

Lemma 21 (A PCA states-matching is a PSIOA states-matching)**.** *If f is a weak (resp. strong) PCA states-matching from X to Y, then f is a PSIOA states-matching from $psioa(X)$ to $psioa(Y)$ (in the sense of definition 112). (The converse is not necessarily true.)*

Proof. The signature preservation immediately comes from the configuration preservation and the hiding preservation. □

Now, all the definitions from definition 113 to definition 115 of previous subsections are the same that is:

Definition 125 (PCA transitions-matching and PCA executions-matching)**.** *Let X and Y be two PCA and let $f : Q'_X \subset Q_X \to Q_Y$ be a PCA states-matching from X to Y.*

- *Let $D'_X \subseteq D_X$ be a subset of transitions, D'_X is* eligible to PCA transitions-matching domain *from f if it is eligible to PSIOA transitions-matching domain from f according to definition 113.*
- *Let $D'_X \subseteq D_X$ be a subset of transitions eligible to PCA transitions-matching domain from f. We define the* PCA transitions-matching *(f, f^{tr}) induced by the PCA states-matching f and the subset of transitions D'_X as the PSIOA transitions-matching induced by the PSIOA states-matching f and the subset of transitions D'_X according to definition 114.*

- *Let $f^{tr} : D'_X \subseteq D_X \to D_Y$ s.t. (f, f^{tr}) is a PCA transitions-matching, we define the PCA executions-matching (f, f^{tr}, f^{ex}) induced by (f, f^{tr}) (resp. by f and $dom(f^{tr})$) as the PSIOA executions-matching (f, f^{tr}, f^{ex}) induced by (f, f^{tr}) (resp. by f and $dom(f^{tr})$) according to definition 115. Furthermore, let $(\mu, \mu') \in Disc(Frags(X)) \times Disc(Frags(Y))$ s.t. for every $\alpha' \in supp(\mu)$, $\alpha' \in dom(f^{ex})$ and $\mu(\alpha) = \mu'(f^{ex}(\alpha'))$. then we say that (f, f^{tr}, f^{ex}) is a* PCA executions-matching *from (X, μ) to (Y, μ') according to definition 116.*
- *The (f^+, D''_X)-continuation of a PCA-executions-matching (f, f^{tr}, f^{ex}) is the (f^+, D''_X)- continuation of (f, f^{tr}, f^{ex}) in the according to definition 117.*

We restate the theorem 8 and 9 for PCA executions-matching:

Theorem 13 (PCA-executions-matching preserves probabilsitic distribution)**.** *Let X and Y be two PCA $(\mu, \mu') \in Disc(Frags(X)) \times Disc(Frags(Y))$. Let (f, f^{tr}, f^{ex}) be a PCA executions-matching from (X, μ) to (Y, μ') . Let $(\tilde{\sigma}, \sigma) \in schedulers(\mathcal{A}) \times schedulers(\mathcal{B})$, s.t. $(\tilde{\sigma}, \sigma)$ are (f, f^{tr}, f^{ex})*-alter egos. *Let $(\alpha, \pi) \in dom(f^{ex}) \times Frags(Y)$.*
If $\pi = f^{ex}(\alpha)$, then $\epsilon_{\tilde{\sigma},\tilde{\mu}}(C_{\tilde{\alpha}}) = \epsilon_{\sigma,\mu}(C_\alpha)$ and $\epsilon_{\tilde{\sigma},\tilde{\mu}}(\tilde{\alpha}) = \epsilon_{\sigma,\mu}(\alpha)$.

Proof. We just re-apply the theorem 8, since (f, f^{tr}, f^{ex}) is a PSIOA executions-matching from $(psioa(X), \mu)$ to $(psioa(Y), \mu')$. □

Theorem 14 (Continued PCA executions-matching preserves general probabilistic distribution)**.** *Let X and Y be two PCA $(\mu, \mu') \in Disc(Frags(X)) \times Disc(Frags(Y))$. Let (f, f^{tr}, f^{ex}) be a PCA executions-matching from (X, μ) to (Y, μ') . Let $((f, f^+), f^{tr,+}, f^{ex,+})$ be a continuation of (f, f^{tr}, f^{ex}). Let $(\tilde{\sigma}, \sigma) \in schedulers(\mathcal{A}) \times schedulers(\mathcal{B})$, s.t. $(\tilde{\sigma}, \sigma)$ are (f, f^{tr}, f^{ex})*-alter egos. *Let $(\alpha, \pi) \in dom(f^{ex,+}) \times Frags(Y)$.*
If $\pi = f^{ex,+}(\alpha)$, then $\epsilon_{\tilde{\sigma},\tilde{\mu}}(C_{\tilde{\alpha}}) = \epsilon_{\sigma,\mu}(C_\alpha)$.

Proof. We just re-apply the theorem, 9 since $((f, f^+), f^{tr,+}, f^{ex,+})$ is a continued PSIOA executions-matching from $(psioa(X), \mu)$ to $(psioa(Y), \mu')$. □

Composability of executions-matching relationship Now we are looking for composability of PCA executions-matching. Here again the notions are the same than the ones for PSIOA excepting for states-matching and for partial-compatibility. Hence we only need to show that i) the $\mathcal{E}$-extension of a PCA states-matching is still a PCA states-matching (see lemma 22), ii) if $f : Q_X \to Q_Y$ is a bijective PCA states-matching and f^{-1} is complete, then for every PCA $\mathcal{E}$ partial-compatible with X, $\mathcal{E}$ is partial-compatible Y (see lemma 23).

Lemma 22 (Composability of PCA states-matching)**.** *Let X and Y be two PCA. Let $\mathcal{E}$ be partially-compatible with both X and Y. Let $f : Q'_X \subset Q_X \to Q_Y$ be a PCA states-matching. Let g be the $\mathcal{E}$-extension of f.*

If $range(g) \subset Q_{Y||\mathcal{E}}$, then g is a PCA states-matching from $X||\mathcal{E}$ to $Y||\mathcal{E}$.

Proof.
- If $(\bar{q}_X, \bar{q}_\mathcal{E}) \in Q_{X||\mathcal{E}}$ then $\bar{q}_X \in Q'_X$ which means $f(\bar{q}_X) = \bar{q}_Y$, thus $g((\bar{q}_X, \bar{q}_\mathcal{E})) = (\bar{q}_\mathcal{E}, \bar{q}_\mathcal{E})$.
- $\forall((q_X, q_\mathcal{E}), (q_Y, q_\mathcal{E})) \in Q'_{X||\mathcal{E}} \times Q_{Y||\mathcal{E}}$ with $(q_Y, q_\mathcal{E}) = g((q_X, q_\mathcal{E}))$, we have
  - Configuration preservation (modulo hiding): if $auts(config(X)(q_X)) = (\mathcal{A}_1, ..., \mathcal{A}_n)$, then $auts(config(Y)(q_Y)) = (\mathcal{A}'_1, ..., \mathcal{A}'_n)$ where $\forall i \in [1:n], \mathcal{A}_i = hide(\mathcal{A}'_i, h_i)$ with h_i defined on $states(\mathcal{A}'_i)$, s. t. $h_i(q_{\mathcal{A}'_i}) \subseteq out(\mathcal{A}'_i)(q_{\mathcal{A}'_i})$ (resp. s.t. $h_i(q_{\mathcal{A}'_i}) = \emptyset$, that is $\mathcal{A}_i = \mathcal{A}'_i$). Hence if $auts(config(X||\mathcal{E})((q_X, q_\mathcal{E})) = (\mathcal{A}_1, ..., \mathcal{A}_n, \mathcal{B}_1, ..., \mathcal{B}_m)$, then $auts(config(Y||\mathcal{E})((q_Y, q_\mathcal{E})) = (\mathcal{A}'_1, ..., \mathcal{A}'_n, \mathcal{B}_1, ..., \mathcal{B}_m)$ where $\forall i \in [1:n], \mathcal{A}_i = hide(\mathcal{A}'_i, h_i)$ with h_i defined on $states(\mathcal{A}'_i)$, s. t. $h_i(q_{\mathcal{A}'_i}) \subseteq out(\mathcal{A}'_i)(q_{\mathcal{A}'_i})$ (resp. s.t. $h_i(q_{\mathcal{A}'_i}) = \emptyset$, that is $\mathcal{A}_i = \mathcal{A}'_i$).

- Hidding preservation (modulo hiding): $hidden\text{-}actions(X)(q_X) = hidden\text{-}actions(Y)(q_Y) \cup h^+(q_Y)$ where h^+ defined on $states(Y)$, s. t. $h^+(q_Y) \subseteq out(Y)(q_Y)$.
  Hence $hidden\text{-}actions(X||\mathcal{E})((q_X, q_\mathcal{E})) = hidden\text{-}actions(X)(q_X) \cup hidden\text{-}actions(\mathcal{E})(q_\mathcal{E}) = hidden\text{-}actions(Y)(q_Y) \cup hidden\text{-}actions(\mathcal{E})(q_\mathcal{E}) \cup h^+(q_Y) = hidden\text{-}actions(Y||\mathcal{E})((q_Y, q_\mathcal{E})) \cup h^{+\prime}((q_Y, q_\mathcal{E}))$ where $h^{+\prime}$ defined on $states(Y||\mathcal{E})$, s. t. $h^{+\prime}((q_Y, q_\mathcal{E})) = h^+(q_Y) \subseteq out(Y)(q_Y) \subseteq out(Y||\mathcal{E})((q_Y, q_\mathcal{E}))$.
- Creation preservation $\forall a \in \widehat{sig}(X)(q_X) = \widehat{sig}(Y)(q_Y)$, $created(X)(q_X)(a) = created(Y)(q_Y)(a)$.
  Hence $\forall a \in \widehat{sig}(X||\mathcal{E})((q_X, q_\mathcal{E})) = \widehat{sig}(Y||\mathcal{E})((q_Y, q_\mathcal{E}))$, either
  - $a \in \widehat{sig}(X)(q_X) = \widehat{sig}(Y)(q_Y)$ but $a \notin \widehat{sig}(\mathcal{E})(q_\mathcal{E})$ and then $created(X||\mathcal{E})((q_X, q_\mathcal{E}))(a) = created(X)(q_X)(a) = created(Y)(q_Y) = created(Y||\mathcal{E})((q_Y, q_\mathcal{E}))(a)$
  - or $a \notin \widehat{sig}(X)(q_X) = \widehat{sig}(Y)(q_Y)$ but $a \in \widehat{sig}(\mathcal{E})(q_\mathcal{E})$ and then $created(X||\mathcal{E})((q_X, q_\mathcal{E}))(a) = created(\mathcal{E})(q_\mathcal{E})(a) = created(Y||\mathcal{E})((q_Y, q_\mathcal{E}))(a)$
  - or $a \in \widehat{sig}(X)(q_X) = \widehat{sig}(Y)(q_Y)$ and $a \in \widehat{sig}(\mathcal{E})(q_\mathcal{E})$ and then $created(X||\mathcal{E})((q_X, q_\mathcal{E}))(a) = created(X)(q_X)(a) \cup created(\mathcal{E})(q_\mathcal{E})(a) = created(Y)(q_Y) \cup created(\mathcal{E})(q_\mathcal{E})(a)$, i.e. $created(X||\mathcal{E})((q_X, q_\mathcal{E}))(a) = created(Y||\mathcal{E})((q_Y, q_\mathcal{E}))(a)$

  Thus, $\forall a \in \widehat{sig}(X||\mathcal{E})((q_X, q_\mathcal{E})) = \widehat{sig}(Y||\mathcal{E})((q_Y, q_\mathcal{E}))$, we have $created(X||\mathcal{E})((q_X, q_\mathcal{E}))(a) = created(Y||\mathcal{E})((q_Y, q_\mathcal{E}))(a)$.

□

We restate the theorem 10 of executions-matching composability.

Theorem 15 (Composability of PCA matching-execution)**.** *Let X and Y be two PCA. Let $\mathcal{E}$ be partially-compatible with both X and Y. Let (f, f^{tr}, f^{ex}) be a PCA executions-matching from X to Y. Let g be the $\mathcal{E}$-extension of f. If $range(g) \subset Q_{Y||\mathcal{E}}$, then the $\mathcal{E}$-extension of (f, f^{tr}, f^{ex}) is a PCA executions-matching (g, g^{tr}, g^{ex}) from $X||\mathcal{E}$ to $Y||\mathcal{E}$ induced by g and $dom(g^{tr})$.*

Proof. This comes immediately from theorem 10. □

We extend the lemma 19 but we have to take a little precaution for the partial-compatibility since here the configurations have to be pairwise compatible, not only the signatures.

Lemma 23 (Some properties preserved by $\mathcal{E}$-extension of a PCA executions-matching)**.** *Let X and Y be two PCA. Let (f, f^{tr}, f^{ex}) be a PCA executions-matching from X to Y.*

1. *If f is complete, then for every PSIOA $\mathcal{E}$ partially-compatible with X, $\mathcal{E}$ is partially-compatible with Y.*
2. *Let $\mathcal{E}$ partially-compatible with both X and Y, let g be the $\mathcal{E}$-extension of f.*
   a) *If f is bijective and f^{-1} is complete, then $range(g) = Q_{Y||\mathcal{E}}$ and so we can talk about the $\mathcal{E}$-extension of (f, f^{tr}, f^{ex})*
   b) *If (f, f^{tr}) is a bijective complete transition-matching, (g, g^{tr}) is a bijective complete transition-matching. (And (f, f^{tr}, f^{ex}) and (g, g^{tr}, g^{ex}) are bijective complete executions-matching.)*
   c) *If f is strong, then g is strong*

Proof. 1. We need to show that every pseudo-execution of $(Y, \mathcal{E})$ ends on a compatible state. Let $\pi = q^0 a^1 q^1 ... a^n q^n$ be a finite pseudo-execution of $(Y, \mathcal{E})$. We note α the alternating sequence $(f^{-1}(q_Y^0), q_\mathcal{E}^0) a^1 (f^{-1}(q_Y^1), q_\mathcal{E}^1) ... a^n (f^{-1}(q_Y^n), q_\mathcal{E}^n)$. The proof is in two steps.

First, we show by induction that $\alpha = (f^{-1}(q_Y^0), q_{\mathcal{E}}^0)a^1(f^{-1}(q_Y^1), q_{\mathcal{E}}^1)...a^n(f^{-1}(q_Y^n), q_{\mathcal{E}}^n)$ is an execution of $X||\mathcal{E}$. Second, we deduce that it means $(f^{-1}(q_Y^n), q_{\mathcal{E}}^n)$ is a compatible state of $(X, \mathcal{E})$ which means that a fortiori, $(q_Y^n, q_{\mathcal{E}}^n)$ is a compatible state of $(Y, \mathcal{E})$ which ends the proof.

- First, we show by induction that α is an execution of $X||\mathcal{E}$. We have $(f^{-1}(\bar{q}_Y), \bar{q}_{\mathcal{E}}) = (\bar{q}_X, \bar{q}_{\mathcal{E}})$ which ends the basis.
  Let assume $(f^{-1}(q_Y^0), q_{\mathcal{E}}^0)a^1(f^{-1}(q_Y^1), q_{\mathcal{E}}^1)...a^k(f^{-1}(q_Y^k), q_{\mathcal{E}}^k)$ is an execution of $X||\mathcal{E}$. Hence $(f^{-1}(q_Y^k), q_{\mathcal{E}}^k)$ is a compatible state of $(X, \mathcal{E})$ which means that a fortiori q^k is a compatible state of $(Y, \mathcal{E})$ because of signature preservation of f.
  Similarly, $\widehat{sig}(Y, \mathcal{E})(q^k) = \widehat{sig}(X||\mathcal{E})((f^{-1}(q_Y^k), q_{\mathcal{E}}^k))$, so $a^{k+1} \in \widehat{sig}(X, \mathcal{E})((f^{-1}(q_Y^k), q_{\mathcal{E}}^k))$. Then we use the completeness of $(f^{-1}, (f^{tr})^{-1})$, to obtain the fact that either $\eta_{(Y,q_Y^k,a^{k+1})} \in dom((f^{tr})^{-1})$ or $a^{k+1} \notin \widehat{sig(Y)}(q_Y^k)$ (and we recall the convention that in this second case $\eta_{(Y,q_Y^k,a^{k+1})} = \delta_{q_Y^k}$). which means either $(f^{-1}(q_Y^k), a^{k+1}, \eta_{(X,f^{-1}(q_Y^k),a^{k+1})})$ is a transition of X that ensures $\forall q'' \in supp(\eta_{(Y,q_Y^k,a^{k+1})}), f^{-1}(q'') \in supp(\eta_{(X,f^{-1}(q_Y^k),a^{k+1})})$ or $a^{k+1} \notin \widehat{sig(X)}(f^{-1}(q_Y^k))$ (and we recall the convention that in this second case $\eta_{(X,f^{-1}(q_Y^k),a^{k+1})} = \delta_{f^{-1}(q_Y^k)}$). Thus for every $(q'', q''') \in supp(\eta_{(Y,\mathcal{E}),q^k,a^{k+1})})$, $(f^{-1}(q''), q''') = g^{-1}((q'', q''')) \in supp(\eta_{(X,\mathcal{E}),g^{-1}(q^k),a^{k+1})})$ namely for $(q'', q''') = (q_Y^{k+1}, q_{\mathcal{E}}^{k+1})$. Hence, $(f^{-1}(q_Y^{k+1}), q_{\mathcal{E}}^{k+1})$ is reachable by $(X, \mathcal{E})$ which means the alternating sequence $(f^{-1}(q_Y^0), q_{\mathcal{E}}^0)a^1(f^{-1}(q_Y^1), q_{\mathcal{E}}^1)...a^k(f^{-1}(q_Y^k), q_{\mathcal{E}}^k)a^k(f^{-1}(q_Y^k), q_{\mathcal{E}}^k)a^{k+1}(f^{-1}(q_Y^{k+1}), q_{\mathcal{E}}^{k+1})$ is an execution of $X||\mathcal{E}$. Thus by induction α is an execution of $X||\mathcal{E}$.
- Since X and $\mathcal{E}$ are partially-compatible $(f^{-1}(q_Y^n), q_{\mathcal{E}}^n)$ is a state of $X||\mathcal{E}$, so $(f^{-1}(q_Y^n), q_{\mathcal{E}}^n)$ is a compatible state of $(X, \mathcal{E})$ which means $(q_Y^k, q_{\mathcal{E}}^k)$ is a fortiori a compatible state of $(Y, \mathcal{E})$. Hence every reachable state of $(Y, \mathcal{E})$ is compatible which means Y and $\mathcal{E}$ are partially compatible which ends the proof.

2. This comes immediately from lemma 19 since (f, f^{tr}, f^{ex}) is a PSIOA executions-matching from $psioa(X)$ to $psioa(Y)$ by construction.

□

Finally, we restate the semantic-equivalence.

A strong complete bijective transitions-matching implies a strong complete bijective executions-matching which means the two automata are completely semantically equivalent.

Definition 126 (PCA semantic equivalence)**.** *Let X an Y be two PCA. We say that X and Y are semantically-equivalent if it exists a complete bijective strong PCA executions-matching from X to Y*

Theorem 16 (composability of semantic equivalence)**.** *Let X and Y be PCA semantically-equivalent. Then for every PSIOA $\mathcal{E}$:*

- *$\mathcal{E}$ is partially-compatible with $X \iff \mathcal{E}$ is partially-compatible with Y*
- *if $\mathcal{E}$ is an environment for both X and Y, then $X||\mathcal{E}$ and $Y||\mathcal{E}$ are PCA semantically-equivalent.*

Proof.
- The first item comes from lemma 23, first item
- The second item comes from lemma 23, second item

□

A weak complete bijective PCA transitions-matching implies a weak complete bijective PCA executions-matching which means the two automata are completely semantically equivalent modulo some hiding operation that implies that some PSIOA are partially-compatible with one of the automaton and not with the other one and that the traces are not necessarily the same ones.

4.1.3. Digest

We have introduced the concept of executions-matching which is an homomorphism between 2 sets of executions of two automata X and Y that preserves (modulo hiding operations):

- starting state
- signature at each pair of matched states
- measure of probability over matched states at each pair of matched transitions
- configuration at each pair of matched states
- created automata by each action of the common signature of each pair of matched states

Such an executions-matching is said:

- "strong" if no hiding opeartion was required for the correspondance, "weak" otherwise.
- "complete" if is domain covers the entire set of executions of X

If two automata are linked by a strong complete bijective executions-matching, we say they are semantically-equivalent.

The key property of an executions-matching $\tilde{f}$ between to automata X and Y is that for every scheduler σ of X, it is easy to define its $\tilde{f}$-alter ego $\sigma' \in \mathit{schedulers}(Y)$ such that the measures of corresponding executions are preserved, i.e. $\epsilon_\sigma(C_\alpha) = \epsilon_{\sigma'}(C_{\alpha'})$ and $\epsilon_\sigma(\alpha) = \epsilon_{\sigma'}(\alpha')$ for each pair of executions $(\alpha, \alpha') \in \mathit{Execs}(X) \times \mathit{Execs}(Y)$ linked by the executions-matching.

When an executions-matching $\tilde{f}$ between to automata X and Y exists, it is possible (under minor technical conditions) to define its $\mathcal{E}$-extension, i.e. the executions-matching $\tilde{g}$ between the automata $X||\mathcal{E}$ and $Y||\mathcal{E}$ such that $\tilde{g}$ deprived of its attributes relative to $\mathcal{E}$ corresponds to $\tilde{f}$. All the good properties are preserved by extension. In a certain sense, we can say that an extensions-matching is composable.

We also introduced the concept of continued executions-matching, that allows the loss of some properties for the very last steps of the concerned executions. This will be a key notion for copy-pasting operations over execution fragments, where an executions-matching might lose its properties upon the (re-)creation of a sub-automaton.

Intuitively, we can say that the operations of continuation and extension commute.

4.2. Projection

This section aims to formalize the idea of a PCA $X_{\mathcal{A}}$ considered without an internal PSIOA $\mathcal{A}$. This PCA will be noted $Y_{\mathcal{A}} = X_{\mathcal{A}} \setminus \{\mathcal{A}\}$. The reader can already take a look on the figures 4.7 and 4.8 to get an intuition on the desired result. This is an important step in our reasoning since we will be able to formalise in which sense $X_{\mathcal{A}}$ and $psioa(X_{\mathcal{A}} \setminus \{\mathcal{A}\})||\mathcal{A}$ are similar.

We first define some notions of projection on configurations on subsection 4.2.1. Then we define the notion of $\mathcal{A}$-fair PCA X in subsection 4.2.2, which will be a sufficient condition to ensure that $Y = X \setminus \{\mathcal{A}\}$ is still a PCA, namely that it ensures the constraints of top/down and bottom/up transition preservation, which is proved in the last subsection 4.2.3.

4.2.1. Projection on Configurations

In this subsection, we want to define formally $\eta' \in Disc(Q_{conf})$ that would be the result of $\eta \in Disc(Q_{conf})$ "deprived of an automaton $\mathcal{A}$". This is achieved in definition 130. This definition requires particular precautions and motivate the next sequence of definitions, from definition 127 to 130.

The next definition captures the idea of a state deprived of a PSIOA $\mathcal{A}$.

Definition 127 (State projection)**.** *Let $\mathbf{A} = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$ be a set of PSIOA compatible at state $q = (q_1, ..., q_n) \in Q_{\mathcal{A}_1} \times ... \times Q_{\mathcal{A}_n}$. Let $\mathbf{A}^s = \{\mathcal{A}_{s^1}, ..., \mathcal{A}_{s^n}\}$. We note :*

- *$q \setminus \{\mathcal{A}_k\} = (q_1, ..., q_{k-1}, q_{k+1}, ..., q_n)$ if $\mathcal{A}_k \in \mathbf{A}$ and $q \setminus \{\mathcal{A}_k\} = q$ otherwise.*
- *$q \setminus \mathbf{A}^s = (q \setminus \{\mathcal{A}_{s^n}\}) \setminus (\mathbf{A}^s \setminus \{\mathcal{A}_{s^n}\})$ (recursive extension of the previous item).*
- *$q \upharpoonright \mathbf{A}^s = q \setminus (\mathbf{A} \setminus \mathbf{A}^s)$ if $\mathbf{A}^s \subset \mathbf{A}$ (recursive extension of the previous item). We can remark that $q \upharpoonright \mathcal{A}_k = q_k$ if $\mathcal{A}_k \in \mathbf{A}$.*

Since, $\upharpoonright$ can be defined with $\setminus$, the next sequence of definitions only handles $\setminus$ but can be adapted to support $\upharpoonright$ in an obvious way.

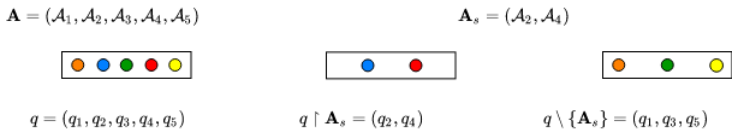


Figure 4.2. – State projection

The next definition captures the idea of a family transition deprived of a PSIOA $\mathcal{A}$.

Definition 128 (Family transition projection)**.** *(see figure 4.3 first for an intuition) Let $\mathbf{A}_1$ be a set of automata compatible at state $q_1 \in Q_{\mathbf{A}_1}$. Let $\mathbf{A}^s, \mathbf{A}_2 = \mathbf{A}_1 \setminus \mathbf{A}^s \neq \emptyset$. Let $q_2 = q_1 \setminus \mathbf{A}^s$. Let a be an action. We note $\eta_{(\mathbf{A}_1,q_1,a)} \setminus \mathbf{A}^s \triangleq \eta_{(\mathbf{A}_2,q_2,a)}$ with the convention $\eta_{(\mathbf{A}_i,q_i,a)} = \delta_{q_i}$ if $a \notin \widehat{sig}(\mathbf{A}_i)(q_i)$ for each $i \in \{1, 2\}$.*

Lemma 24 (family transition projection)**.** *Let $\mathbf{A}_1$ be a set of automata compatible at state $q_1 \in Q_{\mathbf{A}_1}$. Let $\mathbf{A}^s, \mathbf{A}_2 = \mathbf{A}_1 \setminus \mathbf{A}^s \neq \emptyset$. Let $q_2 = q_1 \setminus \mathbf{A}^s$. Let a be an action. Let $\eta_1 = \eta_{(\mathbf{A}_1,q_1,a)}$ and $\eta_2 = \eta_1 \setminus \mathbf{A}^s$ with the convention $\eta_{(\mathbf{A}_1,q_1,a)} = \delta_{q_1}$ if $a \notin \widehat{sig}(\mathbf{A}_1)(q_1)$.*

Then $\forall q_2' \in Q_{\mathbf{A}_2}$, $\eta_2(q_2') = \sum_{q_1' \in Q_{\mathbf{A}_1}, q_1' \setminus \mathbf{A}^s = q_2'} \eta_1(q_1')$

Proof. Comes from total probability law. If $\mathbf{A}^s \cap \mathbf{A}_1 = \emptyset$, $\mathbf{A}_2 = \mathbf{A}_1$, the result is immediate. Assume $\mathbf{A}^s \cap \mathbf{A}_1 \neq \emptyset$. Let $\mathbf{A}_3 = \mathbf{A} \setminus \mathbf{A}_2 = \mathbf{A} \setminus (\mathbf{A} \setminus \mathbf{A}^s) \neq \emptyset$. We note $q_3 = q_1 \setminus \mathbf{A}_2$, $\eta_3 = \eta_1 \setminus \mathbf{A}_2$ Then $\forall q_1' \in Q_{\mathbf{A}_1}$, $\eta_1(q_1') = \eta_2(q_2') \otimes \eta_3(q_3')$ with $q_2' = q_1' \upharpoonright \mathbf{A}_2$ and $q_3' = q_1' \upharpoonright \mathbf{A}_3$. Hence $\forall q_2' \in Q_{\mathbf{A}_2}$, $\sum_{q_1' \in Q_{\mathbf{A}_1}, q_1' \setminus \mathbf{A}^s = q_2'} \eta_1(q_1') = \sum_{q_1' \in Q_{\mathbf{A}_1}, q_1' \upharpoonright \mathbf{A}_2 = q_2'} \eta_2(q_2') \cdot \eta_3(q_1' \upharpoonright \mathbf{A}_3) = \eta_2(q_2') \cdot \sum_{q_3' \in Q_{\mathbf{A}_3}} \eta_3(q_3') = \eta_2(q_2')$, which ends the proof.

□

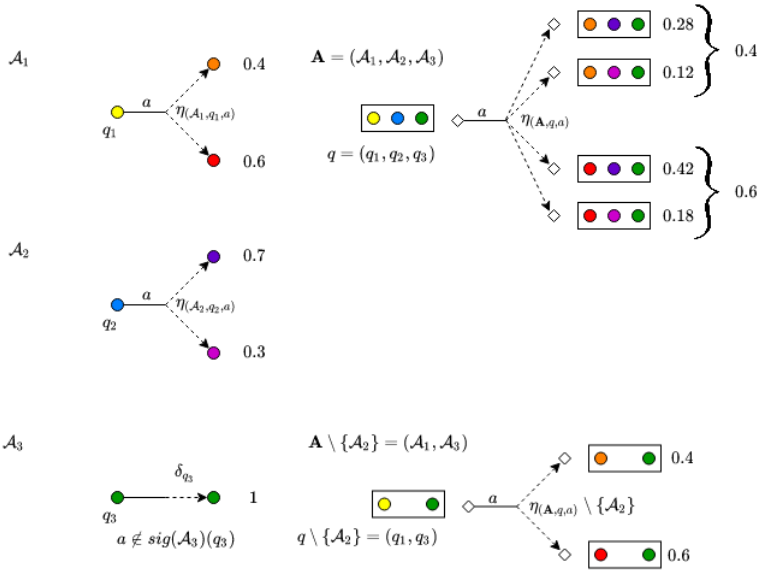


Figure 4.3. – total probability law for family transition projection

Then we apply this notation to preserving distributions.

Definition 129 (preserving transition projection)**.** *(see figure 4.4) Let* $\mathbf{A}$, $\mathbf{A}^s$, $\mathbf{A}_2 = \mathbf{A} \setminus \mathbf{A}^s$ *be set of automata,* $q \in Q_{\mathbf{A}}$*, and* a *be an action. Let* $\eta_p \in Disc(Q_{conf})$ *be the unique preserving distribution s.t.* $\eta_p \overset{TS}{\leftrightarrow} \eta_{(\mathbf{A},q,a)}$ *with the convention* $\eta_{(\mathbf{A},q,a)} = \delta_q$ *if* $a \notin \widehat{sig}(\mathbf{A})(q)$*. We note* $\eta_p \setminus \mathbf{A}^s$ *the unique preserving distribution s.t.* $(\eta_p \setminus \mathbf{A}^s) \overset{TS}{\leftrightarrow} (\eta_{(\mathbf{A},q,a)} \setminus \mathbf{A}^s)$ *if* $\mathbf{A}_2 \neq \emptyset$ *and* $\eta_p = \delta_{(\emptyset,\emptyset)}$ *otherwise.*

Lemma 25 (preserving transition projection)**.** *Let* $\mathbf{A}^s$ *be finite sets of PSIOA. Let* a *be an action. For each* $i \in \{1, 2\}$*, let* $C_i \in Q_{conf}$*,* $C_i \overset{a}{\rightharpoonup} \eta_p^i$ *if* $a \in \widehat{sig}(C_i)$ *and* $\eta_p^i = \delta_{C_i}$ *otherwise. Let* $\tilde{\eta}_p^2 = \eta_p^1 \setminus \mathbf{A}^s$*. Assume* $C_2 = C_1 \setminus \mathbf{A}^s$*. Then,*

- $\eta_p^2 = \tilde{\eta}_p^2$*, i.e.* $(C_1 \setminus \mathbf{A}^s) \overset{a}{\rightharpoonup} (\eta_p^1 \setminus \mathbf{A}^s)$*.*
- *For every* $C_2' \in Q_{conf}, \eta_p^2(C_2') = \Sigma_{(C_1' \in Q_{conf}, C_1' \setminus \mathbf{A}^s = C_2')} \eta_p^1(C_1')$

Proof.

- Immediate by definitions 67 and 129.
- For each $i \in \{1, 2\}$, we note $\mathbf{A}_i = auts(C_i)$, $q_i = TS(C_i)$. By definition, we have $\eta_p^i \overset{TS}{\leftrightarrow} \eta_{(\mathbf{A}_i,q_i,a)}$ with the convention $\eta_{(\mathbf{A}_i,q_i,a)} = \delta_q$ if $a \notin \widehat{sig}(\mathbf{A}_i)(q_i)$. Finally, we apply lemma 24.

□

Now we are able to define intrinsic transition deprived of a PSIOA $\mathcal{A}$.

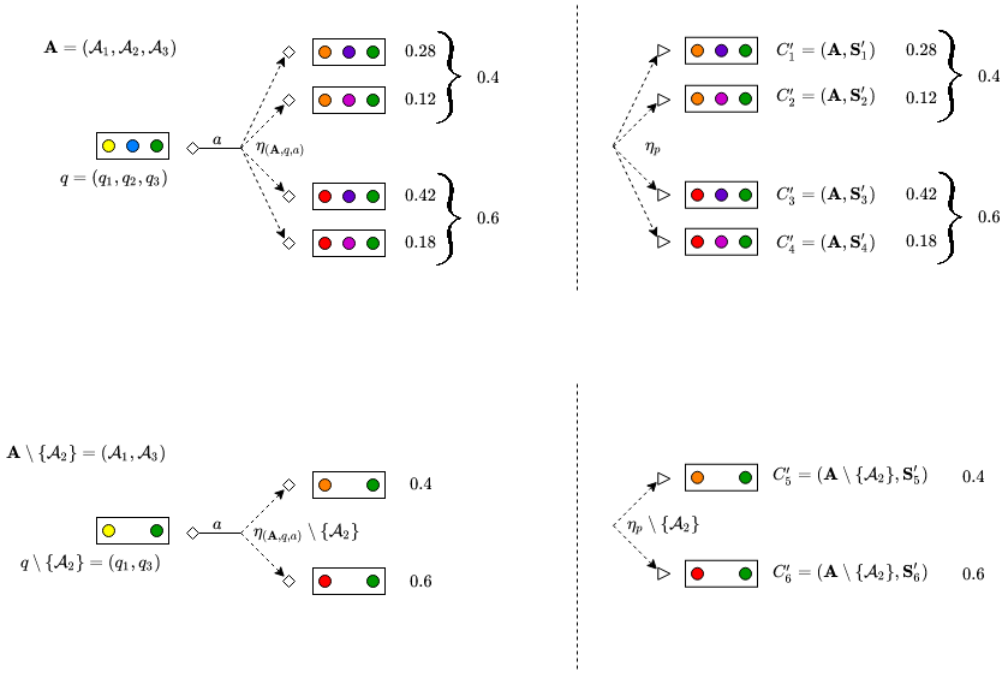


Figure 4.4. – total probability law for preserving configuration

Definition 130 (intrinsic transition projection)**.** *(see figure 4.5) Let* $\mathbf{A}$, $\mathbf{A}^s$ *be finite sets of automata,* $q \in Q_{\mathbf{A}}$*, and* a *be an action. Let* $\eta_p \in Disc(Q_{conf})$ *be the unique preserving distribution s.t.* $\eta_p \overset{TS}{\leftrightarrow} \eta_{(\mathbf{A},q,a)}$ *with the convention* $\eta_{(\mathbf{A},q,a)} = \delta_q$ *if* $a \notin \widehat{sig}(\mathbf{A})(q)$*. Let* φ *be a finite set of PSIOA identifiers with* $aut(\varphi) \cap \mathbf{A} = \emptyset$*. Let* $\eta = reduce(\eta_p \upharpoonright \varphi)$*. We note* $\eta \setminus \mathbf{A}^s = reduce((\eta_p \setminus \mathbf{A}^s) \upharpoonright (\varphi \setminus \mathbf{A}^s))$*.*

Lemma 26 (intrinsic transition projection)**.** *Let* $\mathbf{A}^s$ *be finite sets of PSIOA. Let* a *be an action. For each* $i \in \{1, 2\}$*, let* φ_i *be a finite set of PSIOA identifiers, let* $C_i \in Q_{conf}$*,* $C_i \overset{a}{\Longrightarrow}_{\varphi_i} \eta^i$ *if* $a \in \widehat{sig}(C_i)$ *and* $\eta^i = \delta_{C_i}$ *otherwise. Let* $\tilde{\eta}^2 = \eta^1 \setminus \mathbf{A}^s$ *and* $\tilde{\varphi}^2 = \varphi^1 \setminus \mathbf{A}^s$*. Assume* $C_2 = C_1 \setminus \mathbf{A}^s$*. Then,*

1. $\eta^2 = \tilde{\eta}^2$ *and* $\tilde{\varphi}_2 = \varphi_2$*, i.e.* $(C_1 \setminus \mathbf{A}^s) \overset{a}{\Longrightarrow}_{\varphi_1 \setminus \mathbf{A}^s} (\eta^1 \setminus \mathbf{A}^s)$*.*
2. *For every* $C'_2 \in Q_{conf}$*,* $(\eta^2_p \upharpoonright \varphi_2)(C'_2) = \sum_{(C'_1 \in Q_{conf}, C'_1 \setminus \mathbf{A}^s = C'_2)} (\eta^1_p \upharpoonright \varphi_1)(C'_1)$
3. *For every* $C'_2 \in Q_{conf}$*,* $\eta^2(C'_2) = \Sigma_{(C'_1 \in Q_{conf}, C'_1 \setminus \mathbf{A}^s = C'_2)} \eta^1(C'_1)$

Proof.

1. Immediate by definitions 67, 130 and lemma 25
2. Let $C_3 = C_1 \setminus (auts(C_1) \setminus \mathbf{A}^s)$. We note $\varphi_3 = \varphi_1 \setminus \varphi_2$. By definition 67, for each $i \in \{1, 2, 3\}$, for each $C'_i \in Q_{conf}$, $(\eta^i_p \upharpoonright \varphi_i)(C'_i) = \delta_{C_{\varphi_i}}(C'_i \upharpoonright \varphi_i) \cdot \eta^i_p(C'_i \setminus \varphi_i)$ with $auts(C_{\varphi_i}) = \varphi_i$ and $\forall \mathcal{A} \in \varphi_i, map(C_{\varphi_i})(\mathcal{A}) = \bar{q}_{\mathcal{A}}$. By previous lemma, for every $C''_2 \in Q_{conf}$, $\eta^1_p(C''_2) = \sum_{C''_1, C''_1 \setminus \mathbf{A}^s = C''_2} \eta^1_p(C''_1)$.
   Hence, $(\eta^2_p \upharpoonright \varphi_2)(C'_2) = \delta_{C_{\varphi_2}}(C'_2 \upharpoonright \varphi_2) \cdot \sum_{C''_1, C''_1 \setminus \mathbf{A}^s = (C'_2 \setminus \varphi_2)} \eta^1_p(C''_1)$ and so $(\eta^2_p \upharpoonright \varphi_2)(C'_2) = \sum_{C''_1, C''_1 \setminus \mathbf{A}^s = (C'_2 \setminus \varphi_2)} \delta_{C_{\varphi_2}}(C'_2 \upharpoonright \varphi_2) \cdot \eta^1_p(C''_1)$.
   We remark that the conjunction of $C''_1 \in supp(\eta^1_p)$, $C''_1 \setminus \mathbf{A}^s = (C'_2 \setminus \varphi_2)$ and $C'_2 \upharpoonright \varphi_2 = C_{\varphi_2}$ implies $(C''_1 \cup C_{\varphi_3} \cup C_{\varphi_2}) \setminus \mathbf{A}_s = C'_2$.

Thus,

$$\begin{aligned}(\eta_p^2 \upharpoonright \varphi_2)(C_2') &= \sum_{C_1''',C_1'''\setminus\mathbf{A}^s=(C_2')} \delta_{C_{\varphi_2}}(C_2' \upharpoonright \varphi_2)\cdot\delta_{C_{\varphi_3}}(C_1''' \upharpoonright \varphi_3)\cdot\eta_p^1(C_1''' \setminus \varphi_1)\\ &= \sum_{C_1''',C_1'''\setminus\mathbf{A}^s=(C_2')} \delta_{C_{\varphi_2}}(C_1''' \upharpoonright \varphi_2)\cdot\delta_{C_{\varphi_3}}(C_1''' \upharpoonright \varphi_3)\cdot\eta_p^1(C_1''' \setminus \varphi_1)\\ &= \sum_{C_1''',C_1'''\setminus\mathbf{A}^s=C_2'} \delta_{C_{\varphi_1}}(C_1''' \upharpoonright \varphi_1)\cdot\eta_p^1(C_1''' \setminus \varphi_1)\\ &= \sum_{C_1''',C_1'''\setminus\mathbf{A}^s=C_2'} (\eta_p^1 \upharpoonright \varphi_1)\end{aligned}$$

3. By definition 67, for each $i \in \{1,2\}$, for each $C_i' \in Q_{conf}$, $\eta^i(C_i') = \sum_{C_i'',reduce(C_i'')=C_i'} (\eta_p^i \upharpoonright \varphi_i)(C_i'')$.
By previous lemma, for every $C_2'' \in Q_{conf}, \eta_p^1(C_2'') = \sum_{C_1''',C_1'''\setminus\mathbf{A}^s=C_2''} (\eta_p^1 \upharpoonright \varphi_1)(C_1''')$. Thus,

$$\begin{aligned}\eta^2(C_2') &= \sum_{C_2'',reduce(C_2'')=C_2'} \Big(\sum_{C_1''',C_1'''\setminus\mathbf{A}^s=C_2''} (\eta_p^1 \upharpoonright \varphi_1)(C_1''')\Big)\\ &= \sum_{C_1''',reduce(C_1'''\setminus\mathbf{A}^s)=C_2''} (\eta_p^1 \upharpoonright \varphi_1)(C_1''')\\ &= \sum_{C_1''',reduce(C_1''')\setminus\mathbf{A}^s=C_2''} (\eta_p^1 \upharpoonright \varphi_1)(C_1''')\\ &= \sum_{C_1',C_1'\setminus\mathbf{A}^s=C_2'} \Big(\sum_{C_1'',reduce(C_1'')=C_1'} ((\eta_p^1 \upharpoonright \varphi_1)(C_1''))\Big)\\ &= \sum_{C_1',C_1'\setminus\mathbf{A}^s=C_2'} \eta^1(C_1')\end{aligned}$$

□

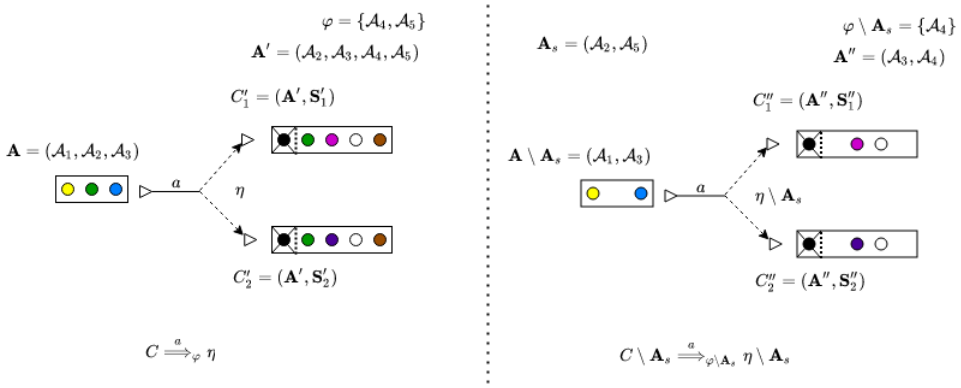


Figure 4.5. – intrinsic transition projection

In next subsection, this lemma 26 will lead to lemma 27 which will be a key lemma to allow the constructive definition 132 of PCA deprived of a (sub) PSIOA.

4.2.2. $\mathcal{A}$-fairness assumption, motivated by our definition of PCA deprived of an internal PSIOA: $X \setminus \{\mathcal{A}\}$

Here we recall in definition 131 the definition 104 of a $\mathcal{A}$-fair PCA. Then we show lemma 27 (via lemma 26) that will be used to enable the constructive definition of $X \setminus \{\mathcal{A}\}$.

Definition 131 ($\mathcal{A}$-fair PCA (recall))**.** *Let $\mathcal{A} \in Autids$. Let X be a PCA. We say that X is $\mathcal{A}$-fair if it verifies the following constraints.*

- *(configuration-conflict-free) X is configuration-conflict-free, that is $\forall q, q' \in Q_X$, s.t. $qR_{conf}q'$ (i.e. $config(X)(q) = config(X)(q')$) then $q = q'$*
- *(no conflict for projection) $\forall q, q' \in Q_X$, s.t. $qR_{conf}^{\setminus\{\mathcal{A}\}}q'$ then $qR_{strict}^{\setminus\{\mathcal{A}\}}q'$. That is if $config(X)(q) \setminus \{A\} = config(X)(q') \setminus \{A\}$, then*
  - *$\forall a \in \widehat{sig}(X)(q) \cap \widehat{sig}(X)(q')$, $created(X)(q)(a) \setminus \{\mathcal{A}\} = created(X)(q')(a) \setminus \{\mathcal{A}\}$*
  - *$hidden\text{-}actions(X)(q) \backslash pot\text{-}out(X)(q)(\mathcal{A}) = hidden\text{-}actions(X)(q') \backslash pot\text{-}out(X)(q')(\mathcal{A})$ where for each $q'' \in Q_X$:*
    - *$pot\text{-}out(X)(q'')(\mathcal{A}) = \emptyset$ if $\mathcal{A} \notin auts(config(X)(q''))$, and*
    - *$pot\text{-}out(X)(q'')(\mathcal{A}) = out(\mathcal{A})(map(config(X)(q''))(\mathcal{A}))$ if $\mathcal{A} \in auts(config(X)(q''))$.*
- *(no exclusive creation by $\mathcal{A}$) $\forall q \in Q_X$, $\forall a \in \widehat{sig}(X)(q)$ $\mathcal{A}$-exclusive in q, $created(X)(q)(a) = \emptyset$ where $\mathcal{A}$-exclusive means $\forall \mathcal{B} \in auts(config(X)(q))$, $\mathcal{B} \neq \mathcal{A}$, $a \notin \widehat{sig}(\mathcal{B})(map(config(X)(q))(\mathcal{B}))$.*

A $\mathcal{A}$-fair PCA is a PCA s.t. we can deduce its current properties from its current configuration deprived of $\mathcal{A}$. This will allow the definition of $X \setminus \{\mathcal{A}\}$, where X is a PCA, to be well-defined.

Now we give the second key lemma (after lemma 26) to allow the definition 132 of PCA deprived of a (sub) PSIOA. Basically, this lemma states that if two states q_X and q_Y are strictly equivalent modulo the deprivation of a (sub) automaton P, noted $q_X R_{strict}^{\setminus\{P\}} q_Y$, then the intrinsic configurations issued from these states deprived of P are equal.

Lemma 27 (equality of intrinsic transition after deprivation of a sub-PSIOA)**.** *Let X_1, X_2 be two PCA, $(q_1, q_2) \in Q_{X_1} \times Q_{X_2}$ s.t. $q_1 R_{strict}^{\setminus\{P\}} q_2$. Let a be an action. For each $i \in \{1, 2\}$, we note $C_i \triangleq config(X)(q_i)$, $\varphi_i \triangleq created(X)(q_i)(a)$, η_i s.t. if $a \in \widehat{sig}(C_i)$, $C_i \overset{a}{\Longrightarrow}_{\varphi_i} \eta_i$ and $\eta_i = \delta_{C_i}$ otherwise. Then,*

- *$C_0 \triangleq C_1 \setminus \{P\} = C_2 \setminus \{P\}$,*
- *$\varphi_0 \triangleq \varphi_1 \setminus \{P\} = \varphi_2 \setminus \{P\}$,*
- *$\eta \triangleq \eta_1 \setminus \{P\} = \eta_2 \setminus \{P\}$,*
- *If $a \in \widehat{sig}(C_0)$, $C_0 \overset{a}{\Longrightarrow}_{\varphi} \eta_0$ and $\eta_0 = \delta_{C_0}$ otherwise.*

Proof. The two first items comes directly from definition of $R_{strict}^{\setminus\{P\}}$. By lemma 26, if $a \in \widehat{sig}(C_0)$, we have both $C_0 \overset{a}{\Longrightarrow}_{\varphi} \eta_1 \setminus \{P\}$ and $C_0 \overset{a}{\Longrightarrow}_{\varphi} \eta_2 \setminus \{P\}$, while if $a \notin \widehat{sig}(C_0)$, we have both $(\eta_1 \setminus \{P\}) = \delta_{C_0}$ and $(\eta_2 \setminus \{P\}) = \delta_{C_0}$. By uniqueness of intrinsic transition, we have $\eta_1 \setminus \{P\} = \eta_2 \setminus \{P\}$. □

Definition 132 ($X \setminus \{P\}$)**.** *(see figure 4.6 for the constructive definition and figures 4.7 and 4.8 for the desired result.) Let $P \in Autids$. Let X be a P-fair PCA, with $psioa(X) = (Q_X, \bar{q_X}, sig(X), D_X)$. We note $X \setminus \{P\}$ the automaton Y equipped with the same attributes than a PCA (psioa, config, hidden-actions, created), $\mu_s^P : Q_X \to Q_Y$ and $\mu_d^P : D_X \setminus \{\eta_{(X,q_X,a)} \in D_X | a \text{ is } P\text{-exclusive in } q_X\} \to D_Y$ that respect systematically the following rules:*

- *P-deprivation: $\forall q_Y \in Q_Y$, $P \notin config(Y)(q_Y)$, $\forall a \in \widehat{sig}(Y)(q_Y)(a), P \notin created(Y)(q_Y)(a)$.*

- μ_s^P-*correspondence:* $\forall (q_X, q_Y) \in Q_X \times Q_Y$ *s.t.* $\mu_s^P(q_X) = q_Y$, *then* $q_X R_{strict}^{\setminus\{P\}} q_Y$.
- μ_d^P-*correspondence:* $\forall ((q_Y, a, \eta_{(Y,q_Y,a_Y)}), (q_X, a, \eta_{(X,q_X,a_X)})) \in D_X \times D_Y$ *s.t.* $(q_Y, a, \eta_{(Y,q_Y,a_Y)}) = \mu_d^P(q_X, a, \eta_{(X,q_X,a_X)})$, *then*
  - $\mu_s^P(q_X) = q_Y$,
  - $a_X = a_Y$ *and*
  - $\forall q'_Y \in Q_Y$, $\eta_{(Y,q_Y,a)}(q'_Y) = \Sigma_{q'_X \in Q_X, \mu_s(q'_X)=q'_Y} \eta_{(X,q_X,a)}(q'_X)$.

and constructed (conjointly with the mapping μ_s^P *and* μ_d^P*) as follows:*

- *(Partitioning):*
  We partition Q_X *in equivalence classes according to the equivalence relation* $R_{conf}^{\setminus\{P\}}$ *that is we obtain a partition* $(C_j)_{j \in J \subset \mathbb{N}}$ *s.t.* $\forall j \in J$, $\forall q_X, q'_X \in C_j$, $q_X R_{conf}^{\setminus\{P\}} q'_X$ *and by P-fair assumption,* $q_X R_{strict}^{\setminus\{P\}} q'_X$
- *(*Q_Y*,* $sig(Y)$ *and* μ_s^P*):*
  $\forall j \in J$, *we construct* $q_Y^j \in Q_Y$ *and conjointly extend* μ_s^P *s.t.* $\forall q_X \in C_j$, $\mu_s^P(q_X) = q_Y^j$, *verifying the P-deprivation-rule and* μ_s^P*-correspondence rule, that is*
  - $config(Y)(q_Y^j) = config(X)(q_X) \setminus \{P\}$,
  - $hidden\text{-}actions(Y)(q_Y^j) = hidden\text{-}actions(X)(q_X) \setminus pot\text{-}out(X)(q_X)(P)$,
  - $sig(Y)(q_Y^j) = hide(sig(config(Y)(q_Y^j)), hidden\text{-}actions(Y)(q_Y^j))$
  - $\forall a \in \widehat{sig}(Y)(q_Y^j)$, $created(Y)(q_Y^j)(a) = created(X)(q_X)(a) \setminus \{P\}$.
  - *Furthermore* $\bar{q}_Y = \mu_s^P(\bar{q}_X)$.
- *(*D_Y *and* μ_d^P*):*
  $\forall q_Y \in Q_Y$, $\forall a \in \widehat{sig}(Y)(q_Y)$ *(and so* $\forall q_X \in (\mu_s^P)^{-1}(q_Y), a \in \widehat{sig}(X)(q_X)$*) we construct* $\eta_{(Y,q_Y,a)}$ *and conjointly extend* μ_d^P *s.t.* $\forall q_X \in (\mu_s^P)^{-1}(q_Y)$, $(q_Y, a, \eta_{(Y,q_Y,a_Y)}) = \mu_d^P(q_X, a, \eta_{(X,q_X,a_X)})$, *verifies the* μ_d^P*-correspondence rule. We show this construction is possible:*
  - *We note* $C_Y = config(Y)(q_Y)$, $\varphi_Y = created(Y)(q_Y)(a)$, η_Y *the unique element of* $Disc(Q_{conf})$ *s.t.* $C_Y \overset{a}{\Longrightarrow}_{\varphi_Y} \eta_Y$. *Let* $(q_X^i)_{i \in I \subset \mathbb{N}} = (\mu_s^P)^{-1}(q_Y)$. *For every* $i \in I$, *we note* $C_X^i = config(X)(q_X^i)$, $\varphi_X^i = created(X)(q_X^i)(a)$, η_X^i *the unique element of* $Disc(Q_{conf})$ *s.t.* $C_X^i \overset{a}{\Longrightarrow}_{\varphi_X^i} \eta_X^i$. *By lemma 27,* $\forall i \in I$, $C_X^i \setminus \{P\} = C_Y$, $\varphi^i \setminus \{P\} = \varphi_Y$ *and* $\eta_X^i \setminus \{P\} = \eta_Y$.
  - *For every* $q_X^i \in (\mu_s^P)^{-1}(q_Y)$, *we partition* $supp(\eta_{(X,q_X^i,a)})$ *in equivalence classes according to the equivalence relation* $R_{conf}^{\setminus\{P\}}$ *that is we obtain a partition* $(C'_j)_{j \in J' \subset \mathbb{N}}$ *s.t.* $\forall j \in J'$, $\forall q'_X, q''_X \in C'_j$, $q'_X R_{conf}^{\setminus\{P\}} q''_X$ *and by P-fair assumption,* $q'_X R_{strict}^{\setminus\{P\}} q''_X$. *For each* $j \in J'$, *we extract an arbitrary* $q'_X \in C'_j$ *and* $q'_Y = \mu_s^P(q'_X)$. *We fix* $\eta_{(Y,q_Y,a)}(q'_Y) := \eta_Y(C'_Y)$ *with* $C'_Y = config(Y)(q'_Y)$.

$$
\begin{aligned}
\eta_Y(C'_Y) &= \sum_{C'_X, C'_Y = C'_X \setminus \{P\}} \eta_X^i(C'_X) && \text{by lemma 26} \\
&= \sum_{q'_X, C'_Y = config(X)(q'_X) \setminus \{P\}} \eta_{(X,q_X^i,a)}(q'_X) && \text{by bottom/up transition preservation} \\
&= \sum_{q'_X, q'_Y = \mu_s^P(q'_X)} \eta_{(X,q_X^i,a)}(q'_X) && \text{By } \mu_s^P\text{-correspondence}
\end{aligned}
$$

Thus, the μ_d^P*-correspondence constraint holds for all the possible* $q_X^i \in (\mu_s^P)^{-1}(q_Y)$.

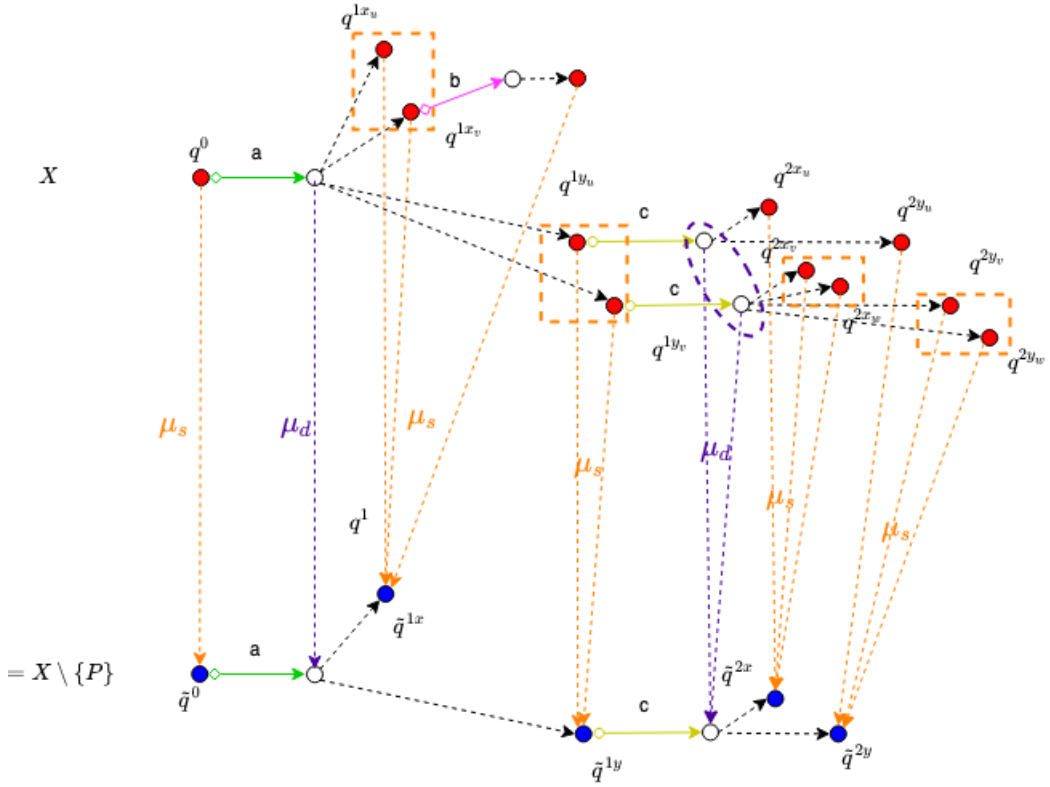


Figure 4.6. – constructive definition of PCA projection

constructive definition of $Y = X \setminus \{P\}$. First, we construct $\tilde{q}^0$ which is the initial state of Y. Then we partition $supp(\eta_{(X,q^0,a)}) = \{q^{1x_u}, q^{1x_v}\} \cup \{q^{1y_u}, q^{1y_v}\}$ s.t. $q^{1x_u} R_{conf}^{\setminus\{P\}} q^{1x_v}$ and $q^{1y_u} R_{conf}^{\setminus\{P\}} q^{1y_v}$. Thereafter we construct $q^{\tilde{1}x} = \mu_s(q^{1x_u}) = \mu_s(q^{1x_v})$ and $q^{\tilde{1}y} = \mu_s(q^{1y_u}) = \mu_s(q^{1y_v})$. Then, $\eta_{(Y,\tilde{q^0},a)}$ is defined s.t. $\eta_{(Y,\tilde{q^0},a)}(\tilde{q}^{1x}) = \eta_{(X,q^0,a)}(q^{1x_u}) + \eta_{(X,q^0,a)}(q^{1x_v})$ and $\eta_{(Y,\tilde{q^0},a)}(\tilde{q}^{1y}) = \eta_{(X,q^0,a)}(q^{1y_u}) + \eta_{(X,q^0,a)}(q^{1y_v})$. We perform another time this procedure. by partitioning $supp(\eta_{(X,q^{1y_u},a)}) = \{q^{2x_u}\} \cup \{q^{2y_u}\}$ or $supp(\eta_{(X,q^{1y_v},a)}) = \{q^{2x_v}, q^{2x_w}\} \cup \{q^{2y_v}, q^{2y_w}\}$ arbitrarily. Indeed the obtai,ed result is the same: (i) $q^{1y_u} R_{conf}^{\setminus\{P\}} q^{1y_v}$ since they are both pre-image of $\tilde{q}^{1y}$ by μ_s, which means (ii) $q^{1y_u} R_{strict}^{\setminus\{P\}} q^{1y_v}$ since X is assumed to be P-fair. If we note $C_u = config(X)(q^{1y_u})$, $C_v = config(X)(q^{1y_v})$, $\varphi_u = created(X)(q^{1y_u})(c)$, $\varphi_v = created(X)(q^{1y_v})(c)$, $C_u \overset{c}{\Longrightarrow}_{\varphi_u} \eta_u$ and $C_v \overset{c}{\Longrightarrow}_{\varphi_v} \eta_v$ we have j) $C_u \setminus \{P\} = C_v \setminus \{P\}$, jj) $C_u \setminus \{P\} \overset{c}{\Longrightarrow}_{\varphi_u \setminus \{P\}} \eta_u \setminus \{P\}$ and jjj) $C_v \setminus \{P\} \overset{c}{\Longrightarrow}_{\varphi_v \setminus \{P\}} \eta_v \setminus \{P\}$ which implies jv) $\eta_u \setminus \{P\} = \eta_v \setminus \{P\}$.

In the remaining, if we consider a PCA X deprived of a PSIOA $\mathcal{A}$ we always implicitly assume that X is $\mathcal{A}$-fair.

4.2.3. $Y = X \setminus \{\mathcal{A}\}$ is a PCA if X is $\mathcal{A}$-fair

Here we prove a sequence of lemma to show that $Y = X \setminus \{P\}$ is indeed a PCA, by verifying all the constraints.

Prepare the top/down transition preservation We show a useful lemma to show $Y = X \setminus \{\mathcal{A}\}$ verifies the constraint 2 of top/down transition preservation.

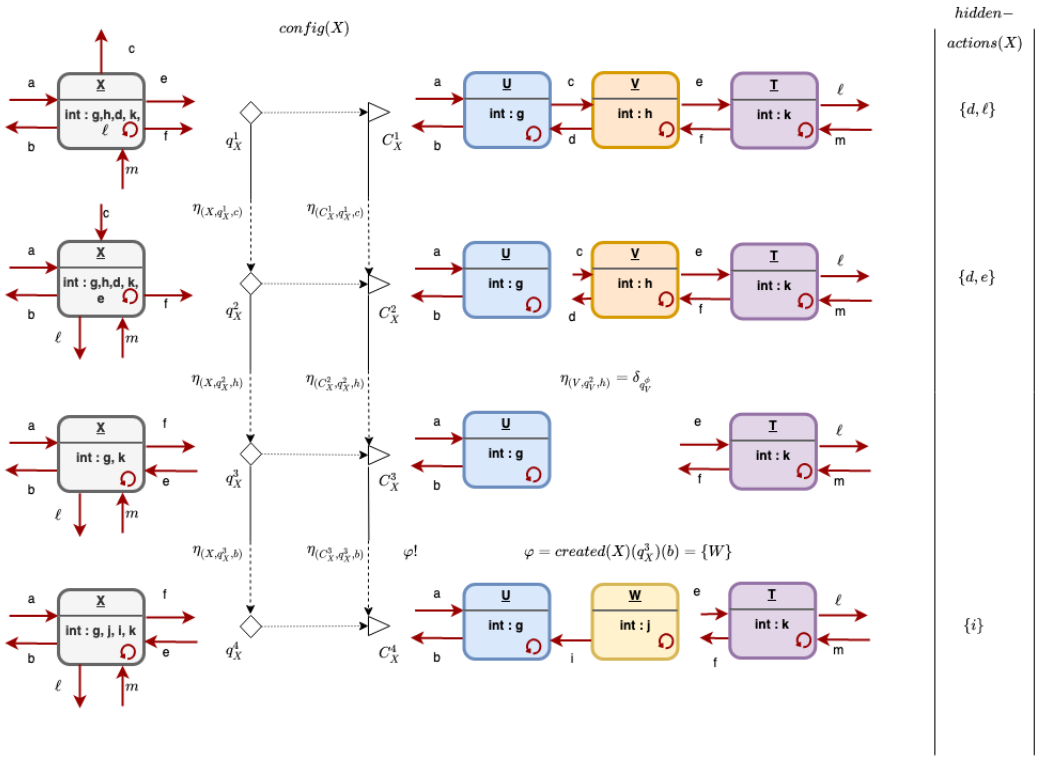


Figure 4.7. – Projection on PCA (part 1/2, the part 2/2 is in figure 4.8): the original PCA X

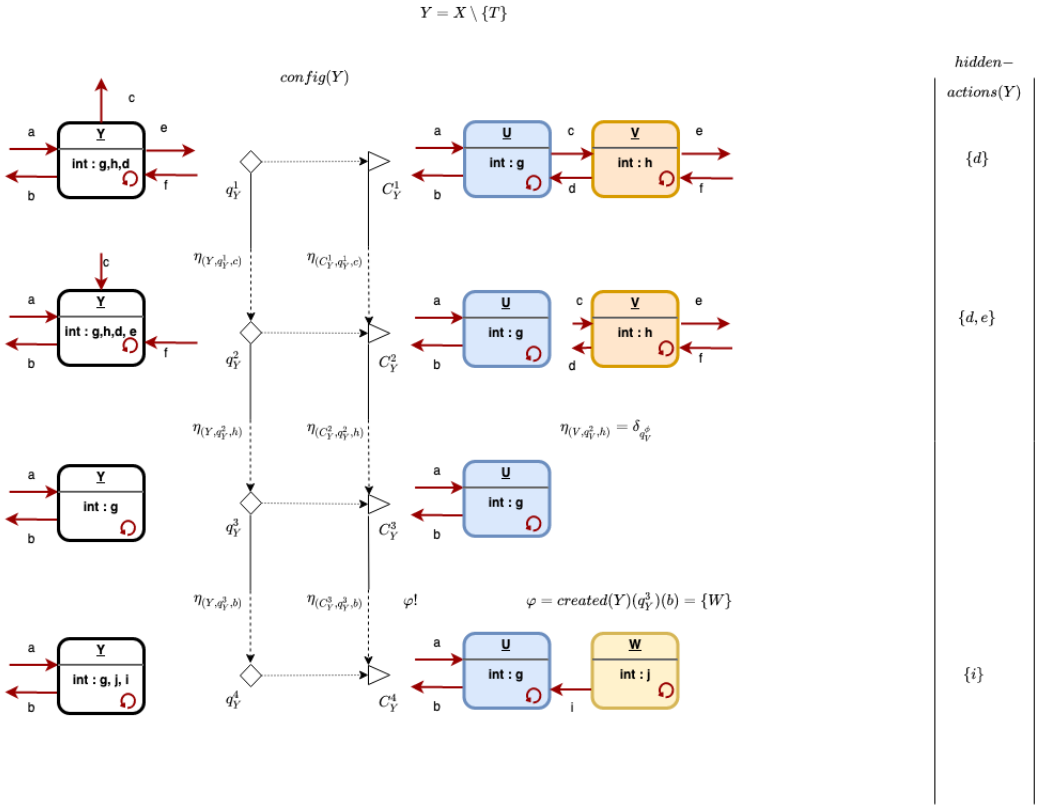


Figure 4.8. – Projection on PCA (part 2/2, the part 1/2 is in figure 4.7): the PCA $Y = X \setminus \{T\}$

Lemma 28 (corresponding transition after projection)**.** *Let $\mathcal{A}$ be a PSIOA. Let X be a $\mathcal{A}$-fair PCA and $Y = X \setminus \mathcal{A}$. $((q_X, a, \eta_X), (q_Y, a, \eta_Y)) \in D_X \times D_Y$, s.t. $(q_Y, a, \eta_{(Y,q_Y,a)}) = \mu_d(q_X, a, \eta_{(X,q_X,a)})$. For each $K \in \{X, Y\}$, we note $C_K = config(K)(q_K)$, $\varphi_K = created(K)(q_K)(a)$.*

Let η'_X the unique element of $Disc(Q_{conf})$ s.t. x0) $\eta_{(X,q_X,a)} \overset{c}{\leftrightarrow} \eta'_X$ with x1) $c = config(X)$ and x2) $C_X \overset{a}{\Longrightarrow}_{\varphi_X} \eta'_X$.

Let $\eta'_Y = \eta'_X \setminus \{\mathcal{A}\}$. Then η'_Y verifies y0) $\eta_{(Y,q_Y,a)} \overset{c'}{\leftrightarrow} \eta'_Y$ with y1) $c' = config(Y)(q_Y)$ and y2) $Config(Y)(q_Y) \overset{a}{\Longrightarrow}_{\varphi_Y} \eta'_Y$.

Proof. We note $(Q_i^X)_{i\in\mathcal{I}}$ the partition of $supp(\eta_{X,q_X,a})$ s.t. $\forall i \in \mathcal{I}, \forall q'_X, q''_X \in Q_i^X, q'_X R_{conf}^{\setminus\{\mathcal{A}\}} q''_X$. $\forall i \in \mathcal{I}$, we note $C_i^{\setminus\{\mathcal{A}\}} = config(q'_X)\setminus\{\mathcal{A}\}$ for an arbitrary element $q'_X \in Q_i^X$ and $C_i = \{C \in supp(\eta'_X) | C \setminus \mathcal{A} = C_i^{\setminus\{\mathcal{A}\}}\}$. Since x0) $\eta_{(X,q_X,a)} \overset{f}{\leftrightarrow} \eta'_X$ with x1) $f = config(X)(q_X)$, $(C_i)_{i\in\mathcal{I}}$ is a partition of $supp(\eta'_X)$.

For every $i \in \mathcal{I}$, we note $q_i^Y = \mu_s(q'_X)$ for an arbitrary element $q'_X \in Q_i^X$. By $\mu_s^{\mathcal{A}}$-correspondance, $config(q_i^Y) = C_i^{\setminus\{\mathcal{A}\}} = config(q'_X) \setminus \{\mathcal{A}\}$

By $\mu_d^{\mathcal{A}}$-correspondence,

$$\begin{aligned}\eta_{(Y,q_Y,a)}(q'_Y) &= \Sigma_{q'_X, \mu_s(q'_X)=q'_Y} \eta_{(X,q_X,a)}(q'_X) \\ &= \Sigma_{i\in I} \Sigma_{q'_X \in Q_i^X, \mu_s(q'_X)=q'_Y} \eta_{(X,q_X,a)}(q'_X)\end{aligned}$$

By assumption x0) and x1), $\eta_{(X,q_X,a)} \overset{c}{\leftrightarrow} \eta'_X$ with $c = config(X)$, thus

$$\begin{aligned}\eta_{Y,q_y,a}(q'_y) &= \Sigma_{i\in\mathcal{I}} \Sigma_{q'_X \in Q_i^X, \mu_s(q'_X)=q'_Y} \eta'_X(config(X)(q'_X)) \\ &= \Sigma_{i\in\mathcal{I}} \Sigma_{C'_X \in C_i, C'_X \setminus \mathcal{A} = config(q'_Y)} \eta'_X(C'_X) \\ &= \Sigma_{C'_X, C'_X \setminus \mathcal{A} = config(q'_Y)} \eta'_X(C'_X)\end{aligned}$$

Therafter, we use the lemma 26 and get $\eta_{(Y,q_y,a)}(q'_Y) = \eta'_Y(config(Y)(q'_Y))$ with $\eta'_Y = \eta'_X \setminus \{\mathcal{A}\}$. By definition of Y, $Config(Y)(q_Y = \mu_s(q_X)) = Config(X)(q_X) \setminus \{\mathcal{A}\}$. We can apply lemma 26. Since $a \in \widehat{sig}(config(X)(q_X) \setminus \{\mathcal{A}\})$, $Config(Y)(q_Y) \overset{a}{\Longrightarrow}_{\varphi_Y} \eta'_Y$ with $\eta'_Y = \eta'_X \setminus \{\mathcal{A}\}$ and $\varphi_Y = (\varphi_X \setminus \{\mathcal{A}\})$. By $\mu_s^{\mathcal{A}}$-correspondance, $created(Y)(q_Y)(a) = created(X)(q_X)(a) \setminus \{\mathcal{A}\}$, thus $\varphi_Y = created(Y)(q_Y)(a)$.

Finally the restriction of $config(Y)$ on $supp(\eta_{(Y,q_Y,a)})$ is a bijection. Indeed, we note $f_1 : q_Y \mapsto Q_i^X$ s.t. $\{q_Y\} = \mu_s(Q_i^X)$, $f_2 : Q_i^X \mapsto C_i$ $f_3 : C_i \mapsto C_i^{\setminus\mathcal{A}}$. By construction, f_1 and f_3 are bijection. By bijectivity of the restriction of $config(X)$ on $supp(\eta_{X,q_X,a})$, f_2 is a bijection too. Moreover, the restriction f' of $config(Y)$ on $supp(\eta_{Y,q_Y,a})$ is $f_1 \circ f_2 \circ f_3$ and hence this is a bijection too.

□

Now we are able to demonstrate that the PCA set is closed under deprivation.

Theorem 17 ($X \setminus \{P\}$ is a PCA)**.** *Let $P \in Autids$. Let X be a P-fair PCA, then $Y = X \setminus \{P\}$ is a PCA.*

Proof. – (Constraint 1) By construction of Y, $\bar{q}_Y = \mu_s^P(\bar{q}_X)$ and by μ_s-correspondence rule, $config(Y)(\bar{q}_Y) = config(X)(\bar{q}_X) \setminus \{P\}$. Since constraint 1 is respected by X, it is a fortiori respected by Y.

– (Constraint 2) Let $(q_Y, a, \eta_{(Y,q_Y,a)}) \in D_Y$. By construction of Y, we know it $\exists (q_X, a, \eta_{(X,q_X,a)}) \in D_X$ with $\eta_{(Y,q_Y,a)} = \mu_d(\eta_{(X,q_X,a)})$ and $q_Y = \mu_s(q_X)$. Then, because of constraint 2 ensured by X, we obtain it exists a reduced configuration distribution $\eta'_X \in Disc(Q_{conf})$ s.t. x0) $\eta_{(X,q_X,a)} \overset{c}{\leftrightarrow} \eta'_X$ with x1) $c = config(X)$ and x2) $Config(X)(q_X) \overset{a}{\Longrightarrow}_{\varphi_X} \eta'_X$ where $\varphi_X = created(X)(q_X)(a)$. We can apply lemma 28 to obtain that $\eta'_Y = \eta'_X \setminus \{P\}$ is a reduced configuration transition that verifies y0) $\eta_{(Y,q_Y,a)} \overset{c'}{\leftrightarrow} \eta'_Y$ with y1) $c' = config(Y)$ and y2) $config(Y)(q_Y) \overset{a}{\Longrightarrow}_{\varphi_Y} \eta'_Y$ where $\varphi_Y = \varphi_X \setminus \{P\} = created(Y)(q_Y)(a)$.
This terminates the proof of constraint 2.

– (Constraint 3) Let $q_Y \in Q_Y, C_Y = config(Y)(q_Y), a \in \widehat{sig}(C_Y), \varphi_Y = created(Y)(q_Y)(a)$, $\eta'_Y \in Disc(Q_{conf})$ s.t. $C_Y \overset{a}{\Longrightarrow} \eta'_Y$.
By construction of $Y = X \setminus \{P\}$, if $q_Y \in Q_Y$, $\exists q_X \in Q_X$, $\mu_s(q_X) = q_Y$, $C_X = config(X)(q_X)$, $C_X \setminus \{P\} = C_Y$. Necessarily, $a \in \widehat{sig}(C_X)$ and by construction of $Y = X \setminus \{P\}$, $\varphi_X \setminus \{P\} = \varphi_Y$ with $\varphi_X = created(X)(q_X)(a)$. We note η'_X verifying $C_X \overset{a}{\Longrightarrow}_{\varphi_X} \eta'_X$. By lemma 26, $\eta'_Y = \eta'_X \setminus \{\mathcal{A}\}$.
Because of constraint 3, it means $(q_X, a, \eta_{X,q_X,a}) \in D_X$ with x0) $\eta_{(X,q_X,a)} \overset{c}{\leftrightarrow} \eta'_X$ with x1) $c = config(X)$. Since $q_Y = \mu_s(q_Y)$ and $a \in \widehat{sig}(Y)(q_Y)$, the construction of D_Y implies $(q_Y, a, \eta_{(Y,q_Y,a)}) \in D_Y$ with $(q_Y, a, \eta_{(Y,q_Y,a)}) = \mu_d^P((q_X, a, \eta_{(X,q_X,a)}))$.
We can apply lemma 28 to obtain that η'_Y verifies y0) $\eta_{(Y,q_Y,a)} \overset{c'}{\leftrightarrow} \eta'_Y$ with y1) $c' = config(Y)$ and y2) $C_Y \overset{a}{\Longrightarrow}_{\varphi_Y} \eta'_Y$.
This terminates the proof of constraint 3.

– (Constraint 4) Verified by construction (We recall that $\forall (q_Y, q_X) \in Q_Y \times Q_X, q_Y = \mu_s^P(q_X), sig(Y)(q_Y) \triangleq hide(sig(config(Y)(q_Y), hidden\text{-}actions(Y)(q_Y))$
where $hidden\text{-}actions(Y)(q_Y) \triangleq hidden\text{-}actions(X)(q_X) \setminus pot\text{-}out(X)(q_X)(P)$.

□

4.3. Reconstruction

In the previous section, we have shown that $Y = X \setminus \mathcal{A}$ is a PCA (as long as X is $\mathcal{A}$-fair). In this section, we will

1. introduce the concept of simpleton wrapper $\tilde{\mathcal{A}}^{sw}$ that is a PCA that encapsulates $\mathcal{A}$.
2. prove that $X \setminus \{\mathcal{A}\}$ and $\tilde{\mathcal{A}}^{sw}$ are partially-compatible (see theorem 18)
3. There is a strong executions-matching from X to $(X \setminus \{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw}$ in a restricted set of executions of X that do not create $\mathcal{A}$ (see theorem 19). Hence it is always possible to transfer a reasoning on X into a reasoning on $(X \setminus \{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw}$ if no re-creation of $\mathcal{A}$ occurs.
4. The operation of projection/deprivation and composition are commutative (see theorem 21).

4.3.1. Simpleton wrapper : $\tilde{A}^{sw}$

Here we introduce simpleton wrapper $\tilde{A}^{sw}$, a PCA that only encapsulates $\tilde{A}^{sw}$

Definition 133 (Simpleton wrapper)**.** *(see figure 4.9) Let $\mathcal{A}$ be a PSIOA. We note $\tilde{\mathcal{A}}^{sw}$ the* simpleton wrapper *of $\mathcal{A}$ as the following PCA:*

- $psioa(\tilde{\mathcal{A}}^{sw}) = \mathcal{A}$
- $config(\tilde{\mathcal{A}}^{sw})(q^{\phi}_{\mathcal{A}}) = (\emptyset, \emptyset)$
- $\forall q \in Q_{\mathcal{A}}, q_{\mathcal{A}} \neq q^{\phi}_{\mathcal{A}}, config(\tilde{\mathcal{A}}^{sw})(q) = (\mathcal{A}, \{(\mathcal{A}, q)\})$
- $\forall q \in Q_{\mathcal{A}}, \forall a \in \widehat{sig}(\tilde{\mathcal{A}}^{sw})(q), created(\tilde{\mathcal{A}}^{sw})(q)(a) = \emptyset$
- $\forall q \in Q_{\mathcal{A}}, hidden\text{-}actions(\tilde{\mathcal{A}}^{sw})(q) = \emptyset$

We can remark that when $\tilde{\mathcal{A}}^{sw}$ enters in $q^{\phi}_{\tilde{\mathcal{A}}^{sw}} = q^{\phi}_{\mathcal{A}}$ where $\widehat{sig}(\tilde{\mathcal{A}}^{sw})(q^{\phi}_{\tilde{\mathcal{A}}^{sw}}) = \emptyset$, this matches the moment where $\mathcal{A}$ enters in $q^{\phi}_{\mathcal{A}}$ where $\widehat{sig}(\mathcal{A})(q^{\phi}_{\mathcal{A}}) = \emptyset$, s.t. the corresponding configuration is the empty one.

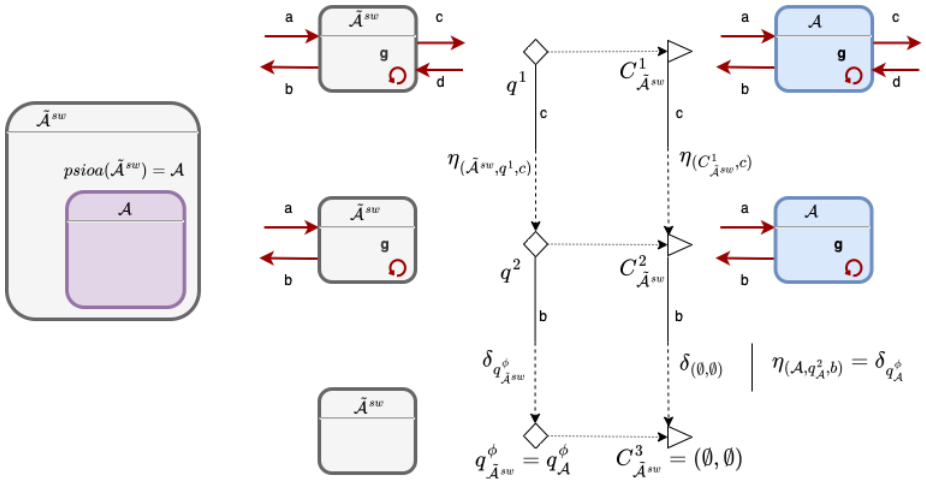


Figure 4.9. – Simpleton wrapper

4.3.2. Partial-compatibility of $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})$ and $\tilde{\mathcal{A}}^{sw}$

In this subsection, we show that $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})$ and $\tilde{\mathcal{A}}^{sw}$ are partially-compatible and that $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw}$ mimics $X_{\mathcal{A}}$ as long as no creation of $\mathcal{A}$ occurs (see figure 4.10).

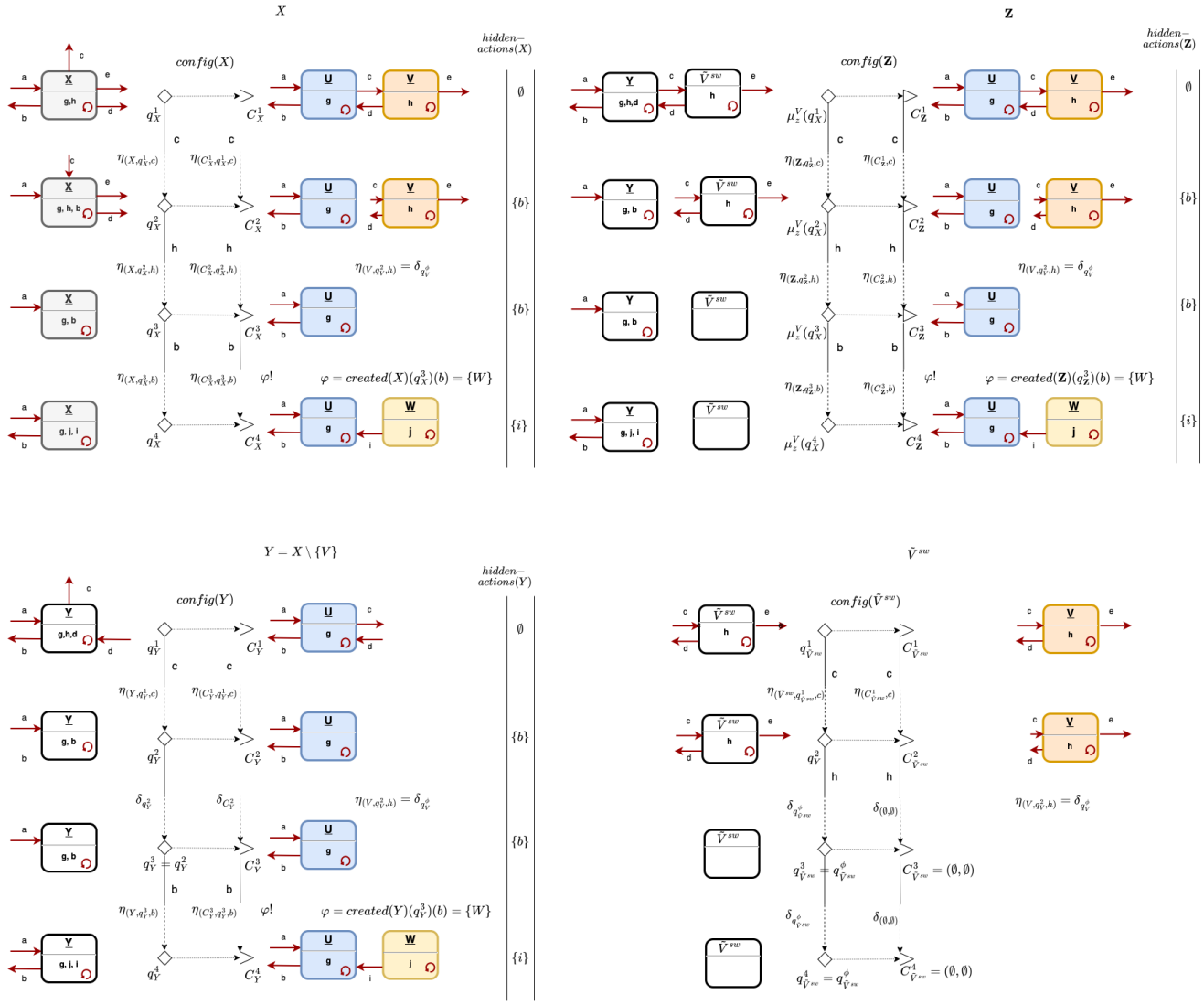


Figure 4.10. – Reconstruction of a PCA via $\mathbf{Z} = (X, X \setminus \{V\})$

Map X and $(X \setminus \{\mathcal{A}\}, \tilde{\mathcal{A}}^{sw})$ We first introduce two functions to map X and $(X \setminus \{\mathcal{A}\}, \tilde{\mathcal{A}}^{sw})$.

Definition 134 ($\mu_z^{\mathcal{A}}$ and $\mu_e^{\mathcal{A}}$: mapping of reconstruction)**.** *Let $\mathcal{A} \in Autids$, X be a $\mathcal{A}$-fair PCA, $Y = X \setminus \mathcal{A}$. Let $\tilde{\mathcal{A}}^{sw}$ be the simpleton wrapper of $\mathcal{A}$. Let $q_{\mathcal{A}}^{\phi} \in Q_{\mathcal{A}}$ the (assumed) unique state s.t. $\widehat{sig}(\mathcal{A})(q_{\mathcal{A}}^{\phi}) = \emptyset$. We note:*

- *The function $X.\mu_z^{\mathcal{A}} : Q_X \to Q_Y \times Q_{\tilde{\mathcal{A}}^{sw}}$ s.t. $\forall q_X \in Q_X$, $X.\mu_z^{\mathcal{A}}(q_X) = (X.\mu_s^{\mathcal{A}}(q_X), q_{\mathcal{A}})$ with $q_{\mathcal{A}} = map(config(X)(q_X))(\mathcal{A})$ if $\mathcal{A} \in (auts(config(X)(q_X)))$ and $q_{\mathcal{A}} = q_{\mathcal{A}}^{\phi}$ otherwise.*
- *The function $X.\mu_e^{\mathcal{A}}$ that maps any alternating sequence $\alpha_X = q_X^0, a^1, q_X^1, a^2 \ldots$ of states and actions of X, to $\mu_e^{\mathcal{A}}(\alpha_X)$ the alternating sequence $\alpha_Z = X.\mu_z^{\mathcal{A}}(q_X^0), a^1, X.\mu_z^{\mathcal{A}}(q_X^1), a^2, \ldots$.*

The symbols $^{\mathcal{A}}$ and $X.$ are omitted when this is clear in the context.

Now, we recall definition 105 of $\mathcal{A}$-conservative PCA, an additional condition to allow the compatibility between $X \setminus \mathcal{A}$ and $\tilde{\mathcal{A}}^{sw}$.

Definition 135 ($\mathcal{A}$-conservative PCA (recall))**.** *Let X be a PCA, $\mathcal{A} \in Autids$. We say that X is $\mathcal{A}$-conservative if it is $\mathcal{A}$-fair and for every state $q_X \in Q_X$, $C_X = (\mathbf{A}_X, \mathbf{S}_X) = config(X)(q_X)$ s.t. $\mathcal{A} \in \mathbf{A}_X$ and $\mathbf{S}_X(\mathcal{A}) \triangleq q_{\mathcal{A}}$, $hidden\text{-}actions(X)(q_X) = hidden\text{-}actions(X)(q_X) \setminus \widehat{ext}(\mathcal{A})(q_{\mathcal{A}})$.*

A $\mathcal{A}$-conservative PCA is a $\mathcal{A}$-fair PCA that does not hide any output action that could be an external action of $\mathcal{A}$.

Preservation of properties Now we start a sequence of lemma (from lemma 29 to lemma 33) about properties preserved after reconstruction to eventually show in theorem 18 that $X \setminus \mathcal{A}$ and $\tilde{\mathcal{A}}^{sw}$ are partially-compatible.

The next lemma shows that reconstruction preserves signature compatibility.

Lemma 29 (preservation of signature compatibility of configurations)**.** *Let $\mathcal{A} \in Autids$. Let X be a $\mathcal{A}$-conservative PCA, $Y = X \setminus \mathcal{A}$. Let $q_X \in Q_X$, $C_X = (\mathbf{A}_X, \mathbf{S}_X) = config(X)(q_X)$. Let $q_Y \in Q_Y, q_Y = \mu_s(q_X)$. Let $C_Y = (\mathbf{A}_Y, \mathbf{S}_Y) = config(Y)(q_Y)$.*

If $\mathcal{A} \in \mathbf{A}_X$ and $q_\mathcal{A} = \mathbf{S}_X(\mathcal{A})$, then $sig(C_Y)$ and $sig(\tilde{\mathcal{A}}^{sw})(q_\mathcal{A})$ are compatible and $sig(C_X) = sig(C_Y) \times sig(\tilde{\mathcal{A}}^{sw})(q_\mathcal{A})$.

If $\mathcal{A} \notin \mathbf{A}_X$, then $sig(C_Y)$ and $sig(\tilde{\mathcal{A}}^{sw})(q_\mathcal{A}^\phi)$ are compatible and $sig(C_X) = sig(C_Y) \times sig(\tilde{\mathcal{A}}^{sw})(q_\mathcal{A}^\phi)$.

Proof. Let $\mathcal{A} \in Autids$ Let X and $Y \setminus \{\mathcal{A}\}$ be PCA. Let $q_X \in Q_X$. Let $C_X = config(X)(q_X)$, $\mathbf{A}_X = auts(C_X)$ and $\mathbf{S}_X = map(C_X)$. Let $q_Y \in Q_Y, q_Y = \mu_s(q_X)$. Let $C_Y = config(Y)(q_Y)$, $\mathbf{A}_Y = auts(C_Y)$ and $\mathbf{S}_Y = map(C_Y)$. By definition of Y, $C_Y = C_X \setminus \{\mathcal{A}\}$.

Case 1: $\mathcal{A} \in \mathbf{A}_X$

Since X is a PCA, C_X is a compatible configuration, thus $((\mathbf{A}_Y, \mathbf{S}_Y) \cup (\mathcal{A}, q_\mathcal{A}))$ is a compatible configuration. Finally $sig(C_Y)$ and $sig(\mathcal{A})(q_\mathcal{A})$ are compatible with $sig(\mathcal{A})(q_\mathcal{A}) = sig(\tilde{\mathcal{A}}^{sw})(q_\mathcal{A}^\phi)$.

By definition of intrinsinc attributes of a configuration, that are constructed with the attributes of the automaton issued from the composition of the family of automata of the configuration, we have $\mathbf{A}_X = \mathbf{A}_Y \cup \{\mathcal{A}\}$ and $sig(C_X) = sig(C_Y) \times sig(\mathcal{A})(q_\mathcal{A})$, that is $sig(C_X) = sig(C_Y) \times sig(\tilde{\mathcal{A}}^{sw})(q_\mathcal{A})$.

Case 2: $\mathcal{A} \notin \mathbf{A}_X$

Since X is a PCA, C_X is a compatible configuration, thus $C_Y = C_X$ is a compatible configuration. Finally $sig(C_Y)$ and $sig(\mathcal{A})(q_\mathcal{A}^\phi) = (\emptyset, \emptyset, \emptyset) = sig(\mathcal{A})(q_\mathcal{A}) = sig(\tilde{\mathcal{A}}^{sw})(q_\mathcal{A}^\phi)$ are compatible.

By definition of intrinsinc attributes of a configuration, that are constructed with the attributes of the automaton issued from the composition of the family of automata of the configuration (here $\mathbf{A}_Y$ and $\mathbf{A}_X = \mathbf{A}_Y$), we have $sig(C_X) = sig(C_Y)$. Furthermore, $sig(\tilde{\mathcal{A}}^{sw})(q_\mathcal{A}^\phi) = sig(\mathcal{A})(q_\mathcal{A}^\phi) = (\emptyset, \emptyset, \emptyset)$. Thus $sig(C_X) = sig(C_Y) \times sig(\tilde{\mathcal{A}}^{sw})(q_\mathcal{A}^\phi)$ □

The next lemma shows that reconstruction preserves signature.

Lemma 30 (preservation of signature)**.** *Let $\mathcal{A} \in Autids$. Let X be a $\mathcal{A}$-conservative PCA, $\mathcal{A} \in Autids$, $Y = X \setminus \{\mathcal{A}\}$. For every $q_X \in Q_X$, we have $sig(X)(q_X) = sig(Y)(q_Y) \times sig(\tilde{\mathcal{A}}^{sw})(q_\mathcal{A})$ with $(q_Y, q_\mathcal{A}) = \mu_z^\mathcal{A}(q_X)$.*

Proof. We note $C_X = config(X)(q_X)$ and $C_Y = config(Y)(q_Y)$. The last lemma 29 tells us for every $q_X \in Q_X$, we have $sig(C_X) = sig(C_Y) \times sig(\tilde{\mathcal{A}}^{sw})(q_\mathcal{A})$ with $(q_Y, q_\mathcal{A}) = \mu_z(q_X)$. Since X is $\mathcal{A}$-conservative, we have (*) $sig(X)(q_X) = hide(sig(C_X), \underline{acts})$ where $\underline{acts} \subseteq (out(X)(q_X) \setminus (ext(\mathcal{A})(q_\mathcal{A}))$. Hence $sig(Y)(q_Y) = hide(sig(C_Y), \underline{acts})$. Since (**) $\underline{acts} \cap ext(\mathcal{A})(q_\mathcal{A}) = \emptyset$, $sig(Y)(q_Y)$ and $sig(\mathcal{A})(q_\mathcal{A})$ are also compatible. We have $sig(C_X) = sig(C_Y) \times sig(\mathcal{A})(q_\mathcal{A}) = sig(C_Y) \times sig(\tilde{\mathcal{A}}^{sw})(q_\mathcal{A})$ which gives because of (*) $hide(sig(C_X), \underline{acts}) = hide(sig(C_Y), \underline{acts}) \times sig(\mathcal{A})(q_\mathcal{A})$, that is $sig(X)(q_X) = sig(Y)(q_Y) \times sig(\mathcal{A})(q_\mathcal{A}) = sig(Y)(q_Y) \times sig(\tilde{\mathcal{A}}^{sw})(q_\mathcal{A})$.

□

The next lemma shows that reconstruction preserves partial-compatibility at any reachable state.

Lemma 31 (preservation of compatibility at any reachable state)**.** *Let $\mathcal{A} \in Autids$, X be a $\mathcal{A}$-conservative PCA, $Y = X \setminus \{\mathcal{A}\}$, $\mathbf{Z} = (Y, \tilde{\mathcal{A}}^{sw})$ Let $q_Z = (q_Y, \tilde{q}_{\tilde{\mathcal{A}}^{sw}}) \in Q_Y \times Q_{\tilde{\mathcal{A}}^{sw}}$ and $q_X \in Q_X$ s.t. $\mu_z^{\mathcal{A}}(q_X) = q_Z$. Then $psioa(Y)$ and $psioa(\tilde{\mathcal{A}}^{sw})$ are compatible. Moreover, by definition of $Y = X \setminus \{\mathcal{A}\}$ and $\tilde{\mathcal{A}}^{sw}$ being the simpleton wrapper of $\mathcal{A}$, the sub-automaton exclusivity and creation exclusivity of definition 70 are necessarily ensured. Hence, $\mathbf{Z}$ is compatible at state q_Z.*

Proof. Since X is a $\mathcal{A}$-conservative PCA, the previous lemma 30 ensures that $sig(Y)(q_Y)$ and $sig(\mathcal{A})(q_{\mathcal{A}}) = sig(\tilde{\mathcal{A}}^{sw})(q_{\mathcal{A}})$ are compatible, thus by definition $\mathbf{Z}$ is compatible at state q_Z. □

Here, we show that reconstruction preserves the probabilistic distribution of the corresponding transition, as long as no creation of the concerned automaton occurs.

Lemma 32 (homomorphic transition without creation)**.** *Let $\mathcal{A} \in Autids$, X be a $\mathcal{A}$-conservative PCA, $Y = X \setminus \{\mathcal{A}\}$, $\mathbf{Z} = (Y, \tilde{\mathcal{A}}^{sw})$. Let $q_Z = (q_Y, \tilde{q}_{\tilde{\mathcal{A}}^{sw}}) \in Q_Y \times Q_{\tilde{\mathcal{A}}^{sw}}$ and $q_X \in Q_X$ s.t. (i) $\mu_z(q_X) = q_Z$. Let $a \in sig(X)(q_X) = sig(Y)(q_Y) \times sig(\tilde{\mathcal{A}}^{sw})(\tilde{q}_{\tilde{\mathcal{A}}^{sw}})$, verifying (ii): No creation from $\mathcal{A}$, i.e. if a is $\mathcal{A}$-exclusive in state q_X,then $created(X)(q_X)(a) = \emptyset$, then*

- *If $\mathcal{A}$ is not created by a, i.e. if either*
  - *$\mathcal{A} \in auts(config(X)(q_X))$, or*
  - *$\mathcal{A} \notin auts(config(X)(q_X))$ and $\mathcal{A} \notin created(X)(q_X)(a)$ (X does not create $\mathcal{A}$ with probability 1)*

  Then $\eta_{(X,q_X,a)} \overset{\mu_z}{\leftrightarrow} \eta_{(\mathbf{Z},q_Z,a)}$
- *If $\mathcal{A}$ is created by a i.e. $\mathcal{A} \notin auts(config(X)(q_X))$ and $\mathcal{A} \in created(X)(q_X)(a)$ (X creates $\mathcal{A}$ with probability 1)*

  Then $\eta_{(X,q_X,a)} \overset{f^{\phi}}{\leftrightarrow} \eta_{(\mathbf{Z},q_Z,a)}$ where $f^{\phi} : q'_X \in supp(\eta_{(X,q_X,a)}) \mapsto (X.\mu_s^{\mathcal{A}}(q'_X), q^{\phi}_{\tilde{\mathcal{A}}^{sw}})$.

Proof. By lemma 30, we have $sig(X)(q_X) = sig(Y)(q_Y) \times sig(\mathcal{A})(q_{\mathcal{A}}) = sig(Y)(q_Y) \times sig(\tilde{\mathcal{A}}^{sw})(\tilde{q}_{\tilde{\mathcal{A}}^{sw}} = q_{\mathcal{A}})$.

We note $C_X = (\mathbf{A}_X, \mathbf{S}_X) = config(X)(q_X)$, $C_Y = (\mathbf{A}_Y, \mathbf{S}_Y) = config(Y)(q_Y)$, $C_{\tilde{\mathcal{A}}^{sw}} = (\mathbf{A}_{\tilde{\mathcal{A}}^{sw}}, \mathbf{S}_{\tilde{\mathcal{A}}^{sw}}) = config(\tilde{\mathcal{A}}^{sw})(q_{\tilde{\mathcal{A}}^{sw}})$. By construction of μ_z, $C_X = C_Y \cup C_{\tilde{\mathcal{A}}^{sw}}$ with C_Y and $C_{\tilde{\mathcal{A}}^{sw}}$ compatible configuration (1).

We note $\varphi_X = created(X)(q_X)(a)$, $\varphi_Y = \varphi_X \setminus \{\mathcal{A}\}$, $\varphi_{\tilde{\mathcal{A}}^{sw}} = \emptyset$, $\varphi_Z = \varphi_Y \cup \varphi_{\tilde{\mathcal{A}}^{sw}} = \varphi_Y$.

- If a is $\mathcal{A}$-exclusive in state q_X, then $\varphi_X = \varphi_Y = \varphi_Z = \emptyset$ by assumption.
- If $\mathcal{A}$ is not created by a, then $\varphi_X = \varphi_Z$,
- If $\mathcal{A}$ is created by a, then $\varphi_X = \varphi_Z \cup \{\mathcal{A}\}$ and $\varphi_Z = \varphi_X \setminus \{\mathcal{A}\}$

We note $(\eta_X, \eta_Y, \eta_{\tilde{\mathcal{A}}^{sw}}) \in Disc(Q_X) \times Disc(Q_Y) \times Disc(Q_{\tilde{\mathcal{A}}^{sw}})$ and $(\eta_X, \eta_Y, \eta_{\tilde{\mathcal{A}}^{sw}}) \in Disc(Q_{conf})^3$ such that:

- $\eta_X = \eta_{(X,q_X,a)}$
- $\eta_Y = \eta_{(Y,q_Y,a)}$ if $a \in sig(Y)(q_Y)$ and $\eta_Y = \delta_{q_Y}$ otherwise
- $\eta_{\tilde{\mathcal{A}}^{sw}} = \eta_{(\tilde{\mathcal{A}}^{sw},q_{\tilde{\mathcal{A}}^{sw}},a)}$ if $a \in sig(\tilde{\mathcal{A}}^{sw})(q_{\tilde{\mathcal{A}}^{sw}})$ and $\eta_{\tilde{\mathcal{A}}^{sw}} = \delta_{q_{\tilde{\mathcal{A}}^{sw}}}$ otherwise

- $C_X \stackrel{a}{\Longrightarrow}_{\varphi_X} \eta'_X$
- $C_Y \stackrel{a}{\Longrightarrow}_{\varphi_Y} \eta'_Y$ if $a \in sig(Y)(q_Y)$ and $\eta'_Y = \delta_{C_Y}$ otherwise
- $C_{\tilde{\mathcal{A}}^{sw}} \stackrel{a}{\Longrightarrow}_{\varphi_{\tilde{\mathcal{A}}^{sw}}} \eta'_{\tilde{\mathcal{A}}^{sw}}$ if $a \in sig(\tilde{\mathcal{A}}^{sw})(q_{\tilde{\mathcal{A}}^{sw}})$ and $\eta'_{\tilde{\mathcal{A}}^{sw}} = \delta_{C_{\tilde{\mathcal{A}}^{sw}}}$ otherwise

The constraint 2 of top/down transition preservation applied to PCA implies that:

- $\eta_X \stackrel{c^X}{\leftrightarrow} \eta'_X$ with $c^X = config(X)$ (2)
- $\eta_Y \stackrel{c^Y}{\leftrightarrow} \eta'_Y$ with $c^Y = config(Y)$ (3)
- $\eta_{\tilde{\mathcal{A}}^{sw}} \stackrel{c^{\tilde{\mathcal{A}}^{sw}}}{\leftrightarrow} \eta'_{\tilde{\mathcal{A}}^{sw}}$ with $c^{\tilde{\mathcal{A}}^{sw}} = config(\tilde{\mathcal{A}}^{sw})$ (4)

Since Y never creates $\mathcal{A}$ and $\tilde{\mathcal{A}}^{sw}$ never contains an automaton different from $\mathcal{A}$, the lemma 6 can be applied with (1), (2), (3), and (4) as preconditions:

- by item 1b of lemma 6: $merge((\eta'_{\tilde{\mathcal{A}}^{sw}}, \eta'_Y)) \stackrel{f^s}{\leftrightarrow} join((\eta'_{\tilde{\mathcal{A}}^{sw}}, \eta'_Y))$ with $f^s : C'_Z \mapsto (C'_Y, C'_{\tilde{\mathcal{A}}^{sw}})$ s.t. i) $C'_Z = C'_Y \cup C'_{\tilde{\mathcal{A}}^{sw}}$, ii) $\mathcal{A} \notin C'_Y$ and iii) $\forall \mathcal{B} \neq \mathcal{A}$, $\mathcal{B} \notin C'_{\tilde{\mathcal{A}}^{sw}}$ (5)
- by item 1d of lemma 6: $C_X \stackrel{a}{\Longrightarrow}_{\varphi_Z} merge((\eta'_{\tilde{\mathcal{A}}^{sw}}, \eta'_Y))$ (6)

Furthermore $\eta_{\mathbf{Z},q_Z,a} = \eta_Y \otimes \eta_{\tilde{\mathcal{A}}^{sw}}$. So by (4), $\eta_{\mathbf{Z},q_Z,a} \stackrel{f^Z}{\longleftrightarrow} join((\eta'_{\tilde{\mathcal{A}}^{sw}}, \eta'_Y))$ (***) with $f^Z : q'_Z = (q'_Y, q'_{\tilde{\mathcal{A}}^{sw}}) \mapsto (config(Y)(q'_{\tilde{\mathcal{A}}^{sw}}), config(\tilde{\mathcal{A}}^{sw})(q'_{\tilde{\mathcal{A}}^{sw}}))$.

Now we deal have to separate the treatment of the two cases:

- If $\mathcal{A}$ is not created by a, since $\varphi_Z = \varphi_X$, because of (6) and (3), $merge((\eta'_{\tilde{\mathcal{A}}^{sw}}, \eta'_Y)) = \eta'_X$ and because of (3) $\eta_{(X,q_X,a)} \stackrel{f^X}{\leftrightarrow} merge((\eta'_{\tilde{\mathcal{A}}^{sw}}, \eta'_Y))$ (7). Because of (7) and (5), $\eta_{(X,q_X,a)} \stackrel{g}{\longleftrightarrow} join((\eta'_{\tilde{\mathcal{A}}^{sw}}, \eta'_Y))$ with $g = f^s \circ f^X$.

  Hence, if $\mathcal{A}$ is not created by a $\eta_{(X,q_X,a)} \stackrel{h}{\longleftrightarrow} \eta_{(\mathbf{Z},q_Z,a)}$ with $h = (f^Z)^{-1} \circ f^s \circ f^X = \mu_z$ which ends the proof for this case.
- If $\mathcal{A}$ is created by a, we have both
  - $C_X \stackrel{a}{\Longrightarrow}_{\varphi_Z} merge((\eta'_{\tilde{\mathcal{A}}^{sw}}, \eta'_Y))$
  - $C_X \stackrel{a}{\Longrightarrow}_{\varphi_Z \cup \{\mathcal{A}\}} \eta'_X$

  which means $C_X \stackrel{a}{\rightharpoonup} \eta'_p$ with
  - $merge((\eta'_{\tilde{\mathcal{A}}^{sw}}, \eta'_Y))$ generated by η'_p and φ_Z and
  - η'_X generated by η'_p and $\varphi_Z \cup \{\mathcal{A}\}$.

  Thus $\eta'_X \stackrel{g^\phi}{\longleftrightarrow} merge((\eta'_{\tilde{\mathcal{A}}^{sw}}, \eta'_Y))$ with $g^\phi : C'_X = C'_Y \cup \bar{C}_{\tilde{\mathcal{A}}^{sw}} \mapsto C'_Y$. where $\bar{C}_{\tilde{\mathcal{A}}^{sw}}(\{\mathcal{A}\}, \mathbf{S}'_{\tilde{\mathcal{A}}^{sw}} : \mathcal{A} \mapsto \bar{q}_{\tilde{\mathcal{A}}^{sw}})$.

  To summarize, we have:
  - $\eta_{(X,q_X,a)} \stackrel{f^X}{\longleftrightarrow} \eta'_X$
  - $\eta'_X \stackrel{g^\phi}{\longleftrightarrow} merge((\eta'_{\tilde{\mathcal{A}}^{sw}}, \eta'_Y))$
  - $merge((\eta'_{\tilde{\mathcal{A}}^{sw}}, \eta'_Y)) \stackrel{f^s}{\longleftrightarrow} join((\eta'_{\tilde{\mathcal{A}}^{sw}}, \eta'_Y))$
  - $\eta_{(\mathbf{Z},q_z,a)} \stackrel{f^Z}{\longleftrightarrow} join((\eta'_{\tilde{\mathcal{A}}^{sw}}, \eta'_Y))$

  Hence $\eta_{(X,q_X,a)} \stackrel{h}{\longleftrightarrow} \eta_{(\mathbf{Z},q_Z,a)}$ with $f^\phi = (f^Z)^{-1} \circ f^s \circ g^\phi \circ f^X$, i.e.
  $f^\phi : q'_X \in supp(\eta_{(X,q_X,a)}) \mapsto (X.\mu_s^{\mathcal{A}}(q'_X), q^\phi_{\tilde{\mathcal{A}}^{sw}})$, which ends the proof for this case.

□

The second case where $\mathcal{A}$ is created will not be used before section 4.5.

We take advantage of the lemma 33 used for theorem 18 to introduce the notion of twin PCA and extends directly the lemma 33 and theorem 18 to twin PCA.

Definition 136 ($X_{\bar{q}_X \to \bar{q}'_X}$)**.** *Let $X = (Q_X, \bar{q}_X, sig(X), D_X)$ be a PSIOA and $\bar{q}'_X \in reachable(X)$. We note $X_{\bar{q}_X \to \bar{q}'_X}$ the PSIOA $X' = (Q_X, \bar{q}'_X, sig(X), D_X)$.*

Two PCA X and X' are $\mathcal{A}$-twin if they differ only by their start state where one of them corresponds to $\mathcal{A}$-creation.

Definition 137 ($\mathcal{A}$-twin)**.** *Let $\mathcal{A} \in Autids$. Let X, X' be PCA. We say that $X' = X_{\bar{q}_X \to \bar{q}_{X'}}$ is a $\mathcal{A}$-twin of X if it differs from X at most only by its start states $\bar{q}_{X'}$ reachable by X s.t. either $X' = X$ or $\mathcal{A} \in config(X')(\bar{q}_{X'})$ and $map(config(X')(\bar{q}_{X'}))(\mathcal{A}) = \bar{q}_{\mathcal{A}}$. If X' is a $\mathcal{A}$-twin of X and $Y = X \setminus \{\mathcal{A}\}$ and $Y' = X' \setminus \{\mathcal{A}\}$, we slightly abuse the notation and say that Y' is a $\mathcal{A}$-twin of Y'.*

Lemma 33 (partial surjectivity 1)**.** *Let $\mathcal{A} \in Autids$. Let X be a PCA $\mathcal{A}$-conservative and X' a $\mathcal{A}$-twin of X. Let $Y' = X' \setminus \{\mathcal{A}\}$. Let Y' be a $\mathcal{A}$-twin of Y. Let $\mathbf{Z}' = (Y', \tilde{\mathcal{A}}^{sw})$.*

Let $\alpha = q^0, a^1, ..., a^k, q^k$ be a pseudo execution of $\mathbf{Z}'$. Let assume the presence of $\mathcal{A}$ in α, i.e. $\forall s \in [0, k-1]$, $q^s_{\tilde{\mathcal{A}}^{sw}} \neq q^{\phi}_{\mathcal{A}}$.

Then $\exists \tilde{\alpha} \in Execs(X')$, s.t. $X'.\mu_e^{\mathcal{A}}(\tilde{\alpha}) = \alpha$.

Proof. By induction on each prefix $\alpha^s = q^0, a^1, ..., a^s, q^s$ with $s \leq k$.

Basis: case 1) $\mathcal{A} \in config(X')(\bar{q}_{X'})$: We have $\mu_z(\bar{q}_{X'}) = (\bar{q}_{Y'}, \bar{q}_{\mathcal{A}})$. Hence $\mu_e(\bar{q}_{X'}) = (\bar{q}_{Y'}, \bar{q}_{\mathcal{A}})$.

case 2) $\mathcal{A} \notin config(X')(\bar{q}_{X'})$, (necessarily $X = X'$): $\mu_z(\bar{q}_{X'}) = (\bar{q}_{Y'}, q^{\phi}_{\mathcal{A}})$. Hence $\mu_e(\bar{q}_{X'}) = (\bar{q}_{Y'}, q^{\phi}_{\mathcal{A}})$.

Induction: we assume this is true for s and we show it implies this is true for $s+1$. We note $\tilde{\alpha}_s$ s.t. $\mu_e(\tilde{\alpha}^s) = \alpha^s$. We also note $\tilde{q}^s = lstate(\tilde{\alpha}^s)$ and we have by induction assumption $\mu_z(\tilde{q}^s) = q^s = (q^s_Y, q^s_{\mathcal{A}})$. Because of preservation of signature compatibility, $sig(X)(\tilde{q}^s)) = sig(Y)(q^s_Y)) \times sig(\mathcal{A})(q^s_{\mathcal{A}}))$. Hence $a^{s+1} \in sig(X)(\tilde{q}^s)$. Thereafter, by construction of $X \setminus \{\mathcal{A}\}$ there exists $\tilde{q}^{s+1}$ s.t. $q^{s+1} = \mu_z^{\mathcal{A}}(\tilde{q}^{s+1})$. Finally, since no creation of and from $\mathcal{A}$ occurs by assumption of presence of $\mathcal{A}$, we can use lemma 32 of homomorphic transition which give $\eta_{(X,\tilde{q}^s,a^{s+1})} \overset{\mu_z}{\leftrightarrow} \eta_{(\mathbf{Z},q^s,a^{s+1})}$ which means $\tilde{q}^{s+1} \in supp(\eta_{(X,\tilde{q}^s,a^{s+1})})$ which ends the induction and so the proof. □

Before using lemma 33 and 31 to demonstrate theorem 18 of partial compatibility after reconstruction, we take the opportunity to extend lemma 33:

Lemma 34 (partial surjectivity 2)**.** *Let $\mathcal{A} \in Autids$. Let X be a PCA $\mathcal{A}$-conservative. Let $Y = X \setminus \mathcal{A}$. Let Y' be a $\mathcal{A}$-twin of Y. Let $\mathcal{Z} = Y' || \tilde{\mathcal{A}}^{sw}$.*

Let $\alpha = q^0, a^1, ..., a^k, q^k$ be a an execution of $\mathcal{Z}$. Let assume (a) $q^s_{\tilde{\mathcal{A}}^{sw}} \neq q^{\phi}_{\mathcal{A}}$ for every $s \in [0, k^*]$ (b) $q^s_{\tilde{\mathcal{A}}^{sw}} = q^{\phi}_{\tilde{\mathcal{A}}^{sw}}$ for every $s \in [k^*+1, k]$ (c) for every $s \in [k^*+1, k-1]$, for every $\tilde{q}^s$, s.t. $\mu_z(\tilde{q}^s) = q^s$, $\mathcal{A} \notin created(X)(\tilde{q}^s)(a^{s+1})$. Then $\exists \tilde{\alpha} \in Frags(X)$, s.t. $\mu_e(\tilde{\alpha}) = \alpha$. If $Y' = Y$, $\exists \tilde{\alpha} \in Execs(X)$, s.t. $\mu_e(\tilde{\alpha}) = \alpha$.

Proof. We already know this is true up to k^* because of lemma 33. We perform the same induction as the one of the previous lemma on partial surjectivity: We note $\tilde{\alpha}_s$ s.t. $\mu_e(\tilde{\alpha}^s) = \alpha^s$. We also note

$\tilde{q}^s = lstate(\tilde{\alpha}^s)$ and we have by induction assumption $\mu_z(\tilde{q}^s) = q^s = (q_Y^s, q_{\mathcal{A}}^s)$. Because of preservation of signature compatibility, $sig(X)(\tilde{q}^s)) = sig(Y)(q_Y^s)) \times sig(\mathcal{A})(q_{\mathcal{A}}^s)$. Hence $a^{k+1} \in sig(X)(\tilde{q}^s)$. Now we use the assumption (c), that says that $\mathcal{A} \notin created(X)(\tilde{q}^s)(a^{s+1})$ to be able to apply preservation of transition since no creation of $\mathcal{A}$ can occurs. □

Now we can use lemma 33 and 31 to demonstrate theorem 18 of partial compatibility after reconstruction.

Theorem 18 (Partial-compatibility after resconstruction)**.** *Let $\mathcal{A} \in Autids$. Let X be a PCA $\mathcal{A}$-conservative s.t. $\forall q_X \in Q_X$, for every action a $\mathcal{A}$-exclusive in q_X, $created(X)(q_X)(a) = \emptyset$. Let X' ba a $\mathcal{A}$-twin of X and $Y' = X' \setminus \{\mathcal{A}\}$. Then Y' and $\tilde{\mathcal{A}}^{sw}$ are partially-compatible.*

Proof. Let $\mathbf{Z}' = (Y', \tilde{\mathcal{A}}^{sw})$. Let α be a pseudo-execution of $\mathbf{Z}'$ with Let $lstate(\alpha) = q_{\mathbf{Z}} = (q_{Y'}, q_{\tilde{\mathcal{A}}^{sw}})$. Case 1) $q_{\tilde{\mathcal{A}}^{sw}} = q^{\phi}_{\tilde{\mathcal{A}}^{sw}}$. The compatibility is immediate since $sig(\tilde{\mathcal{A}}^{sw})(q^{\phi}_{\tilde{\mathcal{A}}^{sw}}) = \emptyset$. Case 2) $q_{\tilde{\mathcal{A}}^{sw}} \neq q^{\phi}_{\tilde{\mathcal{A}}^{sw}}$. Since (*) $\mathcal{A}$ cannot be re-created after destruction by neither Y or $\tilde{\mathcal{A}}^{sw}$ and (**) $\forall q_X \in Q_X$, for every action a $\mathcal{A}$-exclusive in q_X, $created(X)(q_X)(a) = \emptyset$ we can use the previous lemma 33 to show $\exists\ \tilde{\alpha} \in Execs(X')$, s.t. $\mu_e(\tilde{\alpha}) = \alpha$. Thus, $lstate(\alpha) = \mu_z(lstate(\tilde{\alpha}))$ which means $\mathbf{Z}'$ is partially-compatible at $lstate(\alpha)$ by lemma 31. Hence $\mathbf{Z}$ is partially-compatible at every reachable state, which means Y' and $\tilde{\mathcal{A}}^{sw}$ are partially-compatible. We can legitimately note $\mathcal{Z}' = Y'||\tilde{\mathcal{A}}^{sw}$. □

Since $\mathbf{Z}' = (Y', \tilde{\mathcal{A}}^{sw})$ is partially-compatible, we can legitimately note $\mathcal{Z}' = Y'||\tilde{\mathcal{A}}^{sw}$, which will be the standard notation in the remaining.

4.3.3. Executions-matching from X to $(X \setminus \{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw}$

In this subsection, we show in theorem 19 that $X.\mu_e^{\mathcal{A}}$ is a (incomplete) PCA executions-matching from X to $(X \setminus \{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw}$ in a restricted set of executions of X that do not create $\mathcal{A}$.

We start by defining the restricted set of executions of X that do not create $\mathcal{A}$ with definitions 138 and 139.

Definition 138 (execution without creation)**.** *Let $\mathcal{A}$ be a PSIOA. Let X be a PCA , we note execs-without-creation$(X)(\mathcal{A})$ the set of executions of X without creation of $\mathcal{A}$, i.e. execs-without-creation$(X)(\mathcal{A}) = \{\alpha = q^0a^1q^1...a^kq^k \in Execs(X) | \forall i \in [0, |\alpha|], \mathcal{A} \notin auts(config(X)(q^i)) \implies \mathcal{A} \notin auts(config(X)(q^{i+1}))\}$.*

Definition 139 (reachable-by)**.** *Let X be a PSIOA or a PCA. Let $Execs'_X \subseteq Execs(X)$. We note reachable-by$(Execs'_X)$ the set of states of X reachable by an execution of $Execs'_X$, i.e. reachable-by$(Execs'_X) = \{q \in Q_X | \exists \alpha \in Execs'_X, lstate(\alpha) = q\}$*

The next 2 lemmas show that reconstruction preserves configuration and signature. They will be sufficient to show that the restriction of $\mu_e^{\mathcal{A}}$ on *reachable-by(execs-without-creation$(X)(\mathcal{A})$)* is a PCA executions-matching.

Lemma 35 (μ_z configuration preservation)**.** *Let $\mathcal{A} \in Autids$. Let X be a $\mathcal{A}$-conservatiee PCA, $Y = X \setminus \mathcal{A}$, $Z = Y||\mathcal{A}^{sw}$. Let $q_X \in Q_X, q_Z = (q_Y, q_{\tilde{\mathcal{A}}^{sw}}) \in Q_Z$ s.t. $\mu_z(q_X) = q_Z$. Then $config(X)(q_X) = config(Z)(q_Z)$.*

Proof. By definition of composition of PCA, $config(Z)(q_Z) = config(Y)(q_Y) \cup config(\tilde{\mathcal{A}}^{sw})(q_{\tilde{\mathcal{A}}^{sw}})$. (*)

Also, by $\mu_z^{\mathcal{A}}$-correspondence, $config(X)(q_X) \setminus \mathcal{A} = config(Y)(q_Y)$ (**).

We deal with the two cases $\widehat{sig}(\tilde{A}^{sw})(q_{\tilde{\mathcal{A}}^{sw}}) = \emptyset$ or $\widehat{sig}(\tilde{A}^{sw})(q_{\tilde{\mathcal{A}}^{sw}}) \neq \emptyset$

- If $\widehat{sig}(\tilde{A}^{sw})(q_{\tilde{\mathcal{A}}^{sw}}) = \emptyset$, then $\mathcal{A} \notin aut(config(X)(q_X))$ which means, that $config(X)(q_X) = config(X)(q_X) \setminus \mathcal{A}$ (1). Furthermore, $config(\tilde{\mathcal{A}}^{sw})(q_{\tilde{\mathcal{A}}^{sw}}) = (\emptyset, \emptyset)$ (2) .Because of (**) and (1), $config(X)(q_X) = config(Y)(q_Y)$ and because of (*) and (2), $config(X)(q_X) = config(Z)(q_Z)$.
- If $\widehat{sig}(\tilde{A}^{sw})(q_{\tilde{\mathcal{A}}^{sw}}) \neq \emptyset$, then $\mathcal{A} \in aut(config(X)(q_X))$. We note $C_{\mathcal{A}} = config(\tilde{\mathcal{A}}^{sw})(q_{\tilde{\mathcal{A}}^{sw}}) = (\{\mathcal{A}\}, \mathbf{S} : \mathcal{A} \mapsto map(config(X)(q_X))(\mathcal{A}))$. By (*), $config(Z)(q_Z) = config(Y)(q_Y) \cup C_{\mathcal{A}}$ and by (**) $config(Y)(q_Y) \cup C_{\mathcal{A}} = config(X)(q_X) \setminus \mathcal{A} \cup C_{\mathcal{A}} = config(X)(q_X)$. Hence, $config(X)(q_X) = config(Z)(q_Z)$

Thus in all cases, $config(X)(q_X) = config(Z)(q_Z)$ which ends the proof.

□

Lemma 36 (μ_z signature-preservation)**.** *Let $\mathcal{A} \in Autids$. Let X be a $\mathcal{A}$-conservatiee PCA, $Y = X \setminus \mathcal{A}$, $Z = Y||\mathcal{A}^{sw}$. Let $q_X \in Q_X, q_Z = (q_Y, q_{\tilde{\mathcal{A}}^{sw}}) \in Q_Z$ s.t. $\mu_z(q_X) = q_Z$. Then $sig(X)(q_X) = sig(Z)(q_Z)$.*

Proof. By lemma 30 of preservation of signature $sig(X)(q_X) = sig(Y)(q_Y) \times sig(\tilde{\mathcal{A}}^{sw})(q_{\tilde{\mathcal{A}}^{sw}})$. By definition of composition of PCA, $sig(Z)(q_Z) = sig(Y)(q_Y) \times sig(\tilde{\mathcal{A}}^{sw})(q_{\tilde{\mathcal{A}}^{sw}})$ which ends the proof. □

Now we can state our strong PCA executions-matching:

Definition 140. *Let $\mathcal{A}$ be a PSIOA. Let X be a $\mathcal{A}$-conservative PCA. Let $Y = X \setminus \{\mathcal{A}\}$ and $Z = Y||\tilde{\mathcal{A}}^{sw}$.*

We define $(X.\tilde{\mu}_z^{\mathcal{A}}, X.\tilde{\mu}_{tr}^{\mathcal{A}}, X.\tilde{\mu}_e^{\mathcal{A}})$ (noted $(\tilde{\mu}_z^{\mathcal{A}}, \tilde{\mu}_{tr}^{\mathcal{A}}, \tilde{\mu}_e^{\mathcal{A}})$ when it is clear in the context) as follows:

- *$\tilde{\mu}_z^{\mathcal{A}}$ the restriction of $\mu_z^{\mathcal{A}}$ on reachable-by(execs-without-creation$(X)(\mathcal{A})$).*
- *$f^{tr} : (q_X, a, \eta_{(X,q_X,a)}) \in D'_X \mapsto (\tilde{\mu}_z^{\mathcal{A}}(q_X), a, \eta_{(Z,\tilde{\mu}_z^{\mathcal{A}}(q_X),a)})$ where $D'_X = \{(q_X, a, \eta_{(X,q_X,a)}) \in D_X | q_X \in$ reachable-by(execs-without-creation$(X)(\mathcal{A})), (\mathcal{A} \notin auts(config(X)(q_X) \implies \mathcal{A} \notin created(X)(q_X)(a))\}$.*
- *$\tilde{\mu}_e^{\mathcal{A}}$ the restriction of $\mu_e^{\mathcal{A}}$ on execs-without-creation$(X)(\mathcal{A})$.*

Theorem 19 (executions-matching after reconstruction)**.** *Let $\mathcal{A}$ be a PSIOA. Let X be a $\mathcal{A}$-conservative PCA. Let $Y = X \setminus \{\mathcal{A}\}$. The triplet $(\tilde{\mu}_z^{\mathcal{A}}, \tilde{\mu}_{tr}^{\mathcal{A}}, \tilde{\mu}_e^{\mathcal{A}})$ is a strong PCA executions-matching from X to $Y||\tilde{\mathcal{A}}^{sw}$ if $\mathcal{A} \in auts(config(X_{\mathcal{A}})(start(X_{\mathcal{A}})))$ and from X to $Y||\tilde{\mathcal{A}}^{sw}_{\bar{q}_{\tilde{\mathcal{A}}^{sw}} \to q^{\phi}_{\tilde{\mathcal{A}}^{sw}}}$ otherwise.*

Proof. We note $Z = Y||\tilde{\mathcal{A}}^{sw}$ and $Z^{\phi} = Y||\tilde{\mathcal{A}}^{sw}_{\bar{q}_{\tilde{\mathcal{A}}^{sw}} \to q^{\phi}_{\tilde{\mathcal{A}}^{sw}}}$

- $\tilde{\mu}_z^{\mathcal{A}}$ is a strong PCA-state-matching since
  - starting state preservation is ensured by construction:
    - $\mathcal{A} \in auts(config(X_{\mathcal{A}})(start(X_{\mathcal{A}}))) : \tilde{\mu}_z^{\mathcal{A}}(\bar{q}_X) = \bar{q}_Z$
    - $\mathcal{A} \notin auts(config(X_{\mathcal{A}})(start(X_{\mathcal{A}}))) : \tilde{\mu}_z^{\mathcal{A}}(\bar{q}_X) = \bar{q}_{Z^{\phi}}$
  - signature preservation is ensured $\forall (q_X, q_Z) \in Q_X \times Q_Z$ s.t. $q_Z = \tilde{\mu}_z^{\mathcal{A}}(q_X)$, $sig(X)(q_X) = sig(Z)(q_Z)$ by lemma 36 of signature preservation of μ_z.
- $D'_X \triangleq dom(\tilde{\mu}_{tr}^{\mathcal{A}})$ is eligible to PCA transition-matching (and thus $(\tilde{\mu}_z^{\mathcal{A}}, \tilde{\mu}_{tr}^{\mathcal{A}})$ is a strong PCA-transition-matching) since

- matched state preservation is ensured: $\forall \eta_{(X,q_X,a)} \in D'_X$, $q_X \in dom(\tilde{\mu}_z^{\mathcal{A}})$ by construction of D'_X
- equitable corresponding distribution is ensured: $\forall \eta_{(X,q_X,a)} \in D'_X, \forall q'' \in supp(\eta_{(X,q_X,a)})$, $\eta_{(X,q_X,a)}(q'') = \eta_{(Z,\tilde{\mu}_z^{\mathcal{A}}(q_X),a)}(\tilde{\mu}_z^{\mathcal{A}}(q''))$ by lemma 32 of homomorphic transition.

– $(\tilde{\mu}_z^{\mathcal{A}}, \tilde{\mu}_{tr}^{\mathcal{A}}, \tilde{\mu}_e^{\mathcal{A}})$ is the PCA-executions-matching induced by $(\tilde{\mu}_z^{\mathcal{A}}, \tilde{\mu}_{tr}^{\mathcal{A}})$. and correctly verifies:
- For each state q in an execution in *execs-without-creation*$(X)(\mathcal{A})$, $q \in dom(\tilde{\mu}_z^{\mathcal{A}})$.

Then, the triplet $(\tilde{\mu}_z^{\mathcal{A}}, \tilde{\mu}_{tr}^{\mathcal{A}}, \tilde{\mu}_e^{\mathcal{A}})$ is a strong PCA-executions-matching from X to Z if $\mathcal{A} \in auts(config(X_{\mathcal{A}})(\bar{q}_{X_{\mathcal{A}}}))$: $\tilde{\mu}_z^{\mathcal{A}}(\bar{q}_X) = \bar{q}_Z$ and from X to Z^{ϕ} otherwise.

□

extension and continuation of $(\tilde{\mu}_z^{\mathcal{A}}, \tilde{\mu}_{tr}^{\mathcal{A}}, \tilde{\mu}_e^{\mathcal{A}})$ Now, we continue the executions-matching $(\tilde{\mu}_z^{\mathcal{A}}, \tilde{\mu}_{tr}^{\mathcal{A}}, \tilde{\mu}_e^{\mathcal{A}})$ to deal with $\mathcal{A}$ creation at very last action.

Definition 141 (Preparing continuation of PCA executions-matching from X to Z)**.** *Let $\mathcal{A}$ be a PSIOA. Let X be a $\mathcal{A}$-conservative PCA. We define*

– *execs-with-only-one-creation-at-last-action*$(X)(\mathcal{A}) = \{\alpha' = \alpha^\frown q, a, q' \in Execs(X) | \alpha \in$ *execs-without-creation*$(X)(\mathcal{A}) \wedge \alpha' \notin$ *execs-without-creation*$(X)(\mathcal{A})\}$.

– $\tilde{\mu}_z^{\mathcal{A},+} : q_X \in$ *reachable-by*(*execs-with-only-one-creation-at-last-action*$(X)(\mathcal{A})) \mapsto (\tilde{\mu}_s^{\mathcal{A}}(q_{Y_{\mathcal{A}}}), q_{\mathcal{A}}^{\phi})$.

– $\tilde{\mu}_{tr}^{\mathcal{A},+} : (q_X, a, \eta_{(X,q_X,a)}) \in dom(\tilde{\mu}_{tr}^{\mathcal{A}}) \cup D''_X \mapsto (\tilde{\mu}_z^{\mathcal{A}}(q_X), a, \eta_{(X,\tilde{\mu}_z^{\mathcal{A}}(q_X),a)})$ *where*
$D''_X = \{(q_X, a, \eta_{(X,q_X,a)}) \in D_X | q_X \in$ *reachable-by*(*execs-without-creation-at-last-action*$(X)(\mathcal{A}))\wedge$
$\mathcal{A} \notin auts(config(X)(q_X)) \wedge \mathcal{A} \in created(X)(q_X)(a)\}$

We show that $dom(\tilde{\mu}_{tr}^{\mathcal{A},+}) \setminus dom(\tilde{\mu}_{tr}^{\mathcal{A}})$ verifies the equitable corresponding property of definition 117.

Lemma 37 (Continuation of PCA transitions-matching from X to Z)**.** *Let $\mathcal{A}$ be a PSIOA. Let X be a $\mathcal{A}$-conservative PCA. Let $Y = X \setminus \{\mathcal{A}\}$ and $Z = Y||\tilde{\mathcal{A}}^{sw}$.*

$\forall (q_X, a, \eta_{(X,q_X,a)}) \in dom(\tilde{\mu}_{tr}^{\mathcal{A},+}) \setminus dom(\tilde{\mu}_{tr}^{\mathcal{A}})$, $\forall q'_X \in supp(\eta_{(X,q_X,a)})$, $\eta_{(X,q_X,a)}(q'_X) = \eta_{(Z,\tilde{\mu}_z^{\mathcal{A}}(q_X),a)}(\tilde{\mu}_z^{\mathcal{A},+}(q'_X))$

Proof. By configuration preservation, $Conf = config(X)(q_X) = config(Z)(\tilde{\mu}_z^{\mathcal{A}}(q_X))$. We have $Conf \xrightarrow{a} \eta_{(Conf,a),p}$. Moreover, by μ_s-correspondence rule, $\varphi_X \setminus \{\mathcal{A}\} = \varphi_Z$, with $\varphi_X = created(X)(q_X)(a)$ and $\varphi_Z = created(Z)(\tilde{\mu}_z^{\mathcal{A}}(q_X))(a)$.

Hence $Conf \stackrel{a}{\Longrightarrow}_{\varphi_X} \eta'_X$ with η'_X generated by φ_X and $\eta_{(Conf,a),p}$, while $Conf \stackrel{a}{\Longrightarrow}_{\varphi_Z} \eta'_Z$ with η'_Z generated by φ_Z and $\eta_{(Conf,a),p}$.

Since $\mathcal{A}$ is created, for every $Conf'_Z = (\mathbf{A}'_Z, \mathbf{S}'_Z)$ with $\mathcal{A} \notin \mathbf{A}_Z$, for every $Conf'_X = (\mathbf{A}'_X, \mathbf{S}'_X)$ with $\mathbf{A}'_X = \mathbf{A}'_Z \cup \{\mathcal{A}\}$ where $\mathbf{S}'_X(\mathcal{A}) = \bar{q}_{\mathcal{A}}$ and $\mathbf{S}'_X$ agrees with $\mathbf{S}'_Z$ on $\mathbf{A}'_Z$, $\eta'_Z(Conf'_Z) = \eta'_X(Conf'_X)$, while $\eta'_X(Conf''_X) = 0$ for every $Conf''_X = (\mathbf{A}''_X, \mathbf{S}''_X)$ s. t either $\mathcal{A} \notin \mathbf{A}''_X$ or $\mathcal{A} \in \mathbf{A}''_X$ but $\mathbf{S}''_X(\mathcal{A}) \neq \bar{q}_{\mathcal{A}}$. So $\eta_{(Z,\tilde{\mu}_z^{\mathcal{A}}(q_X),a)}(\tilde{\mu}_z^{\mathcal{A},+}(q'_X)) = \eta'_Z(config(Z)(\tilde{\mu}_z^{\mathcal{A},+}(q'_X))) = \eta'_X((config(X)(q'_X))) = \eta_{(X,q_X,a)}(q'_X)$ which ends the proof.

□

Since $dom(\tilde{\mu}_{tr}^{\mathcal{A},+}) \setminus dom(\tilde{\mu}_{tr}^{\mathcal{A}})$ verifies the equitable corresponding property of definition 117, we can define a continuation of $(\tilde{\mu}_z^{\mathcal{A}}, \tilde{\mu}_{tr}^{\mathcal{A}}, \tilde{\mu}_e^{\mathcal{A}})$ that deal with $\mathcal{A}$-creation at very last action.

Definition 142 (Continuation of PCA executions-matching from X to Z)**.** *Let $\mathcal{A}$ be a PSIOA. Let X be a $\mathcal{A}$-conservative PCA. Let $Y = X \setminus \{\mathcal{A}\}$ and $Z = Y||\tilde{\mathcal{A}}^{sw}$. Let $D''_X = dom(\tilde{\mu}_z^{\mathcal{A},+}) \setminus dom(\tilde{\mu}_z^{\mathcal{A}})$. Since $\forall (q_X, a, \eta_{(X,q_X,a)}) \in D''_X$, $\forall q'_X \in supp(\eta_{(X,q_X,a)})$, $\eta_{(X,q_X,a)}(q'_X) = \eta_{Z,\tilde{\mu}_z^{\mathcal{A}}(q_X),a)}(\tilde{\mu}_z^{\mathcal{A},+}(q'_X))$ by previous lemma 37, we can define:*

$((\tilde{\mu}_z^{\mathcal{A}}, \tilde{\mu}_z^{\mathcal{A},+}), \tilde{\mu}_{tr}^{\mathcal{A},+}, \tilde{\mu}_e^{\mathcal{A},+}))$ *the $(\tilde{\mu}_z^{\mathcal{A},+}, D''_X)$-continuation of $(\tilde{\mu}_z^{\mathcal{A}}, \tilde{\mu}_{tr}^{\mathcal{A}}, \tilde{\mu}_e^{\mathcal{A}})$.*

We terminate this subsection by showing the $\mathcal{E}$-extension of our continued PCA executions-matching is always well-defined.

Theorem 20 (extension of continued executions-matching after reconstruction)**.** *Let $\mathcal{A}$ be a PSIOA. Let X be a $\mathcal{A}$-conservative PCA. Let $Y = X \setminus \{\mathcal{A}\}$ and $Z = Y||\tilde{\mathcal{A}}^{sw}$. Let $\tilde{\mathcal{E}}$ partially-compatible with both X and Z. The $\tilde{\mathcal{E}}$-extension of the executions-matching $((X.\tilde{\mu}_z^{\mathcal{A}}, X.\tilde{\mu}_z^{\mathcal{A},+}), X.\tilde{\mu}_{tr}^{\mathcal{A}}, X.\tilde{\mu}_e^{\mathcal{A}})$ from X to Z, noted $(((\tilde{\mathcal{E}}||X).\tilde{\mu}_z^{\mathcal{A}}, (\tilde{\mathcal{E}}||X).\tilde{\mu}_z^{\mathcal{A},+}), (\tilde{\mathcal{E}}||X).\tilde{\mu}_{tr}^{\mathcal{A}}, (\tilde{\mathcal{E}}||X).\tilde{\mu}_e^{\mathcal{A}})$, is a strong continued PCA executions-matching from $\tilde{\mathcal{E}}||X$ to $\tilde{\mathcal{E}}||Z$.*

Proof. By definition of $\tilde{\mu}_z^{\mathcal{A},+}$ and $\tilde{\mu}_z^{\mathcal{A}}$, we have

- $\tilde{E}_{\tilde{\mathcal{E}}||X} = execs\text{-}without\text{-}creation(\tilde{\mathcal{E}}||X)(\mathcal{A})$
- $\tilde{E}^+_{\tilde{\mathcal{E}}||X} = execs\text{-}with\text{-}only\text{-}one\text{-}creation\text{-}at\text{-}last\text{-}action(\tilde{\mathcal{E}}||X)(\mathcal{A})$
- $\tilde{E}_X = execs\text{-}without\text{-}creation(X)(\mathcal{A})$
- $\tilde{E}^+_X = execs\text{-}with\text{-}only\text{-}one\text{-}creation\text{-}at\text{-}last\text{-}action(X)(\mathcal{A})$
- $\tilde{Q}_{\tilde{\mathcal{E}}||X} = reachable\text{-}by(\tilde{E}_{\tilde{\mathcal{E}}||X})$
- $\tilde{Q}^+_{\tilde{\mathcal{E}}||X} = reachable\text{-}by(\tilde{E}^+_{\tilde{\mathcal{E}}||X})$
- $\tilde{Q}_X = reachable\text{-}by(\tilde{E}_X)$
- $\tilde{Q}^+_X = reachable\text{-}by(\tilde{E}^+_X)$
- $dom((\tilde{\mathcal{E}}||X).\tilde{\mu}_z^{\mathcal{A},+}) = \tilde{Q}^+_{\tilde{\mathcal{E}}||X}$
- $dom((\tilde{\mathcal{E}}||X).\tilde{\mu}_z^{\mathcal{A}}) = \tilde{Q}_{\tilde{\mathcal{E}}||X}$
- $dom(X.\tilde{\mu}_z^{\mathcal{A},+}) = \tilde{Q}^+_X$
- $dom(X.\tilde{\mu}_z^{\mathcal{A}}) = \tilde{Q}_X$

This allows us to apply lemma 17 of "sufficient conditions to obtain range inclusion" to both $(\tilde{\mathcal{E}}||X).\tilde{\mu}_z^{\mathcal{A},+}$ and $(\tilde{\mathcal{E}}||X).\tilde{\mu}_z^{\mathcal{A}}$ which gives $range((\tilde{\mathcal{E}}||X).\tilde{\mu}_z^{\mathcal{A},+}) \subseteq Q_{\tilde{\mathcal{E}}||Z}$ and $range((\tilde{\mathcal{E}}||X).\tilde{\mu}_z^{\mathcal{A}}) \subseteq Q_{\tilde{\mathcal{E}}||Z}$ which allows us to apply lemma 20.

The lemma 23 implies that the resulting executions-matching is a strong one.

□

4.3.4. Composition and projection are commutative

This section aims to show in theorem 21 that operation of projection/deprivation and composition are commutative.

Theorem 21 ($(X||\mathcal{E}) \setminus \{\mathcal{A}\}$ and $(X \setminus \{\mathcal{A}\})||\mathcal{E}$ are semantically equivalent)**.** *Let $\mathcal{A}$ be a PSIOA. Let X be a $\mathcal{A}$-fair PCA partially-compatible with $\mathcal{E}$ where both $X, \mathcal{E}$, and $X||\mathcal{E}$ are configuration-conflict-free. The PCA $(X||\mathcal{E}) \setminus \{\mathcal{A}\}$ and $(X \setminus \{\mathcal{A}\})||\mathcal{E}$ are semantically equivalent.*

Proof. We note $W = X||\mathcal{E}$, $U = (X||\mathcal{E})\backslash\{\mathcal{A}\}$, $V = (X\backslash\{\mathcal{A}\})||\mathcal{E}$, $\mu_s^{X,\mathcal{A}} = X.\mu_s^{\mathcal{A}}$, $\mu_s^{W,\mathcal{A}} = W.\mu_s^{\mathcal{A}}$. To stay simple, we note Id the identity function on any domain, that is we note Id for both $Id_{\mathcal{E}} : q_{\mathcal{E}} \in Q_{\mathcal{E}} \mapsto q_{\mathcal{E}}$ and $Id_U : q_U \in Q_U \mapsto q_U$.

The plan of the proof is the following one:

- We will construct two functions, $iso_{UV} : Q_U \to Q_V$ and $iso_{VU} : Q_V \to Q_U$, s.t. $iso_{UV}(q_U)$ is the unique element of $(\mu_s^{X,\mathcal{A}}, Id)((\mu_s^{W,\mathcal{A}})^{-1}(q_U))$ and $iso_{VU}((q_Y, q_{\mathcal{E}}))$ is the unique element of $\mu_s^{W,\mathcal{A}}((\mu_s^{X,\mathcal{A}}, Id)^{-1}((q_Y, q_{\mathcal{E}})))$.
- Then we will show that iso_{UV} and iso_{VU} are two bijections s.t. $iso_{VU} = iso_{UV}^{-1}$.
- Thereafter we will show that for every $(q_U, q_V), (q'_U, q'_V) \in (states(U) \times Q_V)$, s.t. $q_V = iso_{UV}(q_U)$ and $q'_V = iso_{UV}(q'_U)$, then $q_U R_{strict} q_V$, $q'_U R_{strict} q'_V$ and for every $a \in \widehat{sig}(U)(q_U) = \widehat{sig}(V)(q_V)$, $\eta_{(U,q_U,a)}(q'_U) = \eta_{(V,q_V,a)}(q'_V)$.
- Finally, it will allow us to construct a strong complete bijective executions-matching induced by iso_{UV} and D_U (the set of discrete transitions of U) in bijection with a strong complete bijective executions-matching induced by iso_{VU} and D_V (the set of discrete transitions of V) .

First, we show that for every $q_W = (q_X, q_{\mathcal{E}}) \in reachable(W) \subset Q_X \times Q_{\mathcal{E}}$, the state $q_V \triangleq (\mu_s^{X,\mathcal{A}}, Id)(q_W) = (\mu_s^{X,\mathcal{A}}(q_X), q_{\mathcal{E}})$ is an element of $reachable(V)$ (*). We proceed by induction. Basis: $(\mu_s^{X,\mathcal{A}}(\bar{q}_X), \bar{q}_{\mathcal{E}})$ is the initial state of V. Induction: Let $q_W \triangleq (q_X, q_{\mathcal{E}}), q'_W \triangleq (q'_X, q'_{\mathcal{E}}) \in reachable(W), q_V \in reachable(V), a \in \widehat{sig}(W)(q_W)$ s.t. $q'_W \in supp(\eta_{(W,q_W,a)})$, $q_V = (\mu_s^{X,\mathcal{A}}, Id)(q_W)$, and $q'_V = (\mu_s^{X,\mathcal{A}}, Id)(q'_W)$. There are two cases:

case 1) a is $\mathcal{A}$-exclusive in q_W. In this case $q_W R^{\backslash\{\mathcal{A}\}} q'_W$, which means $q'_V = q_V$ and ends the proof

case 2) $a \in \widehat{sig}(V)(q_V) \cap \widehat{sig}(W)(q_W)$

We need to show that $q'_V \in supp(\eta_{(V,q_V,a)})$. This is easy to show. Indeed, $q'_W \in supp(\eta_{(W,q_W,a)})$ means $(q'_X, q'_{\mathcal{E}}) \in supp(\eta_{(X,q_X,a)} \otimes \eta_{(\mathcal{E},q_{\mathcal{E}},a)})$ (with the convention $\eta_{(X,q_X,a)} = \delta_{q_X}$ if $a \notin \widehat{sig}(X)(q_X)$) and $\eta_{(\mathcal{E},q_{\mathcal{E}},a)} = \delta_{q_{\mathcal{E}}}$ if $a \notin \widehat{sig}(\mathcal{E})(q_{\mathcal{E}})$)) which means $q'_X \in supp(\eta_{(X,q_X,a)})$ and $q'_{\mathcal{E}} \in supp(\eta_{(\mathcal{E},q_{\mathcal{E}},a)})$. So $\mu_s^{X,\mathcal{A}}(q'_X) \in supp(\eta_{(Y,\mu_s^{X,\mathcal{A}}(q_X),a)})$ which means $(\mu_s^{X,\mathcal{A}}(q'_X), q'_{\mathcal{E}}) \in supp(\eta_{(Y,\mu_s^{X,\mathcal{A}}(q_X),a)} \otimes \eta_{(\mathcal{E},q_{\mathcal{E}},a)})$, that is $(\mu_s^{X,\mathcal{A}}(q'_X), q'_{\mathcal{E}}) \in supp(\eta_{((Y,\mathcal{E}),(\mu_s^{X,\mathcal{A}}(q_X),q_{\mathcal{E}}),a)})\eta_{(\mathcal{E},q_{\mathcal{E}},a)})$ and thus $q'_V \in supp(\eta_{(V,q_V,a)})$ so $q'_V \in reachable(V)$.

Second, we show that for every $q_V \triangleq (q_Y, q_{\mathcal{E}}) \in reachable(V)$, $\exists\ q_W \triangleq (q_X, q_{\mathcal{E}}) \in reachable(W)$ s.t. $q_V = (\mu_s^{X,\mathcal{A}}, Id)(q_W)$ (**). The reasoning is the same, we proceed by induction. The basis is performed with start state correspondence as before. Induction: Let $q_V \triangleq (q_Y, q_{\mathcal{E}}), q'_V \triangleq (q'_Y, q'_{\mathcal{E}}) \in reachable(V), q_W \in reachable(W), a \in \widehat{sig}(V)(q_V) \cap \widehat{sig}(W)(q_W)$ s.t. $q'_V \in supp(\eta_{(V,q_V,a)})$ with $q_V = (\mu_s^{X,\mathcal{A}}, Id)(q_W)$.

We need to show that $\exists\ q'_W \in supp(\eta_{(W,q_W,a)})$ s.t. $q'_V = (\mu_s^{X,\mathcal{A}}, Id)(q'_W)$. This is easy to show because of $\mu_d^{X,\mathcal{A}}$-correspondence. For every $q'_V \triangleq (q'_Y, q_{\mathcal{E}}) \in supp(\eta_{(V,(q_Y,q_{\mathcal{E}}),a)})$, $q'_Y \in supp(\eta_{(Y,q_Y,a)})$. Because of $\mu_d^{X,\mathcal{A}}$-correspondance, $\exists\ q'_X \in supp(\eta_{(X,q_X,a)})$ with $q'_Y = \mu_s^{X,\mathcal{A}}(q'_X)$, thus $\exists\ q'_W = (q'_X, q'_{\mathcal{E}}) \in supp(\eta_{(W,(q_X,q_{\mathcal{E}}),a)})$ s.t. $q'_V = (\mu_s^{X,\mathcal{A}}(q'_X), q'_{\mathcal{E}})$ which ends the proof of this second point.

Now we can construct iso_{UV} and iso_{VU}.

- iso_{UV}: for every $q_U \in Q_U$, $(\mu_s^{W,\mathcal{A}})^{-1}(q_U) \neq \emptyset$ by construction of U and for every $q_W \triangleq (q_X, q_{\mathcal{E}}), q'_W \triangleq (q'_X, q'_{\mathcal{E}}) \in (\mu_s^{W,\mathcal{A}})^{-1}(q_U)$, $q_W R_{strict}^{\backslash\{\mathcal{A}\}} q'_W$
  [...],
  which means for every $q_W \triangleq (q_X, q_{\mathcal{E}}), q'_W \triangleq (q'_X, q'_{\mathcal{E}}) \in (\mu_s^{W,\mathcal{A}})^{-1}(q_U)$, $(\mu_s^{X,\mathcal{A}}, Id)((q_X, q_{\mathcal{E}})) = (\mu_s^{X,\mathcal{A}}, Id)((q'_X, q'_{\mathcal{E}}))$ and so $(\mu_s^{X,\mathcal{A}}, Id)((\mu_s^{W,\mathcal{A}})^{-1}(q_U)) = \{q_V\}$ where $q_V \triangleq iso_{UV}(q_U) \in Q_V$ by (*).

– iso_{VU}: for every $q_V \triangleq (q_Y, q_\mathcal{E}) \in Q_V$, $(\mu_s^{X,\mathcal{A}}, Id)^{-1}(q_V) \neq \emptyset$ by (**). Furthermore for every $q_W \triangleq (q_X, q_\mathcal{E}), q'_W \triangleq (q'_X, q_\mathcal{E}) \in (\mu_s^{X,\mathcal{A}}, Id)^{-1}(q_V)$, $q_X R_{strict}^{\setminus\{\mathcal{A}\}} q'_X$, which means $q_W R_{strict}^{\setminus\{\mathcal{A}\}} q'_W$ and so $\mu_s^{W,\mathcal{A}}((\mu_s^{X,\mathcal{A}}, Id)^{-1}(q_V)) = \{q_U\}$ where $q_U \triangleq iso_{VU}(q_V) \in Q_U$

Now we can show that iso_{UV} is a bijection with $iso_{VU} = iso_{VU}^{-1}$.

– surjectivity of iso_{UV}: Let $q_V = (q_Y, q_\mathcal{E}) \in reachable(V)$, we will show that $\exists\ q_U \in reachable(U)$ s.t. $iso_{UV}(q_U) = q_V$. Indeed, we already know that (*) $\exists\ q_W = (q_X, q_\mathcal{E}) \in (\mu_s^{X,\mathcal{A}}, Id)^{-1}(q_V) \cap reachable(W)$. Let $q_U = \mu_s^{W,\mathcal{A}}(q_W)$. By construction of U, we have $q_U \in reachable(U)$ and $q_W \in (\mu_s^{W,\mathcal{A}})^{-1}(q_U)$ and $(\mu_s^{X,\mathcal{A}}, Id)(q_W) = q_V$ which means $iso_{UV}(q_U) = q_V$ and ends this item.
– injectivity of iso_{UV}: Let $q_V \in reachable(V)$, Let $q_U, q'_U \in reachable(U)$ s.t. $iso_{UV}(q_U) = iso_{UV}(q'_U)$ then $q_U = q'_U$. Again for every $q_W, q'_W \in (\mu_s^{X,\mathcal{A}}, Id)^{-1}(q_V)$, $q_W R_{strict}^{\setminus\mathcal{A}} q'_W$ and so $\mu_s^{W,\mathcal{A}}(q_W) = \mu_s^{W,\mathcal{A}}(q'_W)$. But for every $q_U, q'_U \in iso_{UV}^{-1}(q_V)$, $q_U, q'_U \in \mu_s^{W,\mathcal{A}}(\mu_s^{X,\mathcal{A}}, Id)^{-1}(q_V)$ which means $q_U = q'_U$.

Let (i) $q_V = iso_{UV}(q_U)$ or (ii) $q_U = iso_{UV}(q_V)$ we will show that in both (i) and (ii) $q_V R_{strict} q_U$. By definition, $\{q_V\} = (\mu_s^{X,\mathcal{A}}, Id)(\mu_s^{W,\mathcal{A}})^{-1}(q_U))$.

In case (i) we note q_W an arbitrary element of $(\mu_s^{W,\mathcal{A}})^{-1}(q_U) \neq \emptyset$, while in case (ii) we note q_W an arbitrary element of $(\mu_s^{X,\mathcal{A}}, Id)^{-1}(q_V) \neq \emptyset$. In both cases, we have 1a) $config(W)(q_W) \setminus \{\mathcal{A}\} = config(U)(q_U)$ and 1b) $config(W)(q_W) \setminus \{\mathcal{A}\} = config(V)(q_V)$, which means 1c) $config(U)(q_U) = config(V)(q_V)$. Then we have 2a) $hidden\text{-}actions(W)(q_W) \setminus pot\text{-}out(W)(q_W)(\mathcal{A}) = hidden\text{-}actions(U)(q_U) \setminus pot\text{-}out(W)(q_W)(\mathcal{A}) = hidden\text{-}actions(U)(q_U)$ and 2b) $hidden\text{-}actions(W)(q_W) \setminus pot\text{-}out(W)(q_W)(\mathcal{A}) = hidden\text{-}actions(V)(q_V) \setminus pot\text{-}out(W)(q_W)(\mathcal{A}) = hidden\text{-}actions(V)(q_V)$, which means 2c) $hidden\text{-}actions(U)(q_U) = hidden\text{-}actions(V)(q_V)$. Thereafter we have 3a) for every action $a \in \widehat{sig}(W)(q_W) \cap \widehat{sig}(U)(q_U), created(W)(q_W)(a) \setminus \{\mathcal{A}\} = created(U)(q_U)(a) \setminus \{\mathcal{A}\} = created(U)(q_U)(a)$ and 3b) for every action $a \in \widehat{sig}(W)(q_W) \cap \widehat{sig}(V)(q_V), created(W)(q_W)(a) \setminus \{\mathcal{A}\} = created(V)(q_V)(a) \setminus \{\mathcal{A}\} = created(V)(q_V)(a)$ which means 3c) for every action $a \in \widehat{sig}(U)(q_U) = \widehat{sig}(V)(q_V), created(U)(q_U)(a) = created(V)(q_V)(a)$. The conjonction of 3a), 3b) and 3c) lead us to $q_V R_{strict} q_U$.

Now we can show that iso_{UV} is the reverse function of iso_{VU}: Let $(q_U, q_V) \in reachable(U) \times reachable(V)$ s.t. $q_V = iso_{UV}(q_U)$. We need to show that $iso_{VU}(q_V) = q_U$. The point is that $\exists!\ q'_U \triangleq iso_{VU}(q_V)$ and we have $q_V R_{strict} q_U$ and $q_V R_{strict} q'_U$ which means $q_U R_{strict} q'_U$ and so $q_U = q'_U$ by assumption of configuration-conflict-free PCA. Hence $iso_{UV} = iso_{VU}^{-1}$.

The last point is to show that that for every $(q_U, q_V), (q'_U, q'_V) \in reachable(U) \times reachable(V)$, s.t. $q_V = iso_{UV}(q_U)$ and $q'_V = iso_{UV}(q'_U)$, then $q_U R_{strict} q_V$, $q'_U R_{strict} q'_V$ and for every $a \in \widehat{sig}(U)(q_U) = \widehat{sig}(V)(q_V)$, $\eta_{(U,q_U,a)}(q'_U) = \eta_{(V,q_V,a)}(q'_V)$.

For every $a \in \widehat{sig}(U)(q_U) = \widehat{sig}(V)(q_V)$ we have a unique η s.t. $C \overset{a}{\Longrightarrow}_\varphi \eta$ with $C = config(U)(q_U) = config(V)(q_V)$ and $\varphi = created(U)(q_U)(a) = created(V)(q_V)(a)$. Hence for every configuration $C' \in supp(\eta)$, $\exists!\ (q'_U, q'_V) \in reachable(U) \times reachable(V)$ s.t. $C' = config(U)(q'_U) = config(V)(q'_V)$. Hence $iso_{UV}(q'_U) = q'_V$ and furthermore $\eta_{(U,q_U,a)}(q'_U) = \eta_{(V,q_V,a)}(q'_V) = \eta(C)$.

Everything is ready to construct the PCA-executions-matching, which is (j) the PCA-executions-matching induced by iso_{UV} and D_U (the set of discrete transition of U) and (jj) the PCA-executions-matching induced by iso_{VU} and D_V (the set of discrete transition of V)

□

4.4. PCA corresponding w.r.t. PSIOA $\mathcal{A}$, $\mathcal{B}$

In the previous section we have shown that $X_{\mathcal{A}}||\mathcal{E}$ and $\tilde{\mathcal{A}}^{sw}||(X_{\mathcal{A}} \setminus \{\mathcal{A}\}||\mathcal{E})$ are linked by a strong PCA executions-matching as long as $\mathcal{A}$ is not re-created by $X_{\mathcal{A}}$. This also means that the probability distribution of $X_{\mathcal{A}}||\mathcal{E}$ is preserved by $\tilde{\mathcal{A}}^{sw}||(X \setminus \{\mathcal{A}\}||\mathcal{E})$, as long as $\mathcal{A}$ is not re-created by $X_{\mathcal{A}}$. We can have the same reasoning to obtain a strong PCA executions-matching from $X_{\mathcal{B}}||\mathcal{E}$ and $\tilde{\mathcal{B}}^{sw}||(X_{\mathcal{B}} \setminus \{\mathcal{B}\}||\mathcal{E})$.

In this section we take an interest in PCA $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ that differ only on the fact that $\mathcal{B}$ supplants $\mathcal{A}$ in $X_{\mathcal{B}}$. Hence, we recall the definitions of section 3.7. Then, we show that under slight assumptions, $X_{\mathcal{A}} \setminus \{\mathcal{A}\}$ and $X_{\mathcal{B}} \setminus \{\mathcal{B}\}$ are semantically equivalent (see theorem 22).

Combined with the result of previous section we will realise that we can obtain a strong PCA executions-matching from (*) $X_{\mathcal{A}}||\mathcal{E}$ to $\tilde{\mathcal{A}}^{sw}||(Y||\mathcal{E})$ and (**) from $X_{\mathcal{B}}||\mathcal{E}$ to $\tilde{\mathcal{B}}^{sw}||(Y||\mathcal{E})$ where Y is semantically equivalent to both $X_{\mathcal{B}} \setminus \{\mathcal{B}\}$ and $X_{\mathcal{A}} \setminus \{\mathcal{A}\}$. Hence if $\mathcal{E}' = \mathcal{E}||Y$ cannot distinguish $\tilde{\mathcal{A}}^{sw}$ from $\tilde{\mathcal{B}}^{sw}$, we will be able to show that $\mathcal{E}$ cannot distinguish $X_{\mathcal{A}}$ from $X_{\mathcal{B}}$ which will be the subject of sections 4.5 to finally prove the monotonicity of p-implementation.

$\lhd_{\mathcal{AB}}$-correspondence between two configurations We formalize the idea that two configurations are the same excepting the fact that the automaton $\mathcal{B}$ supplants $\mathcal{A}$ but with the same external signature. The next definition comes from [AL16].

Definition 143 ($\lhd_{\mathcal{AB}}$-corresponding configurations)**.** *(see figure 4.11) Let $\Phi \subseteq$ Autids, and $\mathcal{A}, \mathcal{B}$ be PSIOA identifiers. Then we define $\Phi[\mathcal{B}/\mathcal{A}] = (\Phi \setminus \mathcal{A}) \cup \{\mathcal{B}\}$ if $\mathcal{A} \in \Phi$, and $\Phi[\mathcal{B}/\mathcal{A}] = \Phi$ if $\mathcal{A} \notin \Phi$. Let C, D be configurations. We define $C \lhd_{\mathcal{AB}} D$ iff (1) $auts(D) = auts(C)[\mathcal{B}/\mathcal{A}]$, (2) for every $\mathcal{A}' \notin auts(C) \setminus \{\mathcal{A}\}$: $map(D)(\mathcal{A}') = map(C)(\mathcal{A}')$, and (3) $ext(\mathcal{A})(s) = ext(\mathcal{B})(t)$ where $s = map(C)(\mathcal{A}), t = map(D)(\mathcal{B})$. That is, in $\lhd_{\mathcal{AB}}$-corresponding configurations, the SIOA other than $\mathcal{A}, \mathcal{B}$ must be the same and must be in the same state. $\mathcal{A}$ and $\mathcal{B}$ must have the same external signature. In the sequel, when we write $\Psi = \Phi[\mathcal{B}/\mathcal{A}]$, we always assume that $\mathcal{B} \notin \Phi$ and $\mathcal{A} \notin \Psi$.*

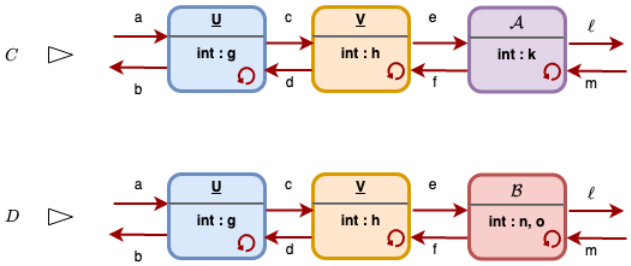


Figure 4.11. – $\lhd_{\mathcal{AB}}$ corresponding-configuration

The next lemma states that $\lhd_{AB}$-corresponding configurations have the same external signature, which is quite intuitive when we see the figure 4.11.

Proposition 10. *Let C, D be configurations such that $C \lhd_{AB} D$. Then $ext(C) = ext(D)$.*

Proof. The proof is in [AL16], section 6, p. 38. We write the proof here to be complete:

If $\mathcal{A} \notin C$ then $C = D$ by definition , and we are done. Now suppose that $\mathcal{A} \in C$, so that $C = (\mathbf{A} \cup \{\mathcal{A}\}, \mathbf{S})$ for some set $\mathbf{A}$ of PSIOA identifiers s.t. $\mathcal{A} \notin \mathbf{A}$, and let $s = \mathbf{S}(\mathcal{A})$. Then, by definition

65 of attributes of configuration, $out(C) = (\bigcup_{\mathcal{A}_i \in \mathbf{A}} out(\mathcal{A}_i)(\mathbf{S}(\mathcal{A}_i))) \cup out(\mathcal{A})(s)$. From $C \lhd_{\mathcal{AB}} D$ and definition , we have $D = (\mathbf{A} \cup \{\mathcal{B}\}, \mathbf{S}')$, where $\mathbf{S}'$ agrees with $\mathbf{S}$ on all $\mathcal{A}_i \in \mathbf{A}$, and $t = \mathbf{S}'(\mathcal{B})$ such that $ext(\mathcal{A})(s) = ext(\mathcal{B})(t)$. Hence $out(\mathcal{A})(s) = out(\mathcal{B})(t)$ and $in(\mathcal{A})(s) = in(\mathcal{B})(t)$. By definition 65 of configuration attributes, $out(D) = (\bigcup_{\mathcal{A}_i \in \mathbf{A}} out(\mathcal{A}_i)(\mathbf{S}'(\mathcal{A}_i))) \cup out(\mathcal{B})(t)$. Finally, $out(C) = out(D)$ since $\mathbf{S}'$ agrees with $\mathbf{S}$ on all $\mathcal{A} \in \mathbf{A}$ and $out(\mathcal{A})(s) = out(\mathcal{B})(t)$. We establish $in(C) = in(D)$ in the same manner, and omit the repetitive details. Hence $ext(C) = ext(D)$. □

Remark 5. *It is possible to have two configurations C, D s.t. $C \lhd_{\mathcal{AA}} D$. That would mean that C and D only differ on the state of $\mathcal{A}$ (s or t) that has even the same external signature in both cases $ext(\mathcal{A})(s) = ext(\mathcal{A})(t)$, while we would potentially have $int(\mathcal{A})(s) \neq int(\mathcal{A})(t)$.*

The next lemma states that $\lhd_{AB}$-corresponding configurations are equal if we omit the automata $\mathcal{A}$ and $\mathcal{B}$.

Lemma 38 (Same configuration)**.** *Let $\mathcal{A}, \mathcal{B} \in Autids$. Let $X_{\mathcal{A}}$, $X_{\mathcal{B}}$ be $\mathcal{A}$-fair and $\mathcal{B}$-fair PCA respectively, where $X_{\mathcal{A}}$ never contains $\mathcal{B}$ and $X_{\mathcal{B}}$ never contains $\mathcal{A}$. Let $Y_{\mathcal{A}} = X_{\mathcal{A}} \setminus \{\mathcal{A}\}$, $Y_{\mathcal{B}} = X_{\mathcal{B}} \setminus \{\mathcal{B}\}$. Let $(x_a, x_b) \in Q_{X_{\mathcal{A}}} \times Q_{X_{\mathcal{B}}}$ s.t. $config(X_{\mathcal{A}})(x_a) \lhd_{AB} config(X_{\mathcal{B}})(x_b)$. Let $y_a = X_{\mathcal{A}}.\mu_s^{\mathcal{A}}(x_a)$, $y_b = X_{\mathcal{A}}.\mu_s^{\mathcal{A}}(x_b)$*

Then $config(Y_A)(y_a) = config(Y_B)(y_b)$.

Proof. By projection, we have $config(Y_{\mathcal{A}})(y_a) \lhd_{AB} config(Y_{\mathcal{B}})(y_b)$ with each configuration that does not contain $\mathcal{A}$ nor $\mathcal{B}$, thus for $config(Y_{\mathcal{A}})(y_a)$ and $config(Y_{\mathcal{B}})(y_b)$ contain the same set of automata ids (rule (1) of $\lhd_{AB}$) and map each automaton of this set to the same state (rule (2) of $\lhd_{AB}$). □

same comportment of two PCA modulo $\mathcal{A}$, $\mathcal{B}$ In this paragraph, we formalize the fact that two PCA have the same comportment, except for $\mathcal{B}$ that supplants $\mathcal{A}$.

First, we formalize the fact that two PCA create some PSIOA in the same manner, excepting for $\mathcal{B}$ that supplants $\mathcal{A}$. Here again, this definition comes from [AL16].

Definition 144 (Creation corresponding configuration automata)**.** *Let X, Y be configuration automata and $\mathcal{A}, \mathcal{B}$ be PSIOA. We say that X, Y are creation-corresponding w.r.t. $\mathcal{A}, \mathcal{B}$ iff*

1. *X never creates $\mathcal{B}$ and Y never creates $\mathcal{A}$.*
2. *$\forall (\alpha, \pi) \in Execs^*(X) \times Execs^*(Y)$ s.t $trace_{\mathcal{A}}(\alpha) = trace_{\mathcal{B}}(\pi)$, for $x = lstate(\alpha), y = lstate(\pi)$, we have Then $\forall a \in \widehat{sig}(X)(x) \cap \widehat{sig}(Y)(y) : created(Y)(y)(a) = created(X)(x)(a)[\mathcal{B}/\mathcal{A}]$.*

Naturally $[\mathcal{B}/\mathcal{A}]$-corresponding sets of created automata are deprived of $\mathcal{A}$ and $\mathcal{B}$ respectively, they become equal, which is formalized in the next lemma.

Lemma 39 (Same creation after projection)**.** *Let $\mathcal{A}, \mathcal{B} \in Autids$. Let $X_{\mathcal{A}}$, $X_{\mathcal{B}}$ be $\mathcal{A}$-fair and $\mathcal{B}$-fair PCA respectively, where $X_{\mathcal{A}}$ never contains $\mathcal{B}$ and $X_{\mathcal{B}}$ never contains $\mathcal{A}$ ($\mathcal{B} \notin UA(X_{\mathcal{A}})$ and $\mathcal{A} \notin UA(X_{\mathcal{B}})$). Let $Y_{\mathcal{A}} = X_{\mathcal{A}} \setminus \{\mathcal{A}\}$, $Y_{\mathcal{B}} = X_{\mathcal{B}} \setminus \{\mathcal{B}\}$. Let $(x_a, x_b) \in Q_{X_{\mathcal{A}}} \times Q_{X_{\mathcal{B}}}$ and $act \in sig(X_{\mathcal{A}})(x_a) \cap sig(X_{\mathcal{B}})(x_b)$ s.t. $created(X_{\mathcal{B}})(x_b)(act) = created(X_{\mathcal{A}})(x_a)(act)[\mathcal{B}/\mathcal{A}]$. Let $y_a = X_{\mathcal{A}}.\mu_s^{\mathcal{A}}(x_a)$, $y_b = X_{\mathcal{B}}.\mu_s^{\mathcal{B}}(x_b)$*

Then $created(Y_{\mathcal{B}})(x_b)(act) = created(Y_{\mathcal{A}})(x_a)(act)$

Proof. By definition of PCA projection, we have $created(Y_{\mathcal{B}})(x_b)(act) = (created(X_{\mathcal{B}})(x_b)(act)) \setminus \mathcal{B} = (created(X_{\mathcal{A}})(x_a)(act)[\mathcal{B}/\mathcal{A}]) \setminus \mathcal{B} = created(X_{\mathcal{A}})(x_a)(act) \setminus \mathcal{A} = created(Y_{\mathcal{A}})(x_a)(act)$. □

Second, we formalize the fact that two PCA hide their actions in the same manner. The definition is strongly inspired by [AL16].

Definition 145 (Hiding corresponding configuration automata)**.** *Let X, Y be configuration automata and $\mathcal{A}, \mathcal{B}$ be PSIOA. We say that X, Y are hiding-corresponding w.r.t. $\mathcal{A}, \mathcal{B}$ iff*

1. *X never creates $\mathcal{B}$ and Y never creates $\mathcal{A}$.*
2. *$\forall (\alpha, \pi) \in Execs^*(X) \times Execs^*(Y)$ s.t $trace_{\mathcal{A}}(\alpha) = trace_{\mathcal{B}}(\pi)$, for $x = lstate(\alpha), y = lstate(\pi)$, we have $hidden\text{-}actions(Y)(y) = hidden\text{-}actions(X)(x)$.*

Naturally if hidden actions of $\lhd_{\mathcal{AB}}$-corresponding states are equal, it remains true after respective deprivation of $\mathcal{A}$ and $\mathcal{B}$ which is formalized in next lemma.

Lemma 40 (Same hidden-actions after projection)**.** *Let $\mathcal{A}, \mathcal{B} \in Autids$. Let $X_{\mathcal{A}}$, $X_{\mathcal{B}}$ be $\mathcal{A}$-fair and $\mathcal{B}$-fair PCA respectively, where $X_{\mathcal{A}}$ never contains $\mathcal{B}$ and $X_{\mathcal{B}}$ never contains $\mathcal{A}$ ($\mathcal{B} \notin UA(X_{\mathcal{A}})$ and $\mathcal{A} \notin UA(X_{\mathcal{B}})$). Let $Y_{\mathcal{A}} = X_{\mathcal{A}} \setminus \{\mathcal{A}\}$, $Y_{\mathcal{B}} = X_{\mathcal{B}} \setminus \{\mathcal{B}\}$. Let $(x_a, x_b) \in Q_{X_{\mathcal{A}}} \times Q_{X_{\mathcal{B}}}$, $y_a = X_{\mathcal{A}}.\mu_s^{\mathcal{A}}(x_a)$, $y_b = X_{\mathcal{B}}.\mu_s^{\mathcal{B}}(x_b)$ s.t.*

- *$x_a R_{conf}^{\setminus\{\mathcal{A}\}} x_b$, i.e. $y_a R_{conf} y_b$*
- *$hidden\text{-}actions(X_{\mathcal{B}})(x_b) = hidden\text{-}actions(X_{\mathcal{A}})(x_a)$*

Then $hidden\text{-}actions(Y_{\mathcal{B}})(y_b) = hidden\text{-}actions(Y_{\mathcal{A}})(y_a)$

Proof. We note $C_{X_{\mathcal{A}}} = config(X_{\mathcal{A}})(x_a)$, $C_{X_{\mathcal{B}}} = config(X_{\mathcal{B}})(x_b)$, $C_{Y_{\mathcal{A}}} = config(Y_{\mathcal{A}})(y_a)$, $C_{Y_{\mathcal{B}}} = config(Y_{\mathcal{B}})(y_b)$. By assumption, $C_{X_{\mathcal{A}}} \setminus \{\mathcal{A}\} = C_{Y_{\mathcal{A}}} = C_{Y_{\mathcal{B}}} = C_{X_{\mathcal{B}}} \setminus \{\mathcal{B}\}$.

We note $h_{X_{\mathcal{A}}} = hidden\text{-}actions(X_{\mathcal{A}})(x_a)$, $h_{X_{\mathcal{B}}} = hidden\text{-}actions(X_{\mathcal{B}})(x_b)$, $h_{Y_{\mathcal{A}}} = hidden\text{-}actions(Y_{\mathcal{A}})(y_a)$, $h_{Y_{\mathcal{B}}} = hidden\text{-}actions(Y_{\mathcal{B}})(y_b)$. By assumption, $h_{X_{\mathcal{A}}} = h_{X_{\mathcal{B}}}$, while by construction, $h_{Y_{\mathcal{A}}} = h_{X_{\mathcal{A}}} \setminus pot\text{-}out(X_{\mathcal{A}})(\mathcal{A})$ and $h_{Y_{\mathcal{B}}} = h_{X_{\mathcal{B}}} \setminus pot\text{-}out(X_{\mathcal{B}})(\mathcal{B})$.

Case 1: $pot\text{-}out(X_{\mathcal{A}})(\mathcal{A})(x_a) = pot\text{-}out(X_{\mathcal{B}})(\mathcal{B})(x_b)$, the result is immediate, Case 2: $pot\text{-}out(X_{\mathcal{A}})(\mathcal{A})(x_a) \cap h_{X_{\mathcal{A}}} = pot\text{-}out(X_{\mathcal{B}})(\mathcal{B})(x_b) \cap h_{X_{\mathcal{B}}} = \emptyset$, the result is immediate.

Case 3: Without loss of generality, we assume $\underline{act} = pot\text{-}out(X_{\mathcal{A}})(\mathcal{A})(x_a) \cap h_{X_{\mathcal{A}}} \neq \emptyset$. For every $\mathcal{C} \in auts(C_{Y_{\mathcal{B}}})$, $\mathcal{C} \in auts(C_{Y_{\mathcal{A}}})$ since $C_{Y_{\mathcal{A}}} = C_{Y_{\mathcal{B}}}$ and $\mathcal{C} \in auts(C_{X_{\mathcal{A}}})$ since $C_{Y_{\mathcal{A}}} = C_{X_{\mathcal{A}}} \setminus \{\mathcal{A}\}$. By compatibility of $C_{X_{\mathcal{A}}}$, $pot\text{-}out(X_{\mathcal{A}})(\mathcal{A})(x_a) \cap pot\text{-}out(X_{\mathcal{A}})(\mathcal{C})(x_a) = \emptyset$.

Case 3a) $\mathcal{B} \notin auts(C_{X_{\mathcal{B}}})$, which means both i) $\underline{act} \subset h_{X_{\mathcal{B}}}$, ii) $\underline{act} \cap out(C_{X_{\mathcal{B}}}) = \emptyset$ and iii) $h_{X_{\mathcal{B}}} \subset out(C_{X_{\mathcal{B}}})$ which is impossible. Thus we only consider

Case 3b) $\mathcal{B} \in auts(C_{X_{\mathcal{B}}})$. Since j) for every $\mathcal{C} \in auts(C_{Y_{\mathcal{B}}})$, $pot\text{-}out(X_{\mathcal{A}})(\mathcal{A})(x_a) \cap pot\text{-}out(X_{\mathcal{A}})(\mathcal{C})(x_a) = \emptyset$ and jj) $h_{X_{\mathcal{B}}} \subset out(C_{X_{\mathcal{B}}})$, we have $\underline{act} \subset pot\text{-}out(X_{\mathcal{B}})(\mathcal{B})(x_b)$.

For symmetrical reason, we have both $pot\text{-}out(X_{\mathcal{A}})(\mathcal{A})(x_a) \cap h_{X_{\mathcal{A}}} \subset pot\text{-}out(X_{\mathcal{B}})(\mathcal{B})(x_b)$ and $pot\text{-}out(X_{\mathcal{B}})(\mathcal{B})(x_b) \cap h_{X_{\mathcal{B}}} \subset pot\text{-}out(X_{\mathcal{A}})(\mathcal{A})(x_A)$, which means $h_{X_{\mathcal{A}}} \setminus pot\text{-}out(X_{\mathcal{B}})(\mathcal{B})(x_b) = h_{X_{\mathcal{B}}} \setminus pot\text{-}out(X_{\mathcal{B}})(\mathcal{B})(x_b)$ and ends the proof

□

Now we are ready to define corresponding PCA w.r.t. PSIOA $\mathcal{A}$, $\mathcal{B}$, that is two PCA $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ that differ only on the fact that B supplants $\mathcal{A}$ in $X_{\mathcal{B}}$. Some additional assumptions are added to ensure monotonicity later. This definition is still inspired by definitions of [AL16].

Definition 146 (corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$)**.** *Let $\mathcal{A}, \mathcal{B} \in Autids$, $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ be PCA we say that $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ are corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$, if they verify:*

- $config(X_{\mathcal{A}})(\bar{q}_{X_{\mathcal{A}}}) \lhd_{AB} config(X_{\mathcal{B}})(\bar{q}_{X_{\mathcal{B}}})$.
- *$X_{\mathcal{A}}$ never contains $\mathcal{B}$ ($\mathcal{B} \notin UA(X_{\mathcal{A}})$), while $X_{\mathcal{B}}$ never contains $\mathcal{A}$ ($\mathcal{A} \notin UA(X_{\mathcal{B}})$).*
- *$X_{\mathcal{A}}, X_{\mathcal{B}}$ are creation-corresponding w.r.t. $\mathcal{A}, \mathcal{B}$.*
- *$X_{\mathcal{A}}, X_{\mathcal{B}}$ are hiding-corresponding w.r.t. $\mathcal{A}, \mathcal{B}$.*
- *$X_{\mathcal{A}}$ (resp. $X_{\mathcal{B}}$) is a $\mathcal{A}$-conservative (resp. $\mathcal{B}$-conservative) PCA.*
- *(No exclusive creation from $\mathcal{A}$ and $\mathcal{B}$)*
  - *$\forall q_{X_{\mathcal{A}}} \in Q_{X_{\mathcal{A}}}$, for every action act $\mathcal{A}$-exclusive, $created(X_{\mathcal{A}})(q_{X_{\mathcal{A}}})(act) = \emptyset$ and similarly*
  - *$\forall q_{X_{\mathcal{B}}} \in Q_{X_{\mathcal{B}}}$, for every action act' $\mathcal{B}$-exclusive, $created(X_{\mathcal{B}})(q_{X_{\mathcal{B}}})(act') = \emptyset$*

equivalent transitions to obtain semantic equivalence after projection In this last paragraph of the section, we show that if two PCA $X_{\mathcal{A}}$ $X_{\mathcal{B}}$ are corresponding w.r.t. $\mathcal{A}$ and $\mathcal{B}$, then there respective projection $Y_{\mathcal{A}} = X_{\mathcal{A}} \setminus \{\mathcal{A}\}$ and $Y_{\mathcal{B}} = X_{\mathcal{B}} \setminus \{\mathcal{B}\}$ are semantically equivalents. To do so, we use notions of equivalent transitions. the idea is to recursively show that any corresponding executions of $Y_{\mathcal{A}}$ and $Y_{\mathcal{B}}$ lead to strictly equivalent transitions to finally build the complete bijective PCA executions-matching from $Y_{\mathcal{A}}$ to $Y_{\mathcal{B}}$.

We start by defining equivalent transitions.

Definition 147 (configuration-equivalence and strict-equivalence between two distributions)**.** *Let K, K' be PCA and $(\eta, \eta') \in Disc(states(K)) \times Disc(states(K'))$.*

- *We say that η and η' are* config-equivalent*, noted $\eta \xleftrightarrow[conf]{f} \eta'$, if there exists $f : Q_K \longrightarrow Q_{K'}$ s.t. $\eta \xleftrightarrow{f} \eta'$ with $\forall q'' \in supp(\eta)$, $q'' R_{conf} f(q'')$.*
- *If additionally, $\forall q'' \in supp(\eta)$, $q'' R_{strict} f(q'')$, then we say that η and η' are* strictly-equivalent*, noted $\eta \xleftrightarrow[strict]{f} \eta'$.*

Basically, equivalent transitions are transitions where the states with a non-zero probability to be reached are mapped by a bijective function that preserves i) the probability to be reached and ii) configuration. A stricter version preserves also iii) future-created automata and hidden actions.

The next lemma states that if we take two corresponding transitions from strict equivalent states, then we obtain configuration equivalent transitions.

Lemma 41. *(strictly-equivalent states implies config-equivalent transition) Let K, K' be PCA and let $(q, q') \in Q_K \times Q_{K'}$ strictly-equivalent, i.e. $q R_{stricit} q'$. Let $a \in \widehat{sig}(K)(q) = \widehat{sig}(K')(q')$ and let $((q, a, \eta_{(K,q,a)}), (q', a, \eta_{(K',q',a)})) \in D_K \times D_{K'}$. Then $\eta_{(K,q,a)}$ and $\eta_{(K',q',a)}$ are config-equivalent, i.e. $\exists f : Q_K \to Q_{K'}$ s.t. $\eta \xleftrightarrow[conf]{f} \eta'$.*

Proof. This is the direct consequence of constraints 2 and 3 of definition 68 of PCA. We note $C = config(K)(q) = config(K')(q')$ and $\varphi = created(K)(q)(a) = created(K')(q')(a)$. By constraint 2, applied to K, there exists η s.t. $\eta_{(K,q,a)} \xleftrightarrow{f^K} \eta$ with $f^K = config(K)$ and $config(K)(q) \stackrel{a}{\Longrightarrow}_{created(K)(q)(a)} \eta$ By constraint 2, applied to K', there exists η' s.t. $\eta_{(K',q',a)} \xleftrightarrow{f^{K'}} \eta'$ with $f^{K'} = config(K')$ and $config(K')(q') \stackrel{a}{\Longrightarrow}_{created(K')(q')(a)} \eta'$.

Since $q R_{strict} q'$, $C \triangleq config(K)(q) = config(K')(q')$ and $\varphi \triangleq created(K)(q)(a) = created(K')(q')(a)$. Hence $C \stackrel{a}{\Longrightarrow}_{\varphi} \eta$ and $C \stackrel{a}{\Longrightarrow}_{\varphi} \eta'$ which means $\eta = \eta'$.

So $\eta_{(K,q,a)} \xleftrightarrow{f} \eta_{(K',q',a)}$ with $\tilde{f} = (\tilde{f}^{K\prime})^{-1} \circ \tilde{f}^{K}$ where $\tilde{f}$ (resp. $\tilde{f}^{K\prime}$, resp. $\tilde{f}^{K}$) is the restriction of f (resp. $f^{K\prime}$, resp. f^{K}) on $supp(\eta_{(K,q,a)})$ (resp. $supp(\eta_{(K',q',a)})$, resp. $supp(\eta_{(K,q,a)})$).

Thus, for every $(\tilde{q}, \tilde{q}') \in supp(\eta_{(K,q,a)}) \times supp(\eta_{(K',q',a)})$ s.t. $\tilde{q}' = f(\tilde{q})$, $f^{K}(\tilde{q}) = f^{K\prime}(\tilde{q}')$, that is $config(K)(\tilde{q}) = config(K')(\tilde{q}')$, i.e. $\tilde{q} R_{conf} \tilde{q}'$.

Hence $\eta_{(K,q,a)} \xleftrightarrow[conf]{f} \eta_{(K',q',a)}$ which ends the proof.

□

Now we start a sequence of lemma (from lemma 42 to lemma 44) to finally show in theorem 22 that if $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ are corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$ then $X_{\mathcal{A}} \setminus \{\mathcal{A}\}$ and $X_{\mathcal{B}} \setminus \{\mathcal{B}\}$ are semantically-equivalent.

The next lemma shows that we can always construct an execution $\tilde{\alpha}_X \in Execs(X)$ from an execution $\alpha_Y \in Execs(Y)$ with $Y = X \setminus \{\mathcal{A}\}$ that preserves the trace.

Lemma 42 ($Execs(X \setminus \{\mathcal{A}\})$ can be obtained by $Execs(X)$)**.** *Let $\mathcal{A} \in Autids$, X a $\mathcal{A}$-fair PCA, $Y = X \setminus \{\mathcal{A}\}$.*

Let $\alpha_Y = q_Y^0, a^1, q_Y^1, ..., q_Y^n \in Execs(Y)$. Then there exists, $\tilde{\alpha}_X = \tilde{q}_X^0, a^1, \tilde{q}_X^1, ..., \tilde{q}_X^n \in Execs(X)$ s.t. $\forall i \in [0, n], q_Y^i = \mu_s^{\mathcal{A}}(\tilde{q}_X^i)$.

Proof. By induction on the size $s = |\alpha_Y^s|$ of prefix $\alpha_Y^s = q_Y^0, a^1, q_Y^1, ..., q_Y^s$.

Basis ($|\alpha_Y^s| = 0$): By definition 132, $\bar{q}_Y = X.\mu_s^{\mathcal{A}}(\bar{q}_X)$

Induction: let assume the proposition is true for prefix $\alpha_Y^s = q_Y^0, a^1, q_Y^1, ..., q_Y^s$ with $s < |\alpha_Y|$. We will show it is true for α_Y^{s+1}. We have $q_Y^s = X.\mu_s^{\mathcal{A}}(q_X^s)$. By construction of D_Y provided by definition 132, there exists $\eta_{(X,q_X^s,a^{s+1})} \in D_X$ s.t. $X.\mu_d^{\mathcal{A}}(\eta_{(X,q_X^s,a^{s+1})}) = \eta_{(Y,q_Y^s,a^{s+1})}$. By $X.\mu_d^{\mathcal{A}}$-correspondence of definition 132, $\eta_{(Y,q_Y^s,a^{s+1})}(q_Y^{s+1}) = \sum_{q'_X \in Q_X, \mu_s(q'_X) = q_Y^{s+1}} \eta_{(X,q_X^s,a^{s+1})}(q'_X)$. By definition of an execution, $q_Y^{s+1} \in supp(\eta_{(Y,q_Y^s,a^{s+1})})$, which means there exists $q_X^{s+1} \in Q_X$ s.t. 1) $\mu_s^{\mathcal{A}}(q_X^{s+1}) = q_Y^{s+1}$ and 2) $q_X^{s+1} \in supp(\eta_{(X,q_X^s,a^{s+1})})$. Thus, it exist $\tilde{\alpha}_X^{s+1} = \tilde{q}_X^0, a^1, \tilde{q}_X^1, ..., \tilde{q}_X^{s+1} \in Execs(X)$ s.t. $\forall i \in [0, s+1], q_Y^i = \mu_s^{\mathcal{A}}(\tilde{q}_X^i)$, which ends the induction and so the proof. □

The next lemma states that, after projection, two configuration-equivalent states obtain via executions with the same trace are strictly equivalent.

Lemma 43 (After projection, configuration-equivalence obtain after same trace implies strict equivalence)**.** *Let $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ be two PCA corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$. Let $Y_{\mathcal{A}} = X_{\mathcal{A}} \setminus \{\mathcal{A}\}$ and $Y_{\mathcal{B}} = X_{\mathcal{B}} \setminus \{\mathcal{B}\}$. Let $(\alpha_{Y_{\mathcal{A}}}, \pi_{Y_{\mathcal{B}}}) \in Execs(Y_{\mathcal{A}}) \times Execs(Y_{\mathcal{B}})$ with $lstate(\alpha_{Y_{\mathcal{A}}}) = q_{Y_{\mathcal{A}}}$ and $lstate(\pi_{Y_{\mathcal{B}}}) = q_{Y_{\mathcal{B}}}$. If*

- *$q_{Y_{\mathcal{A}}} R_{conf} q_{Y_{\mathcal{B}}}$ and*
- *$trace(\alpha_{Y_{\mathcal{A}}}) = trace(\pi_{Y_{\mathcal{B}}}) = \beta$,*

then $q_{Y_{\mathcal{A}}} R_{strict} q_{Y_{\mathcal{B}}}$

Proof. By lemma 42, $\exists(\tilde{\alpha}_{X_{\mathcal{A}}}, \tilde{\pi}_{X_{\mathcal{B}}}) \in Execs(X_{\mathcal{A}}) \times Execs(X_{\mathcal{B}})$ s.t. (i) $trace(\tilde{\alpha}_{X_{\mathcal{A}}}) = trace(\alpha_{Y_{\mathcal{A}}}) = trace(\pi_{Y_{\mathcal{B}}}) = trace(\tilde{\pi}_{X_{\mathcal{B}}})$ and (ii) $q_{Y_{\mathcal{A}}} = X_{\mathcal{A}}.\mu_s^{\mathcal{A}}(\tilde{q}_{X_{\mathcal{A}}})$ and $q_{Y_{\mathcal{B}}} = X_{\mathcal{B}}.\mu_s^{\mathcal{B}}(\tilde{q}_{X_{\mathcal{B}}})$ where $\tilde{q}_{X_{\mathcal{B}}} = lstate(\tilde{\pi}_{X_{\mathcal{B}}})$ and $\tilde{q}_{X_{\mathcal{A}}} = lstate(\tilde{\alpha}_{X_{\mathcal{A}}})$.

Since $trace(\tilde{\alpha}_{X_{\mathcal{A}}}) = trace(\tilde{\pi}_{X_{\mathcal{B}}})$, we have j) $hidden\text{-}actions(X_{\mathcal{A}})(\tilde{q}_{X_{\mathcal{A}}}) = hidden\text{-}actions(X_{\mathcal{B}})(\tilde{q}_{X_{\mathcal{B}}})$ by hiding-correspondence of definition 95 and jj) $\forall a \in \widehat{sig}(X_{\mathcal{A}})(\tilde{q}_{X_{\mathcal{A}}}) \cap \widehat{sig}(X_{\mathcal{B}})(\tilde{q}_{X_{\mathcal{B}}})$, $created(X_{\mathcal{A}})(\tilde{q}_{X_{\mathcal{A}}})(a) = created(X_{\mathcal{B}})(\tilde{q}_{X_{\mathcal{B}}})(a)$.

By lemma 40 we have (*) $hidden\text{-}actions(Y_{\mathcal{A}})(\tilde{q}_{Y_{\mathcal{A}}}) = hidden\text{-}actions(Y_{\mathcal{B}})(\tilde{q}_{Y_{\mathcal{B}}})$, and by lemma 39 we have (**) $\forall a \in \widehat{sig}(Y_{\mathcal{A}})(q_{Y_{\mathcal{A}}}) = \widehat{sig}(Y_{\mathcal{B}})(q_{Y_{\mathcal{B}}})$.

If we combine the definition $q_{Y_{\mathcal{A}}} R_{conf} q_{Y_{\mathcal{B}}}$ with (*) and (**), we obtain $q_{Y_{\mathcal{A}}} R_{strict} q_{Y_{\mathcal{B}}}$, which ends the proof.

□

Finally, the next lemma states that, after projection, two configuration-equivalent states obtain via executions with the same trace lead necessarily to strictly equivalent transitions.

Lemma 44 (After projection, configuration-equivalence obtain after same trace implies strict equivalent transitions)**.** *Let $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ be two PCA corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$. Let $Y_{\mathcal{A}} = X_{\mathcal{A}} \setminus \{\mathcal{A}\}$ and $Y_{\mathcal{B}} = X_{\mathcal{B}} \setminus \{\mathcal{B}\}$. Let $(\alpha_{Y_{\mathcal{A}}}, \pi_{Y_{\mathcal{B}}}) \in Execs(Y_{\mathcal{A}}) \times Execs(Y_{\mathcal{B}})$ with $lstate(\alpha_{Y_{\mathcal{A}}}) = q_{Y_{\mathcal{A}}}$ and $lstate(\pi_{Y_{\mathcal{B}}}) = q_{Y_{\mathcal{B}}}$. If*

- *$q_{Y_{\mathcal{A}}} R_{conf} q_{Y_{\mathcal{B}}}$ and*
- *$trace(\alpha_{Y_{\mathcal{A}}}) = trace(\pi_{Y_{\mathcal{B}}}) = \beta$,*

then for every $a \in \widehat{sig}(Y_{\mathcal{A}})(q_{Y_{\mathcal{A}}}) = \widehat{sig}(Y_{\mathcal{B}})(q_{Y_{\mathcal{B}}})$, $\eta_{(Y_{\mathcal{A}}, q_{Y_{\mathcal{A}}}, a)}$ and $\eta_{(Y_{\mathcal{B}}, q_{Y_{\mathcal{B}}}, a)}$ are strictly equivalent, i.e. $\exists f : Q_K \to Q_{K'}$ s.t. $\eta \xleftrightarrow[strict]{f} \eta'$

Proof. By previous lemma 43, $q_{Y_{\mathcal{A}}}$ and $q_{Y_{\mathcal{B}}}$ are strictly equivalent. Thus by previous lemma 41, there exists f s.t. $\eta_{(Y_{\mathcal{A}}, q_{Y_{\mathcal{A}}}, a)} \xleftrightarrow[conf]{f} \eta_{(Y_{\mathcal{B}}, q_{Y_{\mathcal{B}}}, a)}$. Let two corresponding states $(q'_{Y_{\mathcal{A}}}, q'_{Y_{\mathcal{B}}}) \in supp(\eta_{(Y_{\mathcal{A}}, q_{Y_{\mathcal{A}}}, a)}) \times \eta_{(Y_{\mathcal{B}}, q_{Y_{\mathcal{B}}}, a)}$ s.t. $f(q'_{Y_{\mathcal{A}}}) = q'_{Y_{\mathcal{B}}}$. We have $q'_{Y_{\mathcal{A}}} R_{conf} q'_{Y_{\mathcal{B}}}$(*). Furthermore, since $q_{Y_{\mathcal{A}}} R_{strict} q_{Y_{\mathcal{B}}}$, $sig(Y_{\mathcal{A}})(q_{Y_{\mathcal{A}}}) = sig(Y_{\mathcal{B}})(q_{Y_{\mathcal{B}}})$, namely $ext(Y_{\mathcal{A}})(q_{Y_{\mathcal{A}}}) = ext(Y_{\mathcal{B}})(q_{Y_{\mathcal{B}}})$, which means $trace(\alpha_{\widehat{Y_{\mathcal{A}}}} q_{Y_{\mathcal{A}}} a q'_{Y_{\mathcal{A}}}) = trace(\pi_{\widehat{Y_{\mathcal{B}}}} q_{Y_{\mathcal{B}}} a q'_{Y_{\mathcal{B}}})$. So we can reapply previous lemma to obtain $q'_{Y_{\mathcal{A}}} R_{strict} q'_{Y_{\mathcal{B}}}$ which ends the proof.

□

Now we can finally show that if $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ are corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$ then $X_{\mathcal{A}} \setminus \{\mathcal{A}\}$ and $X_{\mathcal{B}} \setminus \{\mathcal{B}\}$ are semantically-equivalent which was the main aim of this subsection.

Theorem 22 ($X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$ implies $X_{\mathcal{A}} \setminus \{\mathcal{A}\}$ and $X_{\mathcal{B}} \setminus \{\mathcal{B}\}$ semantically-equivalent)**.** *Let $\mathcal{A}, \mathcal{B}$ be PSIOA. Let $X_{\mathcal{A}}$, $X_{\mathcal{B}}$ be PCA. $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$ implies $(X_{\mathcal{A}} \setminus \{\mathcal{A}\})$ and $(X_{\mathcal{B}} \setminus \{\mathcal{B}\})$ are semantically-equivalent.*

Proof. We note $Y_{\mathcal{A}} = X_{\mathcal{A}} \setminus \{\mathcal{A}\}$ and $Y_{\mathcal{B}} = X_{\mathcal{B}} \setminus \{\mathcal{B}\}$. We recursively construct a strong complete bijective PCA executions-matching $(f_s, f_s^{tran}, f_s^{ex})$ where $f_s : reachable_{\leq s}(Y_{\mathcal{A}}) \to reachable_{\leq s}(Y_{\mathcal{B}})$ and $f_s^{ex} : \{\alpha \in Execs(Y_{\mathcal{A}}) || \alpha| \leq s\} \to \{\pi \in Execs(Y_{\mathcal{B}}) || \pi| \leq s\}$ s.t. $f_s^{ex}(\alpha) = \pi$ implies $lstate(\alpha) R_{strict} lstate(\pi)$.

Basis: $s = 0$, $reachable_{\leq 0}(Y_{\mathcal{A}}) = \{\bar{q}_{X_{\mathcal{A}}}\}$, while $reachable_{\leq 0}(Y_{\mathcal{B}}) = \{\bar{q}_{X_{\mathcal{B}}}\}$.

By definition 107 of corresponding automata $config(X_{\mathcal{A}})(\bar{q}_{X_{\mathcal{A}}}) \lhd_{\mathcal{A}\mathcal{B}} config(X_{\mathcal{B}})(\bar{q}_{X_{\mathcal{B}}})$, while $(\bar{q}_{Y_{\mathcal{A}}}, \bar{q}_{Y_{\mathcal{B}}}) = (X_{\mathcal{A}}.\mu_s^{\mathcal{A}}(\bar{q}_{X_{\mathcal{A}}}), X_{\mathcal{B}}.\mu_s^{\mathcal{B}}(\bar{q}_{X_{\mathcal{B}}}))$ by definition 132 of PCA projection, which gives $\bar{q}_{Y_{\mathcal{A}}} R_{conf} \bar{q}_{Y_{\mathcal{B}}}$ by lemma 38. Moreover $trace_{Y_{\mathcal{A}}}(\bar{q}_{Y_{\mathcal{A}}}) = trace_{Y_{\mathcal{B}}}(\bar{q}_{Y_{\mathcal{B}}}) = \lambda$ (λ denotes the empty sequence). Thus we can apply lemma 43 to obtain $\bar{q}_{Y_{\mathcal{A}}} R_{strict} \bar{q}_{Y_{\mathcal{B}}}$. We construct $f_0(\bar{q}_{Y_{\mathcal{A}}}) = \bar{q}_{Y_{\mathcal{B}}}$, $f_0^{ex}(\bar{q}_{Y_{\mathcal{A}}}) = \bar{q}_{Y_{\mathcal{B}}}$. Clearly f_0 is a bijection from $reachable_0(Y_{\mathcal{A}})$ to $reachable_0(Y_{\mathcal{B}})$, while f_0^{ex} is a bijection from $Execs_0(Y_{\mathcal{A}})$ to $Execs_0(Y_{\mathcal{B}})$

Induction: We assume the result to be true for an integer $s \in \mathbb{N}$ and we will show it is then true for $s + 1$. Let $Execs_s(Y_{\mathcal{A}}) = \{\alpha \in Execs(Y_{\mathcal{A}}) || \alpha| = s\}$ and $Execs_s(Y_{\mathcal{B}}) = \{\pi \in Execs(Y_{\mathcal{B}}) || \pi| = s\}$.

We can build f_{s+1} (resp. f^{ex}_{s+1}) s.t. $\forall q \in reachable_{\leq s}(Y_{\mathcal{A}}), f_{s+1}(q) = f_s(q)$ (resp. s.t. $\forall \alpha \in Execs_{\leq s}(Y_{\mathcal{A}})$ $f^{ex}_{s+1}(\alpha) = f^{ex}_s(\alpha)$) and $\forall q^j_{Y_{\mathcal{A}}} \in reachable_{s+1}(Y_{\mathcal{A}}), f_{s+1}(q^*)$ (resp. $\forall \alpha^{a,j} \in Execs_s(Y_{\mathcal{A}})$, $f^{ex}_{s+1}(\alpha')$) is built as follows:

We note $\alpha^{a,j} = \alpha_{Y_{\mathcal{A}}}\widehat{q_{Y_{\mathcal{A}}}aq^j_{Y_{\mathcal{A}}}}$ ($q_{Y_{\mathcal{A}}} = lstate(\alpha_{Y_{\mathcal{A}}})$). We note $\pi_{Y_{\mathcal{B}}} = f^{ex}_s(\alpha_{Y_{\mathcal{A}}})$. By induction assumption, $q_{Y_{\mathcal{A}}} R_{strict} q_{Y_{\mathcal{B}}}$ with $q_{Y_{\mathcal{A}}} = lstate(\alpha_{Y_{\mathcal{A}}})$ and $q_{Y_{\mathcal{B}}} = lstate(\pi_{Y_{\mathcal{B}}})$. Hence $sig(Y_{\mathcal{A}})(q_{Y_{\mathcal{A}}}) = sig(Y_{\mathcal{B}})(q_{Y_{\mathcal{B}}})$ and by previous lemma 44, for every $a \in sig(Y_{\mathcal{A}})(q_{Y_{\mathcal{A}}}) = sig(Y_{\mathcal{B}})(q_{Y_{\mathcal{B}}})$, $\exists g^j_a$, $\eta_{(Y_{\mathcal{A}},q_{Y_{\mathcal{A}}},a)} \overset{g^j_a}{\underset{strict}{\longleftrightarrow}} \eta_{(Y_{\mathcal{B}},q_{Y_{\mathcal{B}}},a)}$.

Hence, we define $f^{ex}_{s+1} : \alpha^{a,j} = \alpha_{Y_{\mathcal{A}}}\widehat{q_{Y_{\mathcal{A}}}aq^j_{Y_{\mathcal{A}}}} \mapsto f^{ex}_{s+1}(\alpha_{Y_{\mathcal{A}}})^\frown f_s(q_{Y_{\mathcal{A}}})ag^j_a(q^j_{Y_{\mathcal{A}}})$, while f_{s+1} is naturally defined via f^{ex}_{s+1}, i.e. for every $q^j_{Y_{\mathcal{A}}} \in reachable_{s+1}(Y_{\mathcal{A}})$, we note $\alpha^{a,j} \in Execs_{s+1}(Y_{\mathcal{A}})$ s.t. $lstate(\alpha^{a,j}) = q^j_{Y_{\mathcal{A}}}$ and $f_{s+1}(q^j_{Y_{\mathcal{A}}}) = g^j_a(q^j_{Y_{\mathcal{A}}}) = lstate(f^{ex}_{s+1}(\alpha^{a,j}))$.

We finally define $f^{ex} : q^0a^1...a^nq^n... \mapsto f_0(q^0)a^1...a^nf_n(q^n)$, $f : q \mapsto f_n(q)$ where $q = lstate(q^0a^1...q^n)$ and $f^{tr} : (q, a, \eta_{(Y_{\mathcal{A}},q,a)}) \mapsto (f(q), a, \eta_{(Y_{\mathcal{B}},f(q),a)})$.

Clearly (f, f^{tr}, f^{ex}) is strong since for every pair $(q_{Y_{\mathcal{A}}}, q_{Y_{\mathcal{B}}})$, s.t. $f(q_{Y_{\mathcal{A}}}) = q_{Y_{\mathcal{B}}}$, $q_{Y_{\mathcal{A}}} R_{strict} q_{Y_{\mathcal{B}}}$.

Moreover, (f, f^{tr}, f^{ex}) is complete since $dom(f) = reachable(Y_{\mathcal{A}}) = Q_{Y_{\mathcal{A}}}$.

Finally, the bijectivity of f^{ex} is given by the inductive bijective construction.

Hence (f, f^{tr}, f^{ex}) is strong complete bijective PCA executions-matching from $Y_{\mathcal{A}}$ to $Y_{\mathcal{B}}$ which ends the proof.

□

4.5. Top/Down corresponding classes

In previous section 4.4, we have shown in theorem 22 that if $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ are corresponding w.r.t. $\mathcal{A}$ and $\mathcal{B}$ (in the sense of definition 107), then $Y_{\mathcal{A}} = X_{\mathcal{A}} \setminus \{\mathcal{A}\}$ and $Y_{\mathcal{B}} = X_{\mathcal{B}} \setminus \{\mathcal{B}\}$ are semantically equivalent. We can note Y an arbitrary PCA semantically equivalent with both $Y_{\mathcal{A}}$ and $Y_{\mathcal{B}}$.

In section 4.3, we have shown in theorem 19 that for every PCA $\mathcal{E}$ environment of both $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$, $X_{\mathcal{A}}||\mathcal{E}$ and $\tilde{\mathcal{A}}^{sw}||Y_{\mathcal{A}}||\mathcal{E}$ (resp. $X_{\mathcal{B}}||\mathcal{E}$ and $\tilde{\mathcal{B}}^{sw}||Y_{\mathcal{B}}||\mathcal{E}$) are linked by a PCA executions-matching

It is time to combine these two results to realize that for every PCA $\mathcal{E}$ environment of both $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$, $X_{\mathcal{A}}||\mathcal{E}$ and $\tilde{\mathcal{A}}^{sw}||\mathcal{E}'$ (resp. $X_{\mathcal{B}}||\mathcal{E}$ and $\tilde{\mathcal{B}}^{sw}||\mathcal{E}'$) are linked by a PCA executions-matching where $\mathcal{E}' = \mathcal{E}||Y$.

Hence (*) if $\mathcal{E}'$ cannot distinguish $\tilde{\mathcal{A}}^{sw}$ from $\tilde{\mathcal{B}}^{sw}$, we will be able to show that $\mathcal{E}$ cannot distinguish $X_{\mathcal{A}}$ from $X_{\mathcal{B}}$.

In this section, we formalise (*) in theorem 23 of monotonicity of implementation relation. However, some assumptions are required to reduce the implementation of $X_{\mathcal{B}}$ by $X_{\mathcal{A}}$ into implementation of $\mathcal{B}$ by $\mathcal{A}$. These are all minor technical assumptions except for one: our implementation relation concerns only a particular subset of schedulers so-called *creation-oblivious*, i.e. in order to compute (potentially randomly) the next transition, they do not take into account the internal actions of a sub-automaton preceding its last destruction.

4.5.1. Creation-oblivious scheduler

Here we recall the definition of creation-oblivious scheduler (already introduced in subsection 3.7.5), which does not take into account previous internal actions of a particular sub-automaton to output its probability over transitions to trigger.

We start by defining *strict oblivious-schedulers* that output the same transition with the same probability for pair of execution fragments that differ only by prefixes in the same class of equivalence. This definition is inspired by the one provided in Segala's thesis but is more restrictive since we require strict equality instead of a correlation (section 5.6.2 in [Seg95b]).

Definition 148 (strict oblivious scheduler (recall))**.** *Let W be a PCA or a PSIOA, let $\sigma \in schedulers(W)$ and let $\equiv$ be an equivalence relation on $Frags^*(W)$ verifying $\forall \alpha_1, \alpha_2 \in Frags^*(W)$ s.t. $\alpha_1 \equiv \alpha_2$, $lstate(\alpha_1) = lstate(\alpha_2)$. We say that σ is $(\equiv)$-strictly oblivious if $\forall \alpha_1, \alpha_2, \alpha_3 \in Frags^*(\tilde{W})$ s.t. 1) $\alpha_1 \equiv \alpha_2$ and 2) $fstate(\alpha_3) = lstate(\alpha_2) = lstate(\alpha_1)$, then $\sigma(\alpha_1 \frown \alpha_3) = \sigma(\alpha_2 \frown \alpha_3)$.*

Now we define the relation of equivalence that defines our subset of creation-oblivious schedulers. Intuitively, two executions fragments ending on $\mathcal{A}$ creation are in the same equivalence class if they differ only in terms of internal actions of $\mathcal{A}$.

Definition 149. *($\tilde{\alpha} \equiv^{cr}_{\mathcal{A}} \tilde{\alpha}'$ (recall)). Let $\tilde{\mathcal{A}}$ be a PSIOA, $\tilde{W}$ be a PCA, $\forall \tilde{\alpha}, \tilde{\alpha}' \in Frags^*(\tilde{W})$, we say $\tilde{\alpha} \equiv^{cr}_{\mathcal{A}} \tilde{\alpha}'$ iff:*

1. *$\tilde{\alpha}, \tilde{\alpha}'$ both ends on $\mathcal{A}$-creation.*
2. *$\tilde{\alpha}$ and $\tilde{\alpha}'$ differ only in the $\mathcal{A}$-exclusive actions and the states of $\mathcal{A}$, i.e. $\mu(\tilde{\alpha}) = \mu(\tilde{\alpha}')$ where $\mu(\tilde{\alpha} = \tilde{q}^0 a^1 \tilde{q}^1 ... a^n \tilde{q}^n) \in Frags^*(\tilde{W})$ is defined as follows:*
   - *remove the $\mathcal{A}$-exclusive actions*
   - *replace each state $\tilde{q}^i$ by its configuration $Config(\tilde{W})(\tilde{q}) = (\mathbf{A}^i, \mathbf{S}^i)$*
   - *replace each configuration $(\mathbf{A}^i, \mathbf{S}^i)$ by $(\mathbf{A}^i, \mathbf{S}^i) \setminus \{\mathcal{A}\}$*

- *replace the (non-alternating) sequences of identical configurations (due to $\mathcal{A}$-exclusiveness of removed actions) by one unique configuration.*

More formally, μ can be recursively defined as follows:

- $\mu(\tilde{\alpha}) = Config(\tilde{W})(\tilde{q}) \setminus \{\mathcal{A}\}$ *if* $\tilde{\alpha} = \tilde{q} \in Q_{\tilde{W}}$
- $\mu(\tilde{\alpha}a\tilde{q}) = \begin{cases} \mu(\tilde{\alpha})a(Config(\tilde{W})(\tilde{q}) \setminus \{\mathcal{A}\}) & \text{if } a \text{ is not } \mathcal{A}\text{-exclusive} \\ \mu(\tilde{\alpha}) & \text{otherwise.} \end{cases}$

3. $lstate(\alpha_1) = lstate(\alpha_2)$

We can remark that the items 3 can be deduced from 1 and 2 if X is configuration-conflict-free. We can also remark that if $\tilde{W}$ is a $\mathcal{A}$-conservative PCA, we can replace $\mu(\tilde{\alpha}) = \mu(\tilde{\alpha}')$, by $\mu_e^{\mathcal{A}}(\tilde{\alpha}) \upharpoonright (\tilde{W} \setminus \{\mathcal{A}\}) = \mu_e^{\mathcal{A}}(\tilde{\alpha}') \upharpoonright (\tilde{W} \setminus \{\mathcal{A}\})$ but we want to be as general as possible for next definition of *creation oblivious scheduler*:

Definition 150 (creation-oblivious scheduler)**.** *Let $\mathcal{A}$ be a PSIOA, W be a PCA, $\sigma \in schedulers(W)$. We say that σ is* $\mathcal{A}$-creation oblivious *if it is $(\equiv_{\mathcal{A}}^{cr})$-strictly oblivious.*

We say that σ is creation-oblivious *if it is $\mathcal{A}$-creation oblivious for every sub-automaton $\mathcal{A}$ of W ($\mathcal{A} \in \bigcup_{q \in Q_W} auts(config(W)(q))$). We note S_o the function that maps every PCA W to the set of creation-oblivious schedulers of W. If W is not a PCA but a PSIOA, $S_o(W) = schedulers(W)$.*

If σ is $\mathcal{A}$-creation oblivious, we can remark that $\forall \alpha, \alpha' \in Execs^*(W), \alpha \equiv_{\mathcal{A}}^{cr} \alpha'$, $\sigma_{|\alpha} = \sigma_{|\alpha'}$ in the sense of definition 151 stated immediately below.

Definition 151 (conditioned scheduler)**.** *Let $\mathcal{A}$ be a PSIOA, $\sigma \in schedulers(\mathcal{A})$ and let $\alpha_1 \in Frags^*(\mathcal{A})$. We note $\sigma_{|\alpha_1} : \{\alpha_2 \in Frags^*(\mathcal{A}) | fstate(\alpha_2) = lstate(\alpha_1)\} \to SubDisc(D_{\mathcal{A}})$ the* sub-scheduler conditioned by σ and α_1 *that verifies $\forall \alpha_2 \in Frags^*(\mathcal{A}), fstate(\alpha_2) = lstate(\alpha_1)$, $\sigma_{|\alpha_1}(\alpha_2) = \sigma(\alpha_1 \frown \alpha_2)$.*

We take the opportunity to state a lemma of conditional probability that will be used later for lemma 58.

Lemma 45 (conditional measure law)**.** *Let $\mathcal{A}$ be a PSIOA, $\sigma \in schedulers(\mathcal{A})$ and let $\alpha_1 \in Frags^*(\mathcal{A})$ and $\sigma_{|\alpha_1}$ the sub-scheduler conditioned by σ and α_1. Let $\alpha_o, \alpha_2 \in Frags^*(\mathcal{A}), fstate(\alpha_2) = lstate(\alpha_1) \triangleq q_{12}$. Then*

$$\epsilon_{\sigma,\alpha_o}(C_{\alpha_1 \frown \alpha_2}) =: \begin{cases} \epsilon_{\sigma,\alpha_o}(C_{\alpha_1}) \cdot \epsilon_{\sigma_{|\alpha_1},q_{12}}(C_{\alpha_2}) & \text{if } \alpha_1 \not\leq \alpha_o \\ \epsilon_{\sigma_{|\alpha_1},\alpha_o'}(C_{\alpha_2}) & \text{if } \alpha_o = \alpha_1 \frown \alpha_o' \end{cases}$$

Proof. We note $\alpha_{12} = \alpha_1 \frown \alpha_2$.

1. $\alpha_1 \not\leq \alpha_o$:
   a) $\alpha_1 \not\leq \alpha_o$ and $\alpha_o \not\leq \alpha_1$:
   This implies $\alpha_{12} \not\leq \alpha_o$ and $\alpha_o \not\leq \alpha_{12}$ thus $\epsilon_{\sigma,\alpha_o}(C_{\alpha_1 \frown \alpha_2}) = \epsilon_{\sigma,\alpha_o}(C_{\alpha_1}) = 0$ which ends the proof.
   b) $\alpha_o \leq \alpha_1$:
   This implies $\alpha_o \leq \alpha_{12}$ By induction on size s of α_2. Basis: $s = 0$, i.e. $\alpha_2 = lstate(\alpha_1) = q_{12}$. Thus, we meet the second case of definition of $\epsilon_{\sigma_{|\alpha_1},q_{12}}(C_{\alpha_2})$: $\alpha_2 \leq q_{12}$, which means $\epsilon_{\sigma_{|\alpha_1},q_{12}}(C_{\alpha_2}) = 1$ and terminates the basis. Induction: We assume the result to be true up to size $s \in \mathbb{N}$ and we want to show it is still true for size $s + 1$. Let $\alpha_2 \in$

$Frags^*(\mathcal{A}), fstate(\alpha_2) = lstate(\alpha_1) \triangleq q_{12}$ with $|\alpha_2| = s+1$. We note $\alpha_2 = \alpha_2'^\frown q'aq$ and $\alpha'_{12} = \alpha_1^\frown \alpha'_2$. We have $|\alpha'_2| = s$ and $\alpha_o \leq \alpha'_{12}$
By definition we have $\epsilon_{\sigma_{|\alpha_1},q_{12}}(C_{\alpha_2}) = \epsilon_{\sigma_{|\alpha_1},q_{12}}(C_{\alpha'_2}) \cdot \sigma(\alpha'_2)(\eta_{(\mathcal{A},q',a)}) \cdot \eta_{(\mathcal{A},q',a)}(q)$.
In Parallel, by definition: $\epsilon_{\sigma,\alpha_o}(C_{\alpha_{12}}) = \epsilon_{\sigma,\alpha_o}(C_{\alpha'_{12}}) \cdot \sigma(\alpha'_{12})(\eta_{(\mathcal{A},q',a)}) \cdot \eta_{(\mathcal{A},q',a)}(q)$ and by induction assumption, $\epsilon_{\sigma,\alpha_o}(C_{\alpha_{12}}) = \epsilon_{\sigma,\alpha_o}(C_{\alpha_1}) \cdot \epsilon_{\sigma_{|\alpha_1},q_{12}}(C_{\alpha'_2}) \cdot \sigma(\alpha'_{12})(\eta_{(\mathcal{A},q',a)}) \cdot \eta_{(\mathcal{A},q',a)}(q)$ and so $\epsilon_{\sigma,\alpha_o}(C_{\alpha_{12}}) = \epsilon_{\sigma,\alpha_o}(C_{\alpha_1}) \cdot \epsilon_{\sigma_{|\alpha_1},q_{12}}(C_{\alpha_2})$, which ends the induction and so the case.

2. $\alpha_o = \alpha_1^\frown \alpha'_o$. By definition, $\epsilon_{\sigma,\alpha_o}(C_{\alpha_1}) = 1$
   a) both $\alpha_{12} \not\leq \alpha_o$ and $\alpha_o \not\leq \alpha_{12}$. This implies $\alpha_2 \not\leq \alpha'_o$ and $\alpha'_o \not\leq \alpha_2$ Then, by definition, $\epsilon_{\sigma,\alpha_o}(C_{\alpha_{12}}) = \epsilon_{\sigma_{|\alpha_1},\alpha'_o}(C_{\alpha_2}) = 0$.
   b) $\alpha_{12} \leq \alpha_o$. This implies $\alpha_2 \leq \alpha'_o$. Then, by definition, $\epsilon_{\sigma,\alpha_o}(C_{\alpha_{12}}) = \epsilon_{\sigma_{|\alpha_1},\alpha'_o}(C_{\alpha_2}) = 1$
   c) $\alpha_o \leq \alpha_{12}$:
   We proceed by induction on size s of α_2.
   Basis: $s = 0$, i.e. $\alpha_2 = q_{12}$. Then by definition $\epsilon_{\sigma,\alpha_o}(C_{\alpha_{12}}) = \epsilon_{\sigma,\alpha_o}(C_{\alpha_1}) = 1$. Moreover $q_{12} \leq \alpha'_o$ which means $\epsilon_{\sigma_{|\alpha_1},\alpha'_o}(C_{\alpha_2}) = 1$, which ends the basis.
   Induction:
   We assume the result to be true up to size $s \in \mathbb{N}$ and we want to show it is still true for size $s+1$. Let $\alpha_2 \in Frags^*(\mathcal{A}), fstate(\alpha_2) = lstate(\alpha_1) \triangleq q_{12}$ with $|\alpha_2| = s+1$. We note $\alpha_2 = \alpha_2'^\frown q'aq$ and $\alpha'_{12} = \alpha_1^\frown \alpha'_2$. We have $|\alpha'_2| = s$ and $\alpha_o \leq \alpha'_{12}$.
   By definition we have $\epsilon_{\sigma_{|\alpha_1},\alpha'_o}(C_{\alpha_2}) = \epsilon_{\sigma_{|\alpha_1},\alpha'_o}(C_{\alpha'_2}) \cdot \sigma(\alpha'_2)(\eta_{(\mathcal{A},q',a)}) \cdot \eta_{(\mathcal{A},q',a)}(q)$.
   In Parallel, by definition: $\epsilon_{\sigma,\alpha_o}(C_{\alpha_{12}}) = \epsilon_{\sigma,\alpha_o}(C_{\alpha'_{12}}) \cdot \sigma(\alpha'_{12})(\eta_{(\mathcal{A},q',a)}) \cdot \eta_{(\mathcal{A},q',a)}(q)$ and by induction assumption, $\epsilon_{\sigma,\alpha_o}(C_{\alpha_{12}}) = \epsilon_{\sigma,\alpha_o}(C_{\alpha_1}) \cdot \epsilon_{\sigma_{|\alpha_1},\alpha'_o}(C_{\alpha'_2}) \cdot \sigma(\alpha'_{12})(\eta_{(\mathcal{A},q',a)}) \cdot \eta_{(\mathcal{A},q',a)}(q)$ and so $\epsilon_{\sigma,\alpha_o}(C_{\alpha_1^\frown \alpha_2}) = \epsilon_{\sigma,\alpha_o}(C_{\alpha_1}) \cdot \epsilon_{\sigma_{|\alpha_1},\alpha'_o}(C_{\alpha_2})$. Finally, since $\epsilon_{\sigma,\alpha_o}(C_{\alpha_1}) = 1$, we have $\epsilon_{\sigma,\alpha_o}(C_{\alpha_{12}}) = \epsilon_{\sigma_{|\alpha_1},\alpha'_o}(C_{\alpha_2})$ which ends the induction, the case and so the proof.

□

We have formally defined our notion of creation-oblivious scheduler. This will be a key property to ensure lemma 56 that allows decomposing the measure of a class of comportment into a function of measures of classes of shorter comportment where no creation of $\mathcal{A}$ or $\mathcal{B}$ occurs excepting potentially at very last action. This reduction is more or less necessary to obtain monotonicity of dynamic creation/destruction with implementation relation.

4.5.2. Tools: proxy function, creation-explicitness, classes

In this subsection, we introduce some tools frequently used during our proof of monotonicity. Later, we will adopt a quite general approach to understand the key properties of a perception function to ensure monotonicity. All these properties will be met by environment projection function $proj_{(.,.)}$, but not by trace function.

First we introduce proxy function, which enables a generic reduction from automata $(\tilde{\mathcal{E}}||X_{\mathcal{A}})$ to automata $((\tilde{\mathcal{E}}||X_{\mathcal{A}} \setminus \{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw})$

Definition 152 (proxy)**.** *Let $\mathcal{A}$ be a PSIOA. Let $f_{(.,.)}$ be an insight function. The $\mathcal{A}$-proxy function of f, noted $f^{\mathcal{A},proxy}_{(.,.)}$, is the insight function s.t. for every $\mathcal{A}$-conservative PCA X, $\forall \tilde{\mathcal{E}} \in env(X)$, $\forall \tilde{\alpha} \in dom((\tilde{\mathcal{E}}||X).\mu_e^{\mathcal{A},+})$, $f^{\mathcal{A},proxy}_{(\tilde{\mathcal{E}},X)}(\tilde{\alpha}) = f_{((\tilde{\mathcal{E}}||X\setminus\{\mathcal{A}\}),\tilde{\mathcal{A}}^{sw})}(\mu_e^{\tilde{\mathcal{A}},+}(\tilde{\alpha}))$*

Second, we define ordinary function, as functions capturing the fact that an environment obtain the exact same insight from $X_{\mathcal{A}}$ or from $((X_{\mathcal{A}} \setminus \{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw})$. Any reasonable insight function is ordinary.

Definition 153 (ordinary)**.** *Let $f_{(.,.)}$ be an insight function. We say $f_{(.,.)}$ is ordinary if for every PSIOA $\mathcal{A}$, for every $\mathcal{A}$-conservative PCA X, $\forall \tilde{\mathcal{E}} \in env(X)$, $\forall \tilde{\alpha} \in dom((\tilde{\mathcal{E}}||X).\mu_e^{\mathcal{A},+})$, $f_{(\tilde{\mathcal{E}},X)}(\tilde{\alpha}) = f_{(\tilde{\mathcal{E}},((X\setminus\{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw}))}(\mu_e^{\tilde{\mathcal{A}},+}(\tilde{\alpha}))$*

It is worthy to remark that for ordinary perception function, a common perception in the reduced world implies a common perception in the original world. This fact will be used in the proof of lemma 54 of partitioning.

Lemma 46 (ordinary perception function)**.** *Let f be an ordinary perception function. Then for every PSIOA $\mathcal{A}$, for every $\mathcal{A}$-conservative PCA X, $\forall \tilde{\mathcal{E}} \in env(X)$, $\forall \tilde{\alpha}, \tilde{\alpha}' \in dom((\tilde{\mathcal{E}}||X).\mu_e^{\mathcal{A},+})$*

$f_{(\tilde{\mathcal{E}},X)}^{\mathcal{A},proxy}(\tilde{\alpha}) = f_{(\tilde{\mathcal{E}},X)}^{\mathcal{A},proxy}(\tilde{\alpha}') \Longrightarrow f_{(\tilde{\mathcal{E}},X)}(\tilde{\alpha}) = f_{(\tilde{\mathcal{E}},X)}(\tilde{\alpha}')$

Proof. By definition of proxy function, $f_{((\tilde{\mathcal{E}}||X\setminus\{\mathcal{A}\}),\tilde{\mathcal{A}}^{sw})}(\mu_e^{\tilde{\mathcal{A}},+}(\tilde{\alpha})) = f_{((\tilde{\mathcal{E}}||X\setminus\{\mathcal{A}\}),\tilde{\mathcal{A}}^{sw})}(\mu_e^{\tilde{\mathcal{A}},+}(\tilde{\alpha}'))$. By definition of perception function, $f_{(\tilde{\mathcal{E}},((X\setminus\{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw}))}(\mu_e^{\tilde{\mathcal{A}},+}(\tilde{\alpha})) = f_{(\tilde{\mathcal{E}},((X\setminus\{\mathcal{A}\})||\tilde{\mathcal{A}}^{sw}))}(\mu_e^{\tilde{\mathcal{A}},+}(\tilde{\alpha}'))$. By definition of ordinary function, $f_{(\tilde{\mathcal{E}},X)}(\tilde{\alpha}) = f_{(\tilde{\mathcal{E}},X)}(\tilde{\alpha}')$. □

Proposition 11. *The environment projection function $proj_{(.,.)}$ (i.e. for each automaton K, $\forall \mathcal{E} \in env(K)$, $proj_{(\mathcal{E},K)} : \alpha \in Execs(\mathcal{E}||K) \mapsto \alpha \upharpoonright \mathcal{E}$) and the trace functions are ordinary function.*

Proof. By definition □

Now, we introduce two new concepts. First, we introduce the notion of creation-explicitness, which states that an automaton has a clear dedicated set of actions to create each sub-automaton. This property of creation-explicitness will clarify the condition to obtain surjectivity of $\tilde{\mu}_e^{\mathcal{A},+}$ since it suffices to consider this function with a restricted range where no action of $creation\text{-}actions(X)(\mathcal{A})$ appears before the last action.

Definition 154 (creation-explicit PCA)**.** *Let $\mathcal{A}$ be a PSIOA and X be a PCA. We say that X is* $\mathcal{A}$-creation-explicit *iff: there exists a set of actions, noted $creation\text{-}actions(X)(\mathcal{A})$, s.t. $\forall q_X \in Q_X$, $\forall a \in \widehat{sig}(X)(q_X)$, if we note $\mathbf{A}_X = auts(config(X)(q_X))$ and $\varphi_X = created(X)(q_X)(a)$, then $\mathcal{A} \notin \mathbf{A}_X \wedge \mathcal{A} \in \varphi_X \Longleftrightarrow a \in creation\text{-}actions(X)(\mathcal{A})$.*

Second, we define classes of equivalence of some executions that imply the exact same perception from the environment.

Definition 155 (class of equivalence)**.** *Let f be an insight function. Let $\mathcal{A}$ be a PSIOA. Let $\mathcal{E} \in env(\mathcal{A})$. Let $\zeta \in \bigcup_{PSIOA\ \mathcal{B}, \mathcal{E} \in env(\mathcal{B})} range(f_{(\mathcal{E},\mathcal{B})})$. We note $Class(\mathcal{E}, \mathcal{A}, f, \zeta) = \{\alpha \in Execs(\mathcal{E}||\mathcal{A})\} | f_{(\mathcal{E},\mathcal{A})}(\alpha) = \zeta\}$.*

4.5.3. Homomorphism between simple classes

In this subsection, we exhibit the conditions such that $\tilde{\mu}_e^{\mathcal{A},+}$ is a homomorphism between the perception after reduction and the original perception. These conditions are met by the projection function.

First, we state that $\tilde{\mu}_e^{\mathcal{A},+}$ is surjective if we consider a range constituted of executions that do not create $\mathcal{A}$ before the very last action.

Lemma 47 (Partial surjectivity with explicit creation)**.** *Let $\mathcal{A}$ be a PSIOA and X be a $\mathcal{A}$-conservative and $\mathcal{A}$-creation-explicit PCA. Let $\tilde{\mathcal{E}}$ be partially-compatible with X. Let $Y = X \setminus \{\mathcal{A}\}$. Let $\mathcal{E}_{\mathcal{A}} = \tilde{\mathcal{E}}||Y$. Let $(((\tilde{\mathcal{E}}||X).\tilde{\mu}_z^{\mathcal{A}}, (\tilde{\mathcal{E}}||X).\tilde{\mu}_z^{\mathcal{A},+}), (\tilde{\mathcal{E}}||X).\tilde{\mu}_{tr}^{\mathcal{A},+}, (\tilde{\mathcal{E}}||X).\tilde{\mu}_e^{\mathcal{A},+})$ be the continued executions matching $\tilde{\mathcal{E}}$-extension of $((X.\tilde{\mu}_z^{\mathcal{A}}, X.\tilde{\mu}_z^{\mathcal{A},+}), X.\tilde{\mu}_{tr}^{\mathcal{A},+}, X.\tilde{\mu}_e^{\mathcal{A},+})$. Let $\alpha, \alpha' \in Execs(\mathcal{E}_{\mathcal{A}}||\tilde{\mathcal{A}}^{sw})$ s.t. creation-actions$(X)(\mathcal{A}) \cap$ actions$(\alpha) = \emptyset$*

1) Then $\exists \tilde{\alpha} \in dom(\tilde{\mu}_e^{\mathcal{A}})$ s.t. $\tilde{\mu}_e^{\mathcal{A},+}(\tilde{\alpha}) = \tilde{\mu}_e^{\mathcal{A}}(\tilde{\alpha}) = \alpha$.

2) If $\alpha' = \alpha^\frown q, a_!, q'$ with $a_! \in$ creation-actions$(X)(\mathcal{A})$, then $\exists \tilde{\alpha}' \in dom(\tilde{\mu}_e^{\mathcal{A},+})$ s.t. $\tilde{\mu}_e^{\mathcal{A},+}(\tilde{\alpha}') = \alpha'$.

Proof. We proof the results in the same order they are stated in the lemma:

1. We note $\alpha = q^0, a^1, ..., a^n, q^n...$ and we proof the result by induction on the prefix size s. Basis: the result trivially holds for any execution α of size 0 by construction of $X \setminus \{\mathcal{A}\}$ that requires $X.\mu_s^{\mathcal{A}}(\bar{q}_X) = \bar{q}_{X\setminus\{\mathcal{A}\}}$. We assume the result holds up to prefix size s and we show it still holds for prefix size $s+1$. We note $\alpha_s = q^0, a^1, ..., a^s, q^s$ and $\tilde{\alpha}^s \in Execs(\tilde{\mathcal{E}}||X)$ s.t. $\tilde{\mu}_e^{\mathcal{A}}(\tilde{\alpha}_s) = \alpha_s$. By lemma 36 of signature preservation $a^{s+1} \in sig(\tilde{\mathcal{E}}||X)(\tilde{q}_s)$. Moreover, by assumption $a^{s+1} \notin$ creation-actions$(X)(\mathcal{A})$ which means the application of lemma 32 of homomorphic transitions leads us to $\eta_{((\tilde{\mathcal{E}}||X),\tilde{q}^s,a^{s+1})} \xleftrightarrow{\mu_z^{\mathcal{A}}} \eta_{((\mathcal{E}_{\mathcal{A}}||\tilde{\mathcal{A}}^{sw}),q^s,a^{s+1})}$. So there exists $\tilde{q}^{s+1} \in supp(\eta_{((\tilde{\mathcal{E}}||X),\tilde{q},a_!)})$ with $\mu_z^{\mathcal{A}}(\tilde{q}) = q$. So $\mu_e^{\mathcal{A}}(\tilde{\alpha}_s^\frown \tilde{q}^s a^{s+1} \tilde{q}^{s+1}) = \alpha_{s+1}$. This ends the induction and so the proof of 1. .
2. We apply 1. and note $\tilde{\alpha} \in Execs(\tilde{\mathcal{E}}||X)$ s.t. $\tilde{\mu}_e^{\mathcal{A}}(\tilde{\alpha}) = \alpha$. By lemma 36 of signature preservation $a_! \in sig(\tilde{\mathcal{E}}||X)(\tilde{q})$ with $\tilde{q} = lstate(\alpha)$. Moreover, by lemma 32 of homomorphic transition, $\eta_{(\tilde{\mathcal{E}}||X),\tilde{q},a_!} \xleftrightarrow{\mu_z^{\mathcal{A},+}} \eta_{(\mathcal{E}_{\mathcal{A}}||\tilde{\mathcal{A}}^{sw}),q,a_!}$. So there exists $\tilde{q}' \in supp(\eta_{(\tilde{\mathcal{E}}||X),\tilde{q},a_!})$ with $\mu_z^{\mathcal{A},+}(\tilde{q}') = q'$. So $\mu_e^{\mathcal{A},+}(\tilde{\alpha}^\frown \tilde{q} a_! \tilde{q}') = \alpha'$ which ends the proof.

□

Since we i) classify executions in some classes according to their projection on an environment and ii) are concerned by the actions of the execution that create $\mathcal{A}$, the next lemma will simplify this classification. It states that if the projection e of an execution $\alpha \in Execs(\mathcal{E}_{\mathcal{A}}||\tilde{\mathcal{A}}^{sw})$ on the environment $\mathcal{E}_{\mathcal{A}}$ ends by an action $a_! \in$ creation-actions$(X)(\mathcal{A})$, then the execution necessarily ends by $a_!$ (without additional suffix).

Then we define Γ-delineated function f that verifies the fact that an execution α perceived in Γ through f implies α does not create $\mathcal{A}$ before very last action.

Definition 156 (delineated function)**.** *Let $\mathcal{A}$ be a PSIOA, X a $\mathcal{A}$-conservative PCA, $\mathcal{E} \in env(X)$, $Y = X \setminus \{\mathcal{A}\}$, $\mathcal{E}_{\mathcal{A}} = \mathcal{E}||Y$. Let $f_{(.,.)}$ be an insight function. Let $\Gamma \subseteq range(f_{(\mathcal{E}_{\mathcal{A}},\tilde{\mathcal{A}}^{sw})})$. We say that f is $(\Gamma, \tilde{\mathcal{E}}, X, \mathcal{A})$-delineated if $\forall \zeta \in \Gamma$, $\forall \alpha \in Execs(\mathcal{E}_{\mathcal{A}}||\tilde{\mathcal{A}}^{sw})$, $f_{(\mathcal{E}_{\mathcal{A}},\tilde{\mathcal{A}}^{sw})}(\alpha) = \zeta$, implies $\alpha \in rangef(\tilde{\mathcal{E}}||X).\mu_e^{\mathcal{A},+}$, i.e $\forall \alpha' < \alpha$, actions$(\alpha') \cap$ creation-actions$(X)(\mathcal{A}) = \emptyset$.*

It is worthy to remark that if the projection e of an execution α does not contain actions dedicated to the creation of $\mathcal{A}$ before very last action, then α does not create $\mathcal{A}$ before the very last action.

Lemma 48 (projection is a delineated function with explicit creation)**.** *Let $\mathcal{A}$ be a PSIOA, X a $\mathcal{A}$-conservative PCA, $\mathcal{E} \in env(X)$, $Y = X \setminus \{\mathcal{A}\}$, $\mathcal{E}_{\mathcal{A}} = \mathcal{E}||Y$. Let $\Gamma \triangleq \{e \in Execs(\mathcal{E}_{\mathcal{A}}) | \forall e' < e,$ actions$(e') \cap$creation-actions$(X)(\mathcal{A}) = \emptyset\}$. The projection function $proj_{(.,.)}$ is $(\Gamma, \tilde{\mathcal{E}}, X, \mathcal{A})$-delineated.*

Proof. Let $\alpha \in Execs(\mathcal{E}_{\mathcal{A}}||\tilde{\mathcal{A}}^{sw})$, $(\alpha \upharpoonright \mathcal{E}_{\mathcal{A}}) = e' \in \Gamma$. Hence either $|e'| = 0$ or $e' = e^\frown q a_! q'$ with actions$(e') \cap$ creation-actions$(X)(\mathcal{A}) = \emptyset$. If actions$(\alpha) \cap$ creation-actions$(X)(\mathcal{A}) = \emptyset$, the result is immediate. Assume the opposite. We note $\alpha = \alpha^{1\frown} q_\ell^1, a_!, q_f^{2\frown} \alpha^2$ with $a_! \in$ creation-actions$(X)(\mathcal{A})$.

We have $q^1_\ell \upharpoonright \tilde{\mathcal{A}}^{sw} = q^\phi_{\tilde{\mathcal{A}}^{sw}}$. Indeed, let us assume the contrary: $q^1_\ell \upharpoonright \tilde{\mathcal{A}}^{sw} \neq q^\phi_{\tilde{\mathcal{A}}^{sw}}$. Then $q \upharpoonright \tilde{\mathcal{A}}^{sw} \neq q^\phi_{\tilde{\mathcal{A}}^{sw}}$ for every state $q \in \alpha^1$. Since $\mathit{creation\text{-}actions}(X)(\mathcal{A}) \cap \mathit{actions}(e') = \emptyset$, $\mathit{creation\text{-}actions}(X)(\mathcal{A}) \cap \mathit{actions}(\alpha^1) = \emptyset$. Thus we apply lemma 47 of partial surjectivity with explicit creation to obtain, there exists $\tilde{\alpha}^1 \in \mathit{Execs}(\tilde{\mathcal{E}}||X)$ s.t. $\tilde{\mu}_e^{\mathcal{A},+}(\tilde{\alpha}^1) = \alpha^1$ with both $\mathcal{A} \in \mathit{auts}(\mathit{config}(X)(\mathit{lstate}(\tilde{\alpha}^1) \upharpoonright X))$ and $a_! \in \mathit{creation\text{-}actions}(X)(\mathcal{A}) \cap \mathit{sig}(X)(\mathit{lstate}(\tilde{\alpha}^1)) \upharpoonright X)$ which is impossible.

Since $q^1_\ell \upharpoonright \tilde{\mathcal{A}}^{sw} = q^\phi_{\tilde{\mathcal{A}}^{sw}}$, $q \upharpoonright \tilde{\mathcal{A}}^{sw} = q^\phi_{\tilde{\mathcal{A}}^{sw}}$ for every state $q \in \alpha^2$. Hence, $\alpha^2 = q^2_f$ to respect $\alpha \upharpoonright \mathcal{E}_\mathcal{A} = e'$, which means $\alpha = \alpha^1 \frown q^1_\ell, a_!, q^2_f$. Since $\mathit{creation\text{-}actions}(X)(\mathcal{A}) \cap \mathit{actions}(e) = \emptyset$, $\mathit{creation\text{-}actions}(X)(\mathcal{A}) \cap \mathit{actions}(\alpha^1) = \emptyset$, which ends the proof.

□

Now, we can clarify when $\tilde{\mu}_e^{\mathcal{A},+}$ is a bijection between "top/down" corresponding classes of equivalence.

Lemma 49. *($\tilde{\mu}_e^{\mathcal{A},+}$ is a bijection from $\tilde{\mathcal{C}}$ to $\mathcal{C}$). Let $\mathcal{A}$ be a PSIOA and let X be a $\mathcal{A}$-conservative and $\mathcal{A}$-creation-explicit PCA. Let $\tilde{\mathcal{E}} \in env(X)$. Let $(((\tilde{\mathcal{E}}||X).\tilde{\mu}_z^{\mathcal{A}}, (\tilde{\mathcal{E}}||X).\tilde{\mu}_z^{\mathcal{A},+}), (\tilde{\mathcal{E}}||X).\tilde{\mu}_{tr}^{\mathcal{A},+}, (\tilde{\mathcal{E}}||X).\tilde{\mu}_e^{\mathcal{A},+})$ be the $\tilde{\mathcal{E}}$-extension of $((X.\tilde{\mu}_z^{\mathcal{A}}, X.\tilde{\mu}_z^{\mathcal{A},+}), X.\tilde{\mu}_{tr}^{\mathcal{A},+}, X.\tilde{\mu}_e^{\mathcal{A},+})$. Let $Y = X \setminus \{\mathcal{A}\}$. Let $\mathcal{E}_\mathcal{A} = \tilde{\mathcal{E}}||Y$.*

Let f be an ordinary perception function, $(\Gamma, \tilde{\mathcal{E}}, X, \mathcal{A})$-delineated.

For every $\zeta \in \Gamma$, $(\tilde{\mathcal{E}}||X).\tilde{\mu}_e^{\mathcal{A},+}$ is a bijection from $\tilde{\mathcal{C}}$ to $\mathcal{C}$, where

- $\tilde{\mathcal{C}} = \mathit{Class}(\tilde{\mathcal{E}}, X, f^{\mathcal{A},proxy}, \zeta)$
- $\mathcal{C} = \mathit{Class}(\mathcal{E}_\mathcal{A}, \tilde{\mathcal{A}}^{sw}, f, \zeta)$

Proof.
- Injectivity is immediate by lemma 14, item (2).
- Surjectivity: Let $\alpha \in \mathcal{C}$. By definition, $f_{(\mathcal{E}_\mathcal{A}, \tilde{\mathcal{A}}^{sw})}(\alpha) = \zeta \in \Gamma$. Since f is $(\Gamma, \tilde{\mathcal{E}}, X, \mathcal{A})$-delineated, then $\forall \alpha' < \alpha$, $(\mathit{actions}(\alpha') \cap \text{creation-actions}(X)(\mathcal{A}) = \emptyset$. Hence, we can apply lemma 47 of partial surjectivity with explicit creation

□

Hence, we obtain an equiprobability of top/down corresponding cones equipped with alter-ego schedulers.

Lemma 50 (equiprobability of top/down corresponding cones)**.** *Let $\mathcal{A}$ be a PSIOA and X be a $\mathcal{A}$-conservative and $\mathcal{A}$-creation-explicit PCA. Let $\tilde{\mathcal{E}} \in env(X)$. Let $Y = X \setminus \{\mathcal{A}\}$. Let $\mathcal{E}_\mathcal{A} = \tilde{\mathcal{E}}||Y$. Let $(((\tilde{\mathcal{E}}||X).\tilde{\mu}_z^{\mathcal{A}}, (\tilde{\mathcal{E}}||X).\tilde{\mu}_z^{\mathcal{A},+}), (\tilde{\mathcal{E}}||X).\tilde{\mu}_{tr}^{\mathcal{A},+}, (\tilde{\mathcal{E}}||X).\tilde{\mu}_e^{\mathcal{A},+})$ the $\tilde{\mathcal{E}}$-extension of $((X.\tilde{\mu}_z^{\mathcal{A}}, X.\tilde{\mu}_z^{\mathcal{A},+}), X.\tilde{\mu}_{tr}^{\mathcal{A},+}, X.\tilde{\mu}_e^{\mathcal{A},+})$.*

Let f be an ordinary perception function, $(\Gamma, \tilde{\mathcal{E}}, X, \mathcal{A})$-delineated. Let $\zeta \in \Gamma$, and

- $\tilde{\mathcal{C}} = \mathit{Class}(\tilde{\mathcal{E}}, X, f^{\mathcal{A},proxy}, \zeta)$
- $\mathcal{C} = \mathit{Class}(\mathcal{E}_\mathcal{A}, \tilde{\mathcal{A}}^{sw}, f, \zeta)$

Then for every $\tilde{\sigma} \in \mathit{schedulers}(\tilde{\mathcal{E}}||X)$, for σ $(((\tilde{\mathcal{E}}||X).\tilde{\mu}_z^{\mathcal{A}}, (\tilde{\mathcal{E}}||X).\tilde{\mu}_z^{\mathcal{A},+}), (\tilde{\mathcal{E}}||X).\tilde{\mu}_{tr}^{\mathcal{A},+}, (\tilde{\mathcal{E}}||X).\tilde{\mu}_e^{\mathcal{A},+})$-alter ego of $\tilde{\sigma}$,

$$\epsilon_{\tilde{\sigma}, \delta_{\bar{q}_{(\tilde{\mathcal{E}}||X)}}}(C_{\tilde{\mathcal{C}}}) = \epsilon_{\sigma, \delta_{\bar{q}_{(\mathcal{E}_\mathcal{A}||\tilde{\mathcal{A}}^{sw})}}}(C_\mathcal{C})$$

Proof. By lemma 49, $\tilde{\mu}_e^{\mathcal{A},+}$ is a bijection from $\tilde{\mathcal{C}}$ to $\mathcal{C}$. We note $\{(\tilde{\alpha}_i, \alpha_i)\}_{i \in I} = \tilde{\mathcal{C}} \times \mathcal{C}$ the related pairs of executions s.t. $\tilde{\mu}_e^{\mathcal{A},+}(\tilde{\alpha}_i) = \alpha_i$. We obtain $\epsilon_{\tilde{\sigma}, \delta_{\bar{q}_{(\tilde{\mathcal{E}}||X)}}}(C_{\tilde{\mathcal{C}}}) = \sum_{i \in I} \epsilon_{\tilde{\sigma}, \delta_{\bar{q}_{(\tilde{\mathcal{E}}||X)}}}(C_{\tilde{\alpha}_i})$ and $\epsilon_{\sigma, \delta_{\bar{q}_{(\mathcal{E}_\mathcal{A}||\tilde{\mathcal{A}}^{sw})}}}(C_\mathcal{C}) = \sum_{i \in I} \epsilon_{\sigma, \delta_{\bar{q}_{(\mathcal{E}_\mathcal{A}||\tilde{\mathcal{A}}^{sw})}}}(C_{\alpha_i})$.

Thus it is enough to show that $\forall i \in I, \epsilon_{\tilde{\sigma},\delta_{\tilde{q}_{(\tilde{\mathcal{E}}||X)}}}(C_{\tilde{\alpha}_i}) = \epsilon_{\sigma,\delta_{\bar{q}_{(\mathcal{E}_{\mathcal{A}}||\bar{\mathcal{A}}^{sw})}}}(C_{\alpha_i})$ which is given by theorem 9 that can be applied since $\tilde{\mu}_e^{\mathcal{A},+}$ is a continued executions-matching by theorem 20.

□

4.5.4. Decomposition, pasting-friendly functions

In the last subsection, the dynamic creation/destruction of $\mathcal{A}$ has been discarded. It is time to generalize the previous approach with dynamic creation/destruction of $\mathcal{A}$.

We first define some tools to describe the decomposition of an execution into segments whose last action is in the dedicated set to create $\mathcal{A}$.

Definition 157. *(n-building-vector for executions). Let α be an alternating sequence of states and actions starting by state and finishing by a state if α is finite. Let $n \in \mathbb{N} \cup \{\infty\}$. A* n-building-vector *of α is a (potentially infinite) vector $\vec{\alpha} = (\alpha^1, ..., \alpha^i, ...)$ of $|\vec{\alpha}| = n$ alternating sequences of states and actions starting by state and finishing by a state (excepting potentially the last one if it is infinite) s.t. $\alpha^1 \frown ... \alpha^{i-1} \frown \alpha^i \frown ... = \alpha$ (with $\forall i \in [1, |\vec{\alpha}| - 1], fstate(\alpha_{i+1}) = lstate(\alpha_i)$). We note Building-vectors(α, n) the set of n-building-vector of α and $\vec{\alpha} \overset{n}{:} \alpha$ to say $\vec{\alpha} \in$ Building-vectors(α, n). We note Building-vectors$(\alpha) = \bigcup_{n \in \mathbb{N} \cup \{\infty\}}$ Building-vectors(α, n) and $\vec{\alpha} : \alpha$ to say $\alpha \in$ Building-vectors(α). We note $\vec{\alpha}[i] = \alpha^i$ and $\vec{\alpha}[: i] = \alpha^1 \frown ... \frown \alpha^{i-1}$. If W is an automaton, $\alpha \in Execs(W)$, $\vec{\alpha} : \alpha$ and f a function with $dom(f) \subseteq Frags(W)$, we note $f(\vec{\alpha}) = [f(\vec{\alpha}[1]), ..., f(\vec{\alpha}[i]), ...]$.*

Definition 158. *($\vec{\alpha} \underset{(X,\mathcal{A})}{:} \alpha$) Let W and X be two PCA s.t. X is $\mathcal{A}$-creation-explicit, $\alpha \in Frags(W)$. We note $\vec{\alpha} \underset{(X,\mathcal{A})}{:} \alpha$ (and $\vec{\alpha} \underset{\mathcal{A}}{:} \alpha$ when X is clear in the context) the (clearly unique) vector $\vec{\alpha} \in$ Building-vectors(α) of execution fragments s.t.*

1. *$\forall i \in [1, n]$, $\forall \alpha' < \vec{\alpha}[i]$, actions$(\alpha') \cap$ creation-actions$(X)(\mathcal{A}) = \emptyset$ and*
2. *$\forall i \in [1, n-1]$, laction$(\vec{\alpha}[i])) \in$ creation-actions$(X)(\mathcal{A})$.*

We write $\vec{\alpha} \overset{n}{\underset{(X,\mathcal{A})}{:}}$ or $\vec{\alpha} \overset{n}{\underset{\mathcal{A}}{:}}$ to indicate that $|\vec{\alpha}| = n$.

Definition 159. *($\mathcal{A}$-decomposition) Let $\mathcal{A}$ be a PSIOA and X be a PCA. Let $\alpha = q^0 a^1 ... a^n q^n ... \in Frags(X)$. We say that*

- *α is a* $\mathcal{A}$-open-portion *iff α does not create $\mathcal{A}$, i.e. $\forall i \in [1, |\alpha|] \mathcal{A} \notin auts(config(X)(q^{i-1})) \Longrightarrow \mathcal{A} \notin auts(config(X)(q^i))$.*
- *α is a* $\mathcal{A}$-closed-portion *iff α does not create $\mathcal{A}$ excepting at very last last action, i.e. $\forall i \in [1, |\alpha|] \mathcal{A} \notin auts(config(X)(q^{i-1})) \wedge \mathcal{A} \in auts(config(X)(q^i)) \Longleftrightarrow i = |\alpha|$.*
- *α is a* $\mathcal{A}$-portion *of X if it is either a $\mathcal{A}$-open-portion or a $\mathcal{A}$-closed-portion.*

We call $\mathcal{A}$-decomposition of α, noted $\mathcal{A}$-decomposition(α), the unique vector $(\alpha^1, ..., \alpha^n, ...) \in$ Building-vectors(α) s.t.

- *$\forall i \in [1, |\mathcal{A}\text{-decomposition}(\alpha)| - 1]$, α^i is a $\mathcal{A}$-closed-portion of X and*
- *if $|\mathcal{A}\text{-decomposition}(\alpha)| = n \in \mathbb{N}$, α^n is a $\mathcal{A}$-portion of X.*

Lemma 51. *($\vec{\alpha} \underset{(X,\mathcal{A})}{:} \alpha$ means $\vec{\alpha} = \mathcal{A}$-decomposition$(\alpha)$). Let $\mathcal{A}$ be a PSIOA and X be a $\mathcal{A}$-creation-explicit PCA. Let $\alpha \in Frags(X)$. Let $\vec{\alpha} = \mathcal{A}$-decomposition$(\alpha)$. Then $\vec{\alpha} \overset{n}{\underset{(X,\mathcal{A})}{:}} \alpha$.*

Proof. By definition, $\vec{\alpha} \in \mathit{Building\text{-}vectors}(\alpha)$. Still by definition, $\forall i \in [1, |\mathcal{A}\text{-}\mathit{decomposition}(\alpha)| - 1]$, α^i is a $\mathcal{A}$-closed-portion of X, i.e. α^i does not create $\mathcal{A}$ excepting at very last last action $\mathit{laction}(\alpha_i)$. By definition of creation-explicitness, the two item of definition 158 are verified for every $i \in [1, |\mathcal{A}\text{-}\mathit{decomposition}(\alpha)| - 1]$. Finally, by definition, if $|\mathcal{A}\text{-}\mathit{decomposition}(\alpha)| = n \in \mathbb{N}$, α^n is a $\mathcal{A}$-portion of X, i.e. α^n does not create $\mathcal{A}$ excepting potentially at the very last action if α^n is finite. Again, by definition of creation-explicitness, the first item of definition 158 is verified.

□

Now, we introduce the crucial property, called *pasting-friendly*, required for a perception function f to ensure monotonicity of $\leq_0^{S_o,f}$. This property allows to cut-paste a general class of equivalence into a composition of smaller classes of equivalence, without the creation of $\mathcal{A}$ before the very last action, where lemma 50 of equiprobability between top-down corresponding cones can be applied to each smaller class.

Definition 160 (pasting friendly)**.** *Let $f_{(.,.)}$ be an insight function. We say that $f_{(.,.)}$ is* pasting-friendly *if for every PSIOA $\mathcal{A}$, for every $\mathcal{A}$-conservative and $\mathcal{A}$-creation-explicit PCA X, $\forall \tilde{\mathcal{E}} \in \mathit{env}(X)$, $\forall \tilde{\zeta} \in \bigcup_{K, \tilde{\mathcal{E}} \in \mathit{env}(K)} \mathit{range}(f_{(\tilde{\mathcal{E}},K)})$, $\forall \vec{\zeta} \in \vec{\mathit{proxy}}(\tilde{\zeta})_{\tilde{\mathcal{E}},X,\mathcal{A}}$ then*

1. *$\forall \tilde{\alpha}, \tilde{\alpha}'$, $\vec{\alpha} = \mathcal{A}\text{-}\mathit{decomposition}(\tilde{\alpha})$, $\vec{\alpha}' = \mathcal{A}\text{-}\mathit{decomposition}(\tilde{\alpha}')$, $f^{\mathcal{A},\mathit{proxy}}_{(\tilde{\mathcal{E}},X)}(\vec{\alpha}) = f^{\mathcal{A},\mathit{proxy}}_{(\tilde{\mathcal{E}},X)}(\vec{\alpha}') \triangleq \vec{\zeta}$ implies $|\vec{\alpha}| = |\vec{\alpha}| = |\vec{\zeta}| \triangleq n \in \mathbb{N} \cup \{\infty\} \wedge\ \forall i \in [1, n-1]$, $\mathit{lstate}(\vec{\alpha}[i]) = \mathit{lstate}(\vec{\alpha}'[i]) \triangleq q_i^\ell$.*
2. *We note $\tilde{\mathcal{E}}^1 = \tilde{\mathcal{E}}$, $X^1 = X$, and $\forall i \in [2, n]$, we note $\tilde{\mathcal{E}}^i = \tilde{\mathcal{E}}_{\bar{q}_{\tilde{\mathcal{E}}} \to (q_{i-1}^\ell \restriction \tilde{\mathcal{E}})}$ (resp $X^i = X_{\bar{q}_X \to (q_{i-1}^\ell \restriction X)}$).*
   $\forall j \in [1, n], \forall \alpha^j \in \mathit{Execs}((\tilde{\mathcal{E}}^j || X^j))$, $f^{\mathcal{A},\mathit{proxy}}_{(\tilde{\mathcal{E}}^j, X^j)}(\alpha^j) = \vec{\zeta}[j]$, then
   a) for every $\alpha'_j < \alpha_j$, $\mathit{actions}(\alpha'_j) \cap \mathit{creation\text{-}actions}(X)(\mathcal{A}) = \emptyset$ and
   b) if $j \in [1, n-1]$, $\alpha_j = \alpha'^\frown_j a^j_! q^j_\ell$ with $a^j_! \in \mathit{creation}(X)(\mathcal{A})$

We state an intermediate lemma to show that projection on the environment is pasting-friendly (see lemma 53).

Lemma 52 (chunks ending on creation)**.** *Let $\mathcal{A}$ be a PSIOA, let X be a $\mathcal{A}$-conservative and $\mathcal{A}$-creation-explicit PCA and $\tilde{\mathcal{E}}$ partially-compatible with X. Let $\tilde{\alpha} \in \mathit{Frags}(\tilde{\mathcal{E}}||X)$ and $e \in \mathit{Frags}(\tilde{\mathcal{E}}||X \setminus \{\mathcal{A}\})$ s.t. $(\tilde{\mathcal{E}}||X).\mu_e^{\mathcal{A},+}(\tilde{\alpha}) \restriction (\tilde{\mathcal{E}}||X \setminus \{\mathcal{A}\}) = e$.*

Then

- *$\mathit{laction}(\tilde{\alpha}) = a_! \in \mathit{creation\text{-}actions}(X)(\mathcal{A}) \implies \mathit{laction}(e) = a_! \in \mathit{creation\text{-}actions}(X)(\mathcal{A})$.*
- *if $\tilde{\alpha} \in \mathit{dom}(\tilde{\mu}_e^{\mathcal{A},+})$,*
  $\mathit{laction}(\tilde{\alpha}) = a_! \in \mathit{creation\text{-}actions}(X)(\mathcal{A}) \impliedby \mathit{laction}(e) = a_! \in \mathit{creation\text{-}actions}(X)(\mathcal{A})$.

Proof. We prove the two implications in the same order.

- $\implies$) Let assume $a_! \triangleq \mathit{laction}(\tilde{\alpha}) \in \mathit{creation\text{-}actions}(X)(\mathcal{A})$. Since X is $\mathcal{A}$-creation-explicit, we have $\tilde{\alpha} = \tilde{\alpha}'^\frown q' a_! q$ with $\mathcal{A} \notin \mathit{auts}(\mathit{config}(X)(q'))$. Thus $\mathit{laction}(e) = a_! \in \mathit{creation\text{-}actions}(X)(\mathcal{A})$.
- $\impliedby$) Let assume $a_! \triangleq \mathit{laction}(e) \in \mathit{creation\text{-}actions}(X)(\mathcal{A})$. Thus $a_! \in \mathit{actions}(\tilde{\alpha})$. Since X is $\mathcal{A}$-creation-explicit, it implies $\tilde{\alpha} = \tilde{\alpha}^1 {}^\frown q^1_\ell, a_!, q^2_f {}^\frown \tilde{\alpha}^2$ where $\mathcal{A} \notin \mathit{auts}(\mathit{config}(X)(q^1_\ell))$ and $\mathcal{A} \in \mathit{auts}(\mathit{config}(X)(q^2_f))$. But $\tilde{\alpha} \in \mathit{dom}((\tilde{\mathcal{E}}||X).\tilde{\mu}_e^{\mathcal{A},+})$, so $\tilde{\alpha}^2 = q^2_f$ and hence $\mathit{laction}(\tilde{\alpha}) = a_! \in \mathit{creation\text{-}actions}(X)(\mathcal{A})$

□

Now, we are ready to show that projection on the environment is pasting-friendly.

Lemma 53. *The projection function proj(.,.) (for each automaton K, $\forall \mathcal{E} \in env(K)$, $proj_{(\mathcal{E},K)} : \alpha \in Execs(\mathcal{E}||K) \mapsto \alpha \upharpoonright \mathcal{E}$ is pasting friendly.*

Proof. 1. Let $\mathcal{A}$ be a PSIOA, let X be a $\mathcal{A}$-conservative PCA, let $\tilde{\mathcal{E}} \in env(X)$, let $\mathcal{E}_{\mathcal{A}} = (\tilde{\mathcal{E}}||(X \setminus \{\mathcal{A}\}))$. We note $q_{\ell,i} = lstate(\overrightarrow{\alpha}[i])$ and $q'_{\ell,i} = lstate(\overrightarrow{\alpha}'[i])$, $C_{\ell,i} = (\mathbf{A}_{\ell,i}, \mathbf{S}_{\ell,i}) = config(\tilde{\mathcal{E}}||X)(q_{\ell,i})$ and $C'_{\ell,i} = (\mathbf{A}'_{\ell,i}, \mathbf{S}'_{\ell,i}) = config(\tilde{\mathcal{E}}||X)(q'_{\ell,i})$. Let $i \in [1, |\alpha|-1]$. By construction of $\mathcal{A}$-decomposition, $\mathbf{S}_{\ell,i}(\mathcal{A}) = \mathbf{S}'_{\ell,i}(\mathcal{A}) = \bar{q}_{\mathcal{A}}$ (1). Moreover, $f^{\mathcal{A},proxy}_{(\tilde{\mathcal{E}},X)}(\overrightarrow{\alpha}) = f^{\mathcal{A},proxy}_{(\tilde{\mathcal{E}},X)}(\overrightarrow{\alpha}') \triangleq \overrightarrow{\zeta}$, i.e. $proj_{(\mathcal{E}_{\mathcal{A}},\tilde{\mathcal{A}}^{sw})}(\overrightarrow{\alpha}[i]) = proj_{(\mathcal{E}_{\mathcal{A}},\tilde{\mathcal{A}}^{sw})}(\overrightarrow{\alpha}'[i])$, which means $q_{\ell,i} \upharpoonright \mathcal{E}_{\mathcal{A}} = q'_{\ell,i} \upharpoonright \mathcal{E}_{\mathcal{A}}$. Hence, $\mathbf{A}_{\ell,i} \setminus \{\mathcal{A}\} = \mathbf{A}'_{\ell,i} \setminus \{\mathcal{A}\} \triangleq \mathbf{A}''_{\ell,i}$ and $\forall \mathcal{B} \in \mathbf{A}''_{\ell,i}$, $\mathbf{S}_{\ell,i}(\mathcal{B}) = \mathbf{S}'_{\ell,i}(\mathcal{B})$ (2). By (1) and (2), $C_{\ell,i} = C'_{\ell,i}$. Since X is configuration-conflict-free, $q_{\ell,i} = q'_{\ell,i}$.

2. Let $j \in [1,n]$, let $\alpha^j \in Execs((\tilde{\mathcal{E}}^j||X^j)), f^{\mathcal{A},proxy}_{(\tilde{\mathcal{E}}^j,X^j)}(\alpha^j) = \overrightarrow{\zeta}[j]$ Let $\tilde{\alpha} \in Execs(\tilde{\mathcal{E}}||X)$, $\overrightarrow{\alpha} = \mathcal{A}$-decomposition$(\tilde{\alpha})$, $\overrightarrow{\alpha} \in (proj^{\mathcal{A},proxy}_{(\tilde{\mathcal{E}},X)})^{-1}(\overrightarrow{\zeta})$.

   a) Let us assume $j \in [1, n-1]$. By construction of $\mathcal{A}$-decomposition, We have $\overrightarrow{\alpha}[j] = \alpha^{*\frown}_j(a^j_! q^j_\ell)$ with $actions(\alpha^*_j) \cap$ creation-actions$(X)(\mathcal{A}) = \emptyset$ and $a^j_! \in$ creation-actions$(X)(\mathcal{A})$. By lemma 52, it implies, $\overrightarrow{\zeta}[j] = e^{*\frown}_j(a^j_! q^j_\ell \upharpoonright \tilde{\mathcal{E}})$ with $actions(e^*_j) \cap$ creation-actions$(X)(\mathcal{A}) = \emptyset$ and $a^j_! \in$ creation-actions$(X)(\mathcal{A})$. By lemma 52, it implies $\alpha_j = \alpha'^{\frown}_j(a^j_!(q^j_\ell))$ with $actions(\alpha'_j) \cap$ creation-actions$(X)(\mathcal{A}) = \emptyset$ and $a^j_! \in$ creation-actions$(X)(\mathcal{A})$ (*). Moreover, let us assume $n \in \mathbb{N}$. For every $\alpha^*_n < \overrightarrow{\alpha}[n]$, $actions(\alpha^*_n) \cap$ creation-actions$(X)(\mathcal{A}) = \emptyset$, hence, for every $e^*_n < \overrightarrow{\zeta}[n]$, $actions(e^*_n) \cap$ creation-actions$(X)(\mathcal{A}) = \emptyset$ and so for every $\alpha^*_n < \alpha_n$, $actions(\alpha^*_n) \cap$ creation-actions$(X)(\mathcal{A}) = \emptyset$.

   b) Let $j \in [1, n-1]$. By previous item, $\alpha_j = \alpha'^{\frown}_j(a^j_! q^j_\ell)$ with $actions(\alpha'_j) \cap$creation-actions$(X)(\mathcal{A})$ $= \emptyset$ and $a^j_! \in$ creation-actions$(X)(\mathcal{A})$ (*). Moreover, by construction, we have $proj^{\mathcal{A},proxy}_{(\tilde{\mathcal{E}},X)})(\alpha_j) = proj^{\mathcal{A},proxy}_{(\tilde{\mathcal{E}},X)})(\overrightarrow{\alpha}[j])$ (**). We can apply the exact same reasoning as in item 1.

□

Before stating our first lemma 54 of decomposition, we define the set of vector proxies. This set contains all the explanations $\overrightarrow{\zeta}^k$, from reduction, of a perception $\tilde{\zeta}$.

Definition 161. *($\overrightarrow{proxy}(\tilde{\zeta})$) Let $f_{(.,.)}$ be an insight function. Let $\mathcal{A}$ be a PSIOA, let X be a $\mathcal{A}$-conservative PCA, let $\tilde{\mathcal{E}} \in env(X)$, Let $\tilde{\zeta} \in \bigcup_{K,\tilde{\mathcal{E}} \in env(K)} range(f_{(\tilde{\mathcal{E}},K)})$. We note*

$$\overrightarrow{proxy}(\tilde{\zeta})_{(\tilde{\mathcal{E}},X,\mathcal{A},f)} = \{\overrightarrow{\zeta}^k \mid \exists \tilde{\alpha} \in f^{-1}_{(\tilde{\mathcal{E}},X)}(\tilde{\zeta}) \wedge f^{\mathcal{A},proxy}_{\tilde{\mathcal{E}},X}(\mathcal{A}\text{-}decomposition(\tilde{\alpha})) = \overrightarrow{\zeta}^k\}.$$

When f is clear in the context, we omit f in the subscript.

Now, we can partition executions with a common perception $\tilde{\zeta}$ into a sub-set of classes with more details related to the reduction.

Lemma 54. *Let f be an ordinary perception function pasting friendly. Let $\mathcal{A}$ be a PSIOA, let X be a $\mathcal{A}$-conservative PCA, let $\tilde{\mathcal{E}} \in env(X)$, Let $\tilde{\zeta} \in \bigcup_{K,\tilde{\mathcal{E}} \in env(K)} range(f_{(\tilde{\mathcal{E}},K)})$. Let $\mathcal{C}^{\tilde{\zeta}} = Class(\tilde{\mathcal{E}}, X, f, \tilde{\zeta})$.*

$$\mathcal{C}^{\tilde{\zeta}} = \biguplus_{\vec{\zeta}^k \in \vec{proxy}(\tilde{\zeta})_{(\tilde{\mathcal{E}},X,\mathcal{A})}} \mathcal{C}^{\vec{\zeta}^k} \text{ with}$$

$$\mathcal{C}^{\vec{\zeta}^k} = Class(\tilde{\mathcal{E}}, X, f^{\mathcal{A},proxy} \circ \mathcal{A}\text{-decomposition}, \vec{\zeta}^k)$$

Proof. The proof is immediate by construction, since $\mathcal{A}$-decomposition is unique.

- (equality) We first show equality by double inclusion.
  - ($\subseteq$) Let $\tilde{\alpha} \in \mathcal{C}^{\tilde{\zeta}}$. We note $\vec{\alpha} = \mathcal{A}$-decomposition$(\tilde{\alpha})$. By construction, we have $\vec{\alpha} \underset{\mathcal{A}}{:} \tilde{\alpha}$. We note $\vec{\zeta} = f^{\mathcal{A},proxy}_{(\tilde{\mathcal{E}},X)}(\vec{\alpha})$. Obviously, $\vec{\zeta} \in \vec{proxy}(\tilde{\zeta})_{(\tilde{\mathcal{E}},X,\mathcal{A})}$.
  - ($\supseteq$) Let $\vec{\zeta}^k \in \vec{proxy}(\tilde{\zeta})_{(\tilde{\mathcal{E}},X,\mathcal{A})}$, with $n \triangleq |\vec{\zeta}^k|$, let $\tilde{\alpha} \in \mathcal{C}^{\vec{\zeta}^k}$. We want to show that $\tilde{\alpha} \in \mathcal{C}^{\tilde{\zeta}}$. Let $\vec{\alpha} = \mathcal{A}$-decomposition$(\tilde{\alpha})$. By definition of $\vec{proxy}(\tilde{\zeta})_{(\tilde{\mathcal{E}},X,\mathcal{A})}$, $\exists \tilde{\alpha}' \in f^{-1}_{(\tilde{\mathcal{E}},X)}(\tilde{\zeta})$ such that $f^{\mathcal{A},proxy}_{\tilde{\mathcal{E}},X}(\mathcal{A}\text{-decomposition}(\tilde{\alpha}')) = \vec{\zeta}^k$. Let fix such a $\tilde{\alpha}'$. Let $\vec{\alpha}' = \mathcal{A}$-decomposition$(\tilde{\alpha}')$. By construction $f^{\mathcal{A},proxy}_{\tilde{\mathcal{E}},X}(\vec{\alpha}) = f^{\mathcal{A},proxy}_{\tilde{\mathcal{E}},X}(\vec{\alpha}')$. Moreover, f is assumed to be pasting friendly, which implies $\forall i \in [1,n]$, $f^{\mathcal{A},proxy}_{\tilde{\mathcal{E}}^i,X^i}(\vec{\alpha}[i]) = f^{\mathcal{A},proxy}_{\tilde{\mathcal{E}}^i,X^i}(\vec{\alpha}'[i])$ where $\tilde{\mathcal{E}}^i$ and X^i are defined as in definition 160 of pasting friendly functions. Since f is an ordinary perception function, we can apply lemma 46, which implies that $\forall i \in [1,n]$, $f_{\tilde{\mathcal{E}},X}(\vec{\alpha}[i]) = f_{\tilde{\mathcal{E}},X}(\vec{\alpha}'[i]))$ and so $f_{\tilde{\mathcal{E}},X}(\tilde{\alpha}) = f_{\tilde{\mathcal{E}},X}(\tilde{\alpha}') = \tilde{\zeta}$, that is $\tilde{\alpha} \in \mathcal{C}^{\tilde{\zeta}}$.
- (partitioning) We show that $\forall (\vec{\zeta}^k, \vec{\zeta}^\ell), \vec{\zeta}^k \neq \vec{\zeta}^\ell, \mathcal{C}^{\vec{\zeta}^k} \cap \mathcal{C}^{\vec{\zeta}^\ell} = \emptyset$. Let $(\tilde{\alpha}, \tilde{\alpha}') \in \mathcal{C}^{\vec{\zeta}^k} \times \mathcal{C}^{\vec{\zeta}^\ell}$. Let $\vec{\alpha} \underset{\mathcal{A}}{:} \alpha$ and $\vec{\alpha}' \underset{\mathcal{A}}{:} \alpha'$. We have $f^{\mathcal{A},proxy}_{(\tilde{\mathcal{E}},X)}(\vec{\alpha}) = \vec{\zeta}^k \neq \vec{\zeta}^\ell = f^{\mathcal{A},proxy}_{(\tilde{\mathcal{E}},X)}(\vec{\alpha}')$. Thus $\vec{\alpha} \neq \vec{\alpha}'$. By lemma 51, $\vec{\alpha} = \mathcal{A}$-decomposition$(\tilde{\alpha})$ and $\vec{\alpha}' = \mathcal{A}$-decomposition$(\tilde{\alpha}')$, and so $\tilde{\alpha} \neq \tilde{\alpha}'$. Hence, $\forall (\vec{\zeta}^k, \vec{\zeta}^\ell), \vec{\zeta}^k \neq \vec{\zeta}^\ell, \mathcal{C}^{\vec{\zeta}^k} \cap \mathcal{C}^{\vec{\zeta}^\ell} = \emptyset$.

□

Then, we perform our decomposition of $\hat{\mathcal{C}}^{\vec{\zeta}} = Class(\tilde{\mathcal{E}}, X, f^{\mathcal{A},proxy} \circ \mathcal{A}\text{-decomposition}, \vec{\zeta}^k)$ into small chunks.

Lemma 55 (decomposition into simple classes)**.** *Let $f_{(.,.)}$ be pasting friendly perception function. Let $\mathcal{A}$ be a PSIOA, X be a $\mathcal{A}$-conservative and $\mathcal{A}$-creation-explicit PCA and $\tilde{\mathcal{E}}$ partially-compatible with X. Let $\mathcal{E}_{\mathcal{A}} = \tilde{\mathcal{E}}||(X \setminus \{\mathcal{A}\})$. Let $\tilde{\zeta} \in \bigcup_{K,\tilde{\mathcal{E}} \in env(K)} range(f_{(\tilde{\mathcal{E}},K)})$. Let $n \in \mathbb{N} \cup \{\infty\}$, let $\vec{\zeta} \in \vec{proxy}(\tilde{\zeta})_{(\tilde{\mathcal{E}},X,\mathcal{A})}$ with $|\vec{\zeta}| = n$. Let $\hat{\mathcal{C}}^{\vec{\zeta}} = Class(\tilde{\mathcal{E}}, X, f^{\mathcal{A},proxy} \circ \mathcal{A}\text{-decomposition}, \vec{\zeta})$.*

Then, $\hat{\mathcal{C}}^{\vec{\zeta}} = \bigotimes_{i=1}^{n} \hat{\mathcal{C}}^{\vec{\zeta}[i]}$ with

1. $\hat{\mathcal{C}}^{\vec{\zeta}[i]} = Class(\tilde{\mathcal{E}}^i, X^i, f^{\mathcal{A},proxy}, \vec{\zeta}[i])$

2. $\forall \alpha^i \in \hat{\mathcal{C}}^{\vec{\zeta}[i]}$ *if* $i \in [1, n-1]$, $\alpha_i = \alpha_i'^\frown a_!^i q_\ell^i$ *with* $a_!^i \in creation(X)(\mathcal{A})$ *and if* $n \in \mathbb{N}$ $\forall \alpha_n' < \alpha_n, actions(\alpha_n') \cap creation\text{-}actions(X)(\mathcal{A}) = \emptyset$ *(ensured by pasting friendship of* f*).*
3. $\forall i \in [1, n-1]$*, we note* q_ℓ^{i-1} *the unique last state of every execution of* $\hat{\mathcal{C}}^{\vec{\zeta}[i]}$ *(ensured by pasting friendship of* f*).*
4. $\tilde{\mathcal{E}}^1 = \tilde{\mathcal{E}}$ *and* $\forall i \in [2, n]$, $\tilde{\mathcal{E}}^i = \tilde{\mathcal{E}}_{\bar{q}_\mathcal{E} \to q_\mathcal{E}^i}$ *, (as per definition 136), with* $q_\mathcal{E}^i = q_\ell^{i-1} \upharpoonright \tilde{\mathcal{E}}$*.*
5. $X^1 = X$ *and* $\forall i \in [2, n]$, $X^i = X_{\bar{q}_X \to q_X^i}$ *(as per definition 136) with* $q_X^i = q_\ell^{i-1} \upharpoonright X$*.*
6. $\bigotimes_i^n \mathcal{C}^i = \mathcal{C}^1 \otimes \mathcal{C}^2 \otimes ... \otimes \mathcal{C}^n$
7. $\mathcal{C}^1 \otimes \mathcal{C}^2 = \{\alpha_1^\frown \alpha_2 | \alpha_1 \in \mathcal{C}^1, \alpha_2 \in \mathcal{C}^2\}$ *(The concatenation is always defined by item 3)*

Proof. The properties are ensured by the fact f is pasting-friendly. We prove equality by double inclusion.

- $\subseteq$) Let $\alpha \in \hat{\mathcal{C}}^{\vec{\zeta}}$, and. $\vec{\alpha} = \mathcal{A}\text{-}decomposition(\alpha)$, i.e. $f_{\tilde{\mathcal{E}},X}^{\mathcal{A},proxy}(\vec{\alpha}) = \vec{\zeta}$. By construction due to $\mathcal{A}$-decomposition, $\forall i \in [2, n]$, $fstate(\vec{\alpha}[i]) = lstate(\vec{\alpha}[i-1])$ where $\vec{\alpha}[i-1]$ ends on $\mathcal{A}$-creation (1). Moreover, since f is assumed to be pasting-friendly, each q_ℓ^i is well defined (2). By (1) and (2), $fstate(\vec{\alpha}[i]) = \bar{q}_{\tilde{\mathcal{E}}^i || X^i}$ where $\tilde{\mathcal{E}}^i$ and X^i are defined like in the lemma (3). By construction due to $\mathcal{A}$-decomposition, $\vec{\alpha}[i]$ does not create $\mathcal{A}$ before its very last action, i.e. $\forall \alpha_i' < \vec{\alpha}[i], actions(\alpha_i') \cap \text{creation-actions}(X)(\mathcal{A}) = \emptyset$ (4). Thus by (3) and (4), $\alpha \in \bigotimes_i^n \hat{\mathcal{C}}^{\vec{\zeta}[i]}$. Hence, $\hat{\mathcal{C}}^{\vec{\zeta}} \subseteq \bigotimes_i^n \hat{\mathcal{C}}^{\vec{\zeta}[i]}$
- $\supseteq$) Let $\alpha \in \bigotimes_i^n \hat{\mathcal{C}}^{\vec{\zeta}[i]}$ Let $\vec{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_i, ...) \in \hat{\mathcal{C}}^{\vec{\zeta}[1]} \times \hat{\mathcal{C}}^{\vec{\zeta}[2]} \times ... \times \hat{\mathcal{C}}^{\vec{\zeta}[i]} \times ...$, s.t. $\vec{\alpha} : \alpha$. By construction, $\forall i \in [1, n]$ $f_{(\tilde{\mathcal{E}}^i, X^i)}^{\mathcal{A},proxy}(\alpha_i) = \vec{\zeta}[i]$. Hence $f_{(\tilde{\mathcal{E}}^i, X^i)}^{\mathcal{A},proxy}(\vec{\alpha}) = \vec{\zeta}$. It remains to show that $\vec{\alpha} = \mathcal{A}\text{-decomposition}(\alpha)$, which comes immediately from item 2.

□

A first trivial analysis of the measure of the big class of equivalence gives the following lemma

Lemma 56 (measure after partitioning and decomposition)**.** *Let* $\mathcal{A}$ *be a PSIOA,* X *be a* $\mathcal{A}$*-conservative and* $\mathcal{A}$*-creation-explicit PCA and* $\tilde{\mathcal{E}}$ *partially-compatible with* X*. Let* $\mathcal{E}_\mathcal{A} = \tilde{\mathcal{E}} || X \setminus \{\mathcal{A}\}$*. Let* $\tilde{\zeta} \in \bigcup_{K, \tilde{\mathcal{E}} \in env(K)} range(f_{(\tilde{\mathcal{E}}, K)})$*. Let* $\tilde{\sigma} \in schedulers(\tilde{\mathcal{E}} || X)$*.*

$$\epsilon_{\tilde{\sigma}}(C_{\hat{\mathcal{C}}^{\tilde{\zeta}}}) = \sum_{\vec{\zeta} \in proxy(\tilde{\zeta})_{(\tilde{\mathcal{E}}, X, \mathcal{A})}} \epsilon_{\tilde{\sigma}}(C_{\bigotimes_{i=1}^{|\vec{\zeta}|} \hat{\mathcal{C}}^{\vec{\zeta}[i]}}).$$

Proof. Immediate by two previous lemma 54 and 55 □

4.5.5. Creation oblivious scheduler applied to decomposition

Now we want to transform the term $\epsilon_{\tilde{\sigma}}(C_{\bigotimes_{i=1}^{|\vec{\zeta}|} \hat{\mathcal{C}}^{\vec{\zeta}[i]}})$ as a function of some terms $\epsilon_{\tilde{\sigma}^i}(C_{\hat{\mathcal{C}}^{\vec{\zeta}[i]}})$ where $\tilde{\sigma}^i$ must be defined. The critical point is that the occurrence of these events might not be independent with (*)

a perfect-information scheduler that chooses the measure of class $\hat{\mathcal{C}}^{\vec{\zeta}[i]}$ as a function of the concrete prefix in class $\hat{\mathcal{C}}^{\vec{\zeta}[j<i]}$. This observation enforced us to weaken the implementation definition to make it monotonic w.r.t. PSIOA creation by handling only creation-oblivious schedulers that cannot make the choice (*).

Here again, we exhibit a key property of a perception function to ensure monotonicity of implementation w.r.t. creation oblivious schedulers.

Definition 162 (creation oblivious function)**.** *Let $f_{(.,.)}$ be an insight function. f is said creation-oblivious, if for every PSIOA $\mathcal{A}$, for every $\mathcal{A}$-conservative and $\mathcal{A}$-creation-explicit PCA X, $\forall \tilde{\mathcal{E}} \in env(X)$, $\forall \tilde{\alpha}, \tilde{\alpha}' \in Execs(\tilde{\mathcal{E}}||X)$, $\tilde{\alpha}, \tilde{\alpha}'$ ends on $\mathcal{A}$-creation, then $f^{\mathcal{A},proxy}_{(\tilde{\mathcal{E}},X)}(\tilde{\alpha}) = f^{\mathcal{A},proxy}_{(\tilde{\mathcal{E}},X)}(\tilde{\alpha}')$ implies $\tilde{\alpha} \equiv^{cr}_{\mathcal{A}} \tilde{\alpha}'$.*

In that case, for every $\mathcal{A}$-creation-oblivious scheduler $\tilde{\sigma}$ of $\tilde{\mathcal{E}}||X$, we can note $\tilde{\sigma}|_{\mathcal{A},\zeta} = \tilde{\sigma}|_{\tilde{\alpha}}$ for any $\tilde{\alpha} \in Execs(\tilde{\mathcal{E}}||X)$ s.t. $f^{\mathcal{A},proxy}_{(\tilde{\mathcal{E}},X)}(\tilde{\alpha}) = \zeta$.

This property is naturally verified by the environment projection function.

Lemma 57. *Let $proj_{(.,.)}$ the environment projection function i.e. for each automaton K, $\forall \mathcal{E} \in env(K)$, $proj_{(\mathcal{E},K)} : \alpha \in Execs(\mathcal{E}||K) \mapsto \alpha \upharpoonright \mathcal{E}$. Then $proj_{(.,.)}$ is creation-oblivious.*

Proof. Let $\mathcal{A}$ be a PSIOA, let X be a $\mathcal{A}$-conservative and $\mathcal{A}$-creation-explicit PCA, let $\tilde{\mathcal{E}} \in env(X)$, let $\tilde{\alpha}, \tilde{\alpha}' \in Execs(\tilde{\mathcal{E}}||X)$, s.t. $\tilde{\alpha}, \tilde{\alpha}'$ ends on $\mathcal{A}$-creation and $proj^{\mathcal{A},proxy}_{(\tilde{\mathcal{E}},X)}(\tilde{\alpha}) = proj^{\mathcal{A},proxy}_{(\tilde{\mathcal{E}},X)}(\tilde{\alpha}')$. Then by definition, $(\tilde{\mathcal{E}}||X).\tilde{\mu}^{\mathcal{A}}_e(\alpha) \upharpoonright (\tilde{\mathcal{E}}||(X \setminus \{\mathcal{A}\}) = (\tilde{\mathcal{E}}||X).\tilde{\mu}^{\mathcal{A}}_e(\alpha') \upharpoonright (\tilde{\mathcal{E}}||(X \setminus \{\mathcal{A}\})$ which meets the definition of $\tilde{\alpha} \equiv^{cr}_{\mathcal{A}} \tilde{\alpha}'$. □

Finally, we can terminate our decomposition argument, assuming a creation oblivious scheduler.

Lemma 58 (measure after decomposition for oblivious creation scheduler)**.** *Let $\mathcal{A}$ be a PSIOA, X be a $\mathcal{A}$-conservative, $\mathcal{A}$-creation-explicit PCA and $\tilde{\mathcal{E}}$ partially-compatible with X. Let f a creation-oblivious insight function.*

Let $\tilde{\zeta} \in \bigcup_{K,\tilde{\mathcal{E}} \in env(K)} range(f_{(\tilde{\mathcal{E}},K)})$. Let $n \in \mathbb{N} \cup \{\infty\}$, let $\vec{\zeta} \in \overrightarrow{proxy}(\tilde{\zeta})_{(\tilde{\mathcal{E}},X,\mathcal{A})}$ with $|\vec{\zeta}| = n$. Let $\tilde{\sigma} \in schedulers(\tilde{\mathcal{E}}||X)$ that is $\mathcal{A}$-creation-oblivious.

Then $\epsilon_{\tilde{\sigma}}(C_{\bigotimes_i^n \hat{\mathcal{C}}^{\vec{\zeta}[i]}}) = \prod_i^n \epsilon_{\tilde{\sigma}^i}(C_{\mathcal{C}^{\vec{\zeta}[i]}})$ with $\forall i \in [1,n], \tilde{\sigma}^i = oblivious_{\mathcal{A},\vec{\zeta}[:i]}(\tilde{\sigma})$.

Proof. We recall the remark of definition 150 of $\mathcal{A}$-creation-oblivious scheduler for a $\mathcal{A}$-conservative PCA that raises the fact that if an execution fragment $\tilde{\alpha} \in Frags^*((\tilde{\mathcal{E}}||X))$ verifying

i) $\tilde{\alpha}$ ends on $\mathcal{A}$-creation and ii) $f^{\mathcal{A},proxy}_{(\tilde{\mathcal{E}},X)}(\tilde{\alpha}) = \zeta$, then $\tilde{\sigma}_{|\mathcal{A},\zeta} = \tilde{\sigma}_{|\tilde{\alpha}}$, the sub-scheduler conditioned by $\tilde{\sigma}$ and $\tilde{\alpha}$ in the sense of definition 151. Then we simply apply lemma 45, which states that for every $\alpha = \alpha_x \frown \alpha_y \in Frags^*(\tilde{\mathcal{E}}||X)$, for $\tilde{\sigma}_{|\alpha_x}$ the sub-scheduler conditioned by $\tilde{\sigma} \in schedulers(\tilde{\mathcal{E}}||X)$ and α_x (in the sense of definition 151), for $\epsilon_{\tilde{\sigma}}$ generated by $\tilde{\sigma}$, $\epsilon_{\tilde{\sigma}}(C_\alpha) = \epsilon_{\tilde{\sigma}}(C_{\alpha_x}) \cdot \epsilon_{\tilde{\sigma}_{|\alpha_x}}(C_{\alpha_y})$ with $\tilde{\sigma}_{|\alpha_x}(\alpha_z) = \tilde{\sigma}(\alpha_x \frown \alpha_z)$ for every α_z with $fstate(\alpha_z) = lstate(\alpha_x)$.

For every $\alpha \in \bigotimes_i^n \hat{\mathcal{C}}^{\vec{\zeta}[i]}$, for $\vec{\alpha} = \mathcal{A} - decomposition(\alpha)$, $\epsilon_{\tilde{\sigma}}(C_\alpha) = \prod_i^n \epsilon_{\tilde{\sigma}_{|\vec{\alpha}[1:i-1]}}(C_{\vec{\alpha}[i]})$, with $\vec{\alpha}[1:i-1] = \alpha^1 \frown ... \frown \alpha^{i-1}$.

By $\mathcal{A}$-creation-oblivious property of $\tilde{\sigma}$ and creation-oblivious of f, $\prod_i^n \epsilon_{\tilde{\sigma}_{|\vec{\alpha}[1:i-1]}}(C_{\vec{\alpha}[i]}) = \prod_i^n \epsilon_{\tilde{\sigma}_{|\vec{\zeta}[1:i-1]}}(C_{\vec{\alpha}[i]})$ with $\vec{\zeta}[1:i-1] = f^{\mathcal{A},proxy}_{(\tilde{\mathcal{E}},X)}(\vec{\alpha}[1:i-1])$.

Hence, for every $i \in [1,n]$ we note $\tilde{\sigma}^i \in schedulers(\tilde{\mathcal{E}}^i||X^i)$ that matches $\tilde{\sigma}_{|\vec{\alpha}[1:i-1]}$ on $\mathcal{C}^{\zeta_j}$ for an arbitrary $\vec{\alpha}[1:i-1]$.

This leads us to: $\forall \alpha \in \bigotimes_i^n \hat{\mathcal{C}}^{\vec{\zeta}[i]}$, for $\vec{\alpha} \underset{(X,\mathcal{A})}{:} \alpha$, $\epsilon_{\tilde{\sigma}}(C_\alpha) = \prod_i^n \epsilon_{\tilde{\sigma}^i}(C_{\vec{\alpha}[i]})$

Thus $\epsilon_{\tilde{\sigma}}(C_{\bigotimes_i^n \hat{\mathcal{C}}^{\vec{\zeta}[i]}}) = \sum_{\vec{\alpha} \underset{(X,\mathcal{A})}{:} \alpha,\ \alpha \in \bigotimes_i^n \hat{\mathcal{C}}^{\vec{\zeta}[i]}} \prod_i^n \epsilon_{\tilde{\sigma}^i}(C_{\vec{\alpha}[i]})$ and by lemma 55,

$$\epsilon_{\tilde{\sigma}}(C_{\bigotimes_i^n \hat{\mathcal{C}}^{\vec{\zeta}[i]}}) = \sum_{\alpha_1 \in \mathcal{C}^{\vec{\zeta}[1]}} ... \sum_{\alpha_i \in \mathcal{C}^{\vec{\zeta}[i]}} ...\prod_i^n \epsilon_{\tilde{\sigma}^i}(C_{\alpha_i}) = \prod_i^n \epsilon_{\tilde{\sigma}^i}(C_{\mathcal{C}^{\vec{\zeta}[i]}})$$

□

4.5.6. Monotonicity of implementation

We use the previous decomposition to state the monotonicity of dynamic creation/destruction with the implementation relationship.

Theorem 23 (monotonicity)**.** *Let $\mathcal{A}$, $\mathcal{B}$ be PSIOA, let $X_\mathcal{A}, X_\mathcal{B}$ be PCA. Let S_o be the scheduler schema of creation-oblivious schedulers. Let $f_{(.,.)} = proj_{(.,.)}$ the environment projection function i.e. for each automaton K, $\forall \mathcal{E} \in env(K)$, $f_{(\mathcal{E},K)} : \alpha \in Execs(\mathcal{E}||K) \mapsto \alpha \upharpoonright \mathcal{E}$.*

If $\mathcal{A} \leq_0^{S_o,f} \mathcal{B}$ and $X_\mathcal{A} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_\mathcal{B}$, then $X_\mathcal{A} \leq_0^{S_o,f} X_\mathcal{B}$.

Proof. Let $\tilde{\mathcal{E}} \in env(X_\mathcal{A}) \cap env(X_\mathcal{B})$. Let $Y_\mathcal{A} = X_\mathcal{A} \setminus \{\mathcal{A}\}$, $Y_\mathcal{B} = X_\mathcal{B} \setminus \{\mathcal{B}\}$, $\mathcal{E}_\mathcal{A} = \tilde{\mathcal{E}}||Y_\mathcal{A}$, $\mathcal{E}_\mathcal{B} = \tilde{\mathcal{E}}||Y_\mathcal{B}$ and $\mathcal{E}$ an arbitrary PCA semantically equivalent to both $\mathcal{E}_\mathcal{A}$ and $\mathcal{E}_\mathcal{B}$ with $\mathcal{E} \in env(\tilde{\mathcal{A}}^{sw}) \cap env(\tilde{\mathcal{B}}^{sw})$ by theorem 22. We note $\mu_{\mathcal{AC}}$ the (complete, strong and bijective) PCA executions-matching from $\mathcal{E}_\mathcal{A}$ to $\mathcal{E}$ and $\mu_{\mathcal{CB}}$ the (complete, strong and bijective) PCA executions-matching from $\mathcal{E}$ to $\mathcal{E}_\mathcal{B}$. We also note $\mu^\times_{\mathcal{AC}}$ the (complete, strong and bijective) PCA executions-matching from $\mathcal{E}_\mathcal{A}||\tilde{A}^{sw}$ to $\mathcal{E}||\tilde{A}^{sw}$ and $\mu^\times_{\mathcal{CB}}$ the (complete, strong and bijective) PCA executions-matching from $\mathcal{E}||\tilde{B}^{sw}$ to $\mathcal{E}_\mathcal{B}||\tilde{B}^{sw}$.

In the remaining we note $(\tilde{\mathcal{E}}||X_\mathcal{A})^{\downarrow\zeta}$ the automaton $(\tilde{\mathcal{E}}||X_\mathcal{A})_{\bar{q}_{(\tilde{\mathcal{E}}||X_\mathcal{A})} \to q}$ (as per definition 136) where q is the unique last state of every execution $\tilde{\alpha}$ s.t. $f^{proxy}_{(\tilde{\mathcal{E}},X_\mathcal{A})}(\tilde{\alpha}) = \zeta$. Respectively, we note $(\tilde{\mathcal{E}}||X_\mathcal{B})^{\downarrow\zeta}$ the automaton $(\tilde{\mathcal{E}}||X_\mathcal{B})_{\bar{q}_{(\tilde{\mathcal{E}}||X_\mathcal{B})} \to q}$ (as per definition 136) where q is the unique last state of every execution $\tilde{\pi}$ s.t. $f^{proxy}_{(\tilde{\mathcal{E}},X_\mathcal{B})}(\tilde{\pi}) = \zeta$. This notation is possible because f is pasting-friendly. Finally, $\forall e \in Execs(\tilde{\mathcal{E}})$, we note $\tilde{\mathcal{E}}^e = \tilde{\mathcal{E}}_{\bar{q}_\mathcal{E} \to lstate(e)}$.

Let $\tilde{\sigma} \in S_o(\tilde{\mathcal{E}}||X_\mathcal{A})$. We need to show there exists $\tilde{\sigma}' \in S_o(\tilde{\mathcal{E}}||X_\mathcal{B})$ s.t.

- $\forall \tilde{\zeta} \in range(f_{(\tilde{\mathcal{E}},X_\mathcal{A})}) \cup range(f_{(\tilde{\mathcal{E}},X_\mathcal{B})})$, $\epsilon_{\tilde{\sigma}}(C_{\tilde{\mathcal{C}}^{\tilde{\zeta}}_{X_\mathcal{A}}}) = \epsilon_{\tilde{\sigma}'}(C_{\tilde{\mathcal{C}}^{\tilde{\zeta}}_{X_\mathcal{B}}})$
- where $\tilde{\mathcal{C}}^{\tilde{\zeta}}_{X_\mathcal{A}} = Class(\tilde{\mathcal{E}}, X_\mathcal{A}, f, \tilde{\zeta})$ and $\tilde{\mathcal{C}}^{\tilde{\zeta}}_{X_\mathcal{B}} = Class(\tilde{\mathcal{E}}, X_\mathcal{B}), f, \tilde{\zeta})$.

Let $\tilde{\zeta} \in range(f_{(\tilde{\mathcal{E}},X_\mathcal{A})}) \cup range(f_{(\tilde{\mathcal{E}},X_\mathcal{B})})$. For every $\vec{\zeta} \in \vec{proxy}(\tilde{\zeta})_{(\tilde{\mathcal{E}},X_\mathcal{A},\mathcal{A})}$, $\forall i \in [1 : |\vec{\zeta}|]$, we note $\sigma_{|\mathcal{A},\vec{\zeta}[:i]}$

the $((\tilde{\mathcal{E}}||X_{\mathcal{A}})^{\downarrow \vec{\zeta}[:i]}).\tilde{\mu}_e^{\mathcal{A},+}$ alter-ego of $\tilde{\sigma}_{|\mathcal{A},\vec{\zeta}[:i]}$. For every $i \in [1 : |\vec{\zeta}|]$ $\tilde{\alpha}', \tilde{\alpha}'' \in (f^{\mathcal{A},proxy}_{(\tilde{\mathcal{E}},X_{\mathcal{A}})})^{-1}(\vec{\zeta}[: i])$, $lstate(\tilde{\alpha}') = lstate(\tilde{\alpha}'') \triangleq q_\ell^{i-1}$ since f is pasting-friendly. We note $\mathcal{E}^{(\vec{\zeta},i)} = \mathcal{E}_{\bar{q}_{\mathcal{E}} \to \mu_{\mathcal{AC}}(q_\ell^{i-1}|\mathcal{E}_{\mathcal{A}})}$

We note $\sigma^c_{|\mathcal{A},\vec{\zeta}[:i]} \in schedulers(\mathcal{E}^{(\vec{\zeta},i)}||\tilde{\mathcal{A}}^{sw})$ the $\mu^{\times}_{\mathcal{AC}}$ alter-ego of $\sigma_{|\mathcal{A},\vec{\zeta}[:i]}$.

(*) Since $\mathcal{A} \leq_0^{S_o,f} \mathcal{B}$, $\exists \sigma^d_{|\mathcal{B},\vec{\zeta}[:i]} \in S_o(\mathcal{E}^{(\vec{\zeta},i)}||\tilde{\mathcal{B}}^{sw})$ balanced with $\sigma^c_{|\mathcal{A},\vec{\zeta}[:i]}$, i.e.

- $\forall \zeta' \in range(f_{(\mathcal{E}^i,\tilde{\mathcal{A}}^{sw})}) \cup range(f_{(\mathcal{E}^i,\tilde{\mathcal{B}}^{sw})})$, $\sigma^c_{|\mathcal{A},\vec{\zeta}[:i]}(C_{\check{\mathcal{C}}^{\zeta'}_{\mathcal{A}}}) = \sigma^d_{|\mathcal{B},\vec{\zeta}[:i]}(C_{\check{\mathcal{C}}^{\zeta'}_{\mathcal{B}}})$
- where $\check{\mathcal{C}}^{\zeta'}_{\mathcal{A}} = Class(\mathcal{E}^i, \tilde{\mathcal{A}}^{sw}, f, \zeta')$ and $\check{\mathcal{C}}^{\zeta'}_{\mathcal{B}} = Class(\mathcal{E}^i, \tilde{\mathcal{B}}^{sw}, f, \zeta')$

We note $\sigma'_{|\mathcal{B},\vec{\zeta}[:i]}$ the $\mu^{\times}_{\mathcal{CB}}$ alter-ego of $\sigma^d_{|\mathcal{B},\vec{\zeta}[:i]}$.

We build $\tilde{\sigma}' \in S_o(\tilde{\mathcal{E}}||X_{\mathcal{B}})$ as follows:

For every $\tilde{\zeta} \in range(f_{(\tilde{\mathcal{E}},X_{\mathcal{B}})}) \setminus range(f_{(\tilde{\mathcal{E}},X_{\mathcal{A}})})$, $\forall \vec{\zeta} \in \vec{proxy}(\tilde{\zeta})_{(\tilde{\mathcal{E}},X_{\mathcal{B}},\mathcal{B})}$, $\forall i \in [1 : |\vec{\zeta}|]$, we require that $\tilde{\sigma}_{|\mathcal{B},\vec{\zeta}[:i]}$ halts (i.e. $\forall \tilde{\alpha}', f^{\mathcal{B},proxy}_{(\tilde{\mathcal{E}},X_{\mathcal{B}})}(\tilde{\alpha}') = \vec{\zeta}[: i]$, $supp(\tilde{\sigma}_{|\mathcal{B},\vec{\zeta}[:i]}(\tilde{\alpha}')) = \emptyset$).

For every $\tilde{\zeta} \in range(f_{(\tilde{\mathcal{E}},X_{\mathcal{A}})}) \cup range(f_{(\tilde{\mathcal{E}},X_{\mathcal{B}})})$, $\forall \vec{\zeta} \in \vec{proxy}(\tilde{\zeta})_{(\tilde{\mathcal{E}},X_{\mathcal{B}},\mathcal{B})}$, $\forall i \in [1 : |\vec{\zeta}|]$, we require that $\tilde{\sigma}_{|\mathcal{B},\vec{\zeta}[:i]}$ and $\sigma'_{|\mathcal{B},\vec{\zeta}[:i]}$ are $((\tilde{\mathcal{E}}||X_{\mathcal{B}})^{\downarrow \vec{\zeta}[:i]}).\tilde{\mu}_e^{\mathcal{B},+}$ alter-ego.

Let $\tilde{\zeta} \in range(f_{(\tilde{\mathcal{E}},X_{\mathcal{A}})}) \cup range(f_{(\tilde{\mathcal{E}},X_{\mathcal{B}})})$, let $\vec{\zeta} \in \vec{proxy}(\tilde{\zeta})_{(\tilde{\mathcal{E}},X_{\mathcal{B}},\mathcal{B})}$ For every $i \in [1 : |\vec{\zeta}|]$ $\tilde{\pi}', \tilde{\pi}'' \in (f^{\mathcal{B},proxy}_{(\tilde{\mathcal{E}},X_{\mathcal{B}})})^{-1}(\vec{\zeta}[: i])$, $lstate(\tilde{\pi}') = lstate(\tilde{\pi}') \triangleq q_\ell^{i-1}$ since f is pasting-friendly. We note $\mathcal{E}'^{(\vec{\zeta},i)} = \mathcal{E}_{\bar{q}_{\mathcal{E}} \to \mu_{\mathcal{BC}}(q_\ell^{i-1}|\mathcal{E}_{\mathcal{B}})}$. Moreover, $\mathcal{E}'^{(\vec{\zeta}',i)} = \mathcal{E}^{(\vec{\zeta},i)}$ for every pair $(\vec{\zeta}, \vec{\zeta}')$, s.t. $\mu^{\times}_{\mathcal{AC}}(\vec{\zeta}) = \mu^{\times}_{\mathcal{BC}}(\vec{\zeta}')$.

Now we show that $\tilde{\sigma}$ and $\tilde{\sigma}'$ are balanced:

Let $\tilde{\zeta} \in range(f_{(\tilde{\mathcal{E}},X_{\mathcal{A}})}) \cup range(f_{(\tilde{\mathcal{E}},X_{\mathcal{B}})})$, $(\tilde{\zeta} \in Execs(\tilde{\mathcal{E}}))$. Let

- $\tilde{\mathcal{C}}^{\tilde{\zeta}}_{\mathcal{A}} = Class(\tilde{\mathcal{E}}, X_{\mathcal{A}}, f, \tilde{\zeta})$ and
- $\tilde{\mathcal{C}}^{\tilde{\zeta}}_{\mathcal{B}} = Class(\tilde{\mathcal{E}}, X_{\mathcal{B}}, f, \tilde{\zeta})$

.

We need to show that $\epsilon_{\tilde{\sigma}}(C_{\tilde{\mathcal{C}}^{\tilde{\zeta}}_{\mathcal{A}}}) = \epsilon_{\tilde{\sigma}'}(C_{\tilde{\mathcal{C}}^{\tilde{\zeta}}_{\mathcal{B}}})$:

We apply lemma 56 to obtain:

- $\epsilon_{\tilde{\sigma}}(C_{\tilde{\mathcal{C}}^{\tilde{\zeta}}_{\mathcal{A}}}) = \sum_{\vec{\zeta}_a \in \vec{proxy}(\tilde{\zeta})_{(\tilde{\mathcal{E}},X_{\mathcal{A}},\mathcal{A})}} \epsilon_{\tilde{\sigma}}(C_{\bigotimes_i^n \hat{\mathcal{C}}^{\vec{\zeta}_a[i]}_{\mathcal{A}}})$.
- $\epsilon_{\tilde{\sigma}'}(C_{\tilde{\mathcal{C}}^{\tilde{\zeta}}_{\mathcal{B}}}) = \sum_{\vec{\zeta}_b \in \vec{proxy}(\tilde{\zeta})_{(\tilde{\mathcal{E}},X_{\mathcal{B}},\mathcal{B})}} \epsilon_{\tilde{\sigma}'}(C_{\bigotimes_i^n \hat{\mathcal{C}}^{\vec{\zeta}_b[i]}_{\mathcal{B}}})$.

Since $\mathcal{E}_{\mathcal{A}}$ and $\mathcal{E}_{\mathcal{B}}$ are semantically equivalent, the sets $\{\zeta^a \in Execs(\mathcal{E}_{\mathcal{A}}) | \zeta^a \upharpoonright \tilde{\mathcal{E}} = \tilde{\zeta}\}$ and $\{\zeta^b \in Execs(\mathcal{E}_{\mathcal{B}}) | \zeta^b \upharpoonright \tilde{\mathcal{E}} = \tilde{\zeta}\}$ are in bijection. Hence, it is enough to show that $\forall (\zeta^{ac}, \zeta^{bc}) \in Execs(\mathcal{E}_{\mathcal{A}}) \times$

$Execs(\mathcal{E}_{\mathcal{B}})$ with $\zeta^{bc} = \mu_{\mathcal{AC}} \circ \mu_{\mathcal{CB}}(\zeta^{ac})$ and $\zeta^{bc} \upharpoonright \tilde{\mathcal{E}} = \zeta^{ac} \upharpoonright \tilde{\mathcal{E}} = \tilde{\zeta}$, for $\overset{\rightarrow ac}{\zeta} \overset{n}{\underset{\mathcal{A}}{\vdots}} \zeta^{ac}$, $\overset{\rightarrow bc}{\zeta} \overset{n}{\underset{\mathcal{A}}{\vdots}} \zeta^{bc}$, then $\epsilon_{\tilde{\sigma}}(C_{\bigotimes_i^n \hat{\mathcal{C}}_{\mathcal{A}}^{\zeta \overset{\rightarrow ac}{}[i]}}) = \epsilon_{\tilde{\sigma}'}(C_{\bigotimes_i^n \hat{\mathcal{C}}_{\mathcal{B}}^{\zeta \overset{\rightarrow bc}{}[i]}})$.

By definition, $\tilde{\sigma}$ is $\mathcal{A}$-creation-oblivious, and by construction, $\tilde{\sigma}'$ is $\mathcal{B}$-creation-oblivious. This allows us to apply lemma 58 to obtain:

- $\epsilon_{\tilde{\sigma}}(C_{\bigotimes_i^n \hat{\mathcal{C}}_{\mathcal{A}}^{\zeta \overset{\rightarrow ac}{}[i]}}) = \prod_i^n \epsilon_{\tilde{\sigma}^i}(C_{\hat{\mathcal{C}}_{\mathcal{A}}^{\zeta \overset{\rightarrow ac}{}[i]}})$ with $\forall i \in [1,n], \tilde{\sigma}^i = oblivious_{\mathcal{A}, \overset{\rightarrow ac}{\zeta}[:i]}(\tilde{\sigma}) = \tilde{\sigma}_{|\mathcal{A}, \overset{\rightarrow ac}{\zeta}[:i]}$.
- $\epsilon_{\tilde{\sigma}'}(C_{\bigotimes_i^n \hat{\mathcal{C}}_{\mathcal{B}}^{\zeta \overset{\rightarrow bc}{}[i]}}) = \prod_i^n \epsilon_{\tilde{\sigma}'^i}(C_{\hat{\mathcal{C}}_{\mathcal{B}}^{\zeta \overset{\rightarrow ac}{}[i]}})$ with $\forall i \in [1,n], \tilde{\sigma}'^i = oblivious_{\mathcal{B}, \overset{\rightarrow bc}{\zeta}[:i]}(\tilde{\sigma}') = \tilde{\sigma}_{|\mathcal{B}, \overset{\rightarrow bc}{\zeta}[:i]}$.
- where $\vec{z}[:i] = \vec{z}[1]^\frown ... ^\frown \vec{z}[i-1]$ for $\vec{z} \in \{\overset{\rightarrow ac}{\zeta}, \overset{\rightarrow bc}{\zeta}\}$
- $\hat{\mathcal{C}}_{\mathcal{A}}^{\zeta \overset{\rightarrow ac}{}[i]} = Class((\tilde{\mathcal{E}}||X_{\mathcal{A}})^{\downarrow \overset{\rightarrow ac}{e}[:i])}, f^{\mathcal{A},proxy}, \overset{\rightarrow ac}{\zeta}[i])$
- $\hat{\mathcal{C}}_{\mathcal{B}}^{\zeta \overset{\rightarrow bc}{}[i]} = Class((\tilde{\mathcal{E}}||X_{\mathcal{B}})^{\downarrow \overset{\rightarrow bc}{e}[:i])}, f^{\mathcal{B},proxy}, \overset{\rightarrow bc}{\zeta}[i])$

Thus it is enough to show that $\forall i \in [1,n]$, $\epsilon_{\tilde{\sigma}^i}(C_{\hat{\mathcal{C}}_{\mathcal{A}}^{\zeta \overset{\rightarrow ac}{}[i]}}) = \epsilon_{\tilde{\sigma}'^i}(C_{\hat{\mathcal{C}}_{\mathcal{B}}^{\zeta \overset{\rightarrow bc}{}[i]}})$. Let $i \in [1,n]$

By lemma 48 combined with lemma 50, we obtain:

- $\epsilon_{\tilde{\sigma}^i}(C_{\hat{\mathcal{C}}_{\mathcal{A}}^{\zeta \overset{\rightarrow ac}{}[i]}}) = \epsilon_{\sigma_{|\mathcal{A}, \overset{\rightarrow ac}{\zeta}[:i]}}(\check{\mathcal{C}}_{(\mathcal{E}_{\mathcal{A}},\mathcal{A})}^{(\overset{\rightarrow ac}{\zeta}[i])})$
- $\epsilon_{\tilde{\sigma}'^i}(C_{\hat{\mathcal{C}}_{\mathcal{B}}^{\zeta \overset{\rightarrow bc}{}[i]}}) = \epsilon_{\sigma'_{|\mathcal{B}, \overset{\rightarrow bc}{\zeta}[:i]}}(\check{\mathcal{C}}_{(\mathcal{E}_{\mathcal{B}},\mathcal{B})}^{(\overset{\rightarrow bc}{\zeta}[i])}$.

where:

- $\check{\mathcal{C}}_{(\mathcal{E}_{\mathcal{A}},\mathcal{A})}^{(\overset{\rightarrow ac}{\zeta}[i])} = Class(\mathcal{E}_{\mathcal{A}}^{\zeta \overset{\rightarrow ac}{}[:i]}, \tilde{\mathcal{A}}^{sw}, f, \overset{\rightarrow ac}{\zeta}[i])$ and
- $\check{\mathcal{C}}_{(\mathcal{E}_{\mathcal{B}},\mathcal{B})}^{(\overset{\rightarrow bc}{\zeta}[i])} = Class(\mathcal{E}_{\mathcal{B}}^{\zeta \overset{\rightarrow bc}{}[:i]}, \tilde{\mathcal{B}}^{sw}, f, \overset{\rightarrow bc}{\zeta}[i])$
- $\sigma_{|\mathcal{A}, \overset{\rightarrow ac}{\zeta}[:i]}$ is the $((\tilde{\mathcal{E}}||X_{\mathcal{A}})^{\downarrow \overset{\rightarrow ac}{\zeta}[:i]}).\tilde{\mu}_e^{\mathcal{A},+}$ alter-ego of $\tilde{\sigma}^i$.
- $\sigma'_{|\mathcal{B}, \overset{\rightarrow bc}{\zeta}[:i]}$ is the $((\tilde{\mathcal{E}}||X_{\mathcal{B}})^{\downarrow \overset{\rightarrow bc}{\zeta}[:i]}).\tilde{\mu}_e^{\mathcal{B},+}$ alter-ego of $\tilde{\sigma}'^i$.

.

Hence it is sufficient to show that $\epsilon_{\sigma_{|\mathcal{A}, \overset{\rightarrow ac}{\zeta}[:i]}}(C_{\check{\mathcal{C}}_{(\mathcal{E}_{\mathcal{A}},\mathcal{A})}^{(\overset{\rightarrow ac}{\zeta}[i])}}) = \epsilon_{\sigma'_{|\mathcal{B}, \overset{\rightarrow bc}{\zeta}[:i]}}(C_{\check{\mathcal{C}}_{(\mathcal{E}_{\mathcal{B}},\mathcal{B})}^{(\overset{\rightarrow bc}{\zeta}[i])}})$.

Finally, we find again our construction (*):

- $\epsilon_{\sigma_{|\mathcal{A}, \overset{\rightarrow ac}{\zeta}[:i]}}(C_{\check{\mathcal{C}}_{(\mathcal{E}_{\mathcal{A}},\mathcal{A})}^{(\overset{\rightarrow ac}{\zeta}[i])}}) = \epsilon_{\sigma^c{}_{|\mathcal{A}, \overset{\rightarrow}{\zeta}[:i]}}(C_{\check{\mathcal{C}}_{(\mathcal{E},\mathcal{A})}^{(\overset{\rightarrow}{\zeta}[i])}})$
- $\epsilon_{\sigma'_{|\mathcal{B}, \overset{\rightarrow bc}{\zeta}[:i]}}(C_{\check{\mathcal{C}}_{(\mathcal{E}_{\mathcal{B}},\mathcal{B})}^{(\overset{\rightarrow bc}{\zeta}[i])}}) = \epsilon_{\sigma^d{}_{|\mathcal{B}, \overset{\rightarrow}{\zeta}[:i]}}(C_{\check{\mathcal{C}}_{(\mathcal{E},\mathcal{B})}^{(\overset{\rightarrow}{\zeta}[i])}})$
- $\epsilon_{\sigma^c{}_{|\mathcal{A}, \overset{\rightarrow}{\zeta}[:i]}}(C_{\check{\mathcal{C}}_{(\mathcal{E},\mathcal{A})}^{(\overset{\rightarrow}{\zeta}[i])}}) = \epsilon_{\sigma^d{}_{|\mathcal{B}, \overset{\rightarrow}{\zeta}[:i]}}(C_{\check{\mathcal{C}}_{(\mathcal{E},\mathcal{B})}^{(\overset{\rightarrow}{\zeta}[i])}})$

where:

- $\overrightarrow{e}$ is the vector of $(Frags^*(\mathcal{E}))^n$ s.t. $\forall j \in [1:n]$, $\overrightarrow{\zeta}[j] = \mu_{\mathcal{AC}}(\overrightarrow{\zeta}^{ac}[j]) = \mu^{-1}_{\mathcal{CB}}(\overrightarrow{\zeta}^{bc}[j])$.
- $\check{\mathcal{C}}^{(\overrightarrow{\zeta}[i])}_{(\mathcal{E},\mathcal{A})} = Class(\mathcal{E}^{\overrightarrow{\zeta}[:i])}, \tilde{\mathcal{A}}^{sw}, f, \overrightarrow{\zeta}[i])$ and
- $\check{\mathcal{C}}^{(\overrightarrow{\zeta}[i])}_{(\mathcal{E},\mathcal{B})} = Class(\mathcal{E}^{\overrightarrow{\zeta}[:i])}, \tilde{\mathcal{B}}^{sw}, f, \overrightarrow{\zeta}[i])$

.

This leads us to $\epsilon_{\sigma}{}_{|\mathcal{A},\,\overrightarrow{\zeta}^{ac}[:i]}(C_{\check{\mathcal{C}}^{(\overrightarrow{\zeta}^{ac}[i])}_{(\mathcal{E}_{\mathcal{A}},\mathcal{A})}}) = \epsilon_{\sigma'}{}_{|\mathcal{B},\,\overrightarrow{\beta}[:i],\,\overrightarrow{\zeta}^{bc}[:i]}(C_{\check{\mathcal{C}}^{(\overrightarrow{\zeta}^{bc}[i])}_{(\mathcal{E}_{\mathcal{B}},\mathcal{B})}})$, which ends the proof.

□

4.6. Task schedule

We have shown in the previous section that dynamic creation/destruction of PSIOA is monotonic with $\leq_0^{S_o,\mathtt{P}}$ relationship. In this section, we explain why, without cautious modifications, an easy-to-use off-line scheduler introduced by Canetti et al. [CCK$^+$06a, CCK$^+$18], so-called task-scheduler, is not a priori creation-oblivious which prevents us from obtaining the same result of monotinicity for this particular schema.

4.6.1. Task-schedulers for PIOA

It is convenient to partition the properties required of a program into safety properties and liveness properties [AS85]. A safety property says that nothing bad will ever happen and a liveness property, says that something good will eventually happen. We can understand safety properties as allowed behavior and liveness properties as required behavior. The same kind of partitioning can be done for hyperproperties [CS08]. In distributed computing, the satisfaction of the most interesting liveness properties can be ensured only if the pure non-determinism is resolved by a *fair scheduler*, which does give the opportunity to every process to make computational progress infinitely often. We can imagine a wide variety of fairness conditions imposed on a scheduler schema. The distinction between input and output actions in the context of I/O automata has simplified the specification of such fairness conditions [Tut87]. Indeed, such a distinction allows us to define local actions and simplify the specification of "having the opportunity to make a computational progress". The original idea was to classify the local actions of an I/O automaton $\mathcal{A}$ into *tasks*, where the set of the tasks of $\mathcal{A}$ is its *task-structure*, noted $\mathcal{R}_{\mathcal{A}}$. The task-structure of the composition of two I/O automata $\mathcal{A}$ and $\mathcal{B}$ is then $\mathcal{R}_{\mathcal{A}} \cup \mathcal{R}_{\mathcal{B}}$. An example of the definition of "(weakly) fair execution" α can then be an execution such as, for every task T of the task structure:

- no action of T is enabled from $lstate(\alpha)$ if α is finite.
- If α is infinite, either actions from T appear infinitely often in α or states from which no action of T is enabled appear infinitely often in α

A fair scheduler is then a scheduler resolving the pure non-determinism such that the probability of an unfair execution is 0. Finally, we can exclusively reason with fair schedulers to prove some liveness properties.

Of course, fairness alone might be not sufficient. For example, a powerful perfect-information scheduler can avoid termination of asynchronous randomized consensus protocols, based on the emulation of a common coin, if it is immediately aware of the results of the local coin draws. Here again, the task structure is useful to limit the power of the scheduler, by requiring him to only output tasks instead of actions, without knowing the exact situation. This idea can be formalized easily with the task-scheduler introduced by Canetti et al. [CCK$^+$06a, CCK$^+$18]. Such a scheduler provides in advance (a.k.a. off-line) a potentially infinite sequence $\rho = T_1, T_2, ...$ of tasks. Then the system continuously makes progress by successively (i) triggering the unique enabled action of the next task if it exists, or (ii) ignoring the task otherwise. This scheduler schema is easy to use and the associated implementation relationships can be proved with well-established simulation relationships [CCK$^+$06a, ML07a].

4.6.2. Discussion on adaptation of task-structure in dynamic setting

We adapt the task structure of [CCK$^+$06a, CCK$^+$18] to the dynamic setting. For any PSIOA $\mathcal{A} = (Q_{\mathcal{A}}, \bar{q}_{\mathcal{A}}, sig(\mathcal{A}), D_{\mathcal{A}})$, we note $acts(\mathcal{A}) = \bigcup_{q \in Q_{\mathcal{A}}} sig(\mathcal{A})(q)$, $UI(\mathcal{A}) = \bigcup_{q \in Q_{\mathcal{A}}} in(\mathcal{A})(q)$, $UO(\mathcal{A}) =$

$$\bigcup_{q\in Q_{\mathcal{A}}} out(\mathcal{A})(q),\ UH(\mathcal{A}) = \bigcup_{q\in Q_{\mathcal{A}}} int(\mathcal{A})(q),\ UL(\mathcal{A}) = \bigcup_{q\in Q_{\mathcal{A}}} \widehat{local}(\mathcal{A})(q),\ UE(\mathcal{A}) = \bigcup_{q\in Q_{\mathcal{A}}} \widehat{ext}(\mathcal{A})(q).$$

In classic *PIOA* formalism [Seg95b], if an action $a \in O_{\mathcal{A}} \cap I_{\mathcal{B}}$ is an output action for $\mathcal{A}$ and an input action for $\mathcal{B}$, then a is an output for $\mathcal{A}||\mathcal{B}$ and this does not depend on the current state of $\mathcal{A}||\mathcal{B}$.

In *PSIOA* formalism, if an action $a \in UO(\mathcal{A}) \cap UI(\mathcal{B})$ is an output action for $\mathcal{A}$ at a certain state $q_{\mathcal{A}}$, without being an input action of $\mathcal{A}$ at any other state, while this is an input action for $\mathcal{B}$ at some state $q_{\mathcal{B}}$, without being an output action of $\mathcal{B}$ at another state, then it does not say that a will never be an input of $\mathcal{A}||\mathcal{B}$ at a certain state $q' = (q'_{\mathcal{A}}, q'_{\mathcal{B}})$ where $a \in in(\mathcal{B})(q'_{\mathcal{B}})$ but $a \notin out(\mathcal{A})(q'_{\mathcal{A}})$.

To summarize, if an action can clearly and definitely be an input or an output in PIOA formalism [Seg95b], this is not the case in PSIOA formalism where an action can be an input and becomes an output and vice-versa.

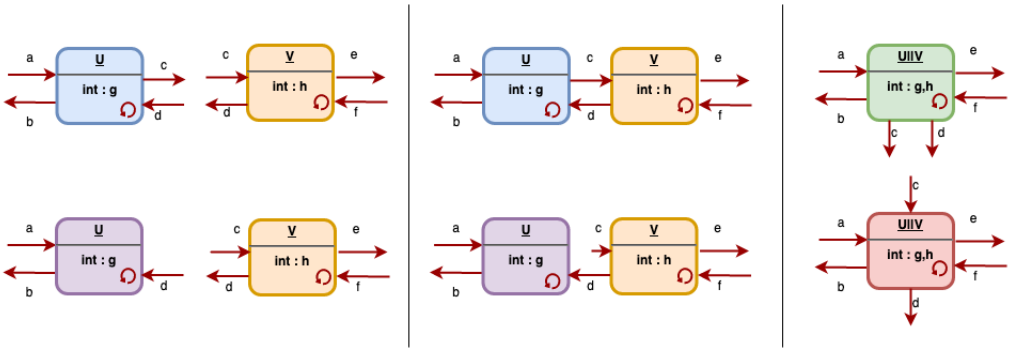


Figure 4.12. – Signature is not fixed

We represents the composition $W = U||V$ of two automata U and V. At two different states $q_W = (q_U, q_V)$ and $q'_W = (q'_U, q'_V)$ where $sig(U)(q'_U) = (in(U)(q_U), out(U)(q_U) \setminus \{c\}, int(U)(q'_U))$. The different states are represented with different colors. The action c is an output of W in q_W but an input of W' in q'_W.

In [CCK+18], a task-structure $\mathcal{R}_{\mathcal{A}}$ of a PIOA $\mathcal{A}$ is an equivalence class on local actions of $\mathcal{A}$ and a task-schedule is a sequence of tasks. The task-structure is assumed to ensure *next-action determinism*, that is for each state $q \in Q_{\mathcal{A}}$, for each task $T \in \mathcal{R}_{\mathcal{A}}$, there exists at most one (local) action $a \in T \cap local(\mathcal{A})(q)$ enabled in q. A task-schedule can hence "resolve the non-determinism", leading to a unique probabilistic measure on the executions. A nice property is that next-action determinism is preserved by composition if the task-structure $\mathcal{R}$ of the parallel composition of task-PIOA $(\mathcal{A}, \mathcal{R}_{\mathcal{A}})$ and $(\mathcal{B}, \mathcal{R}_{\mathcal{B}})$ is defined as $\mathcal{R} = \mathcal{R}_{\mathcal{A}} \cup \mathcal{R}_{\mathcal{B}}$

In PSIOA formalism, the preservation of well-formdness after a composition is less obvious. If we assume that a task is a set of actions ensuring (local action determinism) (that is for each state $q \in Q_{\mathcal{A}}$, for each task $T \in \mathcal{R}_{\mathcal{A}}$, at most one local action $a \in T$ is enabled in q), this property will not be preserved by the composition. Indeed let imagine PISOA $\mathcal{A}$, $\mathcal{B}$, $(q_{\mathcal{A}}, q_{\mathcal{B}}) \in Q_{\mathcal{A}} \times Q_{\mathcal{B}}$ with $sig(\mathcal{A})(q_{\mathcal{A}}) = (\{a\}, \{b\}, \emptyset)$, $sig(\mathcal{B})(q_{\mathcal{B}}) = (\emptyset, \{a\}, \emptyset)$ and $T = \{a, b\}$ is a task of $\mathcal{A}$. Then $sig(\mathcal{A}||\mathcal{B})(q_{\mathcal{A}}, q_{\mathcal{B}}) = (\emptyset, \{a, b\}, \emptyset)$ and both a and b can be enabled.

This observation motivates an additional assumption, called *input partitioning*. We assume the existence of a set of "atomic entities" $Autids_0 \subset Autids$, s.t. for every $\mathcal{A} \in Autids_0$, every action $a \in acts(\mathcal{A})$, $a \in UI(\mathcal{A}) \implies a \notin UO(\mathcal{A})$. Since the vocation of an input a of $\mathcal{A}$ is to be triggered as an output action of a compatible automaton $\mathcal{B}$, this assumption is very conservative. Furthermore, in [AL16], the composition is defined for automata where all the states are compatible. Hence nothing is lost compared to the formalization of [AL16]. Now, we can assume that, for every $\mathcal{A} \in Autids_0$, for

every action $a \in UI(\mathcal{A})$, for every task T of $\mathcal{A}$, $a \notin T$.

This assumption is not preserved by the composition. Indeed, if a is an output of $\mathcal{A} \subset Autids_0$ and an input of $\mathcal{B} \subset Autids_0$, we can have a task $T = \{a\}$ of $\mathcal{A}$, that would become a task of $\mathcal{A}||\mathcal{B}$, where a can be an input of $\mathcal{A}||\mathcal{B}$. In fact, we will assume both input partitioning for $Autids_0$ and local action determinism and we will show that local action determinism is ensured by any PSIOA or PCA built with atomic elements of $Autids_0$.

Another subtlety appears. In static setting, since the signature is unique and compatibility of $\mathcal{A}$ and $\mathcal{B}$ means $UL(\mathcal{A}) \cap UL(\mathcal{B}) = \emptyset$, there is no ambiguity in defining a subset of tasks $\underline{T}' = \{T_{k'}\}_{k' \in K'}$ among the ones of $\mathcal{A}||\mathcal{B}$ composed uniquely of tasks of $\mathcal{A}$ (or $\mathcal{B}$ symmetrically). In the dynamic setting, if a task T is only a set of action labels, T could be a task for different automata (not at the same time). For example, T could be triggered by the $\mathcal{A}$ "contribution" of $\mathcal{A}||\mathcal{B}$ or by the $\mathcal{B}$ "contribution" of $\mathcal{A}||\mathcal{B}$ in alternative execution branches. The confusion can become much greater for a configuration automaton X (formalized in section 4) where each state points to a configuration of dynamic set $\mathbf{A}_X$ of automata (with their own current state). What if the scheduler proposes a task T to a configuration automaton X that goes successively into states q_X and q'_X pointing to configuration C_X and C'_X with different set of automata $\mathbf{A}_X$ and $\mathbf{A}'_X$ where $\mathcal{B} \in \mathbf{A}_X$ and is in its current state $q_{\mathcal{B}}$ and $\mathcal{B}' \in \mathbf{A}'_X$ and is in its current state $q_{\mathcal{B}'}$ with $\mathcal{B} \neq \mathcal{B}'$ but $\widehat{loc}(\mathcal{B})(q_{\mathcal{B}}) \cap \widehat{loc}(\mathcal{B}')(q_{\mathcal{B}'}) \cap T \neq \emptyset$? There are a lot of different ways to deal with this source of ambiguity. To solve it, we have two motivations:

- Reuse the notion of projection of a schedule on an environment as in [CCK+07]
- Obtain our theorem of monocity,. To do so, we need to avoid that a task T that was intended to be triggered by an automaton $\mathcal{A}$ in a certain execution branch α and ignored in another branch α' can be triggered by another automata $\mathcal{A}'$ in an execution branch $\tilde{\alpha}'$ with $trace(\alpha') = trace(\tilde{\alpha}')$ of a configuration automaton X that creates $\mathcal{A}'$ instead of $\mathcal{A}$.

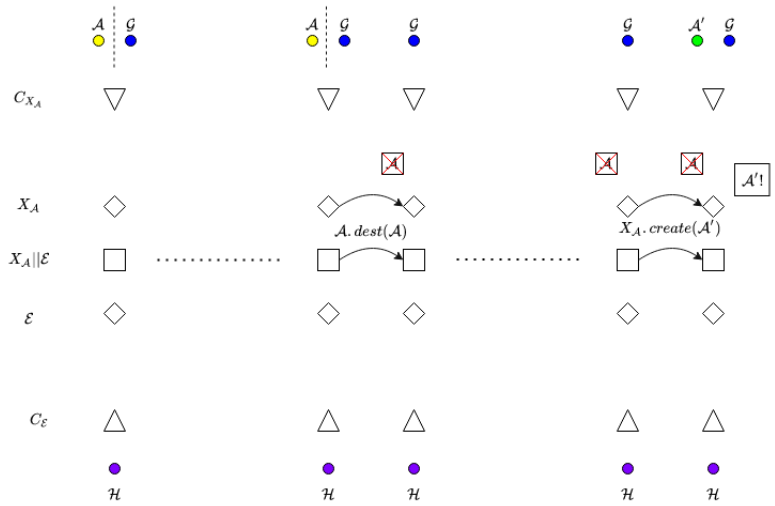


Figure 4.13. – An execution with clones

An example of an execution $\tilde{\alpha}$ of a probabilistic configuration automata (PCA) $X_\mathcal{A}||\mathcal{E}$. At first, $\mathcal{A}$ is a "member" (yellow dot) of $X_\mathcal{A}$, then it is destroyed and finally a clone $\mathcal{A}'$ is created (green dot) in $X_\mathcal{A}$. The formalism of [AL16] allows that $\mathcal{A}$ and $\mathcal{A}'$ are "members" of $X_\mathcal{A}$ in two different states as long as they cannot be members in the same state.

The monocity theorem is based on the fact that $X_\mathcal{A}||\mathcal{E}$ mimics the behaviour of $\tilde{\mathcal{A}}^{sw}||\mathcal{E}''_\mathcal{A}$ with $\mathcal{E}''_\mathcal{A} = X_\mathcal{A} \setminus \{\mathcal{A}\}||\mathcal{E}$ where $\tilde{\mathcal{A}}^{sw}$ is the simpleton wrapper of $\mathcal{A}$ (formalized in definition 133) and $X_\mathcal{A} \setminus \{\mathcal{A}\}$

(formalized in definition 132) is the PCA $X_{\mathcal{A}}$ deprived of $\mathcal{A}$ at each configuration (see figures 4.13 and 4.14). If we examine the succession of reduced configurations (configuration without automata with empty signature) visited in $\tilde{\alpha} \in Execs(X_{\mathcal{A}}||\mathcal{E})$ and in corresponding $\alpha \in Execs(\mathcal{A}||\mathcal{E}''_{\mathcal{A}})$, $\alpha = \mu_e^{\mathcal{A}}(\tilde{\alpha})$, we obtain the same ones (see figure 4.15). Since our theorem takes advantage of the corresponding successions of configurations, it is natural to make appear the ids of $Autids_0$, representing the "atomic" entities among all the entities.

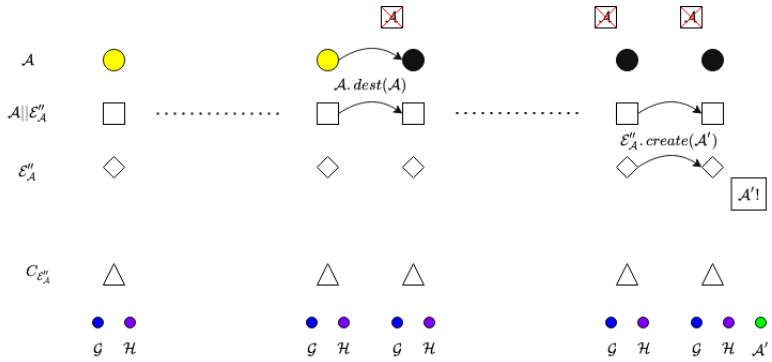


Figure 4.14. – An execution with clones: The perspective of sub-automaton $\mathcal{A}$

The corresponding execution α of $\mathcal{A}||\mathcal{E}''_{\mathcal{A}}$, noted $\alpha = \mu_e^{\mathcal{A}}(\tilde{\alpha})$. At first, $\mathcal{A}$ is "alive" (yellow dot), then it goes forever into a "zombie state" $q_{\mathcal{A}}^{\phi}$ (black dot) where $\widehat{sig}(\mathcal{A})(q_{\mathcal{A}}^{\phi}) = \emptyset$. Finally a clone $\mathcal{A}'$ is created (green dot) in $\mathcal{E}''_{\mathcal{A}}$. The formalism of [AL16] is not supposed to allow this composition since among all the states of $Q_{\mathcal{A}} \times Q_{\mathcal{E}''_{\mathcal{A}}}$, some of them are not compatible. However, it is possible to extend their formalism and define a partial-compatibility where all *reachable* states of $Q_{\mathcal{A}} \times Q_{\mathcal{E}''_{\mathcal{A}}}$ are compatible.

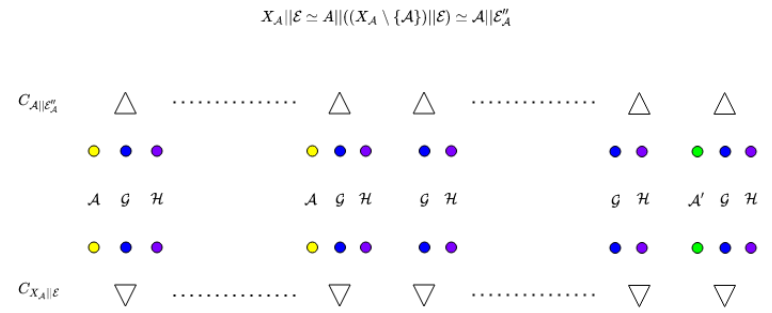


Figure 4.15. – homomorphism between PSIOA and PCA

As long as no creation of $\mathcal{A}$ occurs, the executions $\tilde{\alpha} \in Execs(X_{\mathcal{A}}||\mathcal{E})$ and $\alpha \in Execs(\mathcal{A}||\mathcal{E}''_{\mathcal{A}})$ handle the same succession of reduced configurations.

This formalism avoids the possibility for an atomic entity $\mathcal{A}$ to be a "member" of two different hierarchies as it was already the case in [AL16] which is completely normal in I/O automata formalism. However, contrary to [AL16], the notion of partial-compatibility does not prevent an automaton $\mathcal{A}$ to move from a configuration X to another configuration Y. Indeed we can imagine X and Y that create and destroy $\mathcal{A}$ so that they are partially-compatible (while they cannot be compatible). We can remark we are not dealing with a schedule *of a specific automaton* anymore, which differs from [CCK$^+$07]. However, the restriction of our definition to the "static" setting, where each automaton is the composition of a finite set of automata in $Autids_0$, matches their definition. It will be the responsibility of the task-scheduler to choose a task-schedule $\rho = T_1, ..., T_k, ...$ that produces the probabilistic

distribution that it wants.

According to our understanding, the fact that the set of tasks is not a set of equivalence classes for an equivalence relation is not crucial for the model.

4.6.3. task-schedule for dynamic setting

We formalize the scheduler schema of *task-schedulers* that is a schema of off-line schedulers.

We assume the existence of a subset $Autids_0 \subset Autids$ that represents the "atomic entities" of our formalism. Any automaton is the result of the composition of automata in $Autids_0$.

Definition 163 (Constitution)**.** *For every PSIOA or PCA $\mathcal{A}$, we note*

$$constitution(\mathcal{A}) : \begin{cases} Q_{\mathcal{A}} & \rightarrow & \mathcal{P}(Autids_0) \text{ where } \mathcal{P}(Autids_0) \text{ denotes the power set of } Autids_0 \\ q & \mapsto & constitution(\mathcal{A})(q) \end{cases}$$

The function constitution is defined as follows:

- $\forall \mathcal{A} \in Autids_0$, $\forall q \in Q_{\mathcal{A}}$, $constitution(\mathcal{A})(q) = \{\mathcal{A}\}$.
- $\forall \mathbf{A} = \{\mathcal{A}_1, ..., \mathcal{A}_n\} \in (Autids_0)^n$, $\forall q \in Q_{\mathbf{A}}$, *if* $\mathbf{A}$ *is partially-compatible, then* $constitution(\mathcal{A}_1||...||\mathcal{A}_n)(q) = \mathbf{A}$.
- *The constitution of a PCA is defined recursively through its configuration. For every PCA X, $\forall q \in Q_X$, if we note $(\mathbf{A}, \mathbf{S}) = config(X)(q)$, $constitution(X)(q) = \bigcup_{\mathcal{A} \in \mathbf{A}} constitution(\mathcal{A})(\mathbf{S}(\mathcal{A}))$.*

We can extend the principle of a partial function *map* (attached to a configuration) to the entire constitution of a PCA or PSIOA.

Definition 164 (hierarchy mapping S^H)**.** *Let X be a PCA or a PSIOA. Let $q \in Q_X$ We note $\mathbf{S}^H(X)(q)$ [2] the function that maps any PSIOA $\mathcal{A}_i \in constitution(X)(q)$ to a state $q_{\mathcal{A}_i} \in Q_{\mathcal{A}_i}$ s.t.*

- *if $X = \mathcal{A}_i$, $q_{\mathcal{A}_i} = q$*
- *if $X = \mathcal{A}_1||...||\mathcal{A}_i||...||\mathcal{A}_n$, then $q_{\mathcal{A}_i} = q \upharpoonright \mathcal{A}_i$*
- *if X is a PCA, $q_{\mathcal{A}_i} = S^H(Y)(q_Y)$ where Y is the unique member of $auts(config(X)(q))$ s.t. $\mathcal{A}_i \in constitution(Y)(q_Y)$ with $q_Y = map(config(X)(q))(Y)$*

Anticipating the definition of an enabled task, we extend the definition of tasks of [CCK+18] with an id of $Autids_0$.

Definition 165 (Task)**.** *A task T is a pair $(id, actions)$ where $id \in Autids_0$ and $actions \subset acts(aut(id))$ is a set of action labels. Let $T = (id, actions)$, we note $id(T) = id$ and $actions(T) = actions$.*

Now, we are ready to define the notion of enabled tasks.

Definition 166 (Enabled task)**.** *Let X be a PSIOA or a PCA. A task T is said* enabled *in state $q \in Q_X$ if*

- *$id(T) \in constitution(X)(q)$*
- *there exists a unique local action $a \in \widehat{loc}(\mathcal{A})(q_{\mathcal{A}_i}) \cap actions(T)$ enabled at state $S^H(X)(q)(\mathcal{A})$ [3].*

2. H stands for "hierarchy" and $\mathbf{S}$ refers to notation of mapping function of a configuration $(\mathbf{A}, \mathbf{S})$.

3. action enabling assumption implies that $a \in \widehat{sig}(\mathcal{A}_i)(S^H(X)(q)(\mathcal{A})) \implies a$ enabled at state $S^H(X)(q)(\mathcal{A})$ (i.e. $\exists \eta \in Disc(Q_{\mathcal{A}})$ s.t. $(S^H(X)(q)(\mathcal{A}), a, \eta) \in D_{\mathcal{A}}$)

All previous precautions allow us to define a task-schedule, which is a particular subclass of schedulers, avoiding the technical problems mentioned in the previous subsection. We are not dealing with a task-schedule *of a specific automaton* anymore, which differs from [CCK+18]. However, the restriction of our definition to a "static" setting matches their definition.

Definition 167 (task-schedule)**.** *A task-schedule $\rho = T_1, T_2, T_3, ...$ is a (finite or infinite) sequence of tasks.*

Since our task-schedule is defined, we are ready to solve the non-determinism and define a probability on the executions of a PSIOA. We use the measure of [CCK+18].

Definition 168. *(task-based probability on executions: $apply_{\mathcal{A}}(\mu, \rho) : Frags(\mathcal{A}) \to [0, 1]$) Let $\mathcal{A}$ be a PSIOA. Given $\mu \in Disc(Frags(\mathcal{A}))$ a discrete probability measure on the execution fragments and a task schedule ρ, $apply(\mu, \rho)$ is a probability measure on $Frags(\mathcal{A})$. It is defined recursively as follows.*

1. *$apply_{\mathcal{A}}(\mu, \lambda) := \mu$. Here λ denotes the empty sequence.*
2. *For every T and $\alpha \in Frags^*(\mathcal{A})$, $apply(\mu, T)(\alpha) := p_1(\alpha) + p_2(\alpha)$, where:*
   - $p_1(\alpha) = \begin{cases} \mu(\alpha')\eta_{(\mathcal{A},q',a)}(q) & \text{if } \alpha = \alpha'^\frown(a, q), q' = lstate(\alpha') \text{ and } a \text{ is triggered by } T \text{ enabled after } \alpha' \\ 0 & \text{otherwise} \end{cases}$
   - $p_2(\alpha) = \begin{cases} \mu(\alpha) & \text{if } T \text{ is not enabled after } \alpha \\ 0 & \text{otherwise} \end{cases}$
3. *3. If ρ is finite and of the form $\rho' T$, then $apply_{\mathcal{A}}(\mu, \rho) := apply_{\mathcal{A}}(apply_{\mathcal{A}}(\mu, \rho'), T)$.*
4. *4. If ρ is infinite, let ρ_i denote the length-i prefix of ρ and let pm_i be $apply_{\mathcal{A}}(\mu, \rho_i)$. Then $apply_{\mathcal{A}}(\mu, \rho) := \lim_{i \to \infty} pm_i$.*

Proposition 12. *Let $\mathcal{A}$ be a PSIOA, For each measure μ on $Frags^*(\mathcal{A})$ and task schedule ρ, there exists a scheduler σ for $\mathcal{A}$ such that $apply(\mu, \rho)$ is the generalized probabilistic execution fragment $\epsilon_{\sigma,\mu}$.*

Proof. The result has been proven in [CCK+18], appendix B.4. □

4.6.4. Why a task-scheduler is not creation-oblivious ?

Let us imagine the following example. The class C^x is composed of two executions $\alpha^{x,1}$ and $\alpha^{x,2}$, the class C^y is composed of two executions $\alpha^{y,1}$ and $\alpha^{y,2}$ and the class C^z is composed of four executions $\alpha^{z,11} = \alpha^{x,1}{}^\frown\alpha^{y,1}$, $\alpha^{z,12} = \alpha^{x,1}{}^\frown\alpha^{y,2}$, $\alpha^{z,21} = \alpha^{x,2}{}^\frown\alpha^{y,1}$, $\alpha^{z,22} = \alpha^{x,2}{}^\frown\alpha^{y,2}$. Let $\rho = \rho^1{}^\frown\rho^2$ be a task-schedule. We do not have $apply(., \rho)(C^z)) = apply(., \rho^1)(C^x) \cdot apply(., \rho^2)(C^y)$! Indeed, the executions $\alpha^{x,1}$ and $\alpha^{x,2}$ can differ s.t. they do not ignore the same tasks. Typically, ρ^1 could be written $\rho^1 = \rho^{1,a}{}^\frown\rho^{1,b}$ where the last action of $\alpha^{x,1}$ is triggered by the last task of $\rho^{1,a}$ and $\rho^{1,b}$ is "ignored by $\alpha^{x,1}$. The issue comes if both $apply(., \rho^2)(C^y) \neq \emptyset$ and $apply(., \rho^{1,b}{}^\frown\rho^2)(C^y) \neq \emptyset$. The point is that C^z can be obtained with different cut-paste: cut-paste A: $\rho^{1,a}$ for C^x and $\rho^{1,b}{}^\frown\rho^2$ for C^y ; cut-paste B: ρ^1 for C^x and ρ^2 for C^y.

There is room for finding the appropriate natural assumptions to obtain creation-obliviousness for task-schedules in future work.

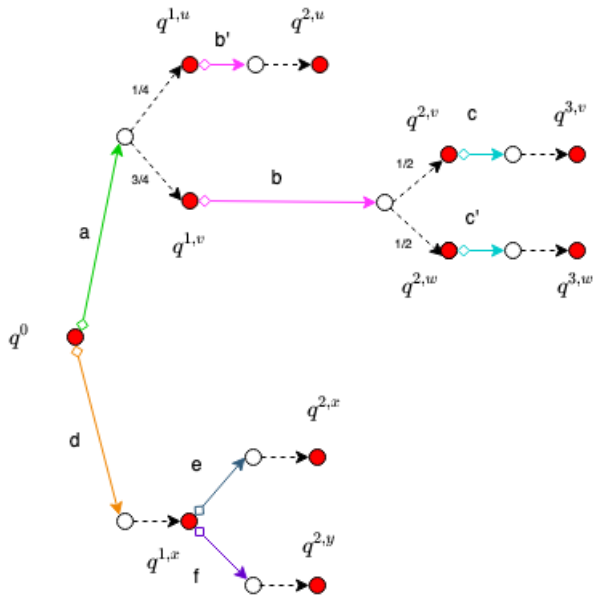


Figure 4.16. – non-deterministic execution requires a (task ?) scheduler

Non-deterministic execution: The scheduler allows us to solve the non-determinism, by triggering an action among the enabled one. We give an example with an automaton $\mathcal{A} = (Q_{\mathcal{A}}, \bar{q}_{\mathcal{A}} = q_0, sig(\mathcal{A}), D_{\mathcal{A}})$ and the tasks T_g, T_o, T_p, T_b (for green, orange, pink, blue) with the respective actions $\{a\}, \{d\}, \{b, b'\}, \{c, c'\}$, and the tasks T_{go}, T_{bo} with the respective actions $\{a, d\}, \{c, c', d\}$. At state q_0, $sig(\mathcal{A})(q_0) = (\emptyset, \{a\}, \{d\})$. Hence both a and d are enabled local action at q_0, which means both T_g and T_o are enabled at state q_0, but T_{go} is not enabled at state q_0 since it does not solve the non-determinism (a and d are enabled local action at q_0). At state q_1, T_p is enabled but neither T_o or T_b. We give some results: $apply(\delta_{q^0}, T_g)(q^0, a, q^{1,v}) = 1$
$apply(\delta_{q^0}, T_gT_p)(q^0, a, q^{1,v}, b, q^{2,w}) = apply(apply(\delta_{q^0}, T_g), T_p)(q^0, a, q^{1,v}, b, q^{2,w}) = 1/2$
$apply(\delta_{q^0}, T_gT_pT_b)(q^0, a, q^{1,v}, b, q^{2,w}, c, q^{3,w}) = apply(apply(\delta_{q^0}, T_gT_p), T_b)(q^0, a, q^{1,v}, b, q^{2,w}, c, q^{3,w}) = 3/8$
$apply(\delta_{q^0}, T_gT_pT_oT_b)(q^0, a, q^{1,v}, b, q^{2,w}, c, q^{3,w}) = 3/8$, since T_o is not enabled at state $q^{2,w}$.

4.7. Monotonicity of Tenacious Task-Implementation

To rescue the task-schedulers, we propose an alternative definition of implementation, called tenacious implementation. In this definition, the equiprobability of corresponding classes for two "balanced" task-schedulers ρ and ρ' has to be preserved even if we "cut" the task-schedulers at some arbitrary points.

First, we anticipate some technical issues and we adapt the definition of environment.

Definition 169 (Independent environment)**.** *Let $\mathcal{V}$ be a PCA (resp a PSIOA). An independent environment $\mathcal{E}$ for $\mathcal{V}$ is a PCA (resp. a PSIOA) partially-compatible with $\mathcal{V}$ s.t. $UA(\mathcal{E}) \cap UA(\mathcal{V}) = \emptyset$. We note $ienv(\mathcal{V})$ the set of independent environment of $\mathcal{V}$.*

At first glance, we might think that such a new definition would prevent from applying the argument of proof of theorem 3 for the composability of implementation. Indeed, if $\mathcal{A}$ and $\mathcal{B}$ are partially compatible and $\mathcal{E} \in ienv(\mathcal{A}||\mathcal{B})$, we cannot say that $\mathcal{E}||\mathcal{B} \in ienv(\mathcal{A})$ since we could have $UA(\mathcal{A}) \cap UA(\mathcal{B}) \neq \emptyset$ (typically, to model the movement of an agent from $\mathcal{A}$ to $\mathcal{B}$ and vice versa. But again, we can apply a renaming operator to $\mathcal{B}$, to obtain $\mathcal{B}' \in ienv(\mathcal{A})$ (and so $\mathcal{E}||\mathcal{B}' \in ienv(\mathcal{A})$) with $\mathcal{A}||\mathcal{B}'$ completely

equivalent to $\mathcal{A}||\mathcal{B}$. Thus, this definition does not limit the expressiveness of the model.

The next definition 170 captures the idea that two schedulers implies the same probability measure for corresponding classes of executions, with the same perception from the environment.

Definition 170 ($S^{\leq\varepsilon}_{(\mathcal{A},\mathcal{B})}$)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA or two PCA. Let (ρ,ρ') be two task-schedulers. Let $\varepsilon\in\mathbb{R}^{\geq 0}$ We say that $\rho S^{\leq\varepsilon}_{(\mathcal{A},\mathcal{B})}\rho'$ if $\forall\mathcal{E}\in ienv(\mathcal{A})\cap ienv(\mathcal{B})$, $\forall e\in Execs(\mathcal{E})$,*
$|apply_{(\mathcal{E}||\mathcal{A})}(\delta_{\bar{q}_{\mathcal{E}||\mathcal{A}}},proj^{-1}_{(\mathcal{E},\mathcal{A})}(e))-apply_{(\mathcal{E}||\mathcal{B})}(\delta_{\bar{q}_{\mathcal{E}||\mathcal{B}}},proj^{-1}_{(\mathcal{E},\mathcal{B})}(e))|\leq\varepsilon$.

For any environment, the probability to make a distinction between the two situations under the respective schedulers ρ and ρ' is not greater than $\frac{1}{2}+\varepsilon$. We stress that the pair (ρ,ρ') is fixed for all the possible environments.

We can remark that schedulers are "perfectly balanced", noted $\rho S^{\leq 0}_{(\mathcal{A},\mathcal{B})}\rho'$, if $\forall\mathcal{E}\in ienv(\mathcal{A})\cap ienv(\mathcal{B})$, $\forall e\in Execs(\mathcal{E})$, $apply_{(\mathcal{E}||\mathcal{A})}(\delta_{\bar{q}_{\mathcal{E}||\mathcal{A}}},proj^{-1}_{(\mathcal{E},\mathcal{A})}(e))=apply_{(\mathcal{E}||\mathcal{B})}(\delta_{\bar{q}_{\mathcal{E}||\mathcal{B}}},proj^{-1}_{(\mathcal{E},\mathcal{B})}(e))$.

Also, we could consider an alternative definition where $\rho S^{\leq\varepsilon}_{(\mathcal{A},\mathcal{B})}\rho'$ if $\forall\mathcal{E}\in ienv(\mathcal{A})\cap ienv(\mathcal{B})$, $\forall E\subseteq Execs(\mathcal{E})$, $|apply_{(\mathcal{E}||\mathcal{A})}(\delta_{\bar{q}_{\mathcal{E}||\mathcal{A}}},proj^{-1}_{(\mathcal{E},\mathcal{A})}(E))-apply_{(\mathcal{E}||\mathcal{B})}(\delta_{\bar{q}_{\mathcal{E}||\mathcal{B}}},proj^{-1}_{(\mathcal{E},\mathcal{B})}(E))|\leq\varepsilon$. Such a definition might be more convenient for monotonicity of dynamic creation/destruction of PSIOA with approximate tenacious implementation.

4.7.1. schedule notations

Here, we give syntactical tools to manipulate easily task-schedules.

Definition 171 (simple schedule notation)**.** *Let $\rho=T^{\ell},T^{\ell+1},...,T^{h},...$ be a schedule, i. e. a sequence of tasks, beginning with T^{ℓ} and terminating by T^{h} if ρ is finite with $\ell,h\in\mathbb{N}^{*}$. For every $q,q'\in[\ell,h],q\leq q'$, we note:*

- *$hi(\rho)=h$ the highest index in ρ ($hi(\rho)=\omega$ if ρ is infinite)*
- *$li(\rho)=\ell$ the lowest index in ρ*
- *$\rho[q]=T^{q}$*
- *$\rho|_{q}=T^{\ell}...T^{q}$*
- *${}_{q}|\rho=T^{q}...T^{h}...$*
- *${}_{q}|\rho|_{q'}=T^{q}...T^{q'}$*

By doing so, we implicitly assume an indexation of ρ, $ind(\rho):ind\in[li(\rho),hi(\rho)]\mapsto T^{ind}\in\rho$. Hence if $\rho=T^{1},T^{2},...,T^{k},T^{k+1},...,T^{q},T^{q+1}...,T^{h},...$, $\rho'=_{k}|\rho$, $\rho''=_{q}|\rho'$, then $\rho''=_{q}|\rho$.

Definition 172 (Schedule partition and index)**.** *Let ρ be a schedule. A partition p of ρ is a sequence of schedules (finite or infinite) $p=(\rho^{m},\rho^{m+1},...,\rho^{n},...)$ so that ρ can be written $\rho=\rho^{m},\rho^{m+1},...,\rho^{n},....$ We note $min(p)=m$ and $max(p)=card(p)+m-1$ (if p is infinite, $max(p)=\omega$).*

A total ordered set $(ind(\rho,p),\prec)\subset\mathbb{N}^{2}$ is defined as follows :

$ind(\rho,p)=\{(k,q)\in(\mathbb{N}^{*})^{2}|k\in[min(p),max(p)],q\in[li(\rho^{k}),hi(\rho^{k})]\}$ For every $\ell=(k,q),\ell'=(k',q')\in ind(\rho,p)$:

- *If $k<k'$, then $\ell\prec\ell'$*
- *If $k=k',q<q'$, then $\ell\prec\ell'$*
- *If $k=k'$ and $q=q'$, then $\ell=\ell'$. If either $\ell\prec\ell'$ or $\ell=\ell'$, we note $\ell\preceq\ell'$.*

For every $\ell = (k, q) \in ind(\rho, p)$, we note $\ell + 1$ the smaller element (according to $\prec$) of $ind(\rho, p)$ that is greater than ℓ. For convenience, we extend $ind(\rho, p)$ with $\{(k, 0) \in (\mathbb{N}^*)^2 | k \leq card(p)\}$, where $(k + 1, 0) \triangleq (k, card(\rho^k))$.

Definition 173 (Schedule notation)**.** *Let ρ be a schedule. Let p be a partition of ρ. For every $\ell = (k, q), \ell' = (k', q') \in ind(\rho, p)^2$, we note (when this is allowed):*

- $\rho[p, \ell] = \rho^k[q]$
- $\rho|_{(p,\ell)} = \rho^1, ..., \rho^k|_q$
- $_{(p,\ell)}|\rho = (_q|\rho^k), ...$
- $_\ell|\rho|_{(p,\ell')} = (_q|\rho^k), ..., (\rho^{k'}|_q)$

The symbol p of the partition is removed when it is clear in the context.

4.7.2. Tenacious Implementation

Before defining tenacious implementation, we take some precautions to properly partition the tasks dedicated to the environment and the task dedicated to the analysed system.

Definition 174 ($\mathcal{V}$-partition of a schedule)**.** *Let $\mathcal{V}$ be a PCA or a PSIOA. Let $\rho_{\mathcal{VE}}$ be a schedule. Let $p = (\rho^1_{\mathcal{V}}, \rho^2_{\mathcal{E}}, \rho^3_{\mathcal{V}}, \rho^4_{\mathcal{E}}...)$ be the obviously unique partition of $\rho_{\mathcal{VE}}$ where (1) each $\rho^{2k+1}_{\mathcal{V}}$ is a sequence of tasks of $UA(\mathcal{V})$ only, (2) each $\rho^{2k}_{\mathcal{E}}$ does not contain any task of $UA(\mathcal{V})$, and (3) each $\rho^{2k}_{\mathcal{E}}$ has a length strictly greater than 0. We call such a partition, the* $\mathcal{V}$-partition *of $\rho_{\mathcal{VE}}$.*

Thus, in the remaining, we say *the* $\mathcal{V}$-partition of a schedule.

Definition 175 (Environment corresponding schedule)**.** *Let $\mathcal{V}$ and $\mathcal{W}$ be two PCA or two PSIOA. Let $\rho_{\mathcal{VE}}$ and $\rho_{\mathcal{WE}}$ be two schedules. Let $(\rho^1_{\mathcal{V}}, \rho^2_{\mathcal{E}}, \rho^3_{\mathcal{V}}, \rho^4_{\mathcal{E}}...)$ (resp. $\rho_{\mathcal{WE}}$: $(\rho^1_{\mathcal{W}}, \rho^{2'}_{\mathcal{E}}, \rho^3_{\mathcal{W}}, \rho^{4'}_{\mathcal{E}}, ...)$) be the $\mathcal{V}$-partition (resp. $\mathcal{W}$-partition) of $\rho_{\mathcal{VE}}$ (resp. $\rho_{\mathcal{WE}}$). We say that $\rho_{\mathcal{VE}}$ and $\rho_{\mathcal{WE}}$ are* $\mathcal{VW}$-environment-corresponding *if for every k, $\rho^{2k}_{\mathcal{E}} = \rho^{2k'}_{\mathcal{E}}$.*

Environment corresponding schedules only differ on the tasks that do not concern the environment.

Definition 176 ($S^{ten,\leq\varepsilon}_{(\mathcal{A},\mathcal{B})}$)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA or two PCA. Let (ρ, ρ') be two task-schedulers. Let $\epsilon \in \mathbb{R}^{\geq 0}$. We say that $\rho S^{ten,\leq\varepsilon}_{(\mathcal{A},\mathcal{B})} \rho'$ if*

- *ρ and ρ' are $(\mathcal{A}, \mathcal{B})$-environment-corresponding*
- *$\forall \ell, \ell' \in ind(\rho, p) \cap ind(\rho', p') \cap (2\mathbb{N}, \mathbb{N})$, for $\tilde{\rho} = (_\ell|\rho|_{\ell'})$, $\tilde{\rho}' = (_\ell|\rho'|_{\ell'})$, $\tilde{\rho} S^{\leq\varepsilon}_{(\mathcal{A},\mathcal{B})} \tilde{\rho}'$*

Definition 177 (Tenacious implementation)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two PSIOA or two PCA. Let $\epsilon \in \mathbb{R}^{\geq 0}$. We say that $\mathcal{A}$ ε-*tenaciously implements *$\mathcal{B}$, noted $\mathcal{A} \leq^{ten}_{\varepsilon} \mathcal{B}$, if for every task-scheduler ρ, there exists a task-scheduler ρ' such that $\rho S^{ten,\leq\varepsilon}_{(\mathcal{A},\mathcal{B})} \rho'$. If $\varepsilon = 0$, we say $\mathcal{A}$* tenaciously implements *$\mathcal{B}$*

4.7.3. sub-classes according to a task schedule

It appeared that the monotonicity of PSIOA creation with $\leq^{ten}_0$ is not straightforward. The next 3 definitions will give us useful tools to address the demonstration in a modular way.

The next definition captures the idea that the last action of a given execution is triggered by the last task of a task-schedule.

Definition 178 (execution matching a schedule)**.** *Let X be an automaton, let α be an execution of X, and $\rho = \rho' T$ be a task-schedule. We say that α matches ρ iff $\alpha \in supp(apply_X(\delta_{fstate(\alpha)}, \rho))$ but $\alpha \notin supp(apply_X(\delta_{fstate(\alpha)}, \rho'))$.*

If $\alpha \in supp(apply_X(\delta_{fstate(\alpha)}, \lambda))$, we say that α match λ (the empty sequence).

Hence, the last action of α is triggered by the last task of ρ.

The next definition formalizes the idea that a prefix of a task-schedule is not useless to provoke an execution. This definition has only a pedagogical interest to stress we do not require this kind of "matching".

Definition 179 (execution strongly matching a schedule)**.** *Let X be an automaton, let $\alpha = q^0 a^1 \frown \alpha'$ be an execution of X, and $\rho = T\rho'$ be a task-schedule. We say that α strongly matches ρ iff α matches ρ and a^1 is triggered by T at state q^0.*

The idea is that T is not useless. In fact, we will see that this definition has no interest to obtain our intermediate results. We introduced it only to explain why this form of matching is not relevant.

Now, we can introduce a notation to represent sub-sets of executions that matches a certain sub-schedule. It is an important step before our partitioning.

Definition 180 (Notations for executions matching a schedule)**.** *Let K be a PCA or a PSIOA, let $\alpha \in execs(K)$ and $\underline{\alpha} \subset execs(K)$, s. t. $\alpha \in \underline{\alpha}$. Let ρ be a schedule, p be a fixed partition of ρ, $\ell_1, \ell_2, \ell_1^-, \ell_2^-, \ell_1^+, \ell_2^+ \in ind(\rho, p)$, we note :*

- $\underline{\alpha}_{(\ell_1,\rho)} = \{\tilde{\alpha} \in \underline{\alpha} | \tilde{\alpha}$ *matches* $\rho|_{\ell_1}\}$
- $\underline{\alpha}_{(\ell_1,\ell_2,\rho)} = \{\tilde{\alpha} \in \underline{\alpha} | \tilde{\alpha}$ *matches* $_{\ell_1}|\rho|_{\ell_2}\}$ *(warning:* $\neq \{\tilde{\alpha} \in \underline{\alpha} | \tilde{\alpha}$ *strongly matches* $_{\ell_1}|\rho|_{\ell_2}\}$ *)*
- $\underline{\alpha}_{(\ell_1,[\ell_2^-,\ell_2^+],\rho)} = \{\tilde{\alpha} \in \underline{\alpha} | \exists \ell^2 \in [\ell_2^-, \ell_2^+], \tilde{\alpha}$ *matches* $_{\ell_1}|\rho|_{\ell_2}\}$

We can remark that if $\ell_1 = min(ind(\rho, p))$, then $\underline{\alpha}_{(\ell_2,\rho)} = \underline{\alpha}_{(\ell_1,\ell_2,\rho)}$ by definition.

The next lemma states that for every task-schedule ρ a set $\underline{\alpha}$ of executions can be partitioned with sub-sets of the form $\underline{\alpha}_{\ell,\rho}$ modulo some executions with zero-probability to occur under task-schedule ρ

Lemma 59 (class-partitioning according to a schedule)**.** *Let X be a PSIOA or a PCA, $\alpha \in execs(X)$ and $\underline{\alpha} \subset execs(X)$, s. t. $\alpha \in \underline{\alpha}$, ρ be a task-schedule, p be a fixed partition of ρ.*

Then $\{\underline{\alpha}_{(\ell^+,\rho)} \cap supp(apply_X(\delta_{fstate(\alpha)}, \rho)) | \ell^+ \in ind(\rho, p)\}$ is a partition of $\underline{\alpha} \cap supp(apply_X(\delta_{fstate(\alpha)}, \rho))\}$.

Proof. – empty intersection: Let $\ell, \ell' \in ind(\rho, p)$. Let $\alpha \in \underline{\alpha}_{\ell,\rho}$, we show that $\alpha \notin \underline{\alpha}_{\ell',\rho}$. By contradiction, we assume the contrary: thus, $\alpha \in supp(apply_X(\delta_{fstate(\alpha)}, \rho|_\ell))$, $\alpha \in supp(apply_X(\delta_{fstate(\alpha)}, \rho|_{\ell'}))$ but $\alpha \notin supp(apply_X(\delta_{fstate(\alpha)}, \rho|_{\ell-1}))$ and $\alpha \notin supp(apply_X(\delta_{fstate(\alpha)}, \rho|_{\ell'-1}))$. If $\ell = \ell' + 1$ or $\ell' = \ell + 1$, the contradiction is immediate.

Without lost of generality, we assume $\ell' \prec \ell + 1$. Since $\alpha \in supp(apply_X(\delta_{fstate(\alpha)}, \rho|_\ell))$, $\alpha \in supp(apply_X(\delta_{fstate(\alpha)}, \rho|_{\ell'}))$, all the tasks in $_{\ell'+1}|\rho|_\ell$ are not enabled in $lstate(\alpha)$, but this is in contradiction with the fact that both $\alpha \in supp(apply_X(\delta_{fstate(\alpha)}, \rho|_{\ell'}))$ and $\alpha \notin supp(apply_X(\delta_{fstate(\alpha)}, \rho|_{\ell-1}))$.

– complete union: Let $\alpha' = \alpha'' \frown a q' \in supp(apply_X(\delta_{fstate(\alpha)}, \rho))$, with $q'' = lstate(\alpha'')$. We show it exists $\ell \in ind(\rho, p)$, so that α' matches $\rho|_\ell$. By contradiction, it means α' matches $\rho|_\ell$ for every $\ell \in ind(\rho, p)$, namely α' matches $\rho|_0 = \lambda$ (the empty sequence) and that for every task T in ρ, T is not enabled in q''. Thus $apply_X(\delta_{fstate(\alpha)}, \lambda)(\alpha') > 0$, which is in contradiction with

$\alpha' \neq fstate(\alpha)$. If $\alpha' = q^0$ and for every task T in ρ, T is not enabled in q^0, then α' matches $\rho_0 = 0$.

□

Hence, for any task-schedule, each class of executions can be partitioned into sub-classes that match one of the prefixes of ρ. This allows us to express a natural lemma of total probability law on all the possible matchings.

Lemma 60 (Total probability law on all possible matchings)**.** *Let X be a PSIOA or a PCA, $\alpha \in execs(X)$ and $\underline{\alpha} \subset execs(X)$, s. t. $\alpha \in \underline{\alpha}$, ρ be a task-schedule, p be a fixed partition of ρ.*

$$apply_X(\delta_{fstate(\alpha)}, \rho)(\underline{\alpha}) = \sum_{\ell^+ \in ind(\rho,p)} apply_X(\delta_{fstate(\alpha)}, \rho)(\underline{\alpha}_{(\ell^+,\rho)})$$

Proof. $\{\underline{\alpha}_{\ell^+,\rho} \cap supp(apply_X(\delta_{fstate(\alpha)}, \rho)) | \ell^+ \in ind(\rho, p)\}$ is a partition of $\underline{\alpha} \cap supp(apply_X(\delta_{fstate(\alpha)}, \rho))\}$, which gives $apply_X(\delta_{fstate(\alpha)}, \rho)(\underline{\alpha}) = \sum_{\ell^+ \in ind(\rho,p)} apply_X(\delta_{fstate(\alpha)}, \rho)(\underline{\alpha}_{(\ell^+,\rho)})$ that is the result. □

We stress that the second argument of function $apply(.,.)(.)$ in the right term is ρ and not $\rho|_{\ell^+}$, otherwise, the result would be $apply_X(\delta_{fstate(\alpha)}, \rho)(\underline{\alpha}) = \sum_{\ell^+ \in ind(\rho,p)} apply_X(\delta_{fstate(\alpha)}, \rho|_{\ell^+})(\underline{\alpha}_{(\ell^+,\rho)} \cap supp(\delta_{fstate(\alpha)}, \rho))$

We slightly anticipate the next subsection to reduce the problem to the probability measures of classes of equivalence of executions without creation before the last action.

Lemma 61 (Express a cut with the entire class)**.** *Let X be a PSIOA or a PCA, $\alpha \in execs(X)$ and $\underline{\alpha} \subset execs(X)$, s. t. $\alpha \in \underline{\alpha}$, ρ be a task-schedule, p be a fixed partition of ρ.*

$$\forall \ell^- \in ind(p, \rho), \sum_{\ell^+ \in ind(p,\rho)}^{\ell^- \prec \ell^+} apply_X(\delta_{fstate(\alpha)}, \rho)(\underline{\alpha}_{\ell^+,\rho}) = apply_X(\delta_{fstate(\alpha)}, \rho)(\underline{\alpha}) - apply_X(\delta_{fstate(\alpha)}, \rho|_{\ell^-})(\underline{\alpha})$$

Proof. We simply apply the previous lemma twice, for both

$apply_X(\delta_{fstate(\alpha)}, \rho)(\underline{\alpha}) = \sum_{\ell \in ind(\rho,p)} apply_X(\delta_{fstate(\alpha)}, \rho)(\underline{\alpha}_{\ell,\rho})$ and

$apply_X(\delta_{fstate(\alpha)}, \rho|_{\ell^-})(\underline{\alpha}) = \sum_{\ell \in ind(\rho|_{\ell^-},p)} apply_X(\delta_{fstate(\alpha)}, \rho)(\underline{\alpha}_{\ell,\rho})$ and the substraction gives the result. □

We finish the current paragraph by introducing a sort of notion of "independence" between successive executions:

Lemma 62 (independent successive executions for a split task-schedule)**.** *Let K be a PSIOA. Let $\rho = \rho^1 \frown \rho^2$ and $\alpha = \alpha^1 \frown \alpha^2 \in execs(K)$ with α^1 matching ρ^1 where $q^0 = fstate(\alpha^1), q^1 = lstate(\alpha^1) = fstate(\alpha^2), q^2 = lstate(\alpha^2)$.*

Then $apply(\delta_{q^0}, \rho)(\alpha) = apply(\delta_{q^0}, \rho^1)(\alpha^1) \cdot apply(\delta_{q^1}, \rho^2)(\alpha^2)$

Proof. By induction on the size of ρ_2.

Special case: $\rho_2 = \lambda$

Case 1) $\alpha^2 = q^1$.

$apply(\delta_{q^0}, \rho)(\alpha) = apply(\delta_{q^0}, \rho^1)(\alpha^1)$ while $apply(\delta_{q^1}, \rho^2)(\alpha^2) = 1$

Case 2) $\alpha^2 = \alpha^{2\prime}{}^\frown q^{2\prime} a q^2$. In this case, $apply(\delta_{q^1}, \rho^2)(\alpha^2) = apply(\delta_{q^0}, \rho)(\alpha) = 0$

This ends the special case, we can start the induction.

We note $\mu_1 = apply(\delta_{q^0}, \rho^1)$

Basis: $\rho_2 = T$

We have:

$apply(\delta_{q^0}, \rho)(\alpha) = apply(\mu_1, T)(\alpha) = p_1(\alpha) + p_2(\alpha)$ with

- $p_1(\alpha) = \begin{cases} \mu_1(\alpha') \cdot \eta_{K,q',a}(q^2) & \text{if } a \text{ triggered by } T \text{ enabled at } q' \text{ with } \alpha = \alpha'^\frown q' a q^2 \\ 0 & \text{otherwise} \end{cases}$
- $p_2(\alpha) = \begin{cases} \mu_1(\alpha) & \text{if } T \text{ is not enabled after } \alpha \\ 0 & \text{otherwise} \end{cases}$

and

$apply(\delta_{q^1}, \rho^2)(\alpha^2) = apply(\delta_{q^1}, T)(\alpha^2) = p'_1(\alpha^2) + p'_2(\alpha^2)$ with

- $p'_1(\alpha^2) = \begin{cases} \delta_{q^1}(\alpha^{2\prime}) \cdot \eta_{K,q^{2\prime},a}(q^2) & \text{if } a \text{ triggered by } T \text{ enabled at } q^{2\prime} \text{ with } \alpha^2 = \alpha^{2\prime}{}^\frown q^{2\prime} a q^2 \\ 0 & \text{otherwise} \end{cases}$
- $p'_2(\alpha^2) = \begin{cases} \delta_{q^1}(\alpha^2) & \text{if } T \text{ is not enabled after } \alpha^2 \\ 0 & \text{otherwise} \end{cases}$

We have T enabled after $\alpha \iff T$ enabled after α^2

Case 1) $\alpha^2 = q^2 = q^1$ i. e. $\alpha = \alpha^1$ and $\delta_{q^1}(\alpha^2) = 1$.

Since α^1 is matching ρ^1, the factor $\mu_1(\alpha')$ of p_1 is necessarily equal to 0, which means $p_1(\alpha) = 0$. Moreover, $p'_1(\alpha^2) = 0$ since $|\alpha^2| = 0$.

So $apply(\delta_{q^1}, \rho^2)(\alpha^2) = p'_2(\alpha^2)$, while $apply(\delta_{q^0}, \rho)(\alpha) = p_2(\alpha)$ and we find $apply(\delta_{q^0}, \rho)(\alpha) = \mu_1(\alpha^1) \cdot apply(\delta_{q^1}, \rho^2)(\alpha^2)$.

Case 2) $\alpha^2 = \alpha^{2\prime}{}^\frown q^{2\prime} a q^2$ i. e. $q' = q^{2\prime}$, $\alpha' = \alpha^1{}^\frown \alpha^{2\prime}$

We have T enabled after $\alpha' \iff T$ enabled after $\alpha^{2\prime}$

If T not enabled after α, $apply(\delta_{q^1}, \rho^2)(\alpha^2) = 0$, while $apply(\delta_{q^0}, \rho)(\alpha) = apply(\delta_{q^0}, \rho^1)(\alpha^1 \frown \alpha^2) = 0$ since α^1 matches ρ^1 and $|\alpha^2| > 0$. and we find $apply(\delta_{q^0}, \rho)(\alpha) = \mu_1(\alpha^1) \cdot apply(\delta_{q^1}, \rho^2)(\alpha^2) = 0$.

If T is enabled after α,

Case 2a) $\alpha^{2\prime} = \alpha^{2\prime\prime}{}^\frown q^{2\prime\prime} a' q'$. For the same reasons than in previous case, $apply(\delta_{q^0}, \rho)(\alpha) = \mu_1(\alpha^1) \cdot apply(\delta_{q^1}, \rho^2)(\alpha^2) = 0$.

Case 2b) $\alpha^2 = q^1 a q^2$, i. e. $\alpha^{2\prime} = q^1$ and $\alpha' = \alpha^1$ and $\delta_{q^1}(\alpha^{2\prime}) = 1$

We have $apply(\delta_{q^1}, \rho^2)(\alpha^2) = p'_1(\alpha^2)$, while $apply(\delta_{q^0}, \rho)(\alpha) = p_1(\alpha) = \mu_1(\alpha^1) \cdot apply(\delta_{q^1}, \rho^2)(\alpha^2)$ which ends the basis.

Induction: We assume the result to be true for task schedule $\rho^{2\prime}$ and we show this also true for $\rho^2 = \rho^{2\prime}T$.

We note $\mu_{12\prime} = apply(\delta_{q^0}, \rho^1\rho^{2\prime})$ and $\mu_{2\prime} = apply(\delta_{q^1}, \rho^{2\prime})$.

By induction assumption, $\mu_{12\prime}(\alpha^1{}^\frown\alpha^2) = apply(\delta_{q^0}, \rho^1)(\alpha^1) \cdot apply(\delta_{q^1}, \rho^{2\prime})(\alpha^2) = \mu_1(\alpha^1) \cdot \mu_{2\prime}(\alpha^2)$

We have:

$apply(\delta_{q^0}, \rho)(\alpha) = apply(apply(\delta_{q^0}, \rho^1\rho^{2\prime}), T)(\alpha) = apply(\mu_{12\prime}, T)(\alpha) = p_1^i(\alpha) + p_2^i(\alpha)$ with

- $p_1^i(\alpha) = \begin{cases} \mu_{12\prime}(\alpha') \cdot \eta_{K,q',a}(q^2) & \text{if } a \text{ triggered by } T \text{ enabled at } q' \text{ with } \alpha = \alpha'{}^\frown q'aq^2 \\ 0 & \text{otherwise} \end{cases}$
- $p_2^i(\alpha) = \begin{cases} \mu_{12\prime}(\alpha) & \text{if } T \text{ is not enabled after } \alpha \\ 0 & \text{otherwise} \end{cases}$

and

$apply(\delta_{q^1}, \rho^2)(\alpha^2) = apply(apply(\delta_{q^1}, \rho^{2\prime}), T)(\alpha^2) = apply(\mu_{2\prime}, T)(\alpha^2) = p_1^j(\alpha^2) + p_2^j(\alpha^2)$ with

- $p_1^j(\alpha^2) = \begin{cases} \mu_{2\prime}(\alpha^{2\prime}) \cdot \eta_{K,q^{2\prime},a}(q^2) & \text{if } a \text{ triggered by } T \text{ enabled at } q^{2\prime} \text{ with } \alpha^2 = \alpha^{2\prime}{}^\frown q^{2\prime}aq^2 \\ 0 & \text{otherwise} \end{cases}$
- $p_2^j(\alpha^2) = \begin{cases} \mu_{2\prime}(\alpha^2) & \text{if } T \text{ is not enabled after } \alpha^2 \\ 0 & \text{otherwise} \end{cases}$

Case 1) T not enabled after α^2 then $apply(\delta_{q^1}, \rho^2)(\alpha^2) = p_2^j(\alpha^2) = \mu_{2\prime}(\alpha^2)$, while $apply(\delta_{q^0}, \rho)(\alpha) = p_2^i(\alpha) = \mu_{12\prime}(\alpha) = \mu_1(\alpha^1) \cdot \mu_{2\prime}(\alpha^2)$ by induction assumption which gives $apply(\delta_{q^0}, \rho)(\alpha) = \mu_1(\alpha^1) \cdot apply(\delta_{q^1}, \rho^2)(\alpha^2)$.

Case 2) T enabled after α^2

Case 2a) $\alpha^2 = q^2$ and so $apply(\delta_{q^0}, \rho)(\alpha) = 0 = apply(\delta_{q^1}, \rho^2)(\alpha^2)$ which gives $apply(\delta_{q^0}, \rho)(\alpha) = \mu_1(\alpha^1) \cdot apply(\delta_{q^1}, \rho^2)(\alpha^2)$

Case 2b) $\alpha^2 = \alpha^{2\prime}{}^\frown q^{2\prime}aq^2$. Here again $apply(\delta_{q^1}, \rho^2)(\alpha^2) = p_1^j(\alpha^2) = \mu_{2\prime}(\alpha^{2\prime})$, while $apply(\delta_{q^0}, \rho)(\alpha) = p_1^i(\alpha) = \mu_{12\prime}(\alpha') = \mu_1(\alpha^1) \cdot \mu_{2\prime}(\alpha')$ by induction assumption which gives $apply(\delta_{q^0}, \rho)(\alpha) = \mu_1(\alpha^1) \cdot apply(\delta_{q^1}, \rho^2)(\alpha^2)$.

This ends the induction for all the cases and so the proof.

□

This lemma can remind the Bayes law for independent events.

Reduction to class without creation Now we want to express a variant of the lemma 58, i. e. being able to express the measure of $\bigotimes_i^n \hat{\mathcal{C}}^{\vec{e}[i]}$ under ρ as a function of measures of the classes $\mathcal{C}^{\vec{e}[i]}$ under the different cuts $\tilde{\rho} =_\ell |\rho|_{\ell'}$. This result is stated in lemma 64 and is the main step to rescuing the monotonicity of dynamic creation/destruction with task-based implementation relationships.

The next definition introduce $\mathcal{A}$-brief partition which is a generalisation of the definition 159 of $\mathcal{A}$-decomposition. This definition facilitates the proof of both lemma 64 and theorem 24 of monotonicity of tenacious implementation.

Definition 181 ($\mathcal{A}$-brief-partition)**.** *Let $\mathcal{A}$ be a PSIOA, X be a PCA, and let ρ be a task-schedule. Let $\alpha \in frags(X)$. Let $\vec{\tilde{\alpha}} = (\tilde{\alpha}^{s^1}, \tilde{\alpha}^{s^2}, ...\tilde{\alpha}^{s^m}) = \mathcal{A}$-decomposition$(\alpha)$. A $\mathcal{A}$-brief-partition of α is a sequence $(\alpha^1, \alpha^2, ..., \alpha^n)$. s. t.*

- $\alpha = \alpha^1 \frown \alpha^2 \frown ... \frown \alpha^n$
- $\forall i \in [1, n], \exists !(\ell_i, h_i) \in [1, m]^2, \alpha^i = \tilde{\alpha}^{s^{\ell_i}} \frown ... \frown \tilde{\alpha}^{s^{h_i}}$
- $\forall i \in [1, n-1], \ell_{i+1} = h_i + 1$

Here, we prepare the lemma 64 by introducing a weaker version that states a total probability law for one unique cut.

Lemma 63 (Total probability law with one cut)**.** *Let $f_{(.,.)} = proj_{(.,.)}$ (a pasting friendly perception function). Let $\mathcal{A}$ be a PSIOA, X be a $\mathcal{A}$-conservative and $\mathcal{A}$-creation-explicit PCA and $\tilde{\mathcal{E}}$ partially-compatible with X. Let $\mathcal{E}_{\mathcal{A}} = \tilde{\mathcal{E}}||(X \setminus \{\mathcal{A}\})$. Let $\tilde{\zeta} \in \bigcup_{K, \tilde{\mathcal{E}} \in env(K)} range(f_{(\tilde{\mathcal{E}}, K)})$. Let $n \in \mathbb{N} \cup \{\infty\}$, let $\vec{\zeta} \in \vec{proxy}(\tilde{\zeta})_{(\tilde{\mathcal{E}}, X, \mathcal{A})}$ with $|\vec{\zeta}| = n$. Let $\hat{\mathcal{C}}^{\vec{\zeta}} = Class(\tilde{\mathcal{E}}, X, f^{\mathcal{A}, proxy} \circ \mathcal{A}\text{-decomposition}, \vec{\zeta}) \neq \emptyset$ and for each $i \in [1:n] \cap \mathbb{N}$, $\hat{\mathcal{C}}^{\vec{\zeta}[i]} = Class(\tilde{\mathcal{E}}^i, X^i, f^{\mathcal{A}, proxy}, \vec{\zeta}[i])$ as defined in lemma 55, that claims that $\hat{\mathcal{C}}^{\vec{\zeta}} = \bigotimes_{i=1}^{n} \hat{\mathcal{C}}^{\vec{\zeta}[i]}$*

Let $k \in [1:n] \cap \mathbb{N}$, $\hat{\mathcal{C}}_1 = \bigotimes_{i=1}^{k} \hat{\mathcal{C}}^{\vec{\zeta}[i]}$ and $\hat{\mathcal{C}}_2 = \bigotimes_{j=k+1}^{n} \hat{\mathcal{C}}^{\vec{\zeta}[j]}$ (trivially, $\hat{\mathcal{C}}^{\vec{\zeta}} = \hat{\mathcal{C}}_1 \otimes \hat{\mathcal{C}}_2$).

Let $(\alpha^1, \alpha^2) \in \hat{\mathcal{C}}_1 \times \hat{\mathcal{C}}_2$. We note $\underline{\alpha}^1 = \hat{\mathcal{C}}_1$, $\underline{\alpha}^2 = \hat{\mathcal{C}}_2$, $\alpha^{12} = \alpha^1 \frown \alpha^2$ and $\underline{\alpha}^{12} = \hat{\mathcal{C}}^{\vec{\zeta}}$.

Let ρ be a task-schedule.

$$apply_X(\delta_{fstate(\alpha^{12})}, \rho)(\underline{\alpha}^{12}) = \sum_{\ell_1}^{0 \prec \ell_1 \prec \ell_2} apply(\delta_{fstate(\alpha^1)}, \rho|_{\ell_1})(\underline{\alpha}^1_{(\ell_1, \rho)}) \cdot apply(\delta_{fstate(\alpha^2), (\ell_1+1)}|\rho)(\underline{\alpha}^2)$$

Proof. We note $\bar{q}^1 = fstate(\alpha^1) = fstate(\alpha^{12})$ and $\bar{q}^2 = fstate(\alpha^2)$. Furthermore, we note $\underline{\alpha}^1_+ = \{\alpha^{1,x} \in \underline{\alpha}^1 | \exists \ell_1^x \in ind(\rho, p), \alpha^{1,x}$ matches $\rho|_{\ell_1^x}\}$ and for every $\alpha^{1,x} \in \underline{\alpha}^1_+$, we note ℓ_1^x s. t. $\ell_1^x \in ind(\rho, p), \alpha^{1,x}$ matches $\rho|_{\ell_1^x}$.

$apply_X(\delta_{\bar{q}^1}, \rho)(\underline{\alpha}^{12}) = \sum_{\alpha^{12,w} \in \underline{\alpha}^{12}} apply(\delta_{\bar{q}^1}, \rho)(\alpha^{12,w}) = \sum_{\alpha^{1,u} \in \underline{\alpha}^1} \sum_{\alpha^{2,v} \in \underline{\alpha}^2} apply(\delta_{\bar{q}^1}, \rho)(\alpha^{1,u} \frown \alpha^{2,v})$ since $\underline{\alpha}^{12} = \underline{\alpha}^1 \otimes \underline{\alpha}^2$ by lemma 55.

Then by lemma 60 we obtain

$$apply_X(\delta_{\bar{q}^1}, \rho)(\underline{\alpha}^{12}) = \sum_{\ell_1 \in ind(p,\rho)} \sum_{\alpha^{1,x} \in \underline{\alpha}^1_{(\ell_1, \rho)}} \sum_{\alpha^{2,v} \in \underline{\alpha}^2} apply(\delta_{\bar{q}^1}, \rho)(\alpha^{1,x} \frown \alpha^{2,v}).$$

By definition $\forall \alpha^{1,x} \in \underline{\alpha}^1_{(\ell_1, \rho)}$, $\alpha^{1,x}$ matches $\rho|_{\ell_1}$.

Hence, we can apply lemma 62 to obtain

$$apply_X(\delta_{\bar{q}^1},\rho)(\underline{\alpha}^{12}) = \sum_{\ell_1\in ind(p,\rho)}\sum_{\alpha^{1,x}\in\underline{\alpha}^1_{(\ell_1,\rho)}}\sum_{\alpha^{2,v}\in\underline{\alpha}^2} apply(\delta_{\bar{q}^1},\rho|_{\ell_1})(\alpha^{1,x})\cdot apply(\delta_{\bar{q}^2,(\ell_1+1)}|\rho)(\alpha^{2,v})$$
$$= \sum_{\ell_1\in ind(p,\rho)}\sum_{\alpha^{1,x}\in\underline{\alpha}^1_{(\ell_1,\rho)}} apply(\delta_{\bar{q}^1},\rho|_{\ell_1})(\alpha^{1,x})\cdot apply(\delta_{\bar{q}^2,(\ell_1+1)}|\rho)(\underline{\alpha}^2)$$
$$= \sum_{\ell_1\in ind(p,\rho)}^{0\prec\ell_1\prec\ell_2}\sum_{\alpha^{1,x}\in\underline{\alpha}^1_{(\ell_1,\rho)}} apply(\delta_{\bar{q}^1},\rho|_{\ell_1})(\alpha^{1,x})\cdot apply(\delta_{\bar{q}^2,(\ell_1+1)}|\rho)(\underline{\alpha}^2)$$

because (*) $\alpha^{1,u}\in\underline{\alpha}^1$ cannot match λ since creation of $\mathcal{A}$ occurs at last action of $\alpha^{1,u}$ and so $|\alpha^{1,u}|\geq 1$ which implies $0\prec\ell_1$ and (**) $\alpha^{2,v}\in\underline{\alpha}^2$ cannot match λ since $|\alpha^{2,v}|\geq 1$ which implies $\ell_1\prec\ell_2$.

Finally $apply_X(\delta_{\bar{q}^1},\rho)(\underline{\alpha}^{12}) = \sum_{\ell_1\in ind(p,\rho)}^{0\prec\ell_1\prec\ell_2} apply(\delta_{\bar{q}^1},\rho|_{\ell_1})(\underline{\alpha}^1_{(\ell_1,\rho)})\cdot apply(\delta_{\bar{q}^2,(\ell_1+1)}|\rho)(\underline{\alpha}^2)$

□

Now, we can recursively use the previous lemma 63 to obtain a total probability law on all the possible cuts s. t. each chunk of the $\mathcal{A}$-brief partition matches one cut.

Lemma 64 (Total probability law with all the possible cuts)**.** *Let $f_{(.,.)}=proj_{(.,.)}$ (a pasting friendly perception function). Let $\mathcal{A}$ be a PSIOA, X be a $\mathcal{A}$-conservative and $\mathcal{A}$-creation-explicit PCA and $\tilde{\mathcal{E}}$ partially-compatible with X. Let $\mathcal{E}_{\mathcal{A}}=\tilde{\mathcal{E}}||(X\backslash\{\mathcal{A}\})$. Let $\tilde{\zeta}\in\bigcup_{K,\tilde{\mathcal{E}}\in env(K)} range(f_{(\tilde{\mathcal{E}},K)})$. Let $n'\in\mathbb{N}\cup\{\infty\}$, let $\vec{\zeta}\in\overrightarrow{proxy}(\tilde{\zeta})_{(\tilde{\mathcal{E}},X,\mathcal{A})}$ with $|\vec{\zeta}|=n'$. Let $\hat{\mathcal{C}}^{\vec{\zeta}}=Class(\tilde{\mathcal{E}},X,f^{\mathcal{A},proxy}\circ\mathcal{A}\text{-}decomposition,\vec{\zeta})\neq\emptyset$ and for each $i\in[1:n']\cap\mathbb{N}$, $\hat{\mathcal{C}}^{\vec{\zeta}[i]}=Class(\tilde{\mathcal{E}}^i,X^i,f^{\mathcal{A},proxy},\vec{\zeta}[i])$ as defined in lemma 55, that claims that $\hat{\mathcal{C}}^{\vec{\zeta}}=\bigotimes_{i=1}^{n'}\hat{\mathcal{C}}^{\vec{\zeta}[i]}$.*

Let $n\in\mathbb{N}$, $k_0=0$, $k_1,k_2,...,k_{n-1}\in[1:n']\cap\mathbb{N}$ and $k_n=n'$ with $k_0<k_1<k_2<...<k_{n-1}<k_n$. For every $j\in[1:n]$, we note $\hat{\mathcal{C}}_j=\bigotimes_{i=k_{j-1}+1}^{k_j}\hat{\mathcal{C}}^{\vec{\zeta}[i]}$. (Obviously, $\hat{\mathcal{C}}^{\vec{\zeta}}=\bigotimes_{j=1}^{n}\hat{\mathcal{C}}_j$)

Let $(\alpha^1,...,\alpha^n)\in\hat{\mathcal{C}}_1\times...\times\hat{\mathcal{C}}_n$. We note $\underline{\alpha}^j=\hat{\mathcal{C}}_j$ for each $j\in[1:n]$ and for every $j',j''\in[1:n]$, $\alpha^{(j',j'')}=\alpha^{j'}\frown...\frown\alpha^{j''}$ and $\underline{\alpha}^{(j',j'')}=\bigotimes_{j=j'}^{j''}\hat{\mathcal{C}}_j$.

Let ρ be a task-schedule. Let $\ell_n=max(ind(\rho,p))$ where p is any partition of ρ.

$apply_X(\delta_{fstate(\alpha^{(1,n)})},\rho)(\underline{\alpha}^{(1,n)}))=$

$$\sum_{\substack{\ell_1,\ell_2,...,\ell_{n-1}\\ 0\prec\ell^1\prec\ell^2\prec...\prec\ell^{n-1}\prec\ell_n}}\Gamma(\alpha^1,\ell^1,\rho)\cdot[\Pi_{i\in[2:n-1]}\Gamma'(\alpha^i,\ell^{i-1},\ell^i,\rho)]\cdot\Gamma''(\alpha^n,\ell^{n-1},\rho)$$

with

- $\Gamma(\alpha^1,\ell^1,\rho)=apply_X(\delta_{fstate(\alpha^1)},\rho|_{\ell_1})(\underline{\alpha}^1_{\ell^1,\rho})$,

– $\Gamma'(\alpha^i, \ell^{i-1}, \ell^i, \rho) = apply_X(\delta_{fstate(\alpha^i),(\ell_{i-1}+1)}\,|\rho|_{\ell_i})(\underline{\alpha}^i_{(\ell^{i-1},\ell^i,\rho)}))$ *and*

– $\Gamma''(\alpha^n, \ell^{n-1}, \rho) = apply_X(\delta_{fstate(\alpha^n),(\ell_{n-1}+1)}\,|\rho)(\underline{\alpha}^n)$

Proof. By induction on the size of the brief-partition. The basis is true by the previous lemma. We assume the predicate true for $n-1$ and we show this implies the predicate is true for integer n.

Let $(\alpha^1, ..., \alpha^{n-1}, \alpha^n)$ be a $\mathcal{A}$-brief-partition of $\alpha^{(1,n)}$.

We note $\alpha^{(1,n)} = \alpha^1{}^\frown\alpha^{(2,n)}$. $(\alpha^2, ..., \alpha^n)$ is clearly a $\mathcal{A}$-brief-partition of $\alpha^{(2,n)}$ of size $n-1$, $(\alpha^1, \alpha^{(2,n)})$ is a $\mathcal{A}$-brief-partition of $\alpha^{(1,n)}$ with size 2 lower or equal than n.

Now $apply_X(\delta_{fstate(\alpha)}, \rho)(\underline{\alpha}^{(1,n)}) =$

$$\sum_{\substack{\ell^1 \\ 0 \prec \ell^1 \prec \ell^n}} apply_X(\delta_{fstate(\alpha^1)}, \rho|_{\ell^1})(\underline{\alpha}^1_{(\ell^1,\rho)}) \cdot apply_X(\delta_{fstate(\alpha^{(2,n)})}, (\ell^1+1|\rho))(\underline{\alpha}^{(2,n)}))$$

(*) by induction hypothesis.

We note $\rho' =_{\ell^1+1} |\rho$, and reuse the induction hypothesis, which gives

$apply_X(\delta_{fstate(\alpha^{(2,n)})}, \rho')(\underline{\alpha}^{(2,n)})) =$

$$\sum_{\substack{\ell_2,...,\ell_{n-1} \\ 0 \prec \ell^2 \prec ... \prec \ell^{n-1} \prec \ell_n}} \Gamma(\alpha^2, \ell^2, \rho')[\Pi_{i\in[3:n-1]}\Gamma'(\alpha^i, \ell^{i-1}, \ell^i, \rho')]\Gamma''(\alpha^n, \ell^{n-1}, \rho')$$

For every ℓ^i s. t. $\ell^1 \prec \ell^i, {}_{\ell^i+1}|\rho' =_{\ell^i+1} |(_{\ell^1+1}|\rho) =_{\ell^i+1} |\rho$, while for every $\ell^2 \preceq \ell^1, \Gamma(\alpha^2, \ell^2, \rho') = 0$ thus

$apply_X(\delta_{fstate(\alpha^{(2,n)})}, \rho')(\underline{\alpha}^{(2,n)})) =$

$$\sum_{\substack{\ell_2,...,\ell_{n-1} \\ \ell^1 \prec \ell^2 \prec ... \prec \ell^{n-1} \prec \ell_n}} \Gamma'(\alpha^2, \ell^1, \ell^2, \rho)[\Pi_{i\in[3:n-1]}\Gamma'(\alpha^i, \ell^{i-1}, \ell^i, \rho)]\Gamma''(\alpha^n, \ell^{n-1}, \rho)$$

(**)

We compose the last two results (*) and (**) to obtain

$apply_X(\delta_{fstate(\alpha^{(1,n)})}, \rho)(\underline{\alpha}^{(1,n)})) =$

$$\sum_{\substack{\ell_1,\ell_2,...,\ell_{n-1} \\ 0 \prec \ell^1 \prec \ell^2 \prec ... \prec \ell^{n-1} \prec \ell_n}} \Gamma(\alpha^1, \ell^1, \rho) \cdot [\Pi_{i\in[2:n-1]}\Gamma'(\alpha^i, \ell^{i-1}, \ell^i, \rho)] \cdot \Gamma''(\alpha^n, \ell^{n-1}, \rho)$$

, which is the desired result.

□

This lemma 64 is very valuable since it will allow us to reduce the problem to classes without creation before the last action.

4.7.4. Preparation before monotonicity

Before starting the proof of monotonicity, we state two easy intermediate results.

The first one states that $\mathcal{AB}$-environment-corresponding schedules are also $X_{\mathcal{A}}X_{\mathcal{B}}$-environment-corresponding schedules for $X_{\mathcal{A}}$, $X_{\mathcal{B}}$ corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$. This is important to use pairs of balanced $\mathcal{AB}$-environment-corresponding schedules as essential elements of the implementation of $X_{\mathcal{B}}$ by $X_{\mathcal{A}}$.

Lemma 65 (Corresponding-environment relation is preserved in the upper level)**.** *(see figure 4.17)*

Let $\mathcal{A}$, $\mathcal{B}$ be PSIOA. Let $X_{\mathcal{A}}$, $X_{\mathcal{B}}$ be PCA corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$. Let ρ, ρ' be $\mathcal{AB}$-environment-corresponding schedules. ρ, ρ' are also $X_{\mathcal{A}}X_{\mathcal{B}}$-environment-corresponding schedules.

Proof. We note $Y_{\mathcal{A}} = X_{\mathcal{A}} \setminus \{\mathcal{A}\}$ and $Y_{\mathcal{B}} = X_{\mathcal{B}} \setminus \{\mathcal{B}\}$. We note $U_Y = UA(Y_{\mathcal{A}}) = UA(Y_{\mathcal{B}}) = UA(X_{\mathcal{A}}) \setminus \{\mathcal{A}\} = UA(X_{\mathcal{B}}) \setminus \{\mathcal{B}\}$, $U_{\mathcal{E}}$ the set of ids not in the ids of U_Y and $U_{\mathcal{E}'}$ the set of ids different that the ones of $\mathcal{A}$ and $\mathcal{B}$. We note $(\rho^1_{\mathcal{A}}, \rho^2_{\mathcal{E}}, \rho^3_{\mathcal{A}}, \rho^4_{\mathcal{E}}, ..., \rho^{2k}_{\mathcal{E}}, \rho^{2k+1}_{\mathcal{A}}, ...)$ the $\mathcal{A}$-partition of ρ and $(\rho^1_{\mathcal{B}}, \rho^2_{\mathcal{E}}, \rho^3_{\mathcal{B}}, \rho^4_{\mathcal{E}}, ..., \rho^{2k}_{\mathcal{E}}, \rho^{2k+1}_{\mathcal{B}}, ...)$ the $\mathcal{B}$-partition of ρ'.

We partition each sub-schedule $\rho^{2k}_{\mathcal{E}}$ into tasks with id in U_Y and tasks with id not in U_Y, i. e. in $U_{\mathcal{E}}$. Hence, $\rho^{2k}_{\mathcal{E}} = \rho^{2k,1}_{\mathcal{E}'}, \rho^{2k,2}_{Y}, ..., \rho^{2k,2i-1}_{\mathcal{E}'}, \rho^{2k,2i}_{Y}, ..., \rho^{2k,2\ell_k-1}_{\mathcal{E}'}, \rho^{2k,2\ell_k}_{Y}$ where empty sequence is allowed for $\rho^{2k,1}_{\mathcal{E}'}$ and $\rho^{2k,2\ell_k}_{Y}$.

So if we note $(\rho^1_{X_{\mathcal{A}}}, \rho^2_{\mathcal{E}'}, \rho^3_{X_{\mathcal{A}}}, \rho^4_{\mathcal{E}'}, ..., \rho^{2q}_{\mathcal{E}'}, \rho^{2q+1}_{X_{\mathcal{A}}}, ...)$ the $X_{\mathcal{A}}$-partition of ρ and $(\rho^1_{X_{\mathcal{B}}}, \rho'^2_{\mathcal{E}'}, \rho^3_{X_{\mathcal{B}}}, \rho'^4_{\mathcal{E}'}, ..., \rho'^{2q}_{\mathcal{E}'}, \rho^{2q+1}_{X_{\mathcal{B}}}, ...)$ the $X_{\mathcal{B}}$-partition of ρ', for every j it exists (k, i) s. t. $\rho^{2j}_{\mathcal{E}'} = \rho'^{2j}_{\mathcal{E}'} = \rho^{2k,2i-1}_{\mathcal{E}'}$.

We can remark that each $\rho^{2q+1}_{X_{\mathcal{A}}}$ is either $\rho^{2k,2i}_{Y}$ for some pair (k, i) or $\rho^{2k,2\ell_k}_{Y}\rho^{2k+1}_{\mathcal{A}}$ for some k or $\rho^{2k,2\ell_k}_{Y}\rho^{2k+1}_{\mathcal{A}}\rho^{2k+2,2}_{Y}$ if $\rho^{2k+2,1}_{\mathcal{E}'}$ is an empty sequence and similarly for $\rho^{2q+1}_{X_{\mathcal{B}}}$.

□

The second lemma of this subsection is a general manipulation of sum and products that will naturally occur when we will compute the difference between the probability of two corresponding classes that can be expressed by the sum of products of elementary probabilities.

Lemma 66 (difference of products)**.** *Let $n \in \mathbb{N}$, $(X_i)_{i\in[1,n]} \in \mathbb{R}^n$, $(Y_i)_{i\in[1,n]} \in \mathbb{R}^n$. $\Pi_{i\in[1,n]}X_i - \Pi_{i\in[1,n]}Y_i = \sum_{k\in[1,n]} (\Pi_{i\in[1,k-1]}X_i)\cdot(X_k - Y_k)\cdot(\Pi_{j\in[k+1,n]}Y_j)$ (with the convention $\Pi_{i\in[1,0]}X_i = \Pi_{j\in[n+1,n]}Y_j = 1$)*

Proof. The case $n = 1$ is immediate. Idem for $n = 2$. By induction.

$\Pi_{i\in[1,n+1]}X_i - \Pi_{i\in[1,n+1]}Y_i = \Pi_{i\in[1,n]}X'_i - \Pi_{i\in[1,n]}Y'_i$ where $\forall i \in [1, n-1] X'_i = X_i$ and $Y'_i = Y_i$ while $X'_n = X_n \cdot X_{n+1}$ and $Y'_{n+1} = Y_n \cdot Y_{n+1}$. By induction assumption, $\Pi_{i\in[1,n]}X'_i - \Pi_{i\in[1,n]}Y'_i =$

$$\sum_{k\in[1,n]} (\Pi_{i\in[1,k-1]}X'_i) \cdot (X'_k - Y'_k) \cdot (\Pi_{j\in[k+1,n]}Y'_j) =$$

$$[\sum_{k\in[1,n-1]} (\Pi_{i\in[1,k-1]}X'_i) \cdot (X'_k - Y'_k) \cdot (\Pi_{j\in[k+1,n]}Y'_j)] + [(\Pi_{i\in[1,n-1]}X'_i) \cdot (X'_n - Y'_n)] =$$

$$\sum_{k\in[1,n-1]} (\Pi_{i\in[1,k-1]}X_i) \cdot (X_k - Y_k) \cdot (\Pi_{j\in[k+1,n+1]}Y_j] + (\Pi_{i\in[1,n-1]}X_i) \cdot ((X_n \cdot X_{n+1}) - (Y_n \cdot Y_{n+1})) =$$

$[\sum_{k\in[1,n-1]}(\Pi_{i\in[1,k-1]}X_i)\cdot(X_k-Y_k)\cdot(\Pi_{j\in[k+1,n+1]}Y_j)]+(\Pi_{i\in[1,n-1]}X_i)\cdot((X_n-Y_n)\cdot Y_{n+1})+X_n\cdot(X_{n+1}-Y_{n+1}))=$

$\sum_{k\in[1,n+1]}(\Pi_{i\in[1,k-1]}X_i)\cdot(X_k-Y_k)\cdot(\Pi_{j\in[k+1,n]}Y_j)$

□

4.7.5. Tenacious Implementation Monotonicity

After the proof of lemma 64, we are ready to obtain the theorem 24 of monotonicity of tenacious implementation. The main step is the next lemma 67 that states that the good balance of two task-schedule for two automata $\mathcal{A}$ and $\mathcal{B}$ is preserved in the upper level for PCA corresponding w. r. t. $\mathcal{A}$, $\mathcal{B}$

Lemma 67 (monotonocity of dynamic creation/destruction with tenacious implementation: main step)**.** *Let $\mathcal{A}$, $\mathcal{B}$ be two PSIOA. Let ρ, ρ' be task-schedules s.t. $\rho S^{ten,\leq 0}_{(\mathcal{A},\mathcal{B})}\rho'$. Let $X_\mathcal{A}$, $X_\mathcal{B}$ be PCA corresponding w.r.t. $\mathcal{A}$, $\mathcal{B}$. Then $\rho S^{ten,\leq 0}_{(X_\mathcal{A},X_\mathcal{B})}\rho'$*

Proof. First, because of lemma 65, ρ and ρ' are also $X_\mathcal{A}X_\mathcal{B}$-environment-corresponding.

Let $f_{(.,.)}=proj_{(.,.)}$ (a pasting friendly perception function). Let $\tilde{\mathcal{E}}\in env(X_\mathcal{A})\cap env(X_\mathcal{B})$. Let $\mathcal{E}_\mathcal{A}=\tilde{\mathcal{E}}||(X\setminus\{\mathcal{A}\})$ and $\mathcal{E}_\mathcal{B}=\tilde{\mathcal{E}}||(X\setminus\{\mathcal{B}\})$. Let $\mathcal{E}\in env(\mathcal{A})\cap env(\mathcal{B})$ semantically equivalent to both $\mathcal{E}_\mathcal{A}$ and $\mathcal{E}_\mathcal{B}$.

Let $\tilde{e}\in\bigcup_{K,\tilde{\mathcal{E}}\in env(K)} range(f_{(\tilde{\mathcal{E}},K)})$. Let $n\in\mathbb{N}\cup\{\infty\}$, let $\vec{e}\in \vec{proxy}(\tilde{e})_{(\tilde{\mathcal{E}},X_\mathcal{A},\mathcal{A})}\cup\vec{proxy}(\tilde{e})_{(\tilde{\mathcal{E}},X_\mathcal{B},\mathcal{B})}$ with $|\vec{e}|=n$.

Let $\hat{\mathcal{C}}^{\vec{e}}_\mathcal{A}=Class(\tilde{\mathcal{E}},X_\mathcal{A},f^{\mathcal{A},proxy}\circ\mathcal{A}\text{-decomposition},\vec{e})\neq\emptyset$ and for each $i\in[1:n]\cap\mathbb{N}$, $\hat{\mathcal{C}}^{\vec{\zeta}[i]}_\mathcal{A}=Class(\tilde{\mathcal{E}}^i,X^i_\mathcal{A},f^{\mathcal{A},proxy},\vec{e}[i])$ as defined in lemma 55, that claims that $\hat{\mathcal{C}}^{\vec{\zeta}}_\mathcal{A}=\bigotimes_{i=1}^{n}\hat{\mathcal{C}}^{\vec{\zeta}[i]}_\mathcal{A}$.

Let $\hat{\mathcal{C}}^{\vec{e}}_\mathcal{B}=Class(\tilde{\mathcal{E}},X_\mathcal{B},f^{\mathcal{B},proxy}\circ\mathcal{A}\text{-decomposition},\vec{e})\neq\emptyset$ and for each $i\in[1:n]\cap\mathbb{N}$, $\hat{\mathcal{C}}^{\vec{\zeta}[i]}_\mathcal{B}=Class(\tilde{\mathcal{E}}^i,X^i_\mathcal{B},f^{\mathcal{B},proxy},\vec{e}[i])$ as defined in lemma 55, that claims that $\hat{\mathcal{C}}^{\vec{\zeta}}_\mathcal{B}=\bigotimes_{i=1}^{n}\hat{\mathcal{C}}^{\vec{\zeta}[i]}_\mathcal{B}$.

Let $(\alpha^1,...,\alpha^j,...)\in\hat{\mathcal{C}}^{\vec{e}[1]}_\mathcal{A}\times...\times\hat{\mathcal{C}}^{\vec{e}[j]}_\mathcal{A}\times....$ Let $(\pi^1,...,\pi^j,...)\in\hat{\mathcal{C}}^{\vec{e}[1]}_\mathcal{B}\times...\times\hat{\mathcal{C}}^{\vec{e}[j]}_\mathcal{B}\times....$ We note $\underline{\alpha}^j=\hat{\mathcal{C}}^{\vec{e}[j]}_\mathcal{A}$ and $\underline{\pi}^j=\hat{\mathcal{C}}^{\vec{e}[j]}_\mathcal{B}$ for each $j\in[1:n]$. We note $\alpha=\alpha^1{}^\frown...{}^\frown\alpha^j{}^\frown...$, $\pi=\pi^1{}^\frown...{}^\frown\pi^j{}^\frown...$, $\underline{\alpha}=\hat{\mathcal{C}}^{\vec{e}}_\mathcal{A}$ and $\underline{\pi}=\hat{\mathcal{C}}^{\vec{e}}_\mathcal{B}$

Let $\tilde{\mathcal{W}}_\mathcal{A}=X_\mathcal{A}||\tilde{\mathcal{E}}$ and $\tilde{\mathcal{W}}_\mathcal{B}=X_\mathcal{B}||\tilde{\mathcal{E}}$.

We note p the $\mathcal{A}$-partition of ρ and p' the $\mathcal{B}$-partition of ρ'. In the remaining, we note for every $\ell^1,\ell^2\in ind(\rho,p)$

- $\Gamma_{\tilde{\mathcal{W}}_\mathcal{A}}(\alpha,\ell^1,\rho)=apply_{\tilde{\mathcal{W}}_\mathcal{A}}(\delta_{fstate(\alpha^1)},\rho|_{\ell^1})(\underline{\alpha}^1_{\ell^1,\rho})$,
- $\Gamma'_{\tilde{\mathcal{W}}_\mathcal{A}}(\alpha,\ell^1,\ell^2,\rho)=apply_{\tilde{\mathcal{W}}_\mathcal{A}}(\delta_{fstate(\alpha),(\ell^1+1)}|\rho|_{\ell^2})(\underline{\alpha}_{(\ell^1,\ell^2,\rho)}))$ and
- $\Gamma''_{\tilde{\mathcal{W}}_\mathcal{A}}(\alpha,\ell^1,\rho)=apply_{\tilde{\mathcal{W}}_\mathcal{A}}(\delta_{fstate(\alpha),(\ell^1+1)}|\rho)(\underline{\alpha})$

and for every $\ell^1, \ell^2 \in ind(\rho', p')$

- $\Gamma_{\tilde{\mathcal{W}}_{\mathcal{B}}}(\pi, \ell^1, \rho') = apply_{\tilde{\mathcal{W}}_{\mathcal{B}}}(\delta_{fstate(\pi^1)}, \rho'|_{\ell^1})(\underline{\pi}^1_{\ell^1,\rho'})$,
- $\Gamma'_{\tilde{\mathcal{W}}_{\mathcal{B}}}(\pi, \ell^1, \ell^2, \rho') = apply_{\tilde{\mathcal{W}}_{\mathcal{B}}}(\delta_{fstate(\pi),(\ell^1+1)}\,|\rho'|_{\ell^2})(\underline{\pi}_{(\ell^1,\ell^2,\rho')}))$ and
- $\Gamma''_{\tilde{\mathcal{W}}_{\mathcal{B}}}(\pi, \ell^1, \rho') = apply_{\tilde{\mathcal{W}}_{\mathcal{B}}}(\delta_{fstate(\pi),(\ell^1+1)}\,|\rho')(\underline{\pi})$

We note $\mathcal{I} = ind(\rho, p) \cap ind(\rho', p') \cap (2\mathbb{N} \times \mathbb{N})$. We fix $\ell, \ell' \in \mathcal{I}$ and $\tilde{\rho} =_\ell |\rho|_{\ell'}$ and $\tilde{\rho}' =_\ell |\rho'|_{\ell'}$.

We need to evaluate $\Delta_{\underrightarrow{e}} = apply_{\tilde{\mathcal{W}}_{\mathcal{A}}}(\delta_{fstate(\alpha)}, \tilde{\rho})(\underline{\alpha}) - apply_{\tilde{\mathcal{W}}_{\mathcal{B}}}(\delta_{fstate(\pi)}, \tilde{\rho}')(\underline{\pi})$.

We have two cases: $n = 1$ and $n > 1$. We treat only $n > 1$ since the former one is trivial.

Since for every $i \in [1, n-1]$, α^i (resp. π^i) ends on $\mathcal{A}$ (resp. $\mathcal{B}$) creation, the last action of α^i is necessarily triggerd by a task $T = \tilde{\rho}[\ell_i]$ with even index $\ell_i \in ind(\rho, p) \cap (2\mathbb{N} \times \mathbb{N})$, that is α^i (resp. π^i) cannot match $\tilde{\rho}|_{\ell_i}$ (resp. $\tilde{\rho'}|_\ell$) if $\ell_i = (2k_i + 1, q_i)$.

We note $\ell_0 + 1$ the first index of $\tilde{\rho}$ and $\tilde{\rho}'$ and ℓ_n the last index of $\tilde{\rho}$ and $\tilde{\rho}'$

We apply lemma 64, to obtain

- $apply(\delta_{fstate(\alpha)}, \tilde{\rho})(\underline{\alpha}) = \sum_{\underline{\ell}^{(1,n-1)} \in \underline{L}^{(1,n-1)}_{(\ell_0,\ell_n)}} \Pi_{i\in[1:n]} \tilde{\Gamma}_{\tilde{\mathcal{W}}_{\mathcal{A}}}(\alpha^i, \ell_{i-1}, \ell_i, \tilde{\rho})$
- $apply(\delta_{fstate(\pi)}, \tilde{\rho}')(\underline{\pi}) = \sum_{\underline{\ell}^{(1,n-1)} \in \underline{L}'^{(1,n-1)}_{(\ell_0,\ell_n)}} \Pi_{i\in[1:n]} \tilde{\Gamma}_{\tilde{\mathcal{W}}_{\mathcal{B}}}(\pi^i, \ell_{i-1}, \ell_i, \tilde{\rho}')$

where

- $\underline{L}^{(u,v)}_{(\ell_x,\ell_y)} = \{\underline{\ell} = (\ell_u, ..., \ell_v) \in ind(\rho, p)^{(v-u+1)} | \ell_x \prec \ell_u \prec \ell_{u+1} \prec ... \prec \ell_v \prec \ell_y\}$
- $\underline{L}'^{(u,v)}_{(\ell_x,\ell_y)} = \{\underline{\ell} = (\ell_u, ..., \ell_v) \in ind(\rho', p')^{(v-u+1)} | \ell_x \prec \ell_u \prec \ell_{u+1} \prec ... \prec \ell_v \prec \ell_y\}$
- $\forall i \in [1, n-1], \tilde{\Gamma}(\alpha^i, \ell_{i-1}, \ell_i, \tilde{\rho}) = apply_{\tilde{\mathcal{W}}_{\mathcal{A}}}(\delta_{fstate(\alpha^i),(\ell_{i-1}+1)}\,|\tilde{\rho}|_{\ell_i})(\underline{\alpha}^i_{(\ell^{i-1}+1,\ell^i,\tilde{\rho})}))$
- $\forall i \in [1, n-1], \tilde{\Gamma}(\pi^i, \ell_{i-1}, \ell_i, \tilde{\rho}') = apply_{\tilde{\mathcal{W}}_{\mathcal{B}}}(\delta_{fstate(\pi^i),(\ell_{i-1}+1)}\,|\tilde{\rho}'|_{\ell_i})(\underline{\pi}^i_{(\ell^{i-1}+1,\ell^i,\tilde{\rho}')}))$
- $\tilde{\Gamma}_{\tilde{\mathcal{W}}_{\mathcal{A}}}(\alpha^n, \ell_{n-1}, \ell_n, \tilde{\rho}) = apply_{\tilde{\mathcal{W}}_{\mathcal{A}}}(\delta_{fstate(\alpha^n),(\ell_{n-1}+1)}\,|\tilde{\rho}|_{\ell_n})(\underline{\alpha}^n)$
- $\tilde{\Gamma}_{\tilde{\mathcal{W}}_{\mathcal{B}}}(\pi^n, \ell_{n-1}, \ell_n, \tilde{\rho}') = apply_{\tilde{\mathcal{W}}_{\mathcal{B}}}(\delta_{fstate(\pi^n),(\ell_{n-1}+1)}\,|\tilde{\rho}'|_{\ell_n})(\underline{\pi}^n)$

Let $\underline{\ell} \in \underline{L}^{(1,n-1)}_{(\ell_0,\ell_n)}$ (resp. $\underline{\ell}' \in \underline{L}'^{(1,n-1)}_{(\ell_0,\ell_n)}$). Let $k \in [1 : n-1]$. Since α^k (resp. π^k) ends on $\mathcal{A}$ (resp. $\mathcal{B}$) creation, the last action of α^k (resp. π^k) can be triggered only by a task $T = \tilde{\rho}[\ell]$ (resp. $T = \tilde{\rho}'[\ell]$) s. t. $\ell \in (2\mathbb{N} \times \mathbb{N})$. Let fix $\ell_{k-1} \in ind(p, \tilde{\rho})$ (resp. $\ell_{k-1} \in ind(p', \tilde{\rho}')$. For every $k \in [1, n-1]$, for every $\ell^k \in ind(\rho, p) \cap (2\mathbb{N}+1, \mathbb{N})$ (resp. $\ell^k \in ind(\rho', p') \cap (2\mathbb{N}+1, \mathbb{N})$), $\tilde{\Gamma}_{\tilde{\mathcal{W}}_{\mathcal{A}}}(\alpha^k, \ell^{k-1}, \ell^k, \tilde{\rho}) = 0$ (resp. $\tilde{\Gamma}_{\tilde{\mathcal{W}}_{\mathcal{B}}}(\pi^k, \ell^{k-1}, \ell^k, \tilde{\rho}') = 0$).

Hence, the previous result can be written

- $apply(\delta_{fstate(\alpha)}, \tilde{\rho})(\underline{\alpha}) = \sum_{\underline{\ell}^{(1,n-1)} \in \underline{L}^{*(1,n-1)}_{(\ell_0,\ell_n)}} \Pi_{i\in[1:n]} \tilde{\Gamma}_{\tilde{\mathcal{W}}_{\mathcal{A}}}(\alpha^i, \ell_{i-1}, \ell_i, \tilde{\rho})$
- $apply(\delta_{fstate(\pi)}, \tilde{\rho}')(\underline{\pi}) = \sum_{\underline{\ell}^{(1,n-1)} \in \underline{L}^{*(1,n-1)}_{(\ell_0,\ell_n)}} \Pi_{i\in[1:n]} \tilde{\Gamma}_{\tilde{\mathcal{W}}_{\mathcal{B}}}(\pi^i, \ell_{i-1}, \ell_i, \tilde{\rho}')$

where

- $\underline{L}^{*(u,v)}_{(\ell_x,\ell_y)} = \{\underline{\ell} = (\ell_u, ..., \ell_v) \in \mathcal{I}^{(v-u+1)} | \ell_x \prec \ell_u \prec \ell_{u+1} \prec ... \prec \ell_v \prec \ell_y\}$

We will apply lemma 66

- $\forall k \in [1,n]$, $\forall y \geq k-1$, $\forall v = y-1$,$\forall \underline{\ell} \in \underline{L}^{*(1,v)}_{\underline{=}(\ell_0,\ell_y)}$, we note
  $P^{k-1}(\underline{\ell}) = \Pi_{i\in[1,k-1]}\tilde{\Gamma}_{\tilde{\mathcal{W}}_{\mathcal{A}}}(\alpha^i, \ell^{i-1}, \ell^i, \tilde{\rho})$ (with the convention $P^0 = 1$)
- $\forall k \in [1,n]$, $\forall x \leq k$, $\forall u = x+1$, $\forall \underline{\ell} \in \underline{L}^{*(u,n-1)}_{\underline{=}(\ell_x,\ell_n)}$, we note
  $Q^{k+1}(\underline{\ell}) = \Pi_{j\in[k+1,n]}\tilde{\Gamma}_{\tilde{\mathcal{W}}_{\mathcal{B}}}(\pi^j, \ell^{j-1}, \ell^j, \tilde{\rho})$ (with the convention $Q^{n+1} = 1$)
- $\forall k \in [1,n]$, $\forall x \leq k-1$, $\forall y \geq k$, $\forall u = x+1$, $\forall v = y-1$, $\forall \underline{\ell} \in \underline{L}^{*(u,v)}_{\underline{=}(\ell_x,\ell_y)}$, we note
  $\Delta^{\Gamma}(k, \underline{\ell}) = (\tilde{\Gamma}_{\tilde{\mathcal{W}}_{\mathcal{A}}}(\alpha^k, \ell^{k-1}, \ell^k, \tilde{\rho}) - \tilde{\Gamma}_{\tilde{\mathcal{W}}_{\mathcal{B}}}(\pi^k, \ell^{k-1}, \ell^k, \tilde{\rho}'))$

We apply the lemma 66 of difference of products to obtain

$$\Delta^{\rightarrow}_{e} = \sum_{\underline{\ell}\in\underline{L}^{*(1,n-1)}_{\underline{=}(\ell_0,\ell_n)}} \sum_{k\in[1,n]} P^{k-1}(\underline{\ell}) \cdot \Delta^{\Gamma}(k,\underline{\ell}) \cdot Q^{k+1}(\underline{\ell}) =$$

$$= \sum_{k\in[1,n]} \sum_{\underline{\ell}^{(k-1,k)}\in\underline{L}^{*(k-1,k)}_{\underline{=}(\ell_0,\ell_n)}} \sum_{\underline{\ell}^{(1,k-2)}\in\underline{L}^{*(1,k-2)}_{\underline{=}(\ell_0,\ell_{k-1})}} \sum_{\underline{\ell}^{(k+1,n-1)}\in\underline{L}^{*(k+1,n-1)}_{\underline{=}(\ell_k,\ell_n)}} P^{k-1}(\underline{\ell}^{(1,k-2)})\cdot\Delta^{\Gamma}(k,\underline{\ell}^{(k-1,k)})\cdot Q^{k+1}(\underline{\ell}^{(k+1,n-1)})$$

and so

$$\Delta^{\rightarrow}_{e} = \sum_{k\in[1,n]} [\sum_{\underline{\ell}^{(k-1,k)}\in\underline{L}^{*(k-1,k)}_{\underline{=}(\ell_0,\ell_n)}} \Delta^{\Gamma}(k,\underline{\ell}^{(k-1,k)})] \cdot \mathfrak{P}(k) \cdot \mathfrak{Q}(k) \tag{4.1}$$

with $\mathfrak{P}(k) = [\sum_{\underline{\ell}^{(1,k-2)}\in\underline{L}^{*(1,k-2)}_{\underline{=}(\ell_0,\ell_{k-1})}} P^{k-1}(\underline{\ell}^{(1,k-2)})]$ and $\mathfrak{Q}(k) = [\sum_{\underline{\ell}^{(k+1,n-1)}\in\underline{L}^{*(k+1,n-1)}_{\underline{=}(\ell_k,\ell_n)}} Q^{k+1}(\underline{\ell}^{(k+1,n-1)})]$

Even if it is not crucial, we can remark that:

- $\mathfrak{P}(k) = apply(\delta_{fstate(\alpha),\ell_0+1}\,|\tilde{\rho}|_{\ell_{k-1}})(\underline{\alpha}^{(1,k-1)}_{\ell_0+1,\ell_{k-1},\tilde{\rho}}) \leq 1$
- $\mathfrak{Q}(k) = apply(\delta_{fstate(\pi^{k+1}),\ell_k+1}\,|\tilde{\rho}|_{\ell_n})(\underline{\alpha}^{(k+1,n)}) \leq 1$

This remark might be relevant for an adaptation to approximate version of the lemma.

Now we will show that $\forall k \in [1,n]$, $\sum_{\underline{\ell}^{(k-1,k)}\in\underline{L}^{*(k-1,k)}_{\underline{=}(\ell_0,\ell_n)}} \Delta^{\Gamma}(k,\underline{\ell}^{(k-1,k)}) = 0$

We begin with $k \in [1, n-1]$.

$$[\sum_{\underline{\ell}^{(k-1,k)}\in\underline{L}^{*(k-1,k)}_{\underline{=}(\ell_0,\ell_n)}} (\tilde{\Gamma}_{\tilde{\mathcal{W}}_{\mathcal{A}}}(\alpha^k, \ell^{k-1}, \ell^k, \tilde{\rho}) - \tilde{\Gamma}_{\tilde{\mathcal{W}}_{\mathcal{B}}}(\pi^k, \ell^{k-1}, \ell^k, \tilde{\rho}'))] =$$

$$[\sum_{\ell_{k-1}\in\mathcal{I}} (\sum_{\ell\in ind(\rho,p)}^{\ell_{k-1}\prec\ell} \tilde{\Gamma}_{\tilde{\mathcal{W}}_{\mathcal{A}}}(\alpha^k, \ell^{k-1}, \ell, \tilde{\rho}) - \sum_{\ell'\in ind(\rho',p')}^{\ell_{k-1}\prec\ell'} \tilde{\Gamma}_{\tilde{\mathcal{W}}_{\mathcal{B}}}(\pi^k, \ell^{k-1}, \ell', \tilde{\rho}'))] =$$

$$[\sum_{\ell_{k-1}\in\mathcal{I}} ((apply_{\tilde{W}_{\mathcal{A}}}(\delta_{fstate(\alpha^k)}, \rho)(\underline{\alpha}^k) - apply_{\tilde{W}_{\mathcal{A}}}(\delta_{fstate(\alpha^k)}, \rho|_{(\ell_{k-1})})(\underline{\alpha}^k)) - (apply_{\tilde{W}_{\mathcal{B}}}(\delta_{fstate(\pi^k)}, \rho')(\underline{\pi}^k) -$$
$$apply_{\tilde{W}_{\mathcal{B}}}(\delta_{fstate(\pi^k)}, \rho'|_{(\ell_{k-1})})(\underline{\pi}^k))] \text{ by lemma 61.}$$

Thus $\sum_{\underline{\ell}^{(k-1,k)}\in\underline{L}^{*(k-1,k)}_{\underline{=}(\ell_0,\ell_n)}} \Delta^{\Gamma}(k,\underline{\ell}^{(k-1,k)}) = \sum_{\ell_{k-1}\in\mathcal{I}} \Delta_{app}(k) - \Delta'_{app}(k,\ell_{k-1})$, with

– $\Delta_{app}(k) = apply_{\tilde{W}_{\mathcal{A}}}(\delta_{fstate(\alpha^k)}, \rho)(\underline{\alpha}^k) - apply_{\tilde{W}_{\mathcal{B}}}(\delta_{fstate(\pi^k)}, \rho')(\underline{\pi}^k)$, and

– $\Delta'_{app}(k, \ell) = apply_{\tilde{W}_{\mathcal{A}}}(\delta_{fstate(\alpha^k)}, \rho|\ell)(\underline{\alpha}^k) - apply_{\tilde{W}_{\mathcal{B}}}(\delta_{fstate(\pi^k)}, \rho'|\ell)(\underline{\pi}^k)$

In the case $k = n$, $\sum_{\underline{\ell}^{(k-1,k)} \in \underline{L}^{*(k-1,k)}_{=(\ell_0,\ell_n)}} \Delta^{\Gamma}(k, \underline{\ell}^{(k-1,k)}) = \sum_{\ell^{n-1} \in \mathcal{I}} \Delta''_{app}(\ell^{n-1})$

with $\Delta''_{app}(\ell) = apply_{\tilde{\mathcal{W}}_{\mathcal{A}}}(\delta_{fstate(\alpha^n),(\ell+1)}\,|\tilde{\rho})(\underline{\alpha}^n) - apply_{\tilde{\mathcal{W}}_{\mathcal{B}}}(\delta_{fstate(\pi^n),(\ell+1)}\,|\tilde{\rho}')(\underline{\pi}^n)$.

It remains to show that $\forall \ell \in \mathcal{I}$, $\forall k \in [1, n-1]$, $\Delta_{app}(k) = \Delta'_{app}(k, \ell) = \Delta''_{app}(\ell) = 0$.

This comes from lemma 50 since (1) $\rho S^{ten,\leq 0}_{(\mathcal{A},\mathcal{B})} \rho'$ (2) A task-scheduler is an alter-ego for himself from $\tilde{\mathcal{W}}_{\mathcal{A}}$ to $\mathcal{E}_{\mathcal{A}}||\tilde{\mathcal{A}}^{sw}$ (resp. from $\tilde{\mathcal{W}}_{\mathcal{B}}$ to $\mathcal{E}_{\mathcal{B}}||\tilde{\mathcal{B}}^{sw}$) (3) The executions-matching holds since we consider executions without creation before last action (4) $\mathcal{E}_{\mathcal{A}}$ and $\mathcal{E}_{\mathcal{B}}$ are semantically equivalent .

Hence, $\Delta_{\overrightarrow{e}} = 0$.

By lemma 54, $\Delta_{\tilde{e}} = apply_{\tilde{\mathcal{W}}_{\mathcal{A}}}(\delta_{\bar{q}_{\tilde{\mathcal{W}}_{\mathcal{A}}}}, \tilde{\rho})(\tilde{\mathcal{C}}^{\tilde{e}}_{\mathcal{A}}) - apply_{\tilde{\mathcal{W}}_{\mathcal{B}}}(\delta_{\bar{q}_{\tilde{\mathcal{W}}_{\mathcal{B}}}}, \tilde{\rho}')(\tilde{\mathcal{C}}^{\tilde{e}}_{\mathcal{B}}) = \sum_{\overrightarrow{e} \in Prox} \Delta_{\overrightarrow{e}} = 0$ with $\tilde{\mathcal{C}}^{\tilde{e}}_{\mathcal{A}} = Class(\tilde{\mathcal{E}}, X_{\mathcal{A}}, f, \tilde{e})$, $\tilde{\mathcal{C}}^{\tilde{e}}_{\mathcal{B}} = Class(\tilde{\mathcal{E}}, X_{\mathcal{B}}, f, \tilde{e})$, and $Prox = \overrightarrow{proxy}(\tilde{e})_{(\tilde{\mathcal{E}}, X_{\mathcal{A}}, \mathcal{A})} \cup \overrightarrow{proxy}(\tilde{e})_{(\tilde{\mathcal{E}}, X_{\mathcal{B}}, \mathcal{B})}$

This ends the proof.

□

Remark 6. *We might think an adaptation of the theorem to the approximate version could be easy with ε being a negligible function compared to $|\mathcal{I}|$. Indeed, for the same reasons we would have $\Delta_{app}(k), \Delta'_{app}(k, \ell), \Delta''_{app}(\ell) \leq \varepsilon$. Moreover $n \leq |\mathcal{I}|$ since the creation of $\mathcal{A}$ and $\mathcal{B}$ comes necessarily from a task $T \in \rho[i]$ for some $i \in \mathcal{I}$. So we would obtain $\Delta_{\overrightarrow{e}} \leq \varepsilon'$ with $\varepsilon' = \varepsilon \cdot |\mathcal{I}|^2$. However, we do not know how to generalize to $\Delta_{\tilde{e}}$.*

Finally we can use the preservation of $S^{ten,\leq 0}$ to show the monotonicity of tenacious implementation $\leq^{ten}_0$.

Theorem 24 (Implementation monotonicity wrt creation/destruction)**.** *Let $\mathcal{A}$, $\mathcal{B}$ be PSIOA. Let $X_{\mathcal{A}}$, $X_{\mathcal{B}}$ be PCA.*
If $\mathcal{A} \leq^{ten}_0 \mathcal{B}$ and $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$, then $X_{\mathcal{A}} \leq^{ten}_0 X_{\mathcal{B}}$.

Proof. Let ρ be a task-schedule, Since $\mathcal{A} \leq^{ten}_0 \mathcal{B}$ there exists a task-schedule ρ' s. t. $\rho S^{ten,\leq 0}_{(\mathcal{A},\mathcal{B})} \rho'$. By previous lemma $\rho S^{ten,\leq 0}_{(X_{\mathcal{A}},X_{\mathcal{B}})} \rho'$. This ends the proof. □

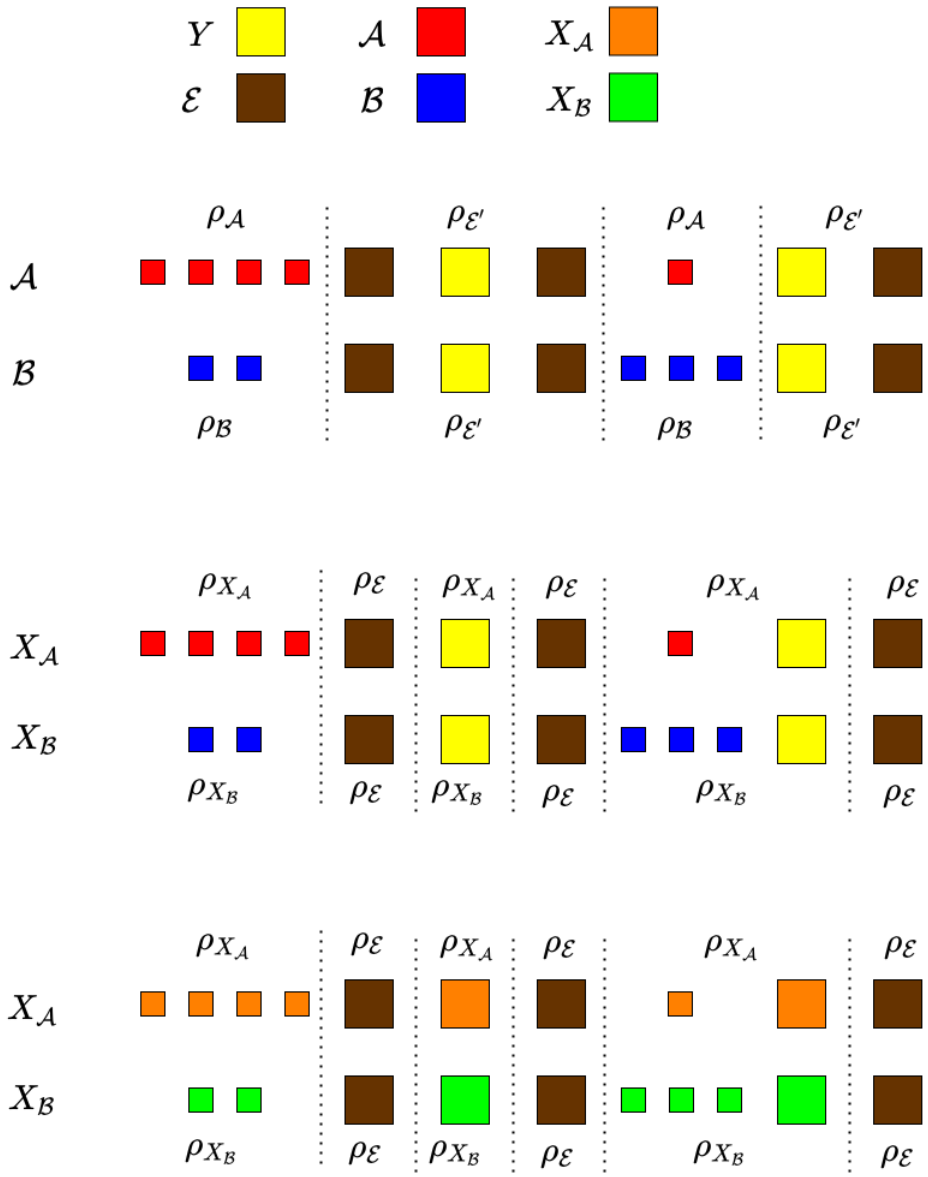


Figure 4.17. – $\mathcal{AB}$-environment-corresponding schedules are also $X_{\mathcal{A}}X_{\mathcal{B}}$-environment-corresponding schedules.

Here, each task is represented by a colored square where the color corresponds to a set of id: yellow for $UA(Y) = UA(X_{\mathcal{A}} \setminus \{\mathcal{A}\}) = UA(X_{\mathcal{B}} \setminus \{\mathcal{B}\})$, red for $\mathcal{A}$, blue for $\mathcal{B}$, orange for $UA(X_{\mathcal{A}}) = UA(Y) \cup \{\mathcal{A}\}$, green for $UA(X_{\mathcal{B}}) = UA(Y) \cup \{\mathcal{B}\}$ and brown for the ones not in $UA(Y) \cup \{\mathcal{A}, \mathcal{B}\}$.

4.8. Summary

We have proved the monotonicity of dynamic creation/destruction of automata with $\leq_0^{S_o,\mathtt{p}}$ precongruence, i.e. if (i) $\mathcal{A}$ implements $\mathcal{B}$ ($\mathcal{A} \leq_0^{S_o,\mathtt{p}} \mathcal{B}$), and (ii) $X_\mathcal{A}$ and $X_\mathcal{B}$ only differ on that $X_\mathcal{A}$ dynamically creates and destroys $\mathcal{A}$ instead of $\mathcal{B}$ as $X_\mathcal{B}$ does, then (iii) $X_\mathcal{A}$ implements $X_\mathcal{B}$ ($X_\mathcal{A} \leq_0^{S_o,\mathtt{p}} X_\mathcal{B}$), where $\mathtt{p}$ is the function that maps each execution to its projection on the environment and S_o represents the set of schedulers that do not take into account the internal past lives before last creation to trigger the next action. Since a naive adaptation of task-schedulers would not be a subset of S_o we discussed an alternative more sophisticated adaptation, the tenacious implementation $\leq_0^{ten}$, which also allows monotonicity of dynamic creation/destruction of automata.

In addition to the results guaranteeing the soundness of modular design based on observational semantics, the proof itself might be even more interesting. First, it is instructive for handling dynamic systems. Second, its modularity allows it to be easily applied to certain variations of the model, typically with the notion of time or security.

Chapter 5

Dynamic Probablistic Timed I/O Automata

This chapter explains how to extend the framework with the notion of time

Overview

5.1. Probabilistic Timed Signature I/O Automata (PTSIOA) . . . 178
5.2. Probabilistic Timed Configuration Automata . . . 181
5.2.1. Measurable space on configurations . . . 181
5.2.2. Probabilistic Timed Configuration Automata . . . 183
5.3. Executions . . . 186
5.4. Global composition . . . 188
5.4.1. PTSIOA composition . . . 188
5.4.2. Configuration composition . . . 189
5.5. Measure over executions, scheduler, implementation . . . 191
5.5.1. Ring on Executions and Traces Definition . . . 191
5.5.2. Probability Measure . . . 196
5.5.3. Implementation . . . 197
5.6. Local Scheduler . . . 198
5.7. Summary . . . 200

Previous chapters 3 and 4 deal with "asynchronous" systems. For the moment, it is impossible to express something like "two events are separated by at most a time t", or apply synchrony assumptions to prove the correctness of an algorithm. In this chapter, we fill this gap by extending PSIOA (resp. PCA) with sets of trajectories to obtain PTSIOA (resp. PTCA). It would have been possible to use the "old-fashioned recipe for real-time" [AL92] as in [Seg95b], where actions can additionally contain real numbers to capture the time elapsing. But, we would miss the opportunity to use the theory of timed I/O automata [KLSV03, KLSV06], which is ready to use. For example, it provides some results to deal with liveness under fairness assumptions (through notions of feasibility, responsiveness, progressiveness,...) that might be more intuitive to express than in [AL92] and benefits of enabled tools to support system development with, for example, specification simulators, code generators, model checking and theorem proving support for analyzing specification, etc [ALL+06, ALL+08]. Also, it has been extended with (continuous) probabilistic setting [Mit07, ML07b]. Hence, the extension of the formalism to Probabilistic Timed Signature I/O automata (PTSIOA) becomes an easy adaptation of PTIOA. We keep the continuous probabilistic setting of Mitra and Lynch [ML07b], since (1) their results are easily adaptable to our dynamic framework and (2) It would be convenient to potentially specify hybrid systems where we would like to model the evolution of some physical quantities (force, mass, velocity, ...) with real numbers, that might be dependent of some algebraic differential equations. The continuous space states require an additional work to verify the measurability of the different operators. However, these details can be left aside for a first reading.

5.1. Probabilistic Timed Signature I/O Automata (PTSIOA)

This section aims to introduce the first brick of our formalism, i.e. the probabilistic timed signature input/output automata (PTSIOA). A PTSIOA is the result of the generalization of probabilistic timed input/output automata (PTIOA) [Mit07, ML07b] and signature input/output automata (SIOA) [AL16]. A PTSIOA is thus an automaton that can deterministically [1] follow a *trajectory* before randomly move from one *state* to another in response to some *actions*. The set of possible actions is the *signature* of the automaton and is partitioned into *input*, *output* and *internal* actions. An action can often be both the input of one automaton and the output of another one to capture the idea that the behavior of an automaton can influence the behavior of another one. As for the SIOA [AL16], the signature of a PTSIOA can change according to the current state of the automaton, which allows us to formalize dynamicity later.

Again, we use the signature approach from [AL16]. We assume the existence of a countable set $Autids$ of unique probabilistic signature input/output automata (PTSIOA) identifiers, an underlying universal set $Auts$ of PTSIOA, and a mapping $aut : Autids \to Auts$. $aut(\mathcal{A})$ is the PTSIOA with identifier $\mathcal{A}$. We use "the automaton $\mathcal{A}$" to mean "the PTSIOA with identifier $\mathcal{A}$". We use the letters $\mathcal{A}, \mathcal{B}$, possibly subscripted or primed, for PTSIOA identifiers.

Definition 182 (PTSIOA)**.** *A pre-PTSIOA* $\mathcal{A} = ((Q_{\mathcal{A}}, \mathcal{F}_{\mathcal{A}}), \bar{q}_{\mathcal{A}}, sig(\mathcal{A}), D_{\mathcal{A}}, \mathcal{T}_{\mathcal{A}})$*, where:*

- $Q_{\mathcal{A}}$ *is the set of* states*,* $(Q_{\mathcal{A}}, \mathcal{F}_{\mathcal{A}})$ *is a measurable space called the* state space*,*
- $\bar{q}_{\mathcal{A}}$ *is the unique* start state*.*
- $sig(\mathcal{A}) : q \in Q_{\mathcal{A}} \mapsto sig(\mathcal{A})(q) = (in(A)(q), out(A)(q), int(A)(q))$ *is the signature function that maps each state to a triplet of mutually disjoint countable set of* actions*, respectively called* input*,* output *and* internal *actions. We note* $acts(\mathcal{A}) = \bigcup_{q \in Q_{\mathcal{A}}} \widehat{sig}(\mathcal{A})(q)$*, which is assumed to be countable.*

1. This determinism will be removed later via a local scheduler that will solve the non-determinism.

- *There exists at most one state, noted* $q_{\mathcal{A}}^{\phi}$ *such that* $\widehat{sig}(\mathcal{A})(q_{\mathcal{A}}^{\phi}) = \emptyset$. *If such a state exists,* $\{q_{\mathcal{A}}^{\phi}\} \in \mathcal{F}_{\mathcal{A}}$.
- $D_{\mathcal{A}} \subset Q_{\mathcal{A}} \times acts(\mathcal{A}) \times \mathcal{P}rob(Q_{\mathcal{A}}, \mathcal{F}_{Q_{\mathcal{A}}})$ *is the set of* probabilistic transitions *where* $\forall (q, a, \eta) \in D_{\mathcal{A}} : a \in \widehat{sig}(\mathcal{A})(q)$. *If* (q, a, η) *is an element of* $D_{\mathcal{A}}$, *we write* $q \xrightarrow{a} \eta$ *and action* a *is said to be* enabled *at* q. *The set of enabled actions at state* q *is noted* $enabled(\mathcal{A})(q)$.*The set of states in which at least one action from the set* $B \subseteq acts(\mathcal{A})$ *is enabled is denoted by* E_B. *In addition* $\mathcal{A}$ *must satisfy the axiom* **E** *of* enabled actions tracking*:* $\widehat{sig}(\mathcal{A}) = enabled(\mathcal{A})$ *We note* $steps(\mathcal{A}) \triangleq \{(q, a, q') \in Q_{\mathcal{A}} \times acts(\mathcal{A}) \times Q_{\mathcal{A}} | \exists (q, a, \eta) \in D_{\mathcal{A}}, q' \in supp(\eta)\}$.
- $\mathcal{T}_{\mathcal{A}}$ *is a set of trajectories in* $Q_{\mathcal{A}}$ *which is (i) closed under prefix, suffix, concatenation, and (ii) contains* $\wp(q)$ *for every* $q \in Q_{\mathcal{A}}$.

A PTSIOA $\mathcal{A}$ is a pre-PTSIOA that satisfies the following conditions:

- **D** (Determinism)
  - $\mathbf{D}_1$ (Trajectory determinism) $\mathcal{T}_{\mathcal{A}}$ is deterministic.
    For every $R \subseteq \mathbb{R}^{>0}$ and every $P \subseteq Q_{\mathcal{A}}$, we define:
    - $E_{R,P} = \{q \in Q_{\mathcal{A}} | maxtime_{\mathcal{T}_{\mathcal{A}}}(q) \in R \wedge maxstate_{\mathcal{T}_{\mathcal{A}}}(q) \in P\}$ the set of states from which there exists a maximal trajectory with length in R and final state in P.
    - $F_{R,P} = \{maxstate_{\mathcal{T}_{\mathcal{A}}}(q) | q \in Q_{\mathcal{A}} \wedge maxtime_{\mathcal{T}_{\mathcal{A}}}(q) \in R\}$ the set of states reachable by a maximal trajectory with length in R and starting from state in P.
  - $\mathbf{D}_2$ (Time-action determinism) For any state q at most one of the following may exist:
    - a local action $a \in acts(\mathcal{A})$ such that $q \in E_a$ and $a \in \widehat{loc}(\mathcal{A})(q)$
    - a non-point trajectory $\tau \in \mathcal{T}_{\mathcal{A}}(q)$.
  - $\mathbf{D_3}$ (Transition determinism) For every $q \in Q_{\mathcal{A}}$ and $a \in enabled(\mathcal{A})(q)$ there exists a unique $\eta \in \mathcal{P}rob(Q_{\mathcal{A}}, \mathcal{F}_{Q_{\mathcal{A}}})$, such that $(q, a, \eta) \in D_{\mathcal{A}}$, and we note $\eta_{(\mathcal{A},q,a)}$ this distribution η. The notation $\eta_{(\mathcal{A},q,a)}$ implicitly means $a \in enabled(\mathcal{A})(q)$.
    For every $P \subseteq Q_{\mathcal{A}}$, we note $H_P = \{q' \in supp(\eta_{\mathcal{A},q,a}) | q \in P \wedge a \in int(\mathcal{A})(q)\}$ the set of reachable states by an internal action triggered from a state in P.
- **M** (Measurability)
  - $\mathbf{M_1}$ (Transitions Measurability) For all $B \subseteq acts(\mathcal{A})$, E_B is measurable.
  - $\mathbf{M_2}$ (Trajectory Measurability) For measurable sets $R \subseteq R^{\geq 0}$, $P \in \mathcal{F}_{Q_{\mathcal{A}}}$, $E_{R,P}$ is measurable, i.e. $E_{R,P} \in \mathcal{F}_{Q_{\mathcal{A}}}$

  Finally, we anticipate the issue of measurability of some perception functions.

  A post-PTSIOA is a PTSIOA that satisfies:
  - $\mathbf{M}'$ (Measurability)
    - $\mathbf{M}_3$ For measurable sets $R \subseteq R^{\geq 0}$, $P \in \mathcal{F}_{Q_{\mathcal{A}}}$, $F_{R,P}$ is measurable, i.e. $F_{R,P} \in \mathcal{F}_{Q_{\mathcal{A}}}$
    - $\mathbf{M}_4$ For measurable set $P \in \mathcal{F}_{Q_{\mathcal{A}}}$, H_P is measurable, i.e. $H_P \in \mathcal{F}_{Q_{\mathcal{A}}}$.

The axiom **E** was already present for PSIOA for sake of simplicity, since for any PTSIOA that does not verify **E**, it is always possible to remove the non-enabled actions of its signature to obtain an equivalent PTSIOA verifying **E**.

The axioms of measurability **M** and $\mathbf{M}'$ are technical axioms to ensure that the required objects during the proofs are well-defined.

The axiom $\mathbf{D}_3$ of transition determinism is the same than the one for PSIOA.

Roughly speaking, the axioms of determinism $\mathbf{D}_1$ and $\mathbf{D}_2$ mean that when a state q is reached by a discrete transition, then the automaton evolves in a deterministic manner until reaching after a certain time t (that can be 0) a state q' with enabled local actions. At that moment, the time cannot elapse without triggering an enabled local action. Such a restriction might appear problematic. Typically, how could we model a partially synchronous distributed algorithm where (i) a decision might be triggered by the expiration of a timer, while (ii) the fact that some messages are in flight means that the reception of these messages are enabled ? In that situation, the time could not elapse with in flight messages. Intuitively, time-elapsing underlines the difference between two different notions of pure non-determinism. The first one captures the lack of knowledge about "what is the next action that will be triggered" ? The second one captures the ignorance about "when the next action will be triggered" ? In fact, these axioms $\mathbf{D}_1$ and $\mathbf{D}_2$ are relaxed in section 5.6 with automata that do not verify $\mathbf{D}_1$ and $\mathbf{D}_2$, while the pure non-determinism is resolved in two steps. First, we resolve the pure non-determinism about "when the next action will be triggered?" by restricting the behaviours of the automata with *local schedulers*, imposing axioms $\mathbf{D}_1$ and $\mathbf{D}_2$. Second, we resolve the pure non-determinism about "what is the next action that will be triggered?" with the classic schedulers. Of course, this resolution could have been performed in one unique step, but this separation allows to reuse the concepts used for untimed PSIOA.

Trivial extension of some definitions We trivially extend definition 60 of compatibility at a state, definition 61 of hiding operator, definition 62 of action renaming operator.

Definition 183 (compatibility at a state)**.** *Let $\mathbf{A} = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$ be a set of PTSIOA. A* state *of $\mathbf{A}$ is an element $q = (q_1, ..., q_n) \in Q_{\mathbf{A}} \triangleq Q_{\mathcal{A}_1} \times ... \times Q_{\mathcal{A}_n}$. We say $\mathcal{A}_1, ..., \mathcal{A}_n$ are (or $\mathbf{A}$ is)* compatible at state q *if $\{sig(\mathcal{A}_1)(q_1), ..., sig(\mathcal{A}_n)(q_n)\}$ is a set of compatible signatures. In this case we note $sig(\mathbf{A})(q) \triangleq sig(\mathcal{A}_1)(q_1) \times ... \times sig(\mathcal{A}_n)(q_n)$ as per definition 59 and we note $\eta_{(\mathbf{A},q,a)} \in Disc(Q_{\mathbf{A}})$, s.t. $\forall a \in \widehat{sig}(\mathbf{A})(q)$, $\eta_{(\mathbf{A},q,a)} = \eta_1 \otimes ... \otimes \eta_n$ where $\forall j \in [1, n]$, $\eta_j = \eta_{(\mathcal{A}_j,q_j,a)}$ if $a \in sig(\mathcal{A}_j)(q_j)$ and $\eta_j = \delta_{q_j}$ otherwise. Moreover, we note $steps(\mathbf{A}) = \{(q, a, q') | q, q' \in Q_{\mathbf{A}}, a \in sig(\mathbf{A})(q), q' \in supp(\eta_{(\mathbf{A},q,a)})\}$. Finally, we note $\bar{q}_{\mathbf{A}} = (\bar{q}_{\mathcal{A}_1}, ..., \bar{q}_{\mathcal{A}_n})$ and $\mathcal{F}_{Q_{\mathbf{A}}} = \mathcal{F}_{Q_{\mathcal{A}_1}} \otimes ... \otimes \mathcal{F}_{Q_{\mathcal{A}_n}}$*

It is convenient for the remaining to define the composition of a set of trajectories of a set of PTSIOA.

Definition 184 (Composition of set of trajectories of a set of PTSIOA)**.** *Let $\mathbf{A} = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$ be a set of PTSIOA. Let $q = (q_1, ..., q_n) \in Q_{\mathbf{A}}$, we note $\mathcal{T}_{\mathbf{A}}(q) \triangleq \mathcal{T}_{\mathcal{A}_1}(q_1)||...||\mathcal{T}_{\mathcal{A}_n}(q_n)$. We note $\mathcal{T}_{\mathbf{A}} \triangleq \mathcal{T}_{\mathcal{A}_1}||...||\mathcal{T}_{\mathcal{A}_n}$ where the composition ($||$) of a set of trajectories is defined in definition 16 in section 2.2.*

The hiding and action renaming operator does not modify the set of trajectories, hence the extension to PTSIOA is immediate.

Definition 185 (hiding and action renaming operator)**.** *Let $\mathcal{A}' = (Q_{\mathcal{A}}, \bar{q}_{\mathcal{A}}, sig(\mathcal{A}), D_{\mathcal{A}})$. Let $\mathcal{A} = (\mathcal{A}', \mathcal{T}_{\mathcal{A}})$ be a PTSIOA with $(Q_{\mathcal{A}}, \bar{q}_{\mathcal{A}}, sig(\mathcal{A}), D_{\mathcal{A}}) = (Q_{\mathcal{A}'}, \bar{q}_{\mathcal{A}'}, sig(\mathcal{A}'), D_{\mathcal{A}'})$.*

- *Let $h : q \in Q_{\mathcal{A}} \mapsto h(q) \subseteq out(\mathcal{A})(q)$. We note $hide(\mathcal{A}, h) \triangleq (hide(\mathcal{A}', h), \mathcal{T}_{\mathcal{A}})$ as per definition 61.*
- *let r be a partial function on $Q_{\mathcal{A}} \times acts(\mathcal{A})$, s.t. $\forall q \in Q_{\mathcal{A}}$, $r(q)$ is an injective mapping with $\widehat{sig}(\mathcal{A})(q)$ as domain. Then $r(\mathcal{A}) \triangleq (r(\mathcal{A}'), \mathcal{T}_{\mathcal{A}})$ as per definition 62.*

5.2. Probabilistic Timed Configuration Automata

Here again, we trivially extend configurations and PCA to their timed version. A

Definition 186 (Timed configuration)**.** *A timed configuration is a pair* $(\mathbf{A}, \mathbf{S})$ *where*

- $\mathbf{A} = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$ *is a finite set of PTSIOA identifiers and*
- $\mathbf{S}$ *maps each* $\mathcal{A}_k \in \mathbf{A}$ *to a state* $s_k \in Q_{\mathcal{A}_k}$*.*

Moreover, we note $TS((\mathbf{A}, \mathbf{S})) = (\mathbf{S}(\mathcal{A}_1), ..., \mathbf{S}(\mathcal{A}_n))$*, the tuple of states of the automata of the configuration.*

Finally, the set of timed configurations is noted Q_{tconf}*.*

All the following definitions of section 3.2

- definition 64 of compatible configuration (with PTSIOA compatible at a state in the sense of definition 183.)
- definition 65 of intrinsic attributes
- definition 66 of reduced configuration

can be adapted in the obvious way by replacing:

- configuration by timed configuration
- PSIOA by PTSIOA

When it is clear in the context, we omit the adjective "timed" for a configuration.

The discrete transitions have already been defined in definition 67 in section 3.2 of chapter 3. Here, we characterize their continuous evolutions.

Definition 187 (intrinsic trajectories)**.** *Let* $C = (\mathbf{A}, \mathbf{S})$ *be a timed configuration. We note* $\mathcal{T}_C \subset traj s(Q_{conf})$ *such that there exists a bijection* u *from* $\mathcal{T}_C$ *to* $\mathcal{T}_{\mathbf{A}}(TS(C))$*, verifying* $\mathcal{T}_C \overset{u,v}{\rightsquigarrow} \mathcal{T}_{\mathbf{A}}(TS(C))$ *with* $v = TS$*, i.e. for every pair* (τ, τ') *with* $\tau' = u(\tau)$*:*

- $dom(\tau) = dom(\tau') \triangleq J$
- $\forall t \in J$, $TS(\tau(t)) = \tau'(t)$

A trajectory starting from a configuration is a trajectory of the associated set of automata starting from their attached states defined by the configuration. In such a trajectory, no automaton creation is allowed.

5.2.1. Measurable space on configurations

The set Q_{tconf} is not countable anymore. In this subsection, we define the appropriate measurable spaces.

Definition 188 (Generators of measurable spaces for timed configurations)**.** *Let* $\mathbf{A} = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$ *be a finite set of PTSIOA. We define:*

- $Q^{\mathbf{A}}_{tconf} = \{C \in Q_{tconf} | auts(C) = \mathbf{A}\}$ *the set of configurations with* $\mathbf{A}$ *as set of automata*
- $Q^{\subseteq \mathbf{A}}_{tconf} = \{C \in Q_{tconf} | auts(C) \subseteq \mathbf{A}\}$ *the set of configurations with a set of automata included in* $\mathbf{A}$*.*
- $\mathscr{C}^{\mathbf{A}}_{tconf} = \{\{C \in Q^{\mathbf{A}}_{tconf} | TS(C) \in F_1 \times ... \times F_n\} | (F_1, ..., F_n) \in \mathcal{F}_{\mathcal{A}_1} \times ... \times \mathcal{F}_{\mathcal{A}_n}\}$ *the collections of subsets of timed configurations matching the parallelepipeds built with elements of respective* σ*-algebra of each automaton in* $\mathbf{A}$*.*

- $\mathscr{C}_{tconf}^{\subseteq \mathbf{A}} = \{F_{tconf}^{\mathbf{A}'} \in \mathscr{C}_{tconf}^{\mathbf{A}'} | \mathbf{A}' \subseteq \mathbf{A}\}$ *the collection of subsets of* $Q_{tconf}^{\subseteq \mathbf{A}}$ *built with all the sets of all the collections* $\mathscr{C}_{tconf}^{\mathbf{A}'}$ *for all the* $\mathbf{A}' \subseteq \mathbf{A}$.
- $\mathscr{C}_{tconf} = \{F_{tconf}^{\mathbf{A}'} \in \mathscr{C}_{tconf}^{\mathbf{A}'} | \mathbf{A}'$ *is a finite set of PTSIOA* $\}$ *the collection of subsets of* $Q_{tconf}^{\mathbf{A}'}$ *built with all the sets of all the collections* $\mathscr{C}_{tconf}^{\mathbf{A}'}$ *for all the finite sets of PTSIOA* $\mathbf{A}'$.
- $\mathcal{F}_{Q_{tconf}} = sigma(\mathscr{C}_{tconf})$ *the smallest* σ*-algebra containing* $\mathscr{C}_{tconf}$.
- $\mathcal{F}_{Q_{tconf}^{\mathbf{A}}} = sigma(\mathscr{C}_{tconf}^{\mathbf{A}})$ *the smallest* σ*-algebra containing* $\mathscr{C}_{tconf}^{\mathbf{A}}$.
- $\mathcal{F}_{Q_{tconf}^{\subseteq \mathbf{A}}} = sigma(\mathscr{C}_{tconf}^{\subseteq \mathbf{A}})$ *the smallest* σ*-algebra containing* $\mathscr{C}_{tconf}^{\subseteq \mathbf{A}}$.

We often want to define union operation, thus we verify its measurability.

Lemma 68 (union is measurable)**.** *Let* $\mathbf{A}^a$,$\mathbf{A}^b$ *and* $\mathbf{A}^c = \mathbf{A}^a \uplus \mathbf{A}^b$ *be finite sets of PTSIOA. The function*

$$union_{(\mathbf{A}^a,\mathbf{A}^b)}: \begin{cases} (Q_{tconf}^{\mathbf{A}^a}, \times Q_{tconf}^{\mathbf{A}^b}, sigma(\{F^a \times F^b | (F^a, F^b) \in \mathscr{C}_{tconf}^{\mathbf{A}^a} \times \mathscr{C}_{tconf}^{\mathbf{A}^b}\}) & \rightarrow & (Q_{tconf}^{\mathbf{A}^c}, \mathcal{F}_{Q_{tconf}^{\mathbf{A}^c}}) \\ (C_1, C_2) & \mapsto & C_1 \cup C_2 \end{cases}$$

is measurable.

Proof. We note $\mathbf{A}^a = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$,$\mathbf{A}^b = \{\mathcal{A}_{n+1}, ..., \mathcal{A}_m\}$. Let $F_{tconf}^{\mathbf{A}^c} \in \mathscr{C}_{tconf}^{\mathbf{A}^c}$, i.e. $F_{tconf}^{\mathbf{A}^c} = \{C \in Q_{tconf}^{\mathbf{A}^c} | TS(C) \in F_1 \times ... \times F_n \times F_{n+1} \times ... \times F_m\}$ for some $(F_1, ..., F_n, F_{n+1}, ..., F_m) \in \mathcal{F}_{\mathcal{A}_1} \times ... \times \mathcal{F}_{\mathcal{A}_n} \times \mathcal{F}_{\mathcal{A}_{n+1}} \times ... \times \mathcal{F}_{\mathcal{A}_m}$. We have $E = union_{(\mathbf{A}^a,\mathbf{A}^b)}^{-1}(F_{tconf}^{\mathbf{A}^c}) = \{(C^a, C^b) \in Q_{tconf}^{\mathbf{A}^a} \times Q_{tconf}^{\mathbf{A}^b} | TS(C^a) \in F_1 \times ... \times F_n \wedge TS(C^b) \in F_{n+1} \times ... \times F_m\}$. Thus $E = \{C^a \in Q_{tconf}^{\mathbf{A}^a} | TS(C^a) \in F_1 \times ... \times F_n\} \times \{C^b \in Q_{tconf}^{\mathbf{A}^b} | TS(C^b) \in F_{n+1} \times ... \times F_m\}$ and so $E = E^a \times E^b$ with $E^a \in \mathscr{C}_{tconf}^{\mathbf{A}^a}$ and $E^b \in \mathscr{C}_{tconf}^{\mathbf{A}^b}$. Thus $\forall F_{tconf}^{\mathbf{A}^c} \in \mathscr{C}_{tconf}^{\mathbf{A}^c}$, $union_{(\mathbf{A}^a,\mathbf{A}^b)}^{-1}(F_{tconf}^{\mathbf{A}^c}) \in \{F^a \times F^b | (F^a, F^b) \in \mathscr{C}_{tconf}^{\mathbf{A}^a} \times \mathscr{C}_{tconf}^{\mathbf{A}^b}\}$. By proposition 3, $union_{(\mathbf{A}^a,\mathbf{A}^b)}$ is measurable. □

In the same way

Lemma 69 (simple union is measurable)**.** *Let* $\mathbf{A}^a$,φ *and* $\mathbf{A}^c = \mathbf{A}^a \uplus \varphi$ *be finite sets of PTSIOA. Let* $C_\varphi = (\varphi, \mathbf{S}_\varphi)$ *with* $\forall \mathcal{A} \in \varphi, \mathbf{S}_\varphi(\mathcal{A}) = q_{\mathcal{A}}^{\phi}$.

The function

$$union_{(\mathbf{A}^a)}^{\varphi}: \begin{cases} (Q_{tconf}^{\mathbf{A}^a}, \mathcal{F}_{Q_{tconf}^{\mathbf{A}^a}}) & \rightarrow & (Q_{tconf}^{\mathbf{A}^c}, \mathcal{F}_{Q_{tconf}^{\mathbf{A}^c}}) \\ C & \mapsto & C \cup C_\varphi \end{cases} \quad \textit{is measurable.}$$

Proof. The proof is exactly the same one as previous lemma. □

We want to define intrinsic transitions on $(Q_{tconf}, \mathcal{F}_{Q_{tconf}})$. This definition requires measurability of *reduce* function.

Lemma 70. *Let* $\mathbf{A}$ *be a finite set of PTSIOA. The function*

$$reduce_{\mathbf{A}}: \begin{cases} (Q_{tconf}^{\mathbf{A}}, \mathcal{F}_{Q_{tconf}^{\mathbf{A}}}) & \rightarrow & (Q_{tconf}^{\subseteq \mathbf{A}}, \mathcal{F}_{Q_{tconf}^{\subseteq \mathbf{A}}}) \\ C & \mapsto & reduce(C) \end{cases} \quad \textit{is measurable.}$$

Proof. We note $\mathbf{A} = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$. Let $F_{tconf}^{\mathbf{A}'} \in \mathscr{C}_{tconf}^{\subseteq \mathbf{A}}$, i.e. $F_{tconf}^{\mathbf{A}'} \in \mathscr{C}_{tconf}^{\mathbf{A}'}$ for some $\mathbf{A}' \subseteq \mathbf{A}$. Without loss of generality, let us assume $\mathbf{A}' = \{\mathcal{A}_1, ..., \mathcal{A}_k\}$ for $k \in [1 : n]$. By definition, $F_{tconf}^{\mathbf{A}'} = \{C' \in$

$Q^{\mathbf{A}'}_{tconf}|TS(C') \in F_1 \times ... \times F_k\}$ for some $(F_1, ..., F_k) \in \mathcal{F}_{\mathcal{A}_1} \times ... \times \mathcal{F}_{\mathcal{A}_k}$ Let $E^{\mathbf{A}}_{tconf} = reduce^{-1}_{\mathbf{A}}(F^{\mathbf{A}'}_{tconf})$. We have $E^{\mathbf{A}}_{tconf} = \{C \in Q^{\mathbf{A}}_{tconf}|TS(C) \in F_1 \setminus \{q^{\phi}_{\mathcal{A}_1}\} \times ... \times F_k \setminus \{q^{\phi}_{\mathcal{A}_k}\} \times \{q^{\phi}_{\mathcal{A}_{k+1}}\} \times ... \times \{q^{\phi}_{\mathcal{A}_n}\}\}$ and so $E^{\mathbf{A}}_{tconf} = \{C \in Q^{\mathbf{A}}_{tconf}|TS(C) \in E_1 \times ... \times E_n\}$ for some $(E_1, ..., E_k) \in \mathcal{F}_{\mathcal{A}_1} \times ... \times \mathcal{F}_{\mathcal{A}_n}$. Hence $E^{\mathbf{A}}_{tconf} \in \mathscr{C}^{\mathbf{A}}_{tconf}$. So $\forall F \in \mathscr{C}^{\subseteq \mathbf{A}}_{tconf}, reduce^{-1}_{\mathbf{A}}(F) \in \mathscr{C}^{\mathbf{A}}_{tconf}$. By proposition 3, $reduce_{\mathbf{A}}$ is measurable. □

Now we are ready to define intrinsic transitions for timed configurations

Definition 189 (From preserving distribution to intrinsic transition (continuous case))**.** *Let* $\mathbf{A}$ *be a finite set of PTSIOA.*

- *(preserving distribution) Let* $\eta_p \in \mathcal{P}rob(Q^{\mathbf{A}}_{tconf}, \mathcal{F}_{Q^{\mathbf{A}}_{tconf}})$*. We say* η_p *is a* preserving distribution *with* family support $\mathbf{A}$*.*
- *(preserving configuration transition* $C \xrightarrow{a} \eta_p$*) Let* $C = (\mathbf{A}, \mathbf{S})$ *be a compatible timed configuration,* $a \in \widehat{sig}(C)$*. Let* $\eta_p \in \mathcal{P}rob(Q^{\mathbf{A}}_{tconf}, \mathcal{F}_{Q^{\mathbf{A}}_{tconf}})$ *such that* $\eta_p \overset{TS}{\leftrightarrow} \eta_{(\mathbf{A},TS(C),a)}$*. We say that* (C, a, η_p) *is a* preserving configuration transition*, noted* $C \xrightarrow{a} \eta_p$*.*
- *(*$\eta_p \uparrow \varphi$*) Let* $\eta_p \in \mathcal{P}rob(Q^{\mathbf{A}}_{tconf}, \mathcal{F}_{Q^{\mathbf{A}}_{tconf}})$ *be a preserving distribution with* $\mathbf{A}$ *as family support. Let* φ *be a finite set of PTSIOA identifiers with* $\mathbf{A} \cap \varphi = \emptyset$*.*
  Let $C_\varphi = (\varphi, S_\varphi) \in Q_{tconf}$ *with* $\forall \mathcal{A}_j \in \varphi, S_\varphi(\mathcal{A}_j) = \bar{q}_{\mathcal{A}_j}$*. We note* $\eta_p \uparrow \varphi$ *the unique element of* $\mathcal{P}rob(Q^{\mathbf{A}\cup\varphi}_{tconf}, \mathcal{F}_{Q^{\mathbf{A}\cup\varphi}_{tconf}})$ *verifying* $\eta_p \overset{u}{\leftrightarrow} (\eta_p \uparrow \varphi)$ *with* $u : F \mapsto \{(C \cup C_\varphi)|C \in F\}$*.*
- *(distribution reduction) Let* $\eta \in \mathcal{P}rob(Q^{\mathbf{A}}_{tconf}, \mathcal{F}_{Q^{\mathbf{A}}_{tconf}})$ *be a preserving distribution. We note* $reduce(\eta)$ *the image measure of* η *by* $reduce_{\mathbf{A}}$*,i.e.* $reduce(\eta) \in \mathcal{P}rob(Q^{\subseteq \mathbf{A}}_{tconf}, \mathcal{F}_{Q^{\subseteq \mathbf{A}}_{tconf}})$ *with* $\forall F \in \mathcal{F}_{Q^{\subseteq \mathbf{A}}_{tconf}}, reduce(\eta)(F) = \eta(reduce^{-1}_{\mathbf{A}}(F)) = \eta(\{C \in Q^{\mathbf{A}}_{tconf}|reduce_{\mathbf{A}}(C) \in F\})$*.*
- *(intrinsic transition* $C \xRightarrow{a}_{\varphi} \eta$*) Let* $C = (\mathbf{A}, \mathbf{S})$ *be a compatible timed configuration, let* $a \in \widehat{sig}(C)$*, let* φ *be a finite set of PTSIOA identifiers with* $\mathbf{A} \cap \varphi = \emptyset$*. We note* $C \xRightarrow{a}_{\varphi} \eta$*, if* $\eta = reduce(\eta_p \uparrow \varphi)$ *with* $C \xrightarrow{a} \eta_p$*. In this case, we say that* η *is* generated by η_p and φ*.*

This definition 189 is nothing more than the extension of definition 67 to timed configurations with continuous state spaces. We have used an heavy formalism to ensure that the definition 189 was well-defined, but the intuition behind definition 67 remains the same. If $C \xRightarrow{a}_{\varphi} \eta$ with η is generated by η_p and φ, it means triggering a lead to a probabilistic distribution that (i) agrees with η_p for PTSIOA originally belonging to C, (ii) creates deterministically the PTSIOA in φ, (iii) destroys the PTSIOA with an empty signature.

5.2.2. Probabilistic Timed Configuration Automata

Here we define our probabilistic timed configuration automata (PTCA). Such an automaton defines a strong link with a dynamic timed configuration.

First we need to extend the intrinsic measure.

The next lemma is not strictly necessary, but might be convenient to simplify mental representation.

Lemma 71. *Let* $\mathbf{A}$ *be a finite set of PTSIOA.* $\forall \eta \in \mathcal{P}rob(Q^{\mathbf{A}}_{tconf}, \mathcal{F}_{Q^{\mathbf{A}}_{tconf}})$*, there exists a unique measure* $\bar{\eta} \in \mathcal{P}rob(Q_{tconf}, \mathcal{F}_{Q_{tconf}})$ *such that for every finite set of PTSIOA* $\mathbf{A}'$*,* $\forall F^{\mathbf{A}'}_{tconf} \in \mathscr{C}^{\mathbf{A}'}_{tconf}$*,* $\bar{\eta}(F^{\mathbf{A}'}_{tconf}) = \begin{cases} \eta(F^{\mathbf{A}'}_{tconf}) \text{ if } \mathbf{A}' = \mathbf{A} \\ 0 \text{ otherwise} \end{cases}$

Proof. The existence and uniqueness comes from the classic construction.

- Let $F_1(\mathscr{C}_{tconf})$ be the be the family containing $\emptyset$, Q_{tconf}, and all $E \subseteq \mathscr{C}_{tconf}$ such that either $E \in \mathscr{C}_{tconf}$ or $Q_{tconf} \setminus E \in \mathscr{C}$.
  There exists a unique extension η^1 of η to $F_1(\mathscr{C})$. Indeed, there is a unique way to extend the measure since for each $E_1 \in F_1(\mathscr{C}_{tconf})$, $\eta_1(E_1) = \eta(E_1)$ if $E_1 \in \mathscr{C}_{tconf}$ and we necessarily have $\eta_1(E_1) = 1 - \eta(E_1^c)$ if $Q_{tconf} \setminus E_1 \in \mathscr{C}$. Moreover $\forall E_1, E_1' \in F_1(\mathscr{C}_{tconf})$, $E_1 \cup E_1' \in F_1(\mathscr{C}_{tconf})$ only if $E_1, E_1' \in \mathscr{C}_{tconf}^{\mathbf{A}'}$ for some finite set of PTSIOA $\mathbf{A}'$. If $\mathbf{A}' = \mathbf{A}$, $\eta^1(E_1 \cup E_1') = \eta(E_1 \cup E_1') = \eta(E_1) + \eta(E_1') = \eta^1(E_1) + \eta^1(E_1')$ by σ-additivity of η. If $\mathbf{A}' \neq \mathbf{A}$, $\eta^1(E_1 \cup E_1') = 0 = 0 + 0 = \eta^1(E_1) + \eta^1(E_1')$ so σ-additivity is preserved.
- Let $F_2(\mathscr{C}_{tconf})$ be the family containing all finite intersections of elements of $F_1(\mathscr{C}_{tconf})$. There exists a unique extension η^2 of η^1 to $F_2(\mathscr{C}_{tconf})$. Let $E_2^1, ..., E_2^n$ a sequence of elements of $F_2(\mathscr{C}_{tconf})$ and E_2 their intersection. If one element E_2^j is an element $\mathscr{C}_{tconf}^{\mathbf{A}'}$ for some $\mathbf{A}'$, then either $E_2 = \emptyset$ or E_2 is an element of $\mathscr{C}_{tconf}^{\mathbf{A}'}$ too. In that case, $\eta^2(E_2) = \eta^1(E_2)$. If all elements E_2^j can be written $Q_{tconf} \setminus E_2'^j$, then $\eta^2(E_2) = 1 - \sum_{j \in [1:n]} \eta^1(E_2'^j)$.
- Let $F_3(\mathscr{C}_{tconf})$ be the family containing all finite unions of disjoint elements of $F_2(\mathscr{C}_{tconf})$. There exists a unique extension η^3 of η^3 to $F_3(\mathscr{C}_{tconf})$. Indeed, there is a unique way of assigning a measure to the finite union of disjoint sets whose measure is known, i.e., adding up their measures i.e. $\eta^3(\biguplus_{j \in [1:n]} E_2^j) = \sum_{j \in [1:n]} \eta^2(E_2^j)$ for every finite set of disjoint elements E_2^j of $F_2(\mathscr{C}_{tconf})$.
- Clearly, $F_3(\mathscr{C}_{tconf})$ is a field on $\mathscr{C}_{tconf}$, i.e. it is a family of subsets of $\mathscr{C}_{tconf}$ that contains $\emptyset, \mathscr{C}_{tconf}$, and that is closed under complementation and finite union. $\mathcal{F}_{Q_{tconf}} \triangleq sigma(\mathscr{C}_{tconf}) = sigma(F_3(\mathscr{C}_{tconf}))$. By famous Carathéodory's extension theorem [Dud04], there exists a unique extension $\bar{\eta}$ of η^3 to the sigma-algebra $\mathcal{F}_{Q_{tconf}}$ (defining $\bar{\eta}(\biguplus_{j \in \mathbb{N}} E_3^j) = \sum_{j \in \mathbb{N}} \eta^3(E_3^j)$ for every countable set of disjoint elements E_3^j of $F_3(\mathscr{C}_{tconf})$).

This detail might appear unnecessarily cumbersome. Moreover, we can remark that the identity function is a bijection between generators of non-zero measurement of η and $\bar{\eta}$ i.e. from $supp(\bar{\eta}) \cap \mathscr{C}_{tconf}$ and $supp(\eta) \cap \mathscr{C}_{tconf}^{\mathbf{A}}$. In the remaining, we often abuse the notation and use η to denote its extension $\bar{\eta}$. □

Lemma 72. *Let $\mathbf{A}$ be a finite set of PTSIOA. $\forall \eta \in \mathcal{P}rob(Q_{tconf}^{\subseteq \mathbf{A}}, \mathcal{F}_{Q_{tconf}^{\subseteq \mathbf{A}}})$, there exists a unique measure $\bar{\eta} \in \mathcal{P}rob(Q_{tconf}, \mathcal{F}_{Q_{tconf}})$ such that for every finite set of PTSIOA $\mathbf{A}'$, $\forall F_{tconf}^{\mathbf{A}'} \in \mathscr{C}_{tconf}^{\mathbf{A}'}$, $\bar{\eta}(F_{tconf}^{\mathbf{A}'}) = \begin{cases} \eta(F_{tconf}^{\mathbf{A}'}) \text{ if } \mathbf{A}' \subseteq \mathbf{A} \\ 0 \text{ otherwise} \end{cases}$*

Proof. The proof is the same as previous lemma. □

Definition 190 (extension)**.** *Let $\mathbf{A}$ be a finite set of PTSIOA.*

- *$\forall \eta \in \mathcal{P}rob(Q_{tconf}^{\mathbf{A}}, \mathcal{F}_{Q_{tconf}^{\mathbf{A}}})$, we note $\bar{\eta}$ the unique measure in $\mathcal{P}rob(Q_{tconf}, \mathcal{F}_{Q_{tconf}})$ such that for every finite set of PTSIOA $\mathbf{A}'$, $\forall F_{tconf}^{\mathbf{A}'} \in \mathscr{C}_{tconf}^{\mathbf{A}'}$, $\bar{\eta}(F_{tconf}^{\mathbf{A}'}) = \begin{cases} \eta(F_{tconf}^{\mathbf{A}'}) \text{ if } \mathbf{A}' = \mathbf{A} \\ 0 \text{ otherwise} \end{cases}$*
- *$\forall \eta \in \mathcal{P}rob(Q_{tconf}^{\subseteq \mathbf{A}}, \mathcal{F}_{Q_{tconf}^{\subseteq \mathbf{A}}})$, we note $\bar{\eta}$ the unique measure in $\mathcal{P}rob(Q_{tconf}, \mathcal{F}_{Q_{tconf}})$ such that for every finite set of PTSIOA $\mathbf{A}'$, $\forall F_{tconf}^{\mathbf{A}'} \in \mathscr{C}_{tconf}^{\mathbf{A}'}$, $\bar{\eta}(F_{tconf}^{\mathbf{A}'}) = \begin{cases} \eta(F_{tconf}^{\mathbf{A}'}) \text{ if } \mathbf{A}' \subseteq \mathbf{A} \\ 0 \text{ otherwise} \end{cases}$*

Now, we are able to define PTCA. The definition is basically the same as the definition 68 of PCA where (i) the associated PSIOA is replaced by a PTSIOA, (ii) the configuration mapping yields timed configurations, but with one additional constraint, so-called *trajectory preservation*.

Definition 191 (Probabilistic Timed Configuration Automaton (PTCA))**.** *A probabilistic timed configuration automaton (PTCA) (resp. pre-PTCA, resp. post-PTCA) X consists of the following components:*

- *1. A probabilistic signature I/O automaton (resp. pre-PTSIOA, resp. post-PTSIOA) $ptsioa(X)$. For brevity, we define $Q_X = Q_{ptsioa(X)}$, $\bar{q}_X = \bar{q}_{ptsioa(X)}$, $sig(X) = sig(ptsioa(X))$ and likewise for all other (sub)components and attributes of $ptsioa(X)$.*
- *2. A configuration mapping $config(X)$ with domain Q_X and such that $config(X)(q)$ is a reduced compatible configuration for all $q \in Q_X$.*
- *3. For each $q \in Q_X$, a mapping $created(X)(q)$ with domain $\widehat{sig}(X)(q)$ and such that $\forall a \in \widehat{sig}(X)(q)$, $created(X)(q)(a) \subseteq Autids$ with $created(X)(q)(a)$ finite.*
- *4. A hidden-actions mapping $hidden\text{-}actions(X)$ with domain Q_X and such that $\forall q \in Q_X$, $hidden\text{-}actions(X)(q) \subseteq out(config(X)(q))$.*

and satisfies the following constraints

- *1. (start states preservation) If $config(X)(\bar{q}_X) = (\mathbf{A}, \mathbf{S})$, then $\forall \mathcal{A}_i \in \mathbf{A}, \mathbf{S}(\mathcal{A}_i) = \bar{q}_{\mathcal{A}_i}$*
- *2. (top/down transition preservation) If $(q, a, \eta_{(X,q,a)}) \in D_X$ then there exists $\bar{\eta}' \in \mathcal{P}rob(Q_{tconf}, \mathcal{F}_{Q_{tconf}})$ s.t. $\eta_{(X,q,a)} \xleftrightarrow{c} \bar{\eta}'$ with i) $c = config(X)$ and ii) $config(X)(q) \stackrel{a}{\Longrightarrow}_{\varphi} \eta'$, where $\varphi = created(X)(q)(a)$*
- *3. (bottom/up transition preservation) If $q \in Q_X$ and $config(X)(q) \stackrel{a}{\Longrightarrow}_{\varphi} \eta'$ for some action a, $\varphi = created(X)(q)(a)$, and reduced compatible probabilistic measure η', then $(q, a, \eta_{(X,q,a)}) \in Q_X$, and $\eta_{(X,q,a)} \xleftrightarrow{c} \bar{\eta}'$ with $c = config(X)$.*
- *4. (signature preservation modulo hiding)*
  For all $q \in Q_X$, $sig(X)(q) = hide(sig(config(X)(q)), hidden\text{-}actions(q))$.
- *5. (trajectory preservation) For every pair $(q, C) \in Q_X \times Q_{conf}$ with $C = config(X)(q)$, $\mathcal{T}_X(q) \overset{u,c}{\rightsquigarrow} \mathcal{T}_C$ with $c = config(X)$, i.e.*
  There is a bijection u from $\mathcal{T}_X(q)$ to $\mathcal{T}_C$ such that for every pair (τ, τ') with $\tau' = u(\tau)$:
  - *$dom(\tau) = dom(\tau') \triangleq J$*
  - *$\forall t \in J$, $config(X)(\tau(t)) = \tau'(t)$*

Compared to definition 68, we only added the constraint of trajectory preservation, which requires that trajectories of the PTCA are linked in an obvious manner with the trajectories of the associated configurations.

We can remark that the requirement of reduced configuration for each state, implies that a continuous trajectory cannot lead to the destruction of a sub-automaton. Hence, sub-automaton destruction can only occur during discrete transitions.

Yet again, compatibility of a set of PTCA at a particular state (resp. hiding on PTCA) is defined as in definition 70 (resp. definition 69), where the PCA are replaced by PTCA and configurations are replaced by timed configurations.

5.3. Executions

In this section, we adapt the definitions of section 3.4 with time elapsing. In this way, executions become alternating sequences of actions and trajectories. It captures a run that is an alternation of discrete and continuous transitions.

Definition 192 (pseudo execution fragment of a finite set of PTSIOA)**.** *Let* $\mathbf{A} = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$ *be a finite set of PTSIOA (resp. PTCA). A* pseudo execution fragment *of* $\mathbf{A}$ *is a finite or infinite sequence* $\alpha = \tau^0 a^1 \tau^1 a^2 ...$ *of alternating trajectories and actions, such that:*

1. *If* α *is finite, it ends with a trajectory. In that case, we note* $lstate(\alpha)$ *the last state of the last trajectory of* α*.*
2. *At the potential exception of* $lstate(\alpha)$ *if* α *is finite, for every trajectory* τ^i*, for every time* $t \in dom(\tau^i)$*,* $\mathbf{A}$ *is compatible at state* $\tau^i(t)$*.*
3. *for ever action* a^i*,* $(\tau^{i-1}.lstate, a^i, \tau^i.fstate) \in steps(\mathbf{A})$

The first state of a pseudo execution fragment α*,* $fstate(\alpha)$*, is* $fstate(\tau^0)$*. A pseudo execution fragment* α *of* $\mathbf{A}$ *is a pseudo execution of* $\mathbf{A}$ *if* $fstate(\alpha) = \bar{q}_{\mathbf{A}}$*. The length* $|\alpha|$ *of a finite pseudo execution fragment* α *is the number of actions in* α*. A pseudo execution fragment* α *is closed if it is a finite sequence and the last trajectory is closed. Given a closed execution fragment* $\alpha = \tau^0 a^1 \tau^1 a^2 ... \tau^n$*, its limit time is defined to be* $\sum_{i=0}^{n} \tau^i.ltime$*. A state* q *of* $\mathbf{A}$ *is said* reachable *if there is a pseudo execution* α *s.t.* $lstate(\alpha) = q$*. We note* $Reachable(\mathbf{A})$ *the set of* reachable *states of* $\mathbf{A}$*. If* $\mathbf{A}$ *is compatible at every reachable state* q*,* $\mathbf{A}$ *is said* partially-compatible*.*

Definition 193 (executions, concatenations)**.** *Let* $\mathcal{A}$ *be an automaton. An* execution fragment *(resp.* execution*) of* $\mathcal{A}$ *is a pseudo execution fragment (resp. pseudo execution) of* $\{\mathcal{A}\}$*. We use* $Frags(\mathcal{A})$ *(resp.,* $Frags^*(\mathcal{A})$*) to denote the set of all (resp., all finite) execution fragments of* $\mathcal{A}$*.* $Execs(\mathcal{A})$ *(resp.* $Execs^*(\mathcal{A})$*) denotes the set of all (resp., all finite) executions of* $\mathcal{A}$*.*

Let α *be an execution fragment of* $\mathcal{A}$*. The length* $|\alpha|$ *of a finite execution fragment* α *is the number of transitions along* α*.*

We define a concatenation operator $\frown$ *for execution fragments as follows:*
If $\alpha = \tau^0 a^1 \tau^1 ... a^n \tau^n \in Frags^*(\mathcal{A})$ *and* $\alpha' = \tau^{0\prime} a^{1\prime} \tau^{1\prime} ... \in Frags^*(\mathcal{A})$*, we define* $\alpha \frown \alpha' = \tau^0 a^1 \tau^1 ... a^n \tau^n \frown \tau^{0\prime} a^{1\prime} \tau^{1\prime} ...$

Let $\alpha, \alpha' \in Frags(\mathcal{A})$*, then* α *is a proper prefix of* α' *iff* $\exists \alpha'' \in Frags(\mathcal{A})$ *such that* $\alpha' = \alpha \frown \alpha''$ *with* $\alpha \neq \alpha'$*. In that case, we note* $\alpha < \alpha'$*. We note* $\alpha \leq \alpha'$ *if* $\alpha < \alpha'$ *or* $\alpha = \alpha'$ *and say that* α *is a prefix of* α'*.*

The trace of an execution represents its externally visible part, namely the external actions and time passage. It is obtained by removing internal actions, concatenating consecutive trajectories, and replacing all the trajectories with their limit times.

Definition 194 (traces)**.** *Let* $\mathcal{A}$ *be a PTSIOA or a PTCA, let* $\alpha \in Execs(\mathcal{A})$*.*

We recursively define the function $trace_{\mathcal{A}}$ *with* $dom(trace_{\mathcal{A}}) = Execs(\mathcal{A})$*, such as* $\forall \alpha \in Execs(\mathcal{A}), \forall \tau \in \mathcal{T}_{\mathcal{A}}, \forall a \in acts(\mathcal{A})$*:*

$trace_{\mathcal{A}}(\alpha) = \tau.ltime$ *if* $\alpha = \tau \in trajs(Q_{\mathcal{A}})$*,*

$$trace_{\mathcal{A}}(\alpha a \tau) = \begin{cases} trace(\alpha) a \tau.ltime \text{ if } a \in \widehat{ext}(\mathcal{A})(lstate(\alpha)) \\ \beta'(\tau' \frown \tau).ltime \text{ where } trace_{\mathcal{A}}(\alpha) = \beta' \tau'.ltime \text{ otherwise.} \end{cases}$$

We say that β is a trace of $\mathcal{A}$ if there is $\alpha \in Execs(\mathcal{A})$ with $\beta = trace(\alpha)$. We note $Traces(\mathcal{A})$ the set of traces of $\mathcal{A}$.

The projection of a pseudo-execution α on an automaton $\mathcal{A}_i$, noted $\alpha \upharpoonright \mathcal{A}_i$, represents the contribution of $\mathcal{A}_i$ to this execution.

Definition 195 (projection)**.** *Let $\mathbf{A}$ be a set of PTSIOA (resp. PTCA), let $\mathcal{A}_i \in \mathbf{A}$. We define projection operator $\upharpoonright$ recursively such as for every pseudo-execution α of $\mathbf{A}$, for every $\tau \in traj s(Q_{\mathbf{A}})$:*

$\alpha \upharpoonright \mathcal{A}_i = \tau \downarrow Q_{\mathcal{A}_i}$ *if* $\alpha = \tau$

$$(\alpha a \tau) \upharpoonright \mathcal{A}_i = \begin{cases} (\alpha \upharpoonright \mathcal{A}_i) a (\tau \upharpoonright \mathcal{A}_i) \text{ if } a \in \widehat{sig}(\mathcal{A}_i)(q \upharpoonright \mathcal{A}_i) \\ \pi'(\tau'^{\frown}(\tau \upharpoonright \mathcal{A}_i)) \text{ where } \alpha \upharpoonright \mathcal{A}_i = \pi' \tau' \text{otherwise.} \end{cases}$$

Having defined executions, traces and projection, we state the final axiom for PTSIOA. This axiom is more or less necessary to prove the measurability of a perception function. Henceforth, we assume that a PTSIOA satisfies

P (Progressive) $\mathcal{A}$ never generates infinitely many locally controlled actions within a finite time interval, i.e. $\forall \alpha \in Execs_{\mathcal{A}}$, with $\alpha.ltime < \infty$, α has finite local actions.

5.4. Global composition

Finally, we can formally define our operation of composition.

5.4.1. PTSIOA composition

Definition 196 (partially-compatible PTSIOA composition)**.** *Let* $\mathbf{A} = \{\mathcal{A}_1, ..., \mathcal{A}_n\}$ *be a set of pre-PTSIOA, with* $\forall i \in [1,n], \mathcal{A}_i = ((Q_{\mathcal{A}_i}, \mathcal{F}_{Q_{\mathcal{A}_i}}), \bar{q}_{\mathcal{A}_i}, sig(\mathcal{A}_i), D_{\mathcal{A}_i}, \mathcal{T}_{\mathcal{A}_i})$*. The automata in* $\mathbf{A}$ *are said* partially-compatible *if* $\mathbf{A}$ *is compatible at each reachable state* q *of* $\mathbf{A}$*. In that case, their partial-composition, noted* $\mathcal{A}_1||...||\mathcal{A}_n$*, is defined to be* $\mathcal{A} = ((Q_{\mathcal{A}}, \mathcal{F}_{Q_{\mathcal{A}}}), \bar{q}_{\mathcal{A}}, sig(\mathcal{A}), D_{\mathcal{A}}, \mathcal{T}_{\mathcal{A}})$*, where:*

- $Q_{\mathcal{A}} = Reachable(\mathbf{A})$
- $\mathcal{F}_{Q_{\mathcal{A}}} = sigma(\{F = F_1 \times ... \times F_n | (F_1, ..., F_n) \in \mathcal{F}_{Q_1} \times ... \times \mathcal{F}_{Q_n} \wedge F \subseteq Reachable(\mathbf{A}))$
- $\bar{q}_{\mathcal{A}} = (\bar{q}_{\mathcal{A}_1}, ..., \bar{q}_{\mathcal{A}_n})$
- $sig(\mathcal{A}) : q \in Q_{\mathcal{A}} \mapsto sig(\mathcal{A})(q) = sig(\mathbf{A})(q)$
- $D_{\mathcal{A}} = \{(q, a, \eta_{(\mathbf{A},q,a)}) | q \in Q_{\mathcal{A}}, a \in \widehat{sig}(\mathbf{A})(q)\}$
- $\mathcal{T}_{\mathcal{A}} = \bigcup_{q \in Q_{\mathcal{A}}} \mathcal{T}_{\mathcal{A}_1}(q \upharpoonright \mathcal{A}_1)||...||\mathcal{T}_{\mathcal{A}_n}(q \upharpoonright \mathcal{A}_n)$

Theorem 25 (pre-PTSIOA closure under composition)**.** *Let* $\mathcal{A}_1, \mathcal{A}_2$ *be partially-compatible pre-PTSIOA. Let* $\mathcal{A} = \mathcal{A}_1||\mathcal{A}_2$*. Then* $\mathcal{A}$ *is a pre-PTSIOA. Furthermore, if* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *satisfy* $\mathbf{D}$*, then* $\mathcal{A}$ *satisfies* $\mathbf{D}$ *too.*

Proof.
- $\forall q \in Q_{\mathcal{A}}$, $sig(\mathcal{A})(q)$ is a triplet of mutually disjoint set by definition of composition of signatures.
- Axiom $\mathbf{E}$ of enabled actions tracking is trivially preserved by construction since $D_{\mathcal{A}}$ is constructed with all the transitions of $\mathcal{A}_i$ for $i \in \{1,2\}$ that also satisfy $\mathbf{E}$.
- Let $\tau \in \mathcal{T}_{\mathcal{A}}$. Then $\tau = (\tau_1||\tau_2)$ for some $(\tau_1, \tau_2) \in \mathcal{T}_{\mathcal{A}_1} \circledast \mathcal{T}_{\mathcal{A}_2}$. A prefix (resp. suffix) τ' of τ, can be written $(\tau_1'||\tau_2')$ where τ_i' is a prefix (resp. suffix) of τ_i for each $i \in \{1,2\}$. Since $\mathcal{T}_{\mathcal{A}_i}$ is closed under prefix (resp. suffix) for each $i \in \{1,2\}$, $\tau' \in \mathcal{T}_{\mathcal{A}}$. Hence $\mathcal{T}_{\mathcal{A}}$ is closed under prefix (resp. suffix). In the same way, $\mathcal{T}_{\mathcal{A}}$ is closed under concatenation and $\forall q \in Q_{\mathcal{A}}, \wp(q) \in \mathcal{T}_{\mathcal{A}}$.
- $\mathbf{D}_1$: Let $q \in Q_{\mathcal{A}}$. Then $\exists (q_1, q_2) \in Q_{\mathcal{A}_1} \times Q_{\mathcal{A}_2}$, $q = (q_1, q_2)$ Let $\tau, \tau' \in \mathcal{T}_{\mathcal{A}}(q)$. Then $\tau = (\tau_1||\tau_2), \tau' = (\tau_1'||\tau_2')$ for some $(\tau_1, \tau_2), (\tau_1', \tau_2') \in \mathcal{T}_{\mathcal{A}_1}(q_1) \circledast \mathcal{T}_{\mathcal{A}_2}(q_2)$. By trajectory determinism satisfied by $\mathcal{A}_1$ and $\mathcal{A}_2$, (1) either $\tau_1' \leq \tau_1$ or $\tau_1 \leq \tau_1'$ and (2) either $\tau_2' \leq \tau_2$ or $\tau_2 \leq \tau_2'$. Moreover, $dom(\tau_1) = dom(\tau_2)$ and $dom(\tau_1') = dom(\tau_2')$. Thus either (a) $\tau_1' \leq \tau_1$ and $\tau_2' \leq \tau_2$ or (b) $\tau_1 \leq \tau_1'$ and $\tau_2 \leq \tau_2'$. Hence either $\tau' \leq \tau$ or $\tau \leq \tau'$.
- $\mathbf{D}_2$: Let $q \in Q_{\mathcal{A}}$. Then $\exists (q_1, q_2) \in Q_{\mathcal{A}_1} \times Q_{\mathcal{A}_2}$, $q = (q_1, q_2)$ Let $\tau \in \mathcal{T}_{\mathcal{A}}(q)$. Then $\tau = (\tau_1||\tau_2)$ for some $(\tau_1, \tau_2) \in \mathcal{T}_{\mathcal{A}_1}(q_1) \circledast \mathcal{T}_{\mathcal{A}_2}(q_2)$.

  Suppose some local action $a \in \widehat{loc}(\mathcal{A})(q) \cap enabled(\mathcal{A})(q)$. Let us assume without loss of generality that $a \in \widehat{loc}(\mathcal{A}_1)(q_1) \setminus \widehat{loc}(\mathcal{A}_2)(q_2)$. Then $a \in enabled(\mathcal{A}_1)(q_1)$ and since $\mathcal{A}_1$ satisfies $\mathbf{D}_2$, there does not exist any non-point trajectory in $\mathcal{T}_{\mathcal{A}_1}(q_1)$ (and therefore in $\mathcal{T}_{\mathcal{A}}(q)$) that starts from q_1. Likewise, if there exists a non-point trajectory starting from q, then no local action is enabled at q_1 or at q_2 ($\widehat{loc}(\mathcal{A}_1)(q_1) \cap enabled(\mathcal{A}_1)(q_1) = \widehat{loc}(\mathcal{A}_1)(q_1) \cap enabled(\mathcal{A}_2)(q_2) = \emptyset$) which leads to $\widehat{loc}(\mathcal{A})(q) \cap enabled(\mathcal{A})(q) = \emptyset$.
- $\mathbf{D}_3$ Immediate by construction

□

Remark 7. *Why* $\mathbf{M_1}$ *is not necessarily preserved by composition? Since* $E_a^{\mathcal{A}_1}$ *(resp.* $E_a^{\mathcal{A}_2}$*) satisfies* $\mathbf{M}_1$*,* $E_a^{\mathcal{A}_1} \in \mathcal{F}_{Q_{\mathcal{A}_1}}$ *(resp.* $E_a^{\mathcal{A}_2} \in \mathcal{F}_{Q_{\mathcal{A}_2}}$*). Moreover, by definition of sigma-field,* $Q_{\mathcal{A}_1} \in \mathcal{F}_{Q_{\mathcal{A}_1}}$ *and* $Q_{\mathcal{A}_2} \in \mathcal{F}_{Q_{\mathcal{A}_2}}$*. Hence (*)* $(E_a^{\mathcal{A}_1} \times Q_{\mathcal{A}_2}), (E_a^{\mathcal{A}_2} \times Q_{\mathcal{A}_1}) \in \{(F_1 \times F_2) | \forall i \in \{1,2\}, F_i \in \mathcal{F}_{Q_{\mathcal{A}_i}}\}$ *. Moreover, (**)* $(E_a^{\mathcal{A}_1} \times Q_{\mathcal{A}_2}) \cup (E_a^{\mathcal{A}_2} \times Q_{\mathcal{A}_1}) \in \mathcal{F}'$ *where* $\mathcal{F}'$ *is a sigma-field generated by* $\{(F_1 \times F_2) | \forall i \in \{1,2\}, F_i \in \mathcal{F}_{Q_{\mathcal{A}_i}}\}$ *.*

Let $E_a^{\mathcal{A}_1} \subseteq Q_{\mathcal{A}_1}$ *(resp.* $E_a^{\mathcal{A}_2} \subseteq Q_{\mathcal{A}_2}$*) be the set of states of* $\mathcal{A}_1$ *(resp.* $\mathcal{A}_2$*) where* a *is enabled. The set of states of* $\mathcal{A}$ *where* a *is enabled is* $E_a^{\mathcal{A}} = \{(q_1, q_2) \in Q_{\mathcal{A}} | q_1 \in E_a^{\mathcal{A}_1} \vee q_2 \in E_a^{\mathcal{A}_2}\}$*. Hence* $E_a^{\mathcal{A}} = [(E_a^{\mathcal{A}_1} \times Q_{\mathcal{A}_2}) \cup (E_a^{\mathcal{A}_2} \times Q_{\mathcal{A}_1})] \cap Q_{\mathcal{A}}$*. However, the reachability of sets* $(E_a^{\mathcal{A}_1} \times Q_{\mathcal{A}_2})$ *and* $(E_a^{\mathcal{A}_2} \times Q_{\mathcal{A}_1})$ *remains unclear and so it is unclear if these sets are in* $\mathcal{F}_{Q_{\mathcal{A}}}$*.*

5.4.2. Configuration composition

Union of timed configurations and compatible timed configurations are defined as in definition 76, where configurations are replaced by timed configurations. Composition of partially-compatible PTCA is defined as in definition 77.

Proposition 13. *Let* $C_1 = (\mathbf{A}_1, \mathbf{S}_1)$ *and* $C_2 = (\mathbf{A}_2, \mathbf{S}_2)$ *be compatible configurations such that* $\mathbf{A}_1 \cap \mathbf{A}_2 = \emptyset$*. Let* $C = (\mathbf{A}, \mathbf{S}) = C_1 \cup C_2$ *be a compatible configuration. Then* $sig(C) = sig(C_1) \times sig(C_2)$*, as per Definition 59.*

Proof. The proof is the same one as the proof of lemma 3. □

The next lemma exhibits a direct homomorphism between (i) the pair of sets of trajectories of a pair of configuration (C_1, C_2) and (ii) the resulting set of trajectories of the configuration $C_1 \cup C_2$.

Lemma 73 (union is an homomorphism from $\mathcal{T}_{C_1} \circledast \mathcal{T}_{C_2}$ to $\mathcal{T}_C$)**.** *Let* $C_1 = (\mathbf{A}_1, \mathbf{S}_1)$ *and* $C_2 = (\mathbf{A}_2, \mathbf{S}_2)$ *be compatible configurations such that* $\mathbf{A}_1 \cap \mathbf{A}_2 = \emptyset$*. Let* $C = (\mathbf{A}, \mathbf{S}) = C_1 \cup C_2$ *be a compatible configuration. Then, there is a bijection* u *from* $\mathcal{T}_{C_1} \circledast \mathcal{T}_{C_2}$ *to* $\mathcal{T}_C$ *such that* $\forall((\tau_1, \tau_2), \tau) \in (\mathcal{T}_{C_1} \circledast \mathcal{T}_{C_2}) \times \mathcal{T}_C$*, with* $\tau = u(\tau_1, \tau_2)$ *we have :*

- $dom(\tau_1) = dom(\tau_2) = dom(\tau) \triangleq J$
- $\forall t \in J$*,* $\tau(t) = \tau_1(t) \cup \tau_2(t)$

Proof. By definition, we have $\mathcal{T}_C \overset{w,v}{\leftrightsquigarrow} \mathcal{T}_{\mathbf{A}}(TS(C))$, $\mathcal{T}_{C_1} \overset{w_1,v_1}{\leftrightsquigarrow} \mathcal{T}_{\mathbf{A}_1}(TS(C_1))$, $\mathcal{T}_{C_2} \overset{w_2,v_2}{\leftrightsquigarrow} \mathcal{T}_{\mathbf{A}_2}(TS(C_2))$ with $v = v_1 = v_2 = TS$.

Also, by definition, $\mathcal{T}_{\mathbf{A}}(TS(C)) = \mathcal{T}_{\mathbf{A}_1}(TS(C_1)) || \mathcal{T}_{\mathbf{A}_2}(TS(C_2)) \triangleq \{(\tau_1' || \tau_2') | (\tau_1', \tau_2') \in \mathcal{T}_{\mathbf{A}_1}(TS(C_1)) \circledast \mathcal{T}_{\mathbf{A}_2}(TS(C_2))\}$.

Moreover $(\tau_1' || \tau_2') = (\tau_1'' || \tau_2'')$ implies $\tau_1' = \tau_1''$ and $\tau_2' = \tau_2''$.

Thus, the function

$$tuple' : \left\{ \begin{array}{lcl} \mathcal{T}_{\mathbf{A}_1}(TS(C_1)) \circledast \mathcal{T}_{\mathbf{A}_2}(TS(C_2)) & \to & \mathcal{T}_{\mathbf{A}}(TS(C)) \\ (\tau_1', \tau_2') & \mapsto & \tau_1' || \tau_2' \end{array} \right. \text{ is a bijection.}$$

$$\text{Hence } f : \left\{ \begin{array}{lcl} \mathcal{T}_{C_1} \circledast \mathcal{T}_{C_2} & \to & \mathcal{T}_C \\ (\tau_1, \tau_2) & \mapsto & w^{-1}(w_1(\tau_1) || w_2(\tau_2)) \end{array} \right. \text{ is a bijection.}$$

By definition, w, w_1, w_2 preserves the domain of related functions.

It remains to show that $\forall(\tau_1, \tau_2) \in \mathcal{T}_{C_1} \circledast \mathcal{T}_{C_2}$, the trajectory $\tau \triangleq w^{-1}(w_1(\tau_1) || w_2(\tau_2))$ verifies $\tau = u(\tau_1, \tau_2)$, i.e. $\forall t \in dom(\tau_1) = dom(\tau_2) \triangleq J$, $\tau(t) = \tau_1(t) \cup \tau(t)$.

Since w is a bijection, it is equivalent to show that $\forall(\tau_1,\tau_2) \in \mathcal{T}_{C_1} \circledast \mathcal{T}_{C_2}$, the trajectory $\tau' \triangleq (w_1(\tau_1)||w_2(\tau_2))$ verifies $\tau' = w(u(\tau))$, i.e.

$\forall t \in J$, $(w_1(\tau_1)||w_2(\tau_2))(t) = w(u(\tau_1,\tau_2))(t)$.

By definition $(w_1(\tau_1)||w_2(\tau_2))(t) = (TS(\tau_1(t)), TS(\tau_2(t)))$ and $w(u(\tau_1,\tau_2))(t) = TS(\tau_1(t) \cup \tau_2(t))$. Finally, the definition of union of configuration ensures $(TS(\tau_1(t)), TS(\tau_2(t))) = TS(\tau_1(t) \cup \tau_2(t))$ which allows us to conclude the expected result.

□

We can extend the theorem 2 of closure of the set of the PCA under composition to pre-PTCA, where we just have to show that trajectory preservation is preserved by composition.

Theorem 26 (pre-PTCA closure under composition)**.** *Let X_1, X_2 be partially-compatible pre-PTCA. Let $X = X_1||X_2$. Then X is a pre-PTCA.*

Proof. By theorem 25, $ptsioa(X)$ is a pre-PTSIOA. We need to show that the constraints of the definition of pre-PTCA are still verified.

- The preservation of constraint of (i) start state preservation, (ii) transition preservation, (iii) signature preservation, have been proved in the proof of theorem 2.
- constraint of trajectory preservation:
  Let $(q = (q_1,q_2), C) \in Q_X \times Q_{conf}$ s.t. $C = config(X)((q_1,q_2))$. We need to show that $\mathcal{T}_X(q) \overset{u,c}{\rightsquigarrow} \mathcal{T}_C$ with $c = config(X)$.
  We note $C_1 = config(X_1)(q_1)$ and $C_2 = config(X_2)(q_2)$. Clearly $C = C_1 \cup C_2$.
  Since X_1 and X_2 are both PTCA, they both satisfy the constraint of trajectory preservation. Hence, $\mathcal{T}_{X_1}(q_1) \overset{u_1,c_1}{\rightsquigarrow} \mathcal{T}_{C_1}$ and $\mathcal{T}_{X_2}(q_2) \overset{u_2,c_2}{\rightsquigarrow} \mathcal{T}_{C_2}$ with $c_1 = config(X_1)$ and $c_2 = config(X_2)$. We will prove the expected result with $u : \tau \in \mathcal{T}_X(q) \mapsto (u_1(\tau \downarrow Q_{X_1}) \cup (u_2(\tau \downarrow Q_{X_2}))$, where $(\tau_1' \cup \tau_2')(t) = \tau_1'(t) \cup \tau_2'(t)$.
  By definition of composition of PTSIOA, $\mathcal{T}_X(q) = \mathcal{T}_{X_1}(q_1)||\mathcal{T}_{X_2}(q_2) \triangleq \{(\tau_1||\tau_2) | (\tau_1,\tau_2) \in \mathcal{T}_{X_1}(q_1) \circledast \mathcal{T}_{X_2}(q_2)\}$.
  Moreover, $\forall(\tau_1,\tau_2),(\tau_1',\tau_2') \in \mathcal{T}_{X_1}(q_1) \circledast \mathcal{T}_{X_2}(q_2)$, $(\tau_1'||\tau_2') = (\tau_1||\tau_2)$ implies $(\tau_1',\tau_2') = (\tau_1,\tau_2)$. Thus,
  $f : \left\{ \begin{array}{lcl} \mathcal{T}_X(q) & \to & \mathcal{T}_{X_1}(q_1) \circledast \mathcal{T}_{X_2}(q_2) \\ \tau & \mapsto & (\tau \downarrow Q_{X_1}, \tau \downarrow Q_{X_2}) \end{array} \right.$ is a bijection.
  By lemma 73
  $g : \left\{ \begin{array}{lcl} \mathcal{T}_{C_1} \circledast \mathcal{T}_{C_2} & \to & \mathcal{T}_C \\ (\tau_1',\tau_2') & \mapsto & \tau_1' \cup \tau_2' \end{array} \right.$ is a bijection.
  Hence $u : \left\{ \begin{array}{lcl} \mathcal{T}_X(q) & \to & \mathcal{T}_C \\ \tau & \mapsto & (u_1(\tau \downarrow Q_{X_1}) \cup (u_2(\tau \downarrow Q_{X_2})) \end{array} \right.$ is a bijection.

Let $(\tau,\tau') \in \mathcal{T}_X(q) \times \mathcal{T}_C$ with $\tau' = u(\tau)$ and $t \in dom(\tau) = dom(\tau') \triangleq J$. We need to show that $c(\tau(t)) = \tau'(t)$. We note $(\tau_1,\tau_2) = f(\tau)$, $\tau_1' = u_1(\tau_1)$, $\tau_2' = u_2(\tau_2)$, $\tau' = g(\tau_1',\tau_2')$ We have $\tau'(t) = \tau_1'(t) \cup \tau_2'(t) = c_1(\tau_1(t)) \cup c_2(\tau_2(t)) = config(X_1)(\tau_1(t)) \cup config(X_2)(\tau_2(t)) = config(X_1)((\tau \downarrow Q_{X_1})(t)) \cup config(X_2)((\tau \downarrow Q_{X_1})(t) = config(X)(\tau(t)) = c(\tau(t))$ which ends the proof.

□

Corollary 1. *Let X_1, X_2 be partially-compatible PTCA (resp. post-PTCA). Let $X = X_1||X_2$. If X satisfies axiom* **M** *(resp.* **M** *and* **M**′*) of measurability, then X is a PTCA (resp. post-PTCA).*

5.5. Measure over executions, scheduler, implementation

In this section, we adapt the section 7.4 of Sayan Mitra PhD thesis [Mit07]. Basically, there are not a lot of differences. We choose a more general scheduler and a more general implementation definition. The sigma-field over executions (generated by cylinders of executions instead of cones of executions) is the same. However, the measurability of a perception function defined on this σ-field is proved only for post-PTCA (i.e. under additional measurability axiom $\mathbf{M}'$) and a particular class of environment, called closing environment, because of the dynamicity of the signature.

5.5.1. Ring on Executions and Traces Definition

The continuity of the involved state spaces and the the continuity of the time axis require more efforts to build a σ-algebra on the executions of the automata. This σ-algebra will be generated by basic sets, a.k.a. cylinders, of executions instead of cones.

Definition 197 (base)**.** *A* base *for a pre-PTSIOA (resp. pre-PTCA) $\mathcal{A}$ is a finite sequence of the form $\Lambda = Q_0R_0Q'_0A_1Q_1R_1Q'_1A_2Q_2...Q'_{m-1}A_mQ_mR_mQ'_m$, where for every $i \in \{0,...,m\}$, $Q_i, Q'_i \in \mathcal{F}_{Q_\mathcal{A}}$, R_i is a measurable set in $R^{\geq 0}$ and for every $i \in \{1,...,m\}$, $A_i \subseteq acts(\mathcal{A})$. The length of a base is the number of sets of actions in the sequence. The set of bases for $\mathcal{A}$ is noted $Bases(\mathcal{A})$.*

Definition 198 (basic set)**.** *Let $\mathcal{A}$ be a pre-PTSIOA (resp. pre-PTCA). The basic set (a.k.a. cylinder) corresponding to a base $\Lambda \in Bases(\mathcal{A})$ is a set of execution fragments of $\mathcal{A}$, $C_\Lambda = \{\tau_0a_1\tau_1...\tau_m\widehat{\ }\alpha | \alpha \in Frags_\mathcal{A}, \tau_0.fstate \in Q_0, \forall i \in \{0,...,m\}, \tau_i.ltime \in R_i, \tau_i.lstate \in Q'_i, \forall i \in \{1,...,m\}, a_i \in A_i\}$. The set of basic sets is noted $Cyls(\mathcal{A}) \triangleq \{C_\Lambda | \Lambda \in Bases(\mathcal{A})\}$.*

If $\Lambda = Q_0R_0Q'_0...Q'_{m-1}A_mQ_mR_mQ'_m$, $\Lambda_1 = Q_0R_0Q'_0...Q'_{m-1}A_mQ_m$, and $\Lambda_2 = Q_0R_0Q'_0...Q'_{m-1}$, we will abbreviate Λ as $\Lambda_1R_mQ'_m$ or as $\Lambda_2A_mQ_mR_mQ'_m$.

Lemma 74 (Ring over $Frags_\mathcal{A}$)**.** *The collection $Cyls(\mathcal{A})$ of all basic sets of $\mathcal{A}$ is a ring.*

Proof. See [Mit07], lemma 7.9, section 7.4, pages 143-144. □

The σ-algebra generated by $Cyls(\mathcal{A})$ is denoted by $\mathcal{F}_{Frags_\mathcal{A}}$. The collection of sets obtained by taking the intersection of each element in $Cyls(\mathcal{A})$ with $Execs_\mathcal{A}$ is a ring in $Execs_\mathcal{A}$. We denote the σ-algebra generated by this ring by $\mathcal{F}_{Execs_\mathcal{A}}$, i.e. $\mathcal{F}_{Execs_\mathcal{A}} = sigma(\{C \cap Execs_\mathcal{A} | C \in Cyls(\mathcal{A})\})$. We define the measurable space of executions of $\mathcal{A}$ to be $(Execs_\mathcal{A}, \mathcal{F}_{Execs_\mathcal{A}})$. By restricting basic sets to finite execution fragments, we define finite basic sets. The collection of all finite basic sets forms a semi-ring over set $Frags^*_\mathcal{A}$ of finite execution fragments of $\mathcal{A}$. The σ-algebra on $Frags^*_\mathcal{A}$, and the measurable space $(Execs^*_\mathcal{A}, \mathcal{F}_{Execs^*_\mathcal{A}})$ of finite executions are defined in a manner identical to the above constructions.

Aside concerning traces It is possible to define a σ-algebra on the traces of an automaton in a similar way

Definition 199. *A* trace base *is a finite sequence of the form $\Lambda = R_0E_1R_1E_2...R_{n-1}E_n$ where $\forall i \in \{O,...,n-1\}$, R_i, is a measurable set in $\mathbb{R}^{\geq 0}$ and $\forall j \in \{1,...,n\}, E_j \subseteq \bigcup_{q \in Q_\mathcal{A}} \widehat{ext}(\mathcal{A})(q)$. The length of such a trace base is defined to be n. The* trace basic set *corresponding to the base $\mathcal{A}$ is a set of traces of $\mathcal{A}$ defined as: $C_\Lambda = \{\tau^0a^1\tau^1...a^n\beta \in Traces_\mathcal{A} | \forall i \in \{0,...,n\}, \tau^i.ltime \in R_i, \forall i \in \{1,...,n\}, a \in E_i\}$.*

The collection $\mathscr{D}$ of all trace basic sets of $\mathcal{A}$ is a semi-ring. The σ-algebra $\mathcal{F}_{Traces}$ on the set of traces of $\mathcal{A}$ is defined as the σ-algebra generated by the collection of trace basic sets; the measurable space of traces is denoted by $(Traces_{\mathcal{A}}, \mathcal{F}_{Traces_{\mathcal{A}}})$.

In Mitra's thesis [Mit07] (theorem 7.14, section 7.4.3, page 151.), it is proved that the trace function $trace_{\mathcal{A}} : (Execs_{\mathcal{A}}, \mathcal{F}_{Execs_{\mathcal{A}}}) \to (Traces_{\mathcal{A}}, \mathcal{F}_{Traces_{\mathcal{A}}})$ is measurable where $\mathcal{A}$ is a PTIOA, i.e. a DPTIOA with a static signature. However, the proof of this theorem cannot be adapted directly. Hence, we propose to prove the measurability of the projection function under additional axioms. We stress that the theorem 27 of measurability of projection, and all the intermediate lemma to prove it are a direct adaptation of Mitra's thesis [Mit07] (section 7.4.3, page 147-151).

Measurability of projection function First, we define the closed environment.

Definition 200 (closed environment)**.** *Let $\mathcal{A}$ be a pre-PTSIOA (resp. PTSIOA, resp. post-PTSIOA,resp. pre-PTCA,resp. PTCA,resp. post-PTCA). A closing environment $\mathcal{E}$ of $\mathcal{A}$ is a pre-PTSIOA (resp. PTSIOA, resp. post-PTSIOA,resp. pre-PTCA,resp. PTCA,resp. post-PTCA) partially-compatible with $\mathcal{A}$ such that $\forall q \in Q_{\mathcal{E}||\mathcal{A}}$, $\forall a \in \widehat{sig}(\mathcal{E}||\mathcal{A})(q)$,*

$a \in \widehat{ext}(\mathcal{A})(q \upharpoonright \mathcal{A}) \Longrightarrow a \in \widehat{ext}(\mathcal{E})(q \upharpoonright \mathcal{E})$.

We note $cenv(\mathcal{A})$, the set of closing environment of $\mathcal{A}$.

Such an environment never misses an external action of $\mathcal{A}$. We can remark that a good start to allow the existence of closing environment of $\mathcal{A}$ is to require that the signature of $\mathcal{A}$ does not increase along a trajectory and after an internal action. The point would be always warning a modification of its signature to its environment.

We can adapt the proof of Mitra [Mit07] to the projection function.

First, we define canonical bases in the same way as Mitra does for canonical trace bases (definition 7.10 p148 in [Mit07]).

Definition 201 (canonical base)**.** *A base of the form $\bar{\Lambda} = Q_0 R_0 Q'_0 A_1 Q_1 R_1 Q'_1 A_2 Q_2 ... Q'_{m-1} A_m Q_m R_m Q'_m$ is said* canonical *if $\forall i \in [1:m]$, $R_i = [0, b_i[$ with $b_i \in \mathbb{R}^{\geq 0}$. We note $Canons(\mathcal{A})$ the set of canonical bases of a PTSIOA (resp. PTCA) $\mathcal{A}$.*

The next lemma 75 (like lemma 7.11 p148 in [Mit07]) will simplify the task.

Lemma 75 (measurability on canonical basic sets is enough)**.** *Let $\mathcal{A}, \mathcal{E}$ be partially-compatible pre-PTSIOA (resp. PTSIOA, resp. post-PTSIOA,resp. pre-PTCA,resp. PTCA,resp. post-PTCA). Let $f : (Execs(\mathcal{E}||\mathcal{A}), \mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}) \to (Execs(\mathcal{E}), \mathcal{F}_{Execs(\mathcal{E})})$. If $\forall \bar{\Lambda} \in Canons(\mathcal{E}||\mathcal{A})$, $f^{-1}(C_{\bar{\Lambda}}) \in \mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}$, then $\forall \Lambda \in Bases(\mathcal{E}||\mathcal{A})$, $f^{-1}(C_\Lambda) \in \mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}$.*

Proof. The proof adapts the strategy of lemma 7.11 p148 in [Mit07].

We note $\mathscr{C} \triangleq \{E \subseteq Execs(\mathcal{E}||\mathcal{A}) | f^{-1}(E) \in \mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}\}$. It is easy to show that $\mathscr{C}$ is a σ-algebra on $Execs(\mathcal{E}||\mathcal{A})$:

- $f^{-1}(Execs(\mathcal{E})) = Execs(\mathcal{E}||\mathcal{A}) \in \mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}$ and $f^{-1}(\emptyset) = \emptyset \in \mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}$. Thus, $\emptyset, Execs(\mathcal{E}) \in \mathscr{C}$
- $\forall E \in \mathscr{C}$, $f^{-1}(Execs(\mathcal{E}) \setminus E) = (Execs(\mathcal{E}||\mathcal{A}) \setminus f^{-1}(E)) \in \mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}$ since $\mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}$ is a σ-algebra, and so, is stable by complementation. Hence $\forall E \in \mathscr{C}, (Execs(\mathcal{E}) \setminus E) \in \mathscr{C}$.

– $\forall (E_i)_{i\in\mathbb{N}} \in (\mathscr{C})^{\mathbb{N}}$, $f^{-1}(\bigcup_{i\in\mathbb{N}} E_i) = \bigcup_{i\in\mathbb{N}} f^{-1}(E_i) \in \mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}$, since $\mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}$ is a σ-algebra, and so, is stable by countable union. Hence $\forall (E_i)_{i\in\mathbb{N}} \in (\mathscr{C})^{\mathbb{N}}$, $\bigcup_{i\in\mathbb{N}} E_i \in \mathscr{C}$.

Next, we show that if $\forall \bar{\Lambda} \in Canons(\mathcal{E}||\mathcal{A})$, $f^{-1}(C_{\bar{\Lambda}}) \in \mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}$, then $\forall \Lambda \in Bases(\mathcal{E}||\mathcal{A})$, $f^{-1}(C_\Lambda) \in \mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}$, that is, if $\forall \bar{\Lambda} \in Canons(\mathcal{E}||\mathcal{A})$, $C_{\bar{\Lambda}} \in \mathscr{C}$, then $\forall \Lambda \in Bases(\mathcal{E}||\mathcal{A})$, $C_\Lambda \in \mathscr{C}$.

We show the result for base length of 1.

The assumption implies that for every canonical base $\bar{\Lambda}_1 = Q_0R_0Q_0'A_1Q_1 \in Canons(\mathcal{E})$, $C_{\bar{\Lambda}_1} \in \mathscr{C}$.

First, we will show that for every base $\Lambda_1^{[b,\infty[} = P_0S_0P_0'B_1P_1 \in Bases(\mathcal{E})$, with $S_0 = [b,\infty[$, $C_{\Lambda_1^{[b,\infty[}} \in \mathscr{C}$. It is enough to remark that $C_{\Lambda_1^{[b,\infty[}} = (\bigcup_{i\in\mathbb{N}} C_{\Lambda_1^{[0,i[}}) \setminus C_{\Lambda_1^{[0,b[}}$ with $\Lambda_1^{[0,x[} = P_0[0,x[P_0'B_1P_1 \in Canons(\mathcal{E})$ for each $x \in \mathbb{N} \cup \{b\}$. So $C_{\Lambda_1^\infty} \in \mathscr{C}$ since $\mathscr{C}$ is a σ-algebra and cylinders corresponding to canonical bases are assumed to belong to $\mathscr{C}$.

Second, we show that for every base $\Lambda_1^{[0,b]} = P_0S_0P_0'B_1P_1 \in Bases(\mathcal{E})$, with $S_0 = [0,b]$, $C_{\Lambda_1^{[0,b]}} \in \mathscr{C}$. To do so, we choose $(b_i)_{i\in\mathbb{N}} \in (\mathbb{R}^{\geq 0})^{\mathbb{N}}$ with $\lim_{i\to\infty} b_i = b$, $\forall i \in \mathbb{N}, b_i < b_{i+1}$. Then, it suffices to remarks that $C_{\Lambda_1^{[0,b]}} = \bigcap_{i\in\mathbb{N}} C_{\Lambda_1^{[0,b_i[}}$, where $\forall i \in \mathbb{N}$, $\Lambda_1^{[0,b_i[} = P_0[0,b_i[P_0'B_1P_1 \in Canons(\mathcal{E})$.

Third, for every base $\Lambda_1^{[a,b]} = P_0S_0P_0'B_1P_1 \in Bases(\mathcal{E})$, with $S_0 = [a,b]$, $C_{\Lambda_1^{[a,b]}} = C_{\Lambda_1^{[0,b]}} \cap C_{\Lambda_1^{[a,\infty[}} \in \mathscr{C}$, where $\Lambda_1^{[0,b]} = P_0[0,b]P_0'B_1P_1$ and $\Lambda_1^{[a,\infty[} = P_0[a,\infty[P_0'B_1P_1$.

Fourth, for every base $\Lambda_1^{[a,b[} = P_0S_0P_0'B_1P_1 \in Bases(\mathcal{E})$, with $S_0 = [a,b[$, $C_{\Lambda_1^{[a,b]}} = C_{\Lambda_1^{[0,b[}} \cap C_{\Lambda_1^{[a,\infty[}} \in \mathscr{C}$, where $\Lambda_1^{[0,b[} = P_0[0,b[P_0'B_1P_1$ and $\Lambda_1^{[a,\infty[} = P_0[a,\infty[P_0'B_1P_1$.

Fifth, for every base $\Lambda_1^{]a,b]} = P_0S_0P_0'B_1P_1 \in Bases(\mathcal{E})$, with $S_0 =]a,b]$, $C_{\Lambda_1^{]a,b]}} = (\bigcap_{i\in\mathbb{N}} C_{\Lambda_1^{[a_i,b]}}) \in \mathscr{C}$, where $\Lambda_1^{[a_i,b]} = P_0[a_i,b]P_0'B_1P_1$ for each i, and $(a_i)_{i\in\mathbb{N}} \in (\mathbb{R}^{\geq 0})^{\mathbb{N}}$ with $\lim_{i\to\infty} a_i = a$, $\forall i \in \mathbb{N}, a_i > a_{i+1}$ and $a_0 = b$.

Sixth, for every base $\Lambda_1^{]a,b[} = P_0S_0P_0'B_1P_1 \in Bases(\mathcal{E})$, with $S_0 =]a,b[$, $C_{\Lambda_1^{]a,b[}} = C_{\Lambda_1^{[a,b[}} \cap C_{\Lambda_1^{]a,b]}} \in \mathscr{C}$, where $\Lambda_1^{[a,b[} = P_0[a,b[P_0'B_1P_1$ and $\Lambda_1^{]a,b]} = P_0]a,b]P_0'B_1P_1$.

So, for every $\Lambda_1 = Q_0R_0Q_0'A_1Q_1 \in Bases(\mathcal{E})$, $C_{\Lambda_1} \in \mathscr{C}$.

The result for any base length $m \in \mathbb{N}$ is obtained with the same construction applied to every R_i. The operations performed in a countable fashion in the case $m = 1$, are then still performed in a countable fashion for every $m \in \mathbb{N}$.

□

Thanks to previous lemma 75, we can focus on canonical basic sets only.

We want to establish the measurability of percpetion function for canonical basic sets of $Cyls(\mathcal{E})$ only. To do so, we need to obtain a counterpart in $Cyls(\mathcal{E}||\mathcal{A})$ such that the executions in this counterpart can "explain" the executions in a canonical basic set of $Cyls(\mathcal{E})$. This counterpart will be built with the set $\mathfrak{Q}_{m,r}$ defined in the next definition 202, strongly inspired by definition 7.11 p 149 in [Mit07]. The dynamic signatures led us to introduce the axioms $\mathbf{M}_3$ and $\mathbf{M}_4$ for post-PTSIOA (resp. post-PTCA).

Definition 202 (m-explanation of a canonical base)**.** *Let $\mathcal{A}$ be a post-PTSIOA or a post-PTCA. Let*

$\mathcal{E} \in cenv(\mathcal{A})$. Let $\bar{\Lambda}_1 = Q^0_{\mathcal{E}} R^0_{\mathcal{E}} Q'^0_{\mathcal{E}} A^1_{\mathcal{E}} Q^1_{\mathcal{E}} \in Canons(\mathcal{E})$ with $R^0_{\mathcal{E}} = [0, r[$ and $r \in \mathbb{R}^{>0}$. For every $m \in \mathbb{N}$ a m-explanation of $\bar{\Lambda}_1$ is a base of length m of the form:

$\Lambda^{m,r}_{q_1,q'_1,\ldots,q_m,q'_m} = Q^0_{\mathcal{E}||\mathcal{A}}[q_1, q'_1] Q'^0_{\mathcal{E}||\mathcal{A}} A^1_{\mathcal{E}||\mathcal{A}} Q^1_{\mathcal{E}||\mathcal{A}}[q_2, q'_2] Q'^1_{\mathcal{E}||\mathcal{A}} A^2_{\mathcal{E}||\mathcal{A}} Q^2_{\mathcal{E}||\mathcal{A}} \ldots [q_m, q'_m] Q'^{(m-1)}_{\mathcal{E}||\mathcal{A}} A^m_{\mathcal{E}||\mathcal{A}} Q^m_{\mathcal{E}||\mathcal{A}}$ *with:*

- *(specified edges)*
  - $A^m_{\mathcal{E}||\mathcal{A}} = A^1_{\mathcal{E}} \cap acts(\mathcal{A})$
  - $Q^0_{\mathcal{E}||\mathcal{A}} = Q^0_{\mathcal{E}} \times Q_{\mathcal{A}}$
  - $Q'^{(m-1)}_{\mathcal{E}||\mathcal{A}} = Q'^0_{\mathcal{E}} \times Q_{\mathcal{A}}$
  - $Q^m_{\mathcal{E}||\mathcal{A}} = Q^1_{\mathcal{E}} \times Q_{\mathcal{A}}$
- *(unspecified intermediate states)*
  - $\forall j \in [0 : m-2]$, $Q'^j_{\mathcal{E}||\mathcal{A}} = F_{[q_{j+1}, q'_{j+1}], Q^j_{\mathcal{E}||\mathcal{A}}}$ *(in the sense of axiom $\mathbf{M}_3$ of post-PTSIOA)*
  - $\forall j \in [1 : m-1]$ $Q^j_{\mathcal{E}||\mathcal{A}} = H_{Q'^{j-1}_{\mathcal{E}||\mathcal{A}}}$ *(in the sense of axiom $\mathbf{M}_4$ of post-PTSIOA)*
- *(no shared action before $A^m_{\mathcal{E}||\mathcal{A}}$)*
  - $\forall j \in [1 : m-1], A^j_{\mathcal{E}||\mathcal{A}} = \bigcup_{q \in Q'^{j-1}_{\mathcal{E}||\mathcal{A}}} int(\mathcal{A})(q \restriction \mathcal{A})$
- *(partially specified time control)*
  - $\forall j \in [1 : m], q_j, q'_j \in \mathbb{Q}^{\geq 0}$
  - $\forall j \in [1 : m], q_j < q'_j$
  - $\sum_{j \in [1:m]} q'_j < r$

We note $\mathfrak{Q}_{m,r} \triangleq \bigcup_{q_1,q'_1,\ldots,q_m,q'_m \in \mathbb{Q}^{\geq 0}} C_{\Lambda^{m,r}_{q_1,q'_1,\ldots,q_m,q'_m}}$

The next lemma shows that perception function is measurable for the canonical basic sets with one action only. This lemma and its proof corresponds to the lemma 7.12 p. 150 in [Mit07].

Lemma 76 (perception is measureable on canonical basic sets with length 1)**.** *Let $\mathcal{A}$ be a post-PTSIOA or a post-PTCA. Let $\mathcal{E} \in cenv(\mathcal{A})$. Let $\bar{\Lambda}_1 = Q^0_{\mathcal{E}} R^0_{\mathcal{E}} Q'^0_{\mathcal{E}} A^1_{\mathcal{E}} Q^1_{\mathcal{E}} \in Canons(\mathcal{E})$ with $R^0_{\mathcal{E}} = [0, r[$ and $r \in \mathbb{R}^{>0}$.*

$proj^{-1}_{(\mathcal{E},\mathcal{A})}(C_{\bar{\Lambda}_1}) = \bigcup_{m \in \mathbb{N}} \mathfrak{Q}_{m,r}$

Proof. First, we show that $\bigcup_{m \in \mathbb{N}} \mathfrak{Q}_{m,r} \subseteq proj^{-1}_{(\mathcal{E},\mathcal{A})}(C_{\bar{\Lambda}_1})$. Let $\alpha \in \mathfrak{Q}_{m,r}$, for some $m \in \mathbb{N}$. The constraints imposes $\alpha = \alpha' a_{\wp}(q^1_{\mathcal{E}||\mathcal{A}})$ with

- $a \in A^1_{\mathcal{E}}$
- $q^1_{(\mathcal{E}||\mathcal{A})} \restriction \mathcal{E} \in Q^1_{\mathcal{E}}$
- $lstate(\alpha) \restriction \mathcal{E} \in Q'^0_{\mathcal{E}}$
- $fstate(\alpha) \restriction \mathcal{E} \in Q^0_{\mathcal{E}}$
- $\alpha.ltime \in [0, \sum_{j \in [1:m]} q'_j]$ and so $\alpha.ltime \in [0, r[$

– $actions(\alpha') \cap \widehat{sig}(\mathcal{E})(fstate(\alpha) \upharpoonright \mathcal{E}) = \emptyset$. This last claim comes from the fact that $\mathcal{E}$ and $\mathcal{A}$ are partially-compatible and so the internal signature of $\mathcal{A}$ and the signature of $\mathcal{E}$ are disjoint for any reachable state.

It follows that $(\alpha \upharpoonright \mathcal{E}) \in C_{\bar{\Lambda}_1}$.

Second, we show that $proj^{-1}_{(\mathcal{E},\mathcal{A})}(C_{\bar{\Lambda}_1}) \subseteq \bigcup_{m\in\mathbb{N}} \mathfrak{Q}_{m,r}$.

Let $\alpha \in Execs((\mathcal{E}||\mathcal{A}))$, such that $\alpha_{\mathcal{E}} \triangleq (\alpha \upharpoonright \mathcal{E}) \in C_{\bar{\Lambda}_1}$ Since $\alpha_{\mathcal{E}}.ltime < r$ and $\alpha.ltime = \alpha_{\mathcal{E}}.ltime$, we have $\alpha.ltime < r$. The axiom $\mathbf{P}_1$ of progressiveness implies that α counts m actions for some $m \in \mathbb{N}$. Moreover, since $\alpha_{\mathcal{E}} \in C_{\bar{\Lambda}_1}$, $\alpha_{\mathcal{E}}$ counts only one action $a^m \in A^1_{\mathcal{E}}$ and so $\alpha = \alpha' a^m \wp(q^1_{\mathcal{E}||\mathcal{A}})$ where

– $q^1_{\mathcal{E}||\mathcal{A}} \upharpoonright \mathcal{E} \in Q^1_{\mathcal{E}}$.
– $fstate(\alpha') \upharpoonright \mathcal{E} \in Q^0_{\mathcal{E}}$
– $lstate(\alpha') \upharpoonright \mathcal{E} \in Q'^0_{\mathcal{E}}$
– $actions(\alpha') = \tau^0 a^1 ... \tau^{m-1}$ with $\forall j \in [1 : m-1] a^j \in int(\mathcal{A})(lstate(\tau^{j-1}) \upharpoonright \mathcal{A})$. This comes from the fact that $\mathcal{E} \in cenv(\mathcal{A})$. Thus if $a^j \in \widehat{ext}(\mathcal{A})(lstate(\tau^{j-1}) \upharpoonright \mathcal{A})$, then $a^j \in \widehat{ext}(\mathcal{A})(lstate(\tau^{j-1}) \upharpoonright \mathcal{E})$ and so $\alpha_{\mathcal{E}} \notin C_{\bar{\Lambda}_1}$.

We note $\alpha = \tau^0 a^1 \tau^1 ... a^m \wp(q^1_{\mathcal{E}||\mathcal{A}})$. We note $\varepsilon = \dfrac{r - \alpha.ltime}{m+1}$. By density of $\mathbb{Q}^{\geq 0}$ in $\mathbb{R}^{\geq 0}$, $\forall j \in [0, m-1]$, $\exists (q_{j+1}, q'_{j+1}) \in \mathbb{Q}^{\geq 0}$, such that $\tau^j.ltime \in [q_{j+1}, q'_{j+1}]$ with $|q'_{j+1} - \tau^j.ltime| < \varepsilon$ and so $\sum_{j\in[1:m]} q'_j < r$.

Hence, $\alpha \in C_\Lambda$ with $\Lambda = \Lambda^{m,r}_{q_1,q'_1,...,q_m,q_{m'}}$, that is $\alpha \in \mathfrak{Q}_{m,r}$. □

The next lemma 77 is just the generalization of lemma 76 for any length m. Here again, this lemma and its proof are strongly inspired by lemma 7.13 p. 150 in [Mit07].

Lemma 77 (perception is measureable on canonical basic sets)**.** *Let $\mathcal{A}$ be a post-PTSIOA or a post-PTCA. Let $\mathcal{E} \in cenv(\mathcal{A})$. Let $\bar{\Lambda}_m = Q^0_{\mathcal{E}} R^0_{\mathcal{E}} Q'^0_{\mathcal{E}} A^1_{\mathcal{E}} Q^1_{\mathcal{E}} R^1_{\mathcal{E}} Q'^1_{\mathcal{E}} A^2_{\mathcal{E}} Q^2_{\mathcal{E}} ... A^m_{\mathcal{E}} Q^m_{\mathcal{E}} \in Canons(\mathcal{E})$ with $\forall j \in [1 : m]$ $R^j_{\mathcal{E}} = [0, r_j[$ and $r_j \in \mathbb{R}^{>0}$.*

$proj^{-1}_{(\mathcal{E},\mathcal{A})}(C_{\bar{\Lambda}_m}) \in \mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}$.

Proof. By induction.

Basis: by previous lemma 76.

Induction: We assume the result to be true up to a particular $m \in \mathbb{N}$ and we show it implies it is true for $m+1$. Let $\bar{\Lambda}_{m+1} = \bar{\Lambda}_m R^m_{\mathcal{E}} Q'^m_{\mathcal{E}} A^{m+1}_{\mathcal{E}} Q^{m+1}_{\mathcal{E}}$ with $R^m_{\mathcal{E}} = [0, r[$ for some $r \in \mathbb{R}^{\geq 0}$. By induction hypothesis, we know that $proj^{-1}_{(\mathcal{E},\mathcal{A})}(C_{\bar{\Lambda}_m}) \in \mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}$, i.e. $proj^{-1}_{(\mathcal{E},\mathcal{A})}(C_{\bar{\Lambda}_m}) = \bigcup_{i\in\mathbb{N}} C_{\Delta_i}$ for some basic sets $(\Delta_i)_{i\in\mathbb{N}} \in Bases(\mathcal{E}||\mathcal{A})^{\mathbb{N}}$. Here, the Δ_i are concatenation of explanation bases for the sets in $\bar{\Lambda}_n$ (one per chunk $Q^j_{\mathcal{E}} R^j_{\mathcal{E}} Q'^j_{\mathcal{E}} A^{j+1}_{\mathcal{E}} Q^{j+1}_{\mathcal{E}}$). Hence, following the construction of the proof of previous lemma, we have:

$$proj^{-1}_{(\mathcal{E},\mathcal{A})}(C_{\bar{\Lambda}_{m+1}}) = \bigcup_{i\in\mathbb{N}} \bigcup_{m\in\mathbb{N}} \bigcup_{(q_1,q'_1,...,q_m,q'_m)\in G_m} C_{\Delta_i \Lambda^{m,r}_{q_1,q'_1,...,q_m,q'_m}}$$

with $G_m = \{(q_1, q'_1, ..., q_m, q'_m) \in (\mathbb{Q}^{\geq 0})^{2m} | \forall j \in [1 : m], q_j < q'_j \wedge \sum_{j\in[1:m]} q'_j < r\}$.

Hence, $proj^{-1}_{(\mathcal{E},\mathcal{A})}(C_{\bar{\Lambda}_{m+1}})$ is a countable union of basic sets which means $proj^{-1}_{(\mathcal{E},\mathcal{A})}(C_{\bar{\Lambda}_{m+1}}) \in \mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}$. □

We finally obtained the measurability of projection function.

Theorem 27 (p is measurable)**.** *Let $\mathcal{A}$ be a post-PTSIOA or a post-PTCA. Let $\mathcal{E} \in cenv(\mathcal{A})$. The function $proj_{\mathcal{E},\mathcal{A}} : (Execs(\mathcal{E}||\mathcal{A}), \mathcal{F}_{Execs(\mathcal{E}||\mathcal{A})}) \to (Execs(\mathcal{E}), \mathcal{F}_{Execs(\mathcal{E})})$ is measurable.*

Proof. By conjunction of lemma 75 and lemma 77. □

5.5.2. Probability Measure

Definition 203 (scheduler)**.** *Let $\mathcal{A}$ be a PTSIOA or a PTCA. A* scheduler *is a function σ that maps each execution fragment $\alpha \in Frags^*_{\mathcal{A}}$ to a probability measure $\eta \in Disc(enabled(\mathcal{A})(lstate(\alpha)))$. The set of schedulers of $\mathcal{A}$ is noted $scheduler(\mathcal{A})$.*

Let σ be a scheduler. We define the probabilistic measure on $(Execs_{\mathcal{A}}, \mathcal{F}_{Execs_{\mathcal{A}}})$ generated by σ, noted ϵ_σ as follows:

1. $\epsilon_\sigma(C_P) = 1$ if $\bar{q}_{\mathcal{A}} \in P$ and 0 otherwise.
2. a) $\epsilon_\sigma(C_{\Lambda RP}) = \int_{\alpha \in C_\Lambda} \mathbb{1}_{E_{R,P}}(lstate(\alpha)) \cdot \epsilon_\sigma(\alpha)\, d\alpha$

   b) $\epsilon_\sigma(C_{\Lambda BP}) = \int_{\alpha \in C_\Lambda} \int_{a \in supp(\sigma(\alpha)) \cap B} \sigma(\alpha)(a) \cdot \eta_{(\mathcal{A},lstate(\alpha),a)}(P) \cdot \epsilon_\sigma(\alpha)\, d\alpha$

where $B \subseteq acts(\mathcal{A})$, $P \subseteq Q_{\mathcal{A}}$ and R is a Borel set of $\mathbb{R}^{\geq 0}$.

Item 1 states that any execution fragment that does not start with the start state has a measure of the probability of 0, while the measure of $Execs_{\mathcal{A}}$ is 1.

Item 2 describes the measure recursively.

The item 2a, says that the measure of cylinder $C_{\Lambda RP}$ is the measure of all executions in cylinder C_Λ that ends in a state so that the trajectory starting from this state ends in a state in P after a time in R. Let us note that the definition implicitly requires axiom $\mathbf{M}_2$ of measurability of $E_{R,P}$.

Item 2b, says that the measure of cylinder $C_{\Lambda BP}$ is a double weighted sum on (1) executions in cylinder C_Λ and (2) actions, triggered by the scheduler after the executions, that are in B. Let us note that the definition implicitly requires axiom $\mathbf{M}_1$.

This double sum is weighted by the product of (i) the probability $\sigma(\alpha)(a)$ that the scheduler triggers a after α, (ii) the probability $\eta_{(\mathcal{A},lstate(\alpha),a)}(P)$ of reaching a state in P after triggering a from $lstate(\alpha)$, and (iii) the measure $\epsilon_\sigma(\alpha)$ of α.

Theorem 28. *ϵ_σ is a probability measure on $(Execs, \mathcal{F}_{Execs})$.*

Proof. For any base Λ, $\epsilon_\sigma(C_\Lambda) \geq 0$, $\epsilon_\sigma(Execs_{\mathcal{A}}) = \epsilon_\sigma(C_{\{\bar{q}_{\mathcal{A}}\}}) = 1$ and if $C_\Lambda = \emptyset$, $\epsilon_\sigma(C_\Lambda) = 0$. Next, we consider a countable disjoint collection of bases $\{\Lambda_i\}_{i \in I}$. For any $i, j \in I$, at least one of the sets in the sequence Λ_i must be disjoint with the corresponding set in Λ_i . (If Λ_i and Λ_j are of different lengths, say Λ_i is shorter than Λ_j, then there must exist a disjoint set in the prefix of the Λ_i that equals in length to Λ_j.) Therefore, $\epsilon_\sigma(\biguplus_{i \in I} C_{\Lambda_i}) = \sum_{i \in I} \epsilon_\sigma(C_{\Lambda_i})$ by additivity of integral. Thus ϵ_σ is a probability measure over the ring $\{C \cap Execs_{\mathcal{A}} | C \in Cyls(\mathcal{A})\}$ defined by the collection of all basic sets of executions. It follows that ϵ_σ is a probability measure on $(Execs, \mathcal{F}_{Execs})$. □

In summary, each scheduler for $\mathcal{A}$ gives rise to a probabilistic execution, which is a probability measure on the space $(Execs, \mathcal{F}_{Execs})$. A set of schedulers gives a set of probabilistic executions.

5.5.3. Implementation

Definition of environment, perception function, $f\text{-}dist$, balanced schedulers, implementation relationship are the same ones than in section 3.6.2. All the results still hold since they are independent of the sigma-field and the definition of executions.

5.6. Local Scheduler

This section is a straightforward adaptation of [Mit07, ML07b] where a probabilistic semantic for the PTSIOA (resp. PTCA) that do not necessarily satisfy the determinism axioms $\mathbf{D}_1$ and $\mathbf{D_2}$, is developed. This development relies on local schedulers which are PTSIOA (resp. PTCA) that do satisfy the determinism axioms $\mathbf{D}_1$ and $\mathbf{D_2}$.

Definition 204 (Generalized PTSIOA)**.** *A* generalized PTSIOA $\mathcal{A}$ *is a pre-PTSIOA, satisfying axiom* $\mathbf{M}_1$ *of transitions measurability, axiom* $\mathbf{D}_3$ *of transition determinism, but not necessarily axiom* $\mathbf{D}_1$ *and* $\mathbf{D}_2$ *of trajectory determinism and time-action determinism.*

Thus, from a given state $q \in Q_{\mathcal{A}}$ of a generalized PTSIOA $\mathcal{A}$, there may be a non-deterministic choice of actions that could be performed and also a choice of distinct trajectories starting from q. A *local scheduler* for generalized PTSIOA $\mathcal{A}$, is a PTSIOA $((Q_A, \mathcal{F}_{Q_A}), \bar{q}_{\mathcal{A}}, sig(\mathcal{A}), D'_{\mathcal{A}}, \mathcal{T}'_{\mathcal{A}})$ that is identical to $\mathcal{A}$ except that $D'_{\mathcal{A}} \subseteq D_{\mathcal{A}}$, $\mathcal{T}'_{\mathcal{A}} \subseteq \mathcal{T}_{\mathcal{A}}$. A local scheduler satisfies $\mathbf{D}_1$ and $\mathbf{D}_2$ and has deterministic trajectories. A probabilistic system captures the notion of possible ways of resolving the non-determinism in a generalized PTSIOA. Formally, a probabilistic-system is a pair $\mathcal{M} = (\mathcal{A}, \mathcal{L})$, where $\mathcal{A}$ is a generalized PTSIOA and $\mathcal{L}$ is a set of local schedulers for $\mathcal{A}$. An environment for $\mathcal{M}$ is any PTSIOA $\mathcal{E} \in env(\mathcal{A})$. We note $env(\mathcal{M})$ the set of environment for $\mathcal{M}$

A probabilistic execution for $\mathcal{M}$ is defined to be any probabilistic execution of L, for any $L \in \mathcal{L}$.

Compatibility and composition of generalized PTSIOA are defined in the same way as in the case of PTSIOA. We remind the reader that the composition $\mathcal{A} = \mathcal{A}_1||\mathcal{A}_2$ of partially-compatible generalized PTSIOA $\mathcal{A}_1$ and $\mathcal{A}_2$, is also a generalized PTSIOA only if $\mathcal{A}$ satisfies $\mathbf{M}_1$ and $\mathbf{M}_2$. In order to avoid restating this condition in all our definitions and results, we assume that whenever two compatible generalized PTSIOA are composed, their composition satisfies $\mathbf{M}_1$ and $\mathbf{M}_2$, and hence is a generalized PTSIOA.

Definition 205 (implementation relationship for probabilistic systems)**.** *Let* $\mathcal{M}_1 = (\mathcal{A}_1, \mathcal{L}_1)$ *and* $\mathcal{M}_2 = (\mathcal{A}_2, \mathcal{L}_2)$ *be probabilistic systems. Let* f *be a perception function,* S *be a scheduler schema and* $\varepsilon \in \mathbb{R}^{\geq 0}$*. Then,* $\mathcal{M}_1 \leq_{\varepsilon}^{S,f} \mathcal{M}_1$ *if* $\forall \mathcal{E} \in env(\mathcal{M}_1) \cap env(\mathcal{M}_2)$, $\forall L_1 \in \mathcal{L}_1$, $\forall \sigma_1 \in S(\mathcal{E}||L_1)$, $\exists L_2 \in \mathcal{L}_2$, $\exists \sigma_2 \in S(\mathcal{E}||L_2)$ *such that* $\sigma_1 B^{\leq \varepsilon}_{(\mathcal{E},L_1,L_2),f} \sigma_2$

This definition can be understood as, for every distinguisher $\mathcal{E}$, for every way of resolving the non-determinism for the first probabilistic system interacting with $\mathcal{E}$, there exists a way of resolving the non-determinism for the second probabilistic system interacting with $\mathcal{E}$, such that the distinguisher is unable to distinguish the two situations with a probability greater than ε.

Two probabilistic systems $\mathcal{M}_1 = (\mathcal{A}_1, \mathcal{L}_1)$ and $\mathcal{M}_2 = (\mathcal{A}_2, \mathcal{L}_2)$ are compatible if $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible, and their composition $\mathcal{M}_1||\mathcal{M}_2$ is defined as $(\mathcal{A}_1||\mathcal{A}_2, \mathcal{L})$, where $\mathcal{L} = \{L_1||L_2 | (L_1, L_2) \in \mathcal{L}_1 \times \mathcal{L}_2\}$.

Theorem 29. *Let* $\varepsilon \in \mathbb{R}^{\geq 0}$*. Let* f *be a perception-function. Let* S *be a scheduler schema.*

Let $\mathcal{M}_1 = (\mathcal{A}_1, \mathcal{L}_1)$*,* $\mathcal{M}_2 = (\mathcal{A}_2, \mathcal{L}_2)$ *and* $\mathcal{M}_3 = (\mathcal{A}_3, \mathcal{L}_3)$ *be 3 probabilistic systems satisfying:* $\mathcal{M}_3$ *is partially compatible with both* $\mathcal{M}_1$ *and* $\mathcal{M}_2$*. Then the following holds.*

If $\mathcal{M}_1 \leq_{\varepsilon}^{S,f} \mathcal{M}_2$*, then* $\mathcal{M}_3||\mathcal{M}_1 \leq_{\varepsilon}^{S,f} \mathcal{M}_3||\mathcal{M}_2$*.*

Proof. Consider any environment $\mathcal{E}$ for $\mathcal{M}_3||\mathcal{M}_1 = (\mathcal{A}_3||\mathcal{A}_1, \mathcal{L}_{31})$ and $\mathcal{M}_3||\mathcal{M}_2 = (\mathcal{A}_3||\mathcal{A}_2, \mathcal{L}_{32})$. We must show that, for every $L_{31} \in \mathcal{L}_{31}$, for every scheduler $\sigma_{31} \in S(\mathcal{E}||L_{31})$, there is a local scheduler $L_{32} \in \mathcal{L}_{32}$ and a scheduler $\sigma_{32} \in S(\mathcal{E}||L_{32})$ such that $\sigma_{31} B^{\leq \varepsilon}_{\mathcal{E},f} \sigma_{32}$. To show this we fix $L_{31} \in \mathcal{L}_{31}$

and $\sigma_{31} \in S(\mathcal{E}||L_{31})$. By definition, $\exists(L_3, L_1) \in \mathcal{L}_3 \times \mathcal{L}_1$, such that $L_{31} = L_3||L_1$. Hence, $\sigma_{31} \in S((\mathcal{E}||L_3)||L_1)$. Yet, $\mathcal{M}_1 \leq_{\varepsilon}^{S,f} \mathcal{M}_2$ and $(\mathcal{E}||L_3)$ is an environment for both $\mathcal{M}_1$ and $\mathcal{M}_2$. Thus, there exists $L_2 \in \mathcal{L}_2$ and $\sigma_{32} \in S((\mathcal{E}||L_3)||L_2)$ such that $\sigma_{31} B^{\leq\varepsilon}_{(\mathcal{E}||L_3),f} \sigma_{32}$. Finally, stability by composition of f gives $\sigma_{31} B^{\leq\varepsilon}_{(\mathcal{E},f)} \sigma_{32}$ which ends the proof.

□

Theorem 30 (Transitivity)**.** *Let $\varepsilon_{12}, \varepsilon_{23}, \varepsilon_{13} \in \mathbb{R}^{\geq 0}$ with $\varepsilon_{13} = \varepsilon_{12} + \varepsilon 23$. Let f be an insight-function. Let S be a scheduler schema.*

Let $\mathcal{M}_1 = (\mathcal{A}_1, \mathcal{L}_1)$, $\mathcal{M}_2 = (\mathcal{A}_2, \mathcal{L}_2)$ and $\mathcal{M}_3 = (\mathcal{A}_3, \mathcal{L}_3)$ be 3 probabilistic systems. If $\mathcal{M}_1 \leq_{\varepsilon_{12}}^{S,f} \mathcal{M}_2$ and $\mathcal{M}_2 \leq_{\varepsilon_{23}}^{S,f} \mathcal{M}_3$, then $\mathcal{M}_1 \leq_{\varepsilon_{13}}^{S,f} \mathcal{M}_3$

Proof. Let $\mathcal{E} \in env(\mathcal{M}_1) \cap env(\mathcal{M}_3)$.

Case 1: Let us assume $\mathcal{E} \in env(\mathcal{M}_2)$. Let $L_1 \in \mathcal{L}_1$, $\sigma_1 \in S(\mathcal{E}||L_1)$. Since $\mathcal{M}_1 \leq_{\varepsilon_{12}}^{S,f} \mathcal{M}_2$, $\exists L_2 \in \mathcal{L}_2$, $\sigma_2 \in S(\mathcal{E}||L_2)$, $\sigma_1 B^{\leq\varepsilon_{12}}_{(\mathcal{E},L_1,L_2),f} \sigma_2$. Since $\mathcal{M}_2 \leq_{\varepsilon_{23}}^{S,f} \mathcal{M}_3$, $\exists L_3 \in \mathcal{L}_3$, $\sigma_3 \in S(\mathcal{E}||L_3)$, $\sigma_2 B^{\leq\varepsilon_{23}}_{(\mathcal{E},L_2,L_3),f} \sigma_3$. By lemma 9, $\sigma_1 B^{\leq\varepsilon_{13}}_{(\mathcal{E},L_1,L_3),f} \sigma_3$., which ends the proof for this case.

Case 2: Let us assume $\mathcal{E} \notin env(\mathcal{M}_2)$. A renaming procedure like in the proof of theorem 4 allows us to fall back to case 1. □

5.7. Summary

We have shown how to merge PSIOA (resp. PCA) with the timed setting of Mitra and Lynch to obtain PTSIOA (resp. PTC). The big picture did not change significantly. Dynamic timed automata have to verify a new constraint of trajectory preservation, that requires a strong link between the set of trajectories of the dynamic automata and the ones of the associated configurations. Theorem 26 shows that the composition of two pre-PTCA is a pre-PTCA itself. The results of substitutability of implementation relationship do not change and do not require an additional treatment. Handling both dynamicity and continuous states spaces requires some precautions to define the associated measures.

Chapter 6

Dynamic Secure Emulation

This chapter explains how to extend the framework with simulation-based security.

Overview

6.1. Polynomial-Bounded automata: formalizing implementation by computational indistinguishability . . . 204
6.1.1. time-bounded automata . . . 204
6.1.2. Boundedness preservation by usual operators . . . 205
6.1.3. Bounded Scheduling . . . 207
6.1.4. Implementation . . . 208
6.2. Security Layer . . . 211
6.2.1. Structured dynamic I/O Automata . . . 211
6.2.2. Adversary for structured automata . . . 212
6.2.3. Dynamic Secure-Emulation . . . 213
6.3. Summary . . . 217

Previous chapters 3 and 4 deal with "unbounded" systems,in the sense, that nothing prevents, a priori, from performing steps that would require a huge amount of computational resources. However, modern applications often rely on sophisticated cryptographic tools, assuming an (a priori polynomially) bounded adversary using a limited amount of computational resources. The achieved security properties are then of the form, "for any adversary with a bounded computational power, the probability of violating the security is negligible". Very often, distributed secure systems are analyzed in the symbolic model [DY83], where cryptographic operations are seen as functions on a space of symbolic (formal) expressions. In such a model, it does not make any sense for an adversary to see a signed message as a sequence of bits. It is just a pair (message, signature) where each element is an atomic entity of the symbolic space. Although such a model is very powerful for automated proof [BPW03], it does not a priori carry any cryptographic computational security guarantees. Several formal analysis frameworks linking the two aspects have been proposed. For example, one approach is to connect symbolic security and computational security with *soundness* (e.g.[AR00, MW05, CW05, CH11]). If soundness is guaranteed, a security proof by a symbolic analysis also holds against computational adversaries. This kind of connection is often delicate and requires a foundational computational model. The most popular approach is simulation-based security introduced in [GM84, GMR85] (a tutorial can be found in [Lin17]).

In the simulation-based security, a.k.a. "real-ideal paradigm", the objective of the adversary is defined only with respect to an "idealized" game, which can be seen as the specification of the task we want to solve. Then, the security properties will be stated in the following form: for any adversary, there exists a "simulator" (which is another "ideal" adversary), so that the "adversary in the real game" and the "simulator in the ideal game" cannot be distinguished with more than a negligible probability by an external observer (our environment which plays the role of a "distinguisher"). The idea is to say that any attack that (i) can be made against the real game (ii) can also be made against the ideal game, while we are satisfied with such an ideal world. Such a paradigm has many interests. First, the absence of an explicitly defined goal for the adversary prevents one from missing any subtle attacks that might occur. Second, it allows for the correct definition of certain properties whose alternative definitions might be cumbersome (e.g., the property of "not learning something more than something").

Composable simulation-based security has been pioneering at the beginning of the century (e.g. [Can01, PW00]), with a lot of extensions and restrictions. A unified model, embedding UC-framework [Can01, Can20] has been proposed [RKC22], using inexhaustible interactive Turing machines (IITM) [KTR20]. This model (1) allows relating a large number of different notions of simulation-based security, (2) enjoys a great expressivity, with its own notion of dynamicity, via an operation of creation, called "bang" operator (noted "!") following the common terminology of process calculus [Mil80]. It is even possible to add the notion of synchrony to the UC framework (which is inherently asynchronous) assuming the access to an ideal functionality $\mathcal{F}_{CLOCK}$ that allows different parties to synchronize [KMTZ13].

In this chapter, we extend DPIOA with simulation-based security as Canetti et al. [CCK$^+$07] do with static PIOA. But what is the interest if the model of IITM [KTR20] is so expressive? Here are some thoughts:

simplicity and abstraction In practice, though entities in the system are defined by interactive Turing machines (ITMs), a precise description would require too many low-level machine details. Hence the entities are usually described using some type of pseudo-code, where it remains unknown how it is supposed to be translated into a Turing Machine. According to Hofheinz and Shoup [HS15], "as long as we restrict ourselves to models that are polynomial-time equivalent to Turing machines, none of these details matter". In fact, detailed execution fragments of a system of ITMs seem to be a non-necessary intricacy, that, combined with the sophistication of distributed algorithms, can lead to an excessively

hard-to-follow analysis. It seems that the overuse of the simulation-based security paradigm, expressed with ITMs, has participated in the respective misunderstanding between the distributed computing and secure multi-party computing (MPC) communities. Labeled Transition Systems (LTS) such as I/O automata allow not to specify the exact computation of the state transition. Furthermore, their manipulation is closer to the human intuition and classic informal reasoning that often appear in distributed computing.

non-sequential scheduling The second interest is that sequential scheduling models are commonly used in many cryptography frameworks (UC, IITM, etc), where multiple processes must not be active simultaneously at any given point in time, and the active process activates the next process by producing and sending some message. Besides being a counter-intuitive modelization, sequential scheduling can artificially introduce some constraints in the ordering of events, and so artificially restrict the power of the adversary in comparison to the real world. For example, Canetti et al. [CCLP07] proved that a beacon protocol preserves security under sequential scheduling (ITM) but not under non-sequential scheduling, while Yoneyama [Yon10] showed that the beacon protocol verifies indifferentiable security (where public channels have to be considered as well as private channels) under the sequential scheduling but not under the non-sequential scheduling. An extension of LTS like our model does not suffer from the problems of sequential scheduling.

time The ITMs-based frameworks are inherently asynchronous. It is possible to add the notion of time and synchrony, assuming the access to an ideal functionality $\mathcal{F}_{CLOCK}$ that allows different parties to synchronize [KMTZ13], but the result is not trivial to handle and does not necessarily correspond to the human intuition of synchrony. Our model proposes something simpler. We can add time in 3 different manners.

1. Use the "old-fashioned recipe for real-time" [AL92], as used in [Seg95b].
2. Use trajectories with discrete time axis ($\mathbb{T} = \mathbb{N}$). In this case, one unit of time corresponds to the greatest common divisor of the clock periods (the reverse of clock rates) of all the processors. Then, we can easily bound the number of steps per unit of time of an object of the model, following a similar approach as in [CCK$^+$08].
3. Use trajectories with a continuous time axis ($\mathbb{T} = \mathbb{R}$) but we need to take precautions not to artificially give extra computational power. Indeed, with such a solution, the set of variables evolving along trajectories, are called clocks and cannot visit a too high number of different states in a certain time window. In this way, the restriction of a trajectory to non-clock variables is a constant function, while the restriction of a trajectory to clock variables is a simple function, where the number of different values that can be taken is again defined by the greatest common divisor of the clock periods (the reverse of clock rates) of all the processors.

Each solution is promising to a simple modelization of time.

In order to analyze cryptographic protocols, Canetti & al. [CCK$^+$07] extends probabilistic Input/Output automata, with the notion of "structures", which classifies communications into two categories: those with a distinguisher environment and those with an adversary (whose respective roles are those of the real-ideal paradigm described above). Then, they are able to define a new implementation relationship, called "secure emulation" in the same way as traditional simulation-based security ([KDMR08, CCLP07, KTR20, RKC22] discuss the relationship between different notions of simulation-based security). Composability of secure emulation is then "easily" derived from the composability of classic implementation relationship for PIOA. Even though the formalism proposed in [CCK$^+$07] has been already used in the verification of various cryptographic protocols [CCLP07, CMP07, YKO07, JMMS10, Yon18], this formalism does not allow to model systems that

"can dynamically create new protocol instances at run time". For example, we would like to cover blockchain systems where subchains can be created or destroyed at run time [RPG19]. Hence, on top of dynamic probabilistic I/O automata described in detail in chapters 3 and 4, we propose an extension of the composable secure emulation of Canetti et al. [CCK+07] to dynamic settings. Our framework, composable dynamic secure emulation, enjoys the composability properties of secure emulation of [CCK+07]. In terms of dynamicity, our work closes a modelization problem left open by the work of Canetti et al. [CCK+07]. That is, our framework allows us to model environments (or structures) that can dynamically create new protocol instances at run time.

Most of the results of this chapter are straightforward adaptations of their "static" counterparts in [CCK+07]. The main source of modification comes from dealing with compatibility at every reachable state instead of compatibility at each element of the Cartesian product of respective sets of states. For sake of completeness, we repeat the results with the required small modifications in the proofs.

6.1. Polynomial-Bounded automata: formalizing implementation by computational indistinguishability

In this section, we extend the approximate implementation relation defined in [CCK+07] to dynamic settings, to express the idea that every behavior of one family of dynamic automata is computationally indistinguishable from some behavior of another dynamic automata family.

We adopt a standard bit-representation where we note $\langle q \rangle$, $\langle a \rangle$, $\langle tr \rangle$, $\langle C \rangle$ the respective bit-string representations of state q, action a, discrete transition tr and configuration C.

Definition 206 (bounded Turing Machine)**.** *Let $b \in \mathbb{N}$. A (probabilistic or deterministic) Turing Machine is said b-bounded if it always runs in time at most b and can be described using a bit string length of at most b, according to some standard encoding of Turing machines. The formal definitions associated to Turing Machines, can be found in Section 2.4.*

6.1.1. time-bounded automata

In the following, we extend the definition of time-bounded PIOA [CCK+07] to dynamic settings. We define bounded PSIOA and then bounded PCA. The idea is to both limit the memory and the computational power of the concerned PSIOA. Typically, we prohibit transitions that would implicitly violate some computational hardness assumptions. Details about the meaning of a Turing Machine deciding a predicate can be found in Section 2.4, namely in definitions 55 and 56 in subsection 2.4.4.

Definition 207 (PSIOA b-bounded)**.** *PSIOA $\mathcal{A}$ is said to be b-bounded, where $b \in \mathbb{N}^{\geq 0}$, provided that:*

1. *Automaton parts: The length of the bit-string representation of every action, state, and the transition is at most b.*
2. *Decoding: There exist b-bounded deterministic Turing Machines M_{states}, M_{start}, M_{act}, M_{in}, M_{out}, M_{int}, M_{trans}, M_{step} so that:*
   - *given the representation $\langle q \rangle$ of a candidate state q, M_{states} decides whether $q \in Q_{\mathcal{A}}$,*
   - *given the representation $\langle q \rangle$ of a state q, M_{start} decides whether q is the unique start state $\bar{q}_{\mathcal{A}}$ of $\mathcal{A}$,*
   - *given the representation $\langle a \rangle$ of a candidate action a, M_{act} decides whether $a \in acts(\mathcal{A})$,*
   - *given the representation $\langle q \rangle$ of a state $q \in Q_{\mathcal{A}}$, given the representation $\langle a \rangle$ of an action that is a candidate input action, (resp. output action, resp. internal action) M_{in} (resp. M_{out},*

resp. M_{int}) decides whether $a \in in(\mathcal{A})(q)$ (resp. $a \in out(\mathcal{A})(q)$, resp. $a \in int(\mathcal{A})(q)$). we note M_{sig} the deterministic Turing Machine that given the representation $\langle q \rangle$ of a state $q \in Q_{\mathcal{A}}$, that given the representation $\langle a \rangle$ of an action that is a candidate enabled action, decides whether $a \in \widehat{sig}(\mathcal{A})(q)$.

- *given the representation $\langle q \rangle$ of a state $q \in Q_{\mathcal{A}}$, the representation $\langle a \rangle$ of an action $a \in \widehat{sig}(\mathcal{A})(q)$ and the representation $\langle tr \rangle$ of $tr = (q, a, \eta)$, M_{trans} decides whether $tr \in D_{\mathcal{A}}$ and*

3. *Determining the next state: There is a b-bounded probabilistic Turing machine M_{state} that, given the representation $\langle q \rangle$ of a state q of $\mathcal{A}$, and the representation $\langle a \rangle$ of an action $a \in \widehat{sig}(\mathcal{A})(q)$ that is enabled in q, produces the representation $\langle q' \rangle$ of the next state q' resulting from the unique transition of $\mathcal{A}$ of the form (q, a, η).*

We naturally extend the last definition 207 to PCA, that are PSIOA equipped with additional tools to define constraints.

Definition 208 (PCA b-bounded)**.** *PCA X is said to be b-bounded, where $b \in \mathbb{N}^{\geq 0}$, provided that: $psioa(X)$ is b-bounded as per definition 207.*

We extend these definitions 207 and 208 to automata families in the obvious manner. An automata family $(\mathcal{A}_k)_{k\in\mathbb{N}}$ is $\underline{b}$-bounded with $\underline{b} \in \mathbb{N}^{\mathbb{N}}$, if $\forall k \in \mathbb{N}$, $\mathcal{A}_k$ is $\underline{b}(k)$-bounded. Finally, An automata family $(\mathcal{A}_k)_{k\in\mathbb{N}}$ is polynomially-bounded if it is $\underline{b}$-bounded with a polynomial $\underline{b}$.

For sake of conciseness, we introduce the following opeartor.

Definition 209 (universal set)**.** *For every PSIOA or PCA $\mathcal{A}$ for every mapping $m(\mathcal{A})$ with $dom(m(\mathcal{A})) = Q_{\mathcal{A}}$, we note $\breve{m}(\mathcal{A}) = \bigcup_{q\in Q_A} m(\mathcal{A})(q)$.*

6.1.2. Boundedness preservation by usual operators

As for bounded PIOA [CCK+07], the composition of two time-bounded PSIOA (resp. PCA) is also time-bounded, with a bound that is a linear combination of the bounds for the two components.

Lemma 78 (composition of bounded PSIOA is bounded)**.** *There exists a constant c_{comp} such that the following holds. Suppose $\mathcal{A}_1$ is a b_1-time-bounded PSIOA (resp. PCA) and $\mathcal{A}_2$ is a b_2-time-bounded PSIOA (resp. PCA), where $b_1, b_2 \geq 1$. Then $\mathcal{A}_1||\mathcal{A}_2$ is a $c_{comp}\cdot(b_1+b_2)$-bounded PSIOA (resp. PCA).*

Proof. We describe how the different bounds of definition 207 combine when we compose $\mathcal{A}_1$ and $\mathcal{A}_2$. Details about encoding of the cartesian product (resp. union) of two encoded sets can be found in 2.4.3.1, namely in proposition 5 (resp. 6).

1. Automaton parts: Every action has a standard representation which is the same as its representation in $\mathcal{A}_1$ or $\mathcal{A}_2$. The length of this representation is, therefore, at most $max(b_1, b_2)$. Every state of $\mathcal{A}_1||\mathcal{A}_2$ can be represented with a $2 \cdot (b_1 + b_2) + 2 \leq 3 \cdot (b_1 + b_2)$-bit string, by following each bit of the bit-string representations of the states of $\mathcal{A}_1$ and $\mathcal{A}_2$ with a zero, and then concatenating the results, separating them with the string 11. Likewise, every transition of $\mathcal{A}_1||\mathcal{A}_2$ can be represented as a $3 \cdot (b_1 + b_2)$-bit string, by combining the representations of transitions of one or both of the component automata.
2. Decoding: It is possible to decide whether a candidate state $q = (q_1, q_2)$ is the start state of $\mathcal{A}_1||\mathcal{A}_2$ by checking if q_1 is the start state of $\mathcal{A}_1$ and q_2 is the start state of $\mathcal{A}_2$. Given the representation $\langle (q_1, q_2) \rangle$ of a state $(q_1, q_2) \in Q_{\mathcal{A}_1||\mathcal{A}_2}$, it is possible to decide if a candidate

input action is an element of $in(\mathcal{A}_1||\mathcal{A}_2)(q_1,q_2)$ by checking if it is an element of $in(\mathcal{A}_1)(q_1)$ or $in(\mathcal{A}_2)(q_2)$ but not an element of $in(\mathcal{A}_1)(q_1)$ or $in(\mathcal{A}_2)(q_2)$. Given the representation $\langle(q_1,q_2)\rangle$ of a state $(q_1,q_2) \in Q_{\mathcal{A}_1||\mathcal{A}_2}$, it is possible to decide if a candidate action is an element of $out(\mathcal{A}_1||\mathcal{A}_2)(q_1,q_2)$ (resp. $int(\mathcal{A}_1||\mathcal{A}_2)(q_1,q_2)$) by checking if it is either an element of $out(\mathcal{A}_1)(q_1)$ or $out(\mathcal{A}_2)(q_2)$ (resp. $int(\mathcal{A}_1)(q_1)$ or $int(\mathcal{A}_2)$ (q_2)) All these verifications can be done in time $O(b_1+b_2)$.

Given the representations $\langle(q_1,q_2)\rangle$ and $\langle a\rangle$ of a state $(q_1,q_2) \in Q_{\mathcal{A}_1||\mathcal{A}_2}$ and an action $a \in \widehat{sig}(\mathcal{A}_1||\mathcal{A}_2)((q_1,q_2))$, it is possible to decide whether a candidate transition $tr = ((q_1,q_2),a,\eta_1 \otimes \eta_2)$ is a transition of $\mathcal{A}_1||\mathcal{A}_2$ by checking if $tr_1 = (q_1,a,\eta_1)$ is a transition of $\mathcal{A}_1$ or $tr_2 = (q_2,a,\eta_2)$ is a transition of $\mathcal{A}_2$ after having extracted the bit-string representation of q_1,q_2,tr_1,tr_2 with time $O(b_1+b_2)$.

3. Determining the next state: Assume M_{state1} and M_{state2} are the probabilistic Turing Machines described in last item of definition 207 for $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively. We define M_{state} for $\mathcal{A}_1||\mathcal{A}_2$ as the probabilistic Turing machine that, given state $q=(q_1,q_2)$ of $\mathcal{A}_1||\mathcal{A}_2$ where $q_1 = q \upharpoonright \mathcal{A}_1$ and $q_2 = q \upharpoonright \mathcal{A}_2$ and action $a \in \widehat{sig}(\mathcal{A})(q)$, outputs the next state of $\mathcal{A}_1||\mathcal{A}_2$ as $q' = (q_1',q_2')$, where q_1' is the next state of $\mathcal{A}_1$ and q_2' is the next state of $\mathcal{A}_2$. The state q' is computed as follows: If $a \in \widehat{sig}(\mathcal{A}_1)(q_1)$, then q_1' is the output of $M_{state1}(q_1,a)$, while $q_1' = q_1$ otherwise. If $a \in \widehat{sig}(\mathcal{A}_2)(q_2)$ then q_2' is the output of $M_{state2}(q_2,a)$, while $q_2' = q_2$ otherwise. M_{state} always operates within time $O(b_1+b_2)$: this time is sufficient to determine whether $a \in \widehat{sig}(\mathcal{A}_1)(q_1)$ and/or $a \in \widehat{sig}(\mathcal{A}_2)(q_2)$, to extract the needed parts of q to run M_{state1} and/or M_{state2}. Using standard Turing machine encoding, each of the needed Turing machines can be represented using $O(b_1+b_2)$ bits.

□

Of course, the result holds for PCA.

Lemma 79 (composition of bounded PCA is a bounded)**.** *There exists a constant c'_{comp} such that the following holds. Suppose X_1 is a b_1-time-bounded PCA and X_2 is a b_2-time-bounded PCA, where $b_1,b_2 \geq 1$. Then $X_1||X_2$ is a $c'_{comp}\cdot(b_1+b_2)$-bounded PCA.*

Proof. $psioa(X_1||X_2) = psioa(X_1)||psioa(X_2)$ which implies $psioa(X_1||X_2)$ is a $c\cdot(b_1+b_2)$-bounded PSIOA.

□

The hiding operator does not affect the boundedness of an automaton.

Definition 210 (b-recognizable function)**.** *Let $h : Q \to \mathcal{P}(A)$ where Q and A are encoded sets. The function h is b-recognizable if there exists a b-bounded Turing Machine M, such that, given the representation $\langle q\rangle$ of a state $q \in Q$, given the representation $\langle a\rangle \in \{0,1\}^*$ of a candidate action, M decides if $a \in h(q)$.*

Details on decision can be found in subsection 2.4.4, namely in definition 55.

Lemma 80 (hiding of bounded automata is bounded)**.** *There exists a constant c_{hide} such that the following holds. Suppose $\mathcal{A}$ is a b-time-bounded PSIOA (resp. PCA), where $b \in \mathbb{N}$, $b \geq 1$. Let h be a b'-time recognizable function with $Q_{\mathcal{A}}$ as domain. Then $hide(\mathcal{A},h)$ is a $c_{hide}\cdot(b+b')$-time-bounded PSIOA (resp. PCA).*

Proof. All properties for $\mathcal{B} = hide(\mathcal{A}, S)$ are straightforward to check, except for the output actions and the internal actions. Let $\langle q \rangle$ be the bit-string representation of $q \in Q_{\mathcal{B}}$. Let $\langle a \rangle$ be the bit-string representation of a candidate action a.

1. Output actions: To check whether a is an element of $out(\mathcal{B})(q)$, we use the fact that a is an element of $out(\mathcal{B})(q)$ if and only if a is an element of $out(\mathcal{A})(q)$ and is not in $S(q)$. So, to determine whether a is an element of $out(\mathcal{B})(q)$, we can use the procedure for checking whether a is an element of $out(\mathcal{A})(q)$, followed by checking whether a is in $S(q)$.

2. Internal actions: To check whether a is an element of $int(\mathcal{B})(q)$, we use the fact that a is an element of $int(\mathcal{B})(q)$ if and only if a is an element of $int(\mathcal{A})(q)$ or is in $S(q)$. So, to determine whether a is an element of $int(\mathcal{B})(q)$, we can use the procedure for checking whether a is an element of $int(\mathcal{A})(q)$, followed by checking whether a is in $S(q)$.

using a reserved special bit constant-sized sequence of bits b^* for concatenation, it the bit-string representation of $hidden\text{-}actions(\mathcal{B})(q)$ can easily have a size of $O(b + b')$ bits.

In all cases, the total time is proportional to $b + b'$. Using standard Turing machine encodings, each of the needed Turing machines can be represented using a number of bits that is proportional to $b + b'$. □

6.1.3. Bounded Scheduling

In the previous section, we adapted the boundness of I/O automata from [CCK$^+$07] into the dynamic setting. However, at the moment, there is no bound imposed on the number of transitions that a PSIOA or a PCA may perform, which could lead to a potentially unbounded behavior and so to a potentially too important computational power.

Therefore [CCK$^+$07] introduced a final restriction on runtime imposed only for comparison of the behaviors of different PSIOA (resp. PCA) using implementation relations, by adding bounds on the number of activations.

In this thesis, we are slightly less restrictive than [CCK$^+$07], since we tolerate a broader set of schedulers instead of only accepting task-schedulers [CCK$^+$18] which generalizes fully off-line schedulers that decide in advance order of "tasks" to perform, where a task is an equivalence class on actions. In addition to the obtained generality, the advantages are as follows:

- We do not have to formalize the extension of task-structures to the dynamic setting with the attached issues mentioned in section 4.6.2.
- We can define a scheduler schema that is *oblivious* in the sufficient sense to ensure the correctness of the studied emulation candidate.
- We can define a *creation-oblivious* scheduler schema. This property has been shown in section 4.5 to be necessary to ensure that the implementation relation is monotonic w.r.t. PSIOA creation, i.e. if PCA $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ differ only in that $X_{\mathcal{A}}$ dynamically creates and destroys PSIOA $\mathcal{A}$ instead of creating and destroying PSIOA $\mathcal{B}$ as $X_{\mathcal{B}}$ does, and if $\mathcal{A}$ implements $\mathcal{B}$ (in the sense they cannot be distinguished by any external observer), then $X_{\mathcal{A}}$ implements $X_{\mathcal{B}}$. This property will allow us to obtain monotonicity w.r.t. PSIOA creation for the relation of secure emulation in future work.

Definition 211 (bounded scheduler)**.** *Let $\mathcal{A}$ be an automaton, let $b \in \mathbb{N}$. Let $\sigma \in schedulers(\mathcal{A})$, we say that σ is b-time bounded if $\forall \alpha \in Execs(\mathcal{A})$ with $|\alpha| > b$, $supp(\sigma(\alpha)) = \emptyset$, i.e. the scheduler never executes more than b actions.*

We could require that (*) the scheduler has to be a bounded automaton, but this precision is at the discretion of the designers of the solution. The results remain true if (*) is required or not.

6.1.4. Implementation

In this section, we adapt the approximate implementation to the bounded setting following the same methodology as in [CCK+07].

Strongly inspired by classic literature of cryptography, the choice of perception-function $f_{(.,.)}$ in [CCK+07] is the function *accept* that, given an execution α, outputs 1 if a special action *acc* appears in $trace(\alpha)$ and 0 otherwise. This function captures the idea that the environment distinguishes the real world from the idealized one. This is at the discretion of the user of the framework to choose a particular perception-function. But we let it as general as possible to be also able to use the perception-function of environment projection $proj_{(.,.)}$ that is particularly well-suited to obtain the monotonicity w.r.t PSIOA creation of implementation (see section 4.5). In future work, we want to use the perception-function $proj_{(.,.)}$ to extend this monotonicity result to secure emulation.

We recall that a scheduler of a PSIOA (resp. PCA) $\mathcal{A}$ is a function that maps each execution fragment to a probability on the enabled transitions, i.e. $\sigma : Frags^*(\mathcal{A}) \to SubDisc(D_{\mathcal{A}})$ such that $(q, a, \eta) \in supp(\sigma(\alpha))$ implies $q = lstate(\alpha)$.

Definition 212 (scheduler)**.** *A* scheduler *of a PSIOA (resp. PCA) $\mathcal{A}$ is a function*

$\sigma : Frags^*(\mathcal{A}) \to SubDisc(D_{\mathcal{A}})$ *such that* $(q, a, \eta) \in supp(\sigma(\alpha))$ *implies* $q = lstate(\alpha)$. *Here* $SubDisc(D_{\mathcal{A}})$ *is the set of discrete sub-probability distributions on* $D_{\mathcal{A}}$. *Loosely speaking,* σ *decides (probabilistically) which transition to take after each finite execution fragment* α. *Since this decision is a discrete sub-probability measure, it may be the case that* σ *chooses to halt after* α *with non-zero probability:* $1 - \sigma(\alpha)(D_{\mathcal{A}}) > 0$. *We note* $schedulers(\mathcal{A})$ *the set of schedulers of* $\mathcal{A}$.

Now we are ready to extend the approximate implementation of [CCK+07] to both bounded and dynamic settings.

Definition 213 (approximate implementation)**.** *Let* S *be a scheduler schema and* f *be an insight function. Let* $\mathcal{A}$ *and* $\mathcal{B}$ *be two PSIOA (resp. PCA). Let* $\varepsilon \in \mathbb{R}^{\geq 0}$, $(p, q_1, q_2) \in \mathbb{N}^3$. *We note* $\mathcal{A} \leq^{S,f}_{p,q_1,q_2,\varepsilon} \mathcal{B}$ *if for every* p*-bounded* $\mathcal{E} \in env(\mathcal{A}) \cap env(\mathcal{B})$, *for every* q_1*-bounded* $\sigma \in S(\mathcal{E}||\mathcal{A})$, *there exists a* q_2*-bounded* $\sigma' \in S(\mathcal{E}||\mathcal{B})$ *s.t.* $\sigma B^{\leq \varepsilon}_{(\mathcal{E},\mathcal{A},\mathcal{B}),f} \sigma'$ *in the sense of definition 176. We extend this definition to scheduler families as follows:*

Let $\underline{\mathcal{A}} = (\mathcal{A}_k)_{k\in\mathbb{N}}$ *and* $\underline{\mathcal{B}} = (\mathcal{B}_k)_{k\in\mathbb{N}}$ *be two PSIOA (resp. PCA) families,* $p, q_1, q_2 \in \mathbb{N}^{\mathbb{N}}$ *and* $\varepsilon : \mathbb{N} \to \mathbb{R}^{\geq 0}$. *We note* $\underline{\mathcal{A}} \leq^{S,f}_{p,q_1,q_2,\varepsilon} \underline{\mathcal{B}}$ *if* $\forall k \in \mathbb{N}$, $\mathcal{A}_k \leq^{S,f}_{p(k),q_1(k),q_2(k),\varepsilon(k)} \mathcal{B}_k$.

Finally, we note $\underline{\mathcal{A}} \leq^{S,f}_{neg,pt} \underline{\mathcal{B}}$ *if there exists polynomial functions* p, q_1, q_2 *and a negligible function* ε *s.t.* $\underline{\mathcal{A}} \leq^{S,f}_{p,q_1,q_2,\varepsilon} \underline{\mathcal{B}}$.

Lemma 81 (Implementation transitivity)**.** *Let* S *be a scheduler schema. Let* $\varepsilon_{12}, \varepsilon_{23}, \varepsilon_{13} \in \mathbb{R}^{\leq 0}, p, q_1, q_2, q_3 \in \mathbb{N}$ *with* $\varepsilon_{13} = \varepsilon_{12} + \varepsilon_{23}$, *Let* $f_{(.,.)}$ *be an insight-function. Let* $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ *be PSIOA (resp. PCA), s.t.* $\mathcal{A}_1 \leq^{S,f}_{p,q_1,q_2,\varepsilon_{12}} \mathcal{A}_2$ *and* $\mathcal{A}_2 \leq^{S,f}_{p,q_2,q_3,\varepsilon_{23}} \mathcal{A}_3$, *then* $\mathcal{A}_1 \leq^{S,f}_{p,q_1,q_3,\varepsilon_{13}} \mathcal{A}_3$.

Proof. Let $\mathcal{E} \in env(\mathcal{A}_1) \cap env(\mathcal{A}_3)$ be p-bounded.

Case 1: $\mathcal{E} \in env(\mathcal{A}_2)$. Let $\sigma_1 \in S(\mathcal{E}||\mathcal{A}_1)$ q_1-bounded, then, since $\mathcal{A}_1 \leq^{S,f}_{p,q_1,q_2,\varepsilon_{12}} \mathcal{A}_2$ there exists $\sigma_2 \in S(\mathcal{E}||\mathcal{A}_2)$ q_2-bounded s.t. $\sigma_1 B^{\leq \varepsilon_{12}}_{(\mathcal{E},\mathcal{A}_1,\mathcal{A}_2),f} \sigma_2$. and since $\mathcal{A}_2 \leq^{S,f}_{p,q_2,q_3,\varepsilon_{23}} \mathcal{A}_3$, there exists $\sigma_3 \in$

$S(\mathcal{E}||\mathcal{A}_3)$ q_3-bounded s.t. $\sigma_2 B^{\leq \varepsilon_{23}}_{(\mathcal{E},\mathcal{A}_2,\mathcal{A}_3),f}\sigma_3$ and so, by lemma 9 of transitivity of relation $B^{\leq\varepsilon}_{(\mathcal{E},.,.)f}$, for every $\sigma_1 \in S(\mathcal{E}||\mathcal{A}_1)$ q_1-bounded, there exists $\sigma_3 \in S(\mathcal{E}||\mathcal{A}_3)$ q_3-bounded s.t. $\sigma_1 B^{\leq\varepsilon_{13}}_{(\mathcal{E},\mathcal{A}_1,\mathcal{A}_3),f}\sigma_3$, i.e. $\mathcal{A}_1 \leq^{S,f}_{p,q_1,q_3,\varepsilon_{13}} \mathcal{A}_3$.

Case 2: $\mathcal{E} \notin env(\mathcal{A}_2)$. The same renaming procedure as in the proof of theorem 4 can be applied before applying Case 1. □

Theorem 31 (Implementation transitivity)**.** *Let S be a scheduler schema. Let $f_{(.,.)}$ be an insight-function.*

Let $\underline{\mathcal{A}}_1$, $\underline{\mathcal{A}}_2$ and $\underline{\mathcal{A}}_3$ be PSIOAs (resp. PCA) families satisfying:

$\underline{\mathcal{A}}_1 \leq^{S,f}_{neg,pt} \underline{\mathcal{A}}_2$, and $\underline{\mathcal{A}}_2 \leq^{S,f}_{neg,pt} \underline{\mathcal{A}}_3$ then $\underline{\mathcal{A}}_1 \leq^{S,f}_{neg,pt} \underline{\mathcal{A}}_3$.

Proof. Immediate by lemma 81. □

Lemma 82 (composability $\leq^{S,f}_{p,q_1,q_2,\varepsilon}$)**.** *Let $\varepsilon \in \mathbb{R}^{\geq 0}$ and p, p_3, q_1, $q_2 \in \mathbb{N}$ be given. Let f be a perception-function.*

Let S be a scheduler schema. Let $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ be 3 PSIOA (resp. PCA) satisfying: $\mathcal{A}_3$ has p_3-bounded description and is partially compatible with both $\mathcal{A}_1$ and $\mathcal{A}_2$. Then the following holds:

If $\mathcal{A}_1 \leq^{S,f}_{c_{comp}(p+p_3),q_1,q_2,\varepsilon} \mathcal{A}_2$, where c_{comp} is the constant factor associated with description bounds in parallel composition (see lemma 78), then $\mathcal{A}_3||\mathcal{A}_1 \leq^{S,f}_{p,q_1,q_2,\varepsilon} \mathcal{A}_3||\mathcal{A}_2$.

Proof. Fix $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ and all the constants as in the hypotheses. Consider any p-time-bounded environment $\mathcal{E}$ for $\mathcal{A}_3||\mathcal{A}_1$ and $\mathcal{A}_3||\mathcal{A}_2$. We must show that, for every q_1-time-bounded scheduler $\sigma_1 \in S(\mathcal{E}||\mathcal{A}_3||\mathcal{A}_1)$, there is a q_2-time-bounded scheduler $\sigma_2 \in S(\mathcal{E}||\mathcal{A}_3||\mathcal{A}_2)$ such that $\sigma B^{\leq\varepsilon}_{(\mathcal{E},\mathcal{A}_3||\mathcal{A}_1,\mathcal{A}_3||\mathcal{A}_2),f}\sigma'$ To show this, fix σ_1 to be any q_1-time-bounded scheduler in $S(\mathcal{E}||\mathcal{A}_3||\mathcal{A}_1)$. The composition $\mathcal{E}||\mathcal{A}_3$ is an environment for both $\mathcal{A}_1$ and $\mathcal{A}_2$. Moreover, lemma 78 implies that $\mathcal{E}||\mathcal{A}_3$ is $c_{comp} \cdot (p + p_3)$-time-bounded. Since $\mathcal{A}_1 \leq^{S,f}_{c_{comp}\cdot(b+b_3),b_1,b_2} \mathcal{A}_2$, $\mathcal{E}||\mathcal{A}_3$ is a $c_{comp}(b+b_3)$-time-bounded environment for $\mathcal{A}_1$ and $\mathcal{A}_2$, and σ_1 is a q_1-time-bounded scheduler for $\mathcal{E}||\mathcal{A}_3||\mathcal{A}_1$, we know that there is a q_2-time-bounded scheduler σ_2 for $\mathcal{E}||\mathcal{A}_3||\mathcal{A}_2$ such that $\sigma_1 B^{\leq\varepsilon}_{(\mathcal{E}||\mathcal{A}_3,\mathcal{A}_2,\mathcal{A}_1),f}\sigma_2$. Finally, the stability by composition of f gives $\sigma_1 B^{\leq\varepsilon}_{(\mathcal{E},\mathcal{A}_3||\mathcal{A}_1,\mathcal{A}_3||\mathcal{A}_2),f}\sigma_2$ which ends the proof.

□

Lemma 83 (composability $\leq^{S,f}_{\underline{p},\underline{q}_1,\underline{q}_2,\underline{\varepsilon}}$)**.** *Let $\varepsilon : \mathbb{N} \to \mathbb{R}^{\geq 0}$ and $p, p_3, q_1, q_2 \in \mathbb{N}^{\mathbb{N}}$ be given. Let S be a scheduler schema. Let $f_{(.,.)}$ be a perception-function. Let $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ be 3 PSIOA (resp. PCA) families satisfying: $\mathcal{C}$ has p_3-bounded description and is partially compatible with both $\mathcal{A}$ and $\mathcal{B}$. Let c_{comp} be the constant factor associated with description bounds in parallel composition (see lemma 78) Then the following holds.*

If $\mathcal{A} \leq^{S,f}_{c_{comp}(p+p_3),q_1,q_2,\varepsilon} \mathcal{B}$, then $\mathcal{C}||\mathcal{A} \leq^{S,f}_{p,q_1,q_2,\varepsilon} \mathcal{C}||\mathcal{B}$.

Proof. Fix $\underline{\mathcal{A}} = (\mathcal{A}_k)_{k\in\mathbb{N}}$, $\underline{\mathcal{B}} = (\mathcal{B}_k)_{k\in\mathbb{N}}$, $\underline{\mathcal{C}} = (\mathcal{C}_k)_{k\in\mathbb{N}}$ and all the functions as in the hypotheses.

By definition 213, for every $k \in \mathbb{N}$, $\mathcal{A}_k \leq^{S,f}_{p'(k),q_1(k),q_2(k)} \mathcal{B}_k$ with $p' = (c_{comp} \cdot (p + p_3))$,

Thus, $\forall k \in \mathbb{N}$, $(\mathcal{C}_k||\mathcal{A}_k) \leq^{S,f}_{p(k),q_1(k),q_2(k)} (\mathcal{C}_k||\mathcal{B}_k)$ by lemma 82.

Finally, we obtain that $\underline{\mathcal{C}}||\underline{\mathcal{A}} \leq_{S,p,q_1,q_2,\varepsilon} \underline{\mathcal{C}}||\underline{\mathcal{B}}$, as needed, applying definition 213 once again. □

Theorem 32 (composability $\leq^{S,f}_{neg,pt}$)**.** *Let S be a scheduler schema. Let $f_{(.,.)}$ be a perception-function.*

Let $\underline{\mathcal{A}}_1$, $\underline{\mathcal{A}}_2$ and $\underline{\mathcal{A}}_3$ be PSIOAs (resp. PCA) families satisfying: $\underline{\mathcal{A}}_3$ has p_3-bounded description where p_3 is a polynomial and is partially compatible with both $\underline{\mathcal{A}}_1$ and $\underline{\mathcal{A}}_2$. Then the following holds.

If $\underline{\mathcal{A}}_1 \leq^{S,f}_{neg,pt} \underline{\mathcal{A}}_2$, then $\underline{\mathcal{A}}_3||\underline{\mathcal{A}}_1 \leq^{S,f}_{neg,pt} \underline{\mathcal{A}}_3||\underline{\mathcal{A}}_2$. Observe that, by induction, the theorem generalizes to any constant number of substitutions.

Proof. Suppose $\underline{\mathcal{A}}_1$, $\underline{\mathcal{A}}_2$, $\underline{\mathcal{A}}_3$ and all the functions as in the hypotheses. Fix polynomial p_3 such that $\underline{\mathcal{A}}_3$ is p_3-time-bounded. To show that $\underline{\mathcal{A}}_3||\underline{\mathcal{A}}_1 \leq^{S,f}_{neg,pt} \underline{\mathcal{A}}_3||\underline{\mathcal{A}}_2$, we fix polynomials p and q_1; we must obtain a polynomial q_2 and a negligible function ε such that $\underline{\mathcal{A}}_3||\underline{\mathcal{A}}_1 \leq^{S,f}_{p,q_1,q_2,\varepsilon} \underline{\mathcal{A}}_3||\underline{\mathcal{A}}_2$. Define p' to be the polynomial $c_{comp}(p + p_3)$. Since $\underline{\mathcal{A}}_1 \leq^{S,f}_{neg,pt} \underline{\mathcal{A}}_2$, there exist a polynomial q_2 and a negligible function ε such that $\underline{\mathcal{A}}_1 \leq^{S,f}_{p',q_1,q_2,\varepsilon} \underline{\mathcal{A}}_2$. Lemma 83 then implies that $\underline{\mathcal{A}}_3||\underline{\mathcal{A}}_1 \leq^{S,f}_{p',q_1,q_2,\varepsilon} \underline{\mathcal{A}}_3||\underline{\mathcal{A}}_2$, as needed.

□

Theorem 33 (substitutability of $\leq^{S,f}_{neg,pt}$)**.** *Let S be a scheduler schema. Let f be a perception function.*

Let $(\underline{\mathcal{A}}_1, \underline{\mathcal{A}}_2, \underline{\mathcal{A}}_3, \underline{\mathcal{A}}_4)$ be a quadruplet of families of PSIOA (resp. PCA).

If $\underline{\mathcal{A}}_1 \leq^{S,f}_{neg,pt} \underline{\mathcal{A}}_2$, $\underline{\mathcal{A}}_3 \leq^{S,f}_{neg,pt} \underline{\mathcal{A}}_4$ and $\underline{\mathcal{A}}_3, \underline{\mathcal{A}}_4$ are both partially compatible with both $\underline{\mathcal{A}}_1, \underline{\mathcal{A}}_2$, then $\underline{\mathcal{A}}_3||\underline{\mathcal{A}}_1 \leq^{S,f}_{neg,pt} \underline{\mathcal{A}}_4||\underline{\mathcal{A}}_2$.

Observe that, by induction, the theorem generalizes to any constant number of substitutions.

Proof. By theorem of composability $\underline{\mathcal{A}}_3||\underline{\mathcal{A}}_1 \leq^{S,f}_{neg,pt} \underline{\mathcal{A}}_3||\underline{\mathcal{A}}_2$ and $\underline{\mathcal{A}}_4||\underline{\mathcal{A}}_1 \leq^{S,f}_{neg,pt} \underline{\mathcal{A}}_4||\underline{\mathcal{A}}_2$. By theorem of transitivity $\underline{\mathcal{A}}_3||\underline{\mathcal{A}}_1 \leq^{S,f}_{neg,pt} \underline{\mathcal{A}}_4||\underline{\mathcal{A}}_2$. □

6.2. Security Layer

In this section, we adapt the security layer of [CCK+07], on top of the foundational layer introduced in previous sections. This layer follows the general outline of simulation-based security. In this approach, computational security is captured within the model itself.

6.2.1. Structured dynamic I/O Automata

First, we extend the PSIOA definition with an additional attribute called the environment action mapping $EAct_{\mathcal{A}} : q \in Q_{\mathcal{A}} \mapsto EAct_{\mathcal{A}}(q) \subseteq \widehat{ext}(\mathcal{A})(q)$, that captures the idea that some actions are intended to be accessible by the environment while others by the adversary.

Definition 214 (Structured PSIOA)**.** *A structured PSIOA $\mathcal{A}$ is a n-uplet $((Q_{\mathcal{A}}, \bar{q}_{\mathcal{A}}, sig(\mathcal{A}), D_{\mathcal{A}}), EAct_{\mathcal{A}})$ where $(Q_{\mathcal{A}}, \bar{q}_{\mathcal{A}}, sig(\mathcal{A}), D_{\mathcal{A}})$ is a PSIOA and $EAct_{\mathcal{A}}$ is a mapping function with domain $Q_{\mathcal{A}}$ such that $\forall q \in Q_{\mathcal{A}}, EAct_{\mathcal{A}}(q) \subseteq \widehat{ext}(\mathcal{A})(q)$.*

The adversary action mapping of $\mathcal{A}$ is $AAct_{\mathcal{A}} : q \in Q_{\mathcal{A}} \mapsto \widehat{ext}(\mathcal{A})(q) \setminus EAct_{\mathcal{A}}(q)$.

For convenience, we also define:

1. *$EI_{\mathcal{A}} : q \in Q_{\mathcal{A}} \mapsto EAct_{\mathcal{A}}(q) \cap in(\mathcal{A})(q)$ (environment inputs),*
2. *$EO_{\mathcal{A}} : q \in Q_{\mathcal{A}} \mapsto EAct_{\mathcal{A}}(q) \cap out(\mathcal{A})(q)$ (environment outputs),*
3. *$AI_{\mathcal{A}} : q \in Q_{\mathcal{A}} \mapsto AAct_{\mathcal{A}}(q) \cap in(\mathcal{A})(q)$ (adversary inputs) and*
4. *$AO_{\mathcal{A}} : q \in Q_{\mathcal{A}} \mapsto AAct_{\mathcal{A}}(q) \cap out(\mathcal{A})(q)$ (adversary outputs).*

Let $S_{\mathcal{A}} : q \in Q_{\mathcal{A}} \mapsto S_{\mathcal{A}}(q) \subseteq out(\mathcal{A})(q)$. We note $EAct_{\mathcal{A}} \setminus S_{\mathcal{A}} : q \in Q_{\mathcal{A}} \mapsto EAct_{\mathcal{A}}(q) \setminus S_{\mathcal{A}}(q))$. We note $hide((\mathcal{A}, EAct_{\mathcal{A}}), S_{\mathcal{A}}) = (hide(\mathcal{A}, S_{\mathcal{A}}), EAct_{\mathcal{A}} \setminus S_{\mathcal{A}})$

When this is clear in the context, we slightly abuse the notation and call a structured PSIOA a PSIOA.

Observe that nothing prevent us to require that $(\bigcup_{q \in Q_{\mathcal{A}}} EAct_{\mathcal{A}}(q), \bigcup_{q \in Q_{\mathcal{A}}} AAct_{\mathcal{A}}(q))$ is a partition of $acts(\mathcal{A})$ s.t. an action a cannot be an environment action in a state and become an adversary action in another state.

We state the compatibility conditions and the composability operation for compatible structured PSIOA.

Definition 215 (Compatible structured PSIOA)**.** *Two structured PSIOA $(\mathcal{A}_1, EAct_{\mathcal{A}_1})$ and $(\mathcal{A}_2, EAct_{\mathcal{A}_2})$ are compatible at state $(q_1, q_2) \in Q_{\mathcal{A}_1} \times Q_{\mathcal{A}_2}$ if $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible at state $(q_1, q_2) \in Q_{\mathcal{A}_1} \times Q_{\mathcal{A}_2}$ and $\widehat{sig}(\mathcal{A}_1)(q_1) \cap \widehat{sig}(\mathcal{A}_2)(q_2) = EAct_{\mathcal{A}_1}(q_1) \cap EAct_{\mathcal{A}_2}(q_2)$. Furthermore, they are partially-compatible if $\mathcal{A}_1$ and $\mathcal{A}_2$ are partially-compatible and $(\mathcal{A}_1, EAct_{\mathcal{A}_1})$ and $(\mathcal{A}_2, EAct_{\mathcal{A}_2})$ are compatible at every reachable state of $(\mathcal{A}_1, \mathcal{A}_2)$. That is, every shared action must be an environment action of both structured PSIOA.*

Definition 216 (Structured PSIOA composition)**.** *Given partially-compatible structured PSIOA $(\mathcal{A}_1, EAct_{\mathcal{A}_1})$ and $(\mathcal{A}_2, EAct_{\mathcal{A}_2})$, their partial-composition $(\mathcal{A}_1, EAct_{\mathcal{A}_1})||(\mathcal{A}_2, EAct_{\mathcal{A}_2})$ is the structured PSIOA $(\mathcal{A}_1||\mathcal{A}_2, EAct_{\mathcal{A}_1} \cup EAct_{\mathcal{A}_2})$ where $EAct_{\mathcal{A}_1} \cup EAct_{\mathcal{A}_2} : (q_1, q_2) \in Q_{\mathcal{A}_1||\mathcal{A}_2} \mapsto EAct_{\mathcal{A}_1}(q_1) \cup EAct_{\mathcal{A}_2}(q_2)$.*

We can also extend the previous definition to PCA:

Definition 217 (Structured configuration)**.** *A structured configuration is a pair $(\mathbf{A}, \mathbf{S})$ where $\mathbf{A}$ is a family of structured PSIOA and $\mathbf{S}$ is a mapping function with domain $\mathbf{A}$ such that for every*

$\mathcal{A} \in \mathbf{A}, \mathbf{S}(\mathcal{A}) \in Q_{\mathcal{A}}$. *Furthermore, in addition of attributes of definition 65, we note* $EAct(C) = \bigcup_{\mathcal{A} \in \mathbf{A}} EAct_{\mathcal{A}}(\mathbf{S}(\mathcal{A}))$.

Definition 218 (Compatible structured configuration)**.** *A structured configuration* $(\mathbf{A}, \mathbf{S})$ *is compatible iff, for all* $\mathcal{A}, \mathcal{B} \in \mathbf{A}, \mathcal{A} \neq \mathcal{B}$, $\mathcal{A}, \mathcal{B}$ *are compatible at state* $(\mathbf{S}(\mathcal{A}), \mathbf{S}(\mathcal{B}))$

Definition 219 (Structured PCA)**.** *A structured PCA* X *is a PCA s.t. (1) the attached PSIOA is replaced by a structured PSIOA, (2)* $config(X)$ *is a function that maps every* $q \in Q_X$ *to a compatible structured configuration* $config(X)(q)$ *and (3)* X *is associated with the mapping* $EAct_X$ *with domain* Q_X *s.t.* $\forall q \in Q_X$, $EAct_X(q) = EAct_{psioa(X)}(q) = EAct(config(X)(q)) \setminus hidden\text{-}actions(X)(q)$

We can naturally define the composition of partially-compatible structured PCA which is the same as the one of definition 216. Such a composition naturally yields a structured PCA.

Lemma 84 (Closure of structured PCA under composition)**.** *Let* X_1 *and* X_2 *be partially-compatible structured PCA. Then* $X_1||X_2$ *is a structured PCA.*

Proof. Let $X = X_1||X_2$. For every $q \in Q_X$ (resp. $q_i \in Q_{X_i}$ with $i \in \{1,2\}$), we note $h(q) = hidden\text{-}actions(X)(q)$ (resp. $h_i(q_i) = hidden\text{-}actions(X_i)(q_i)$ with $i \in \{1,2\}$). In the same way, for every $q \in Q_X$ (resp. $q_i \in Q_{X_i}$ with $i \in \{1,2\}$), we note $C(q) = Config(X)(q)$ (resp. $C_i(q_i) = C_i(X_i)(q_i)$ with $i \in \{1,2\}$).

The closure under composition is ensured if the new restriction $\forall (q_1, q_2) \in Q_X \cap (Q_{X_1} \times Q_{X_2})$, $EAct_X((q_1, q_2)) = EAct(C((q_1, q_2))) \setminus h((q_1, q_2))$ is still ensured after composition.

Let $(q_1, q_2) \in Q_X \cap (Q_{X_1} \times Q_{X_2})$, $EAct_X((q_1, q_2)) = EAct(C((q_1, q_2))) \setminus h((q_1, q_2))$. We need to show that $EAct_{X_1}(q_1) \cup EAct_{X_2}(q_2) = (EAct(C_1(q_1)) \setminus h_1(q_1)) \cup (EAct(C_2(q_2)) \setminus h_2(q_2)) = EAct(C_1(q_1) \cup C_2(q_2)) \setminus (h_1(X_1)(q_1) \cup h_2(q_2))$. This constraint is ensured for the same reason that the fourth one. Indeed, let α be a pseudo execution of (X_1, X_2) ending on state (q_1, q_2). Since X_1 and X_2 are partially-compatible by assumption, (i) the signatures $sig(C_1(q_1))$ and $sig(C_2(q_2))$ are compatible and (ii) $sig(X_1)(q_1)$ and $sig(X_2)(q_2)$ are compatible. The conjonction of (i) and (ii) implies $h_1(q_1) \cap \widehat{sig}(X_2)(q_2) = h_2(q_2) \cap \widehat{sig}(X_1)(q_1) = \emptyset$. This is enough to ensure $(EAct(C_1(q_1)) \setminus h_1(q_1)) \cup (EAct(C_2(q_2)) \setminus h_2(q_2)) = EAct(C_1(q_1) \cup C_2(q_2)) \setminus (h_1(q_1) \cup h_2(q_2))$ which terminates the proof. □

6.2.2. Adversary for structured automata

In the following, we extend the notion of adversary introduced in [CCK+07] to the adversary for structured PSIOA.

Definition 220 (Adversary for structured automaton)**.** *An adversary* Adv *for a structured PSIOA (resp. PCA)* $(\mathcal{A}, EACT_{\mathcal{A}})$ *is a PSIOA (resp. PCA) s.t.*

- Adv *is partially-compatible with* $\mathcal{A}$
- *For every* $q = (q_{\mathcal{A}}, q_{Adv}) \in Q_{(\mathcal{A}||Adv)}$,
  - $IA_{\mathcal{A}}(q_{\mathcal{A}}) \subseteq out(Adv)(q_{Adv})$
  - $EAct_{\mathcal{A}}(q_{\mathcal{A}}) \cap \widehat{sig}(Adv)(q_{Adv}) = \emptyset$

We extend the definition to an automata family: An adversary $\underline{Adv}$ *for a structured PSIOA (resp. PCA) family* $(\underline{\mathcal{A}}, EAct_{\underline{\mathcal{A}}}) = (\mathcal{A}_k, EAct_{\mathcal{A}_k})_{k \in \mathbb{N}}$ *is a family* $(Adv_k)_{k \in \mathbb{N}}$ *of PSIOA (resp. PCA) s.t.* $\forall k \in \mathbb{N}$, Adv_k *is an adversary of* $\mathcal{A}_k$

Lemma 85 (An advesary for $\mathcal{A}||\mathcal{B}$ is an adversary for $\mathcal{A}$)**.** *Suppose $\mathcal{A}$ and $\mathcal{B}$ are compatible structured PSIOA (resp. PCA), and Adv is an adversary for $\mathcal{A}||\mathcal{B}$. Then Adv is an adversary for $\mathcal{A}$. Also, if $\underline{\mathcal{A}}$ and $\underline{\mathcal{B}}$ are compatible structured PSIOA (resp. PCA) families, and $\underline{Adv}$ is an adversary family for $\underline{\mathcal{A}}||\underline{\mathcal{B}}$. Then $\underline{Adv}$ is an adversary family for $\underline{\mathcal{B}}$.*

Proof. Suppose $\mathcal{A}$ and $\mathcal{B}$ are compatible structured PSIOA and Adv is an adversary for $\mathcal{A}||\mathcal{B}$. We observe that the conditions of definition 220 are satisfied.

1. Adv is compatible with $\mathcal{A}$. This follows from the fact that Adv is compatible with $\mathcal{A}||\mathcal{B}$.
2. Let $q' = (q_{\mathcal{A}}, q_{Adv}) \in Q_{(\mathcal{A}||Adv)}$. Then there exists $q = (q_{\mathcal{A}}, q_{\mathcal{B}}, q_{Adv}) \in Q_{(\mathcal{A}||\mathcal{B}||Adv)}$. Since Adv is an adversary for $\mathcal{A}||\mathcal{B}$ we know that:
   - $IA_{\mathcal{A}}(q_{\mathcal{A}}) \cup IA_{\mathcal{B}}(q_{\mathcal{B}}) \subseteq out(Adv)(q_{Adv})$, which means that $IA_{\mathcal{A}}(q_{\mathcal{A}}) \subseteq out(Adv)(q_{Adv})$.
   - $EAct_{\mathcal{A}}(q_{\mathcal{A}}) \cup EAct_{\mathcal{B}}(q_{\mathcal{B}}) \cap \widehat{sig}(Adv)(q_{Adv}) = \emptyset$, which means that $EAct_{\mathcal{A}}(q_{\mathcal{A}}) \cap \widehat{sig}(Adv)(q_{Adv}) = \emptyset$.

The extension to structured automata families and adversaries families is straightforward. □

6.2.3. Dynamic Secure-Emulation

The framework is finally expressive enough to define secure-emulation [CCK+07] for distributed systems with (i) non-sequential scheduling and (ii) dynamic creation/destruction of automata. This relation will be shown to be (iii) composable, which is the main contribution of this chapter.

Definition 221 (Secure Emulation)**.** *Let $\underline{\mathcal{A}}$ and $\underline{\mathcal{B}}$ be two structured PSIOA (resp. PCA) families. Let S be a scheduler schema and $f_{(.,.)}$ be an insight function.*

We say that $\underline{\mathcal{A}}$ secure-emulates $\underline{\mathcal{B}}$ w.r.t. S and f (denoted $\underline{\mathcal{A}} \leq_{SE}^{S,f} \underline{\mathcal{B}}$) if, for every polynomially bounded adversary family $\underline{Adv}$ for $\underline{\mathcal{A}}$ with, there is a polynomially bounded adversary family $\underline{Sim}$ for $\underline{\mathcal{B}}$ such that:
$hide(\underline{\mathcal{A}}||\underline{Adv}, AAct_{\underline{\mathcal{A}}}) \leq_{neg,pt}^{S,f} hide(\underline{\mathcal{B}}||\underline{Sim}, AAct_{\underline{\mathcal{B}}})$. Transitivity of $\leq_{SE}^{S,f}$ follows immediately from transitivity of $\leq_{neg,pt}^{S,f}$.

Dummy Adversary To prove the composability of secure-emulation, we use the well-known technique, introduced by Canetti [Can01], based on dummy-adversary which plays the role of a forwarder between a structured PSIOA (resp. PCA) $\mathcal{A}$ and another (potentially more sophisticated) adversary of $g(\mathcal{A})$ where g is an action-renaming function.

Let $\underline{\mathcal{A}}$ be a structured PSIOA family and, for each $k \in \mathbb{N}$, let g_k be a partial function defined on $Q_{\mathcal{A}} \times acts(\mathcal{A})$ s.t. $\forall q \in Q_{\mathcal{A}}$, $g_k(q)$ is a bijection from $AAct_{\mathcal{A}_k}$ to a set of fresh action names. We refer to $\underline{g} = \{g_k\}_{k \in \mathbb{N}}$ as a renaming of adversary actions for $\mathcal{A}$, and we write $\underline{g}(\underline{\mathcal{A}})$ for the result of applying g_k to $\mathcal{A}_k$ for every $k \in \mathbb{N}$.

Definition 222 (Dummy Adversary)**.** *Let $\mathcal{A}$ be a PSIOA (resp. PCA) and g be a bijection from $AAct_{\mathcal{A}}$ to a set of fresh action names. Then $Dummy(\mathcal{A}, g)$ is the PSIOA (resp. PCA) Adv' defined as follows:*

- *(States) Every state q of Adv' is described by its unique variable $q.pending \in \breve{AO}_{\mathcal{A}} \cup g(\breve{AI}_{\mathcal{A}}) \cup \{\bot\}$*
- *(Start state) $\bar{q}_{Adv'}.pending = \bot$*
- *(Signature) $\forall q \in Q_{Adv'}$:*
  - $in(Adv')(q) = \breve{in}(Adv') = \breve{AO}_{\mathcal{A}} \cup g(\breve{AI}_{\mathcal{A}})$

- $int(Adv')(q) = \emptyset$
- $out(Adv')(q) = \begin{cases} \{a\} & \text{if } g(a) = q.pending \in g(AI_{\mathcal{A}}) \\ \{g(a)\} & \text{if } a = q.pending \in AO_{\mathcal{A}} \\ \emptyset & \text{if } q.pending = \perp \end{cases}$

– *(Transition)* $\forall q \in Q_{Adv'}$, $\forall a \in \widehat{sig}(Adv')(q)$, $supp(\eta_{(Adv',q,a)}) = \{q'\}$ *s.t.:*
- *if* $a \in in(Adv')(q)$, $q'.pending = a$
- *if* $a \in out(Adv')(q)$, $q'.pending = \perp$

We extend this definition to families: Let $\underline{\mathcal{A}}$ *be a structured PSIOA (resp. PCA) family* $\{\mathcal{A}_k\}_{k\in\mathbb{N}}$*. Let* $\underline{g} = \{g_k\}_{k\in\mathbb{N}}$ *be a family of bijection from* $\breve{A}Act_{\mathcal{A}_k}$ *to a set of fresh action names. Then* $Dummy(\underline{\mathcal{A}}, \underline{g}) = \{Dummy(\mathcal{A}_k, g_k)\}_{k\in\mathbb{N}}$ *is a dummy adversary family for* $\underline{\mathcal{A}}$*.*

The following lemma 86 shows that dummy adversaries can be transparently added between a structured PSIOA (resp. PCA) and an adversary for that structured PSIOA (resp. PCA). This fact is used in the proof of composability of secure-emulation, with the classic decomposition technique introduced by Canetti [Can01]. Additional work is required compared to [CCK+07] since we have to deal with a more general definition that enables schedulers that are not task-schedules. However, the approach follows the same methodology, i.e. when the scheduler $\underline{\sigma}$ instructs to trigger an action shared by $\underline{g}(\underline{\mathcal{A}})$ and $\underline{Adv}$, the corresponding balanced scheduler $\underline{\sigma'}$ successively orders to trigger the corresponding action (modulo a potential renaming) and the attached forward by the dummy adversary. The proof of this lemma 86 introduces two natural constructions $Forward^e_{(\underline{\mathcal{A}},\underline{g},\underline{Adv})}$ and $Forward^s_{(\underline{\mathcal{A}},\underline{g},\underline{Adv})}$, to formalise the fact that the dummy adversary forwards the actions between $\underline{\mathcal{A}}$ and $\underline{Adv}$. A pair (S, f) made up of a scheduler and an insight function that allow these very natural constructions, is said *brave*. For sake of simplicity, these natural constructions $Forward^e_{(\underline{\mathcal{A}},\underline{g},\underline{Adv})}$ and $Forward^s_{(\underline{\mathcal{A}},\underline{g},\underline{Adv})}$ are defined in the proof itself.

Lemma 86 (Dummy adversary insertion)**.** *Let* (S, f) *be a brave pair made of a scheduler schema* S *and an insight-function* f*. Let* $\underline{\mathcal{A}}$ *be a structured PSIOA (resp. PCA) family* $\{\mathcal{A}_k\}_{k\in\mathbb{N}}$*. Let* $\underline{g} = \{g_k\}_{k\in\mathbb{N}}$ *be a family of bijection from* $\breve{A}Act_{\mathcal{A}_k}$ *to a set of fresh action names. Let* $\underline{Adv} = \{Adv_k\}_{k\in\mathbb{N}}$ *be an adversary for both* $\underline{g}(\underline{\mathcal{A}})$ *and* $hide(\underline{\mathcal{A}}||Dummy(\underline{\mathcal{A}}, \underline{g}), AAct_{\underline{\mathcal{A}}})$*. Then,*

$$\underline{g}(\underline{\mathcal{A}})||\underline{Adv} \leq^{S,f}_{neg,pt} hide(\underline{\mathcal{A}}||Dummy(\underline{\mathcal{A}}, \underline{g}), AAct_{\underline{\mathcal{A}}})||\underline{Adv}$$

Proof. Let q_1 be any polynomial and set $q_2 := 2 \cdot q_1$. Let p, q be any polynomials and ϵ be the constant zero function, i. e. $\forall k \in \mathbb{N}, \epsilon(k) = 0$. Fix $k \in \mathbb{N}$, we note $D_k = Dummy(\mathcal{A}_k, g_k)$, $H_k = hide(\mathcal{A}_k||D_k, AAct_{\mathcal{A}_k})$ and $\underline{D} = (D_k)_{k\in\mathbb{N}}$ and $\underline{H} = (H_k)_{k\in\mathbb{N}}$. Let $\mathcal{E}$ be an environment for $g_k(\mathcal{A}_k)||Adv_k$ and for $\mathcal{A}_k||H_k||Adv_k$. Let $\sigma \in Sch(\mathcal{E}||g_k(\mathcal{A}_k)||Adv_k)$ be a q_1 bounded scheduler.

We are going to construct $\sigma' \in Sch(\mathcal{E}||H_k||Adv_k)$ balanced with σ and q_2 bounded scheduler in the intuitive way.

First, we partition the functions depending if they are triggered by the environment or by the adversary. Hence, $\forall q = (q_\mathcal{E}, q_\mathcal{A}, q_{Adv}) \in Q_{(\mathcal{E}||g_k(\mathcal{A}_k)\ ||\ Adv_k)}$, for every $q^+ = (q_\mathcal{E}, q_\mathcal{A}, q_D, q_{Adv}) \in Q_{(\mathcal{E}||H_k||Adv_k)}$, we note:

– $E(q) = E(q^+) = \widehat{sig}(\mathcal{E}||g_k(\mathcal{A}_k)||Adv_k)((q_\mathcal{E}, q_\mathcal{A}, q_{Adv})) \setminus ([\widehat{ext}(g_k(\mathcal{A}_k))(q_\mathcal{A}) \cap \widehat{ext}(Adv_k)(q_{Adv})] \cup \widehat{ext}(D_k)(q_D))$, i. e. the actions not dedicated to the dummy adversary.

–
- $F_\mathcal{A}(q) = [out(g_k(\mathcal{A}_k))(q_\mathcal{A}) \cap in(Adv_k)(q_{Adv})] \cap in(D_k)(q_D)$,
- $F_{Adv}(q) = [in(g_k(\mathcal{A}_k))(q_\mathcal{A}) \cap out(Adv_k)(q_{Adv})] \cap in(D_k)(q_D)$
- $F(q) = F_\mathcal{A}(q) \cup F_{Adv}(q)$

- $F^+_{\mathcal{A}}(q^+) = in(D_k)(q_D) \cap out(\mathcal{A}_k)(q_{\mathcal{A}})$
- $F^+_{Adv}(q^+) = in(D_k)(q_D) \cap out(Adv)(q_{Adv})$
- $F^+(q) = F^+_{\mathcal{A}}(q) \cup F^+_{Adv}(q)$

each set holds actions that have to be forwarded by the dummy adversary.

- $\forall a \in F^+_{\mathcal{A}}(q^+), origin(a) = g_k(a)$ and $forward(a) = g_k(a)$. $\forall g_k(b) \in F^+_{Adv}(q^+), origin(g_k(b)) = g_k(b)$ and $forward(g_k(b)) = b$. When an action a is received by the dummy adversary, $origin(a)$ returns the corresponding action shared by $g_k(\mathcal{A}_k)$ and Adv_k, while $forward(a)$ returns the action forwarded by the dummy adversary with potential g_k-renaming.
- $G^+(q^+) = \widehat{sig}(\mathcal{E}||\mathcal{A}_k)||D_k||Adv_k)((q_{\mathcal{E}}, q_{\mathcal{A}}, q_D q_{Adv})) \setminus [E^+(q^+) \cup F^+(q^+)]$. These actions corresponds to a scenario where a new action is received by the dummy adversary before the appropriate forward. These actions will never be triggered by $\underline{\sigma}'$.

Now, we define a relationship between executions, noted $\alpha \sim \alpha'$, that captures the fact that the latter member α' of the relation corresponds to the former one α when each action shared by $g_k(\mathcal{A}_k)$ and Adv_k is correctly forwarded by dummy adversary in α'.

$\forall(\alpha, \alpha') \in frags^*(\mathcal{E}||g_k(\mathcal{A}_k)||Adv_k) \times frags^*(\mathcal{E}||H_k||Adv_k)$ we note $\alpha \sim \alpha'$ iff:

- (initialisation): $\alpha' = start(\mathcal{E}||H_k, AAct_{\mathcal{A}_k})||Adv_k)$ and $\alpha = start(\mathcal{E}||g(\mathcal{A}_k)||Adv_k)$
- (environment side) $\alpha = (q_{\mathcal{E}}, q_{\mathcal{A}}, q_{Adv})\ a\ (q'_{\mathcal{E}}, q'_{\mathcal{A}}, q'_{Adv})$, $\alpha' = (q_{\mathcal{E}}, q_{\mathcal{A}}, q_D, q_{Adv})\ a\ (q'_{\mathcal{E}}, q'_{\mathcal{A}}, q'_D, q'_{Adv})$ with $a \in E((q_{\mathcal{E}}, q_{\mathcal{A}}, q_{Adv}))$
- (forward) $\alpha = (q_{\mathcal{E}}, q_{\mathcal{A}}, q_{Adv})\ a\ (q_{\mathcal{E}}, q'_{\mathcal{A}}, q'_{Adv})$, $\alpha' = (q_{\mathcal{E}}, q_{\mathcal{A}}, q_D, q_{Adv})\ b\ (q_{\mathcal{E}}, q'_{\mathcal{A}}, q'_D, q'_{Adv})\ b'\ (q_{\mathcal{E}}, q''_{\mathcal{A}}, q''_D, q''_{Adv})$ with $a = origin(b) \in F((q_{\mathcal{E}}, q_{\mathcal{A}}, q_{Adv}))$, $b \in F^+((q_{\mathcal{E}}, q_{\mathcal{A}}, q_D, q_{Adv}))$, $b' = forward(b)$.
- (generalization) $\alpha = \alpha^1{}^\frown\alpha^2{}^\frown ...{}^\frown\alpha^n$, $\alpha' = \alpha'^1{}^\frown\alpha'^2{}^\frown ...{}^\frown\alpha'^n$ and $\forall i \in [1, n]$, $\alpha^i \sim \alpha'^i$

We note $Forward^e_{(\underline{\mathcal{A}}, \underline{g}, \underline{Adv})}(\alpha)$ the (clearly) unique α' s.t. $\alpha \sim \alpha'$.

Now we recursively define $\sigma' = Forward^s_{(\underline{\mathcal{A}}, \underline{g}, \underline{Adv})}(\sigma)$ as follows: Let $(\alpha, \alpha') \in frags^*(\mathcal{E}||g_k(\mathcal{A}_k)||Adv_k) \times frags^*(\mathcal{E}||H_k||Adv_k)$, σ' mimics σ, i. e.

- if $\alpha \sim \alpha'$, $\forall b \in \widehat{sig}(\mathcal{E}||H_k||Adv_k)(lstate(\alpha'))$,
  - if $b \in E^+(lstate(\alpha'))$, $\sigma'(\alpha')(b) = \sigma(\alpha)(b)$
  - if $b \in G^+(lstate(\alpha'))$, $\sigma'(\alpha')(b) = 0$
  - if $b \in F^+(lstate(\alpha'))$, $\sigma'(\alpha')(b) = \sigma(\alpha)(origin(b))$
- if $\alpha' = \alpha''{}^\frown bq'$ with $\alpha \sim \alpha''$ and $b \in F^+(lstate(\alpha''))$, then $\sigma'(\alpha') = \delta_{forward(b)}$.

By construction, for every α' s.t. $|\alpha'| > q_2(k)$, $\sigma'(\alpha') = 0$.

The construction ensures $\epsilon_\sigma(\alpha) = \epsilon_{\sigma'}(\alpha')$ for $\alpha \sim \alpha'$ and $\epsilon_{\sigma'}(\alpha') = 0$ if there is no α'' with α' as prefix with $|\alpha''| = |\alpha'| + 1$ s.t. there exists an execution α verifying $\alpha \sim \alpha''$. Since, by bravery property, for every pair (α, α') with $\alpha \sim \alpha'$, we have $f_{(\mathcal{E}_k, H_k||Adv_k)}(\alpha) = f_{(\mathcal{E}_k, g_k(\mathcal{A}_k)||Adv_k)}(\alpha')$, we obtain $f\text{-}dist_{(\mathcal{E}_k, g_k(\mathcal{A}_k)||Adv_k)}(\sigma) = f\text{-}dist_{(\mathcal{E}_k, H_k||Adv_k)}(\sigma')$.

□

We can use the previous lemma 86 to use the technique of reduction to dummy adversary introduced by Canetti [Can01].

Theorem 34 (Composability of dynamic secure-emulation)**.** *Let (S, f) be a brave pair made of a scheduler schema S and a perception-function f. Let $b \in \mathbb{N}$. Let $\underline{\mathcal{A}}^1, \underline{\mathcal{A}}^2, ..., \underline{\mathcal{A}}^b$ and $\underline{\mathcal{B}}^1, \underline{\mathcal{B}}^2, ..., \underline{\mathcal{B}}^b$ be pair-wise partially-compatible polynomial-time-bounded structured PSIOA (resp. PCA) families, with $\underline{\mathcal{A}}^i \leq^{S,f}_{SE} \underline{\mathcal{B}}^i$ for every $i \in [1, b]$. Then, we have $\widehat{\underline{\mathcal{A}}} \leq_{SE} \widehat{\underline{\mathcal{B}}}$ with $\widehat{\underline{\mathcal{A}}} = \underline{\mathcal{A}}^1||\underline{\mathcal{A}}^2||...||\underline{\mathcal{A}}^b$ and $\widehat{\underline{\mathcal{B}}} = \underline{\mathcal{B}}^1||\underline{\mathcal{B}}^2||...||\underline{\mathcal{B}}^b$.*

Proof. Let $\underline{Adv}$ be an adversary family for $\widehat{\underline{\mathcal{A}}}$ with polynomially bounded description. We need to construct an adversary family $\underline{Sim}$ for $\widehat{\underline{\mathcal{B}}}$ with polynomially bounded description such that: $hide(\widehat{\underline{\mathcal{A}}}||\underline{Adv}, AAct_{\widehat{\underline{\mathcal{A}}}}) \leq^{S,f}_{neg,pt} hide(\widehat{\underline{\mathcal{B}}}||\underline{Sim}, AAct_{\widehat{\underline{\mathcal{B}}}})$.

For every $(i,k) \in [1,b] \times \mathbb{N}$. We note g^i_k an arbitrary bijection from $AAct_{\mathcal{A}^i_k}$ to a set of fresh action names, i. e. $\underline{g}^i = \{g^i_k\}_{k\in\mathbb{N}}$ is a renaming of adversary actions for $\underline{\mathcal{A}}^i$. These functions induce a renaming for $\widehat{\underline{\underline{\mathcal{A}}}}$: $\underline{g} = \{g_k\}_{k\in\mathbb{N}}$ with $\forall k \in \mathbb{N}$, $g_k = g^1_k \cup ... \cup g^b_k$, i. e. $\forall q \in Q_{\widehat{\underline{\mathcal{A}}}_k}$, $\forall a \in AAct_{\widehat{\underline{\underline{\mathcal{A}}}}}(q)$, $g_k(a) = g^i_k(a)$ iff $a \in AAct_{\mathcal{A}^i_k}(q \upharpoonright \mathcal{A}^i_k)$. We recall that cthe ompatibility definition for structured PSIOA requires that shared actions of two automata cannot be adversary actions of their composition.

Let $\widehat{\underline{Dum}} = Dummy(\underline{\mathcal{A}}^1, \underline{g}^1)||...Dummy(\underline{\mathcal{A}}^b, \underline{g}^b)$. Let $i \in [1,b]$. Since $\underline{\mathcal{A}}^i \leq_{SE} \underline{\mathcal{B}}^i$, then $\exists \underline{DSim}^i$ s.t. $hide(\underline{\mathcal{A}}^i||Dummy(\underline{\mathcal{A}}^i, \underline{g}^i), AAct^i_{\underline{\underline{\mathcal{A}}}})$

$\leq_{neg,pt} hide(\underline{\mathcal{B}}^i||\underline{DSim}^i, AAct^i_{\underline{\underline{\mathcal{B}}}})$.

We note $\widehat{\underline{DSim}} = \underline{DSim}^1||...||\underline{DSim}^b$.

We observe the following:

$hide(\widehat{\underline{\mathcal{A}}}||\underline{Adv}, AAct_{\widehat{\underline{\mathcal{A}}}}) \equiv^{S,f}_{neg,pt}$

$hide(\underline{g}(\widehat{\underline{\mathcal{A}}})||\underline{g}(\underline{Adv}), \underline{g}(AAct_{\widehat{\underline{\mathcal{A}}}})) \leq^{S,f}_{neg,pt}$

$hide(\widehat{\underline{\mathcal{A}}}||\widehat{\underline{Dum}}||\underline{g}(\underline{Adv}), \underline{g}(AAct_{\widehat{\underline{\mathcal{A}}}}) \cup AAct_{\widehat{\underline{\mathcal{A}}}}) \leq^{S,f}_{neg,pt}$

$hide(\widehat{\underline{\mathcal{B}}}||\widehat{\underline{DSim}}||\underline{g}(\underline{Adv}), \underline{g}(AAct_{\widehat{\underline{\mathcal{A}}}}) \cup AAct_{\widehat{\underline{\mathcal{B}}}}) \equiv^{S,f}_{neg,pt}$

$hide(\widehat{\underline{\mathcal{B}}}||hide(\widehat{\underline{DSim}}||\underline{g}(\underline{Adv}), \underline{g}(AAct_{\widehat{\underline{\mathcal{A}}}})), AAct_{\widehat{\underline{\mathcal{B}}}})$

Here, the first relation follows from the property of renaming, the second from lemma 86, the third from theorem 33, and the last from the properties of the hiding operator.

We define $\underline{Sim} = hide(\widehat{\underline{DSim}}||\underline{g}(\underline{Adv}), \underline{g}(AAct_{\widehat{\underline{\mathcal{A}}}}))$

Hence we have shown that for every adversary family $\underline{Adv}$ for $\widehat{\underline{\mathcal{A}}}$ with polynomially bounded description there exists a polynomially bounded adversary $\underline{Sim}$ for $\widehat{\underline{\mathcal{B}}}$ such that: $hide(\widehat{\underline{\mathcal{A}}}||\underline{Adv}, AAct_{\widehat{\underline{\mathcal{A}}}}) \leq^{S,f}_{neg,pt}$ $hide(\widehat{\underline{\mathcal{B}}}||\underline{Sim}, AAct_{\widehat{\underline{\mathcal{B}}}})$, which ends the proof.

□

6.3. Summary

We have shown how to merge PSIOA (resp. PCA) with computational restrictions and structures of Canetti et al. [CCK+07] to enable simulation-based security. Here again, the good foundations of chapter 3 have prevented any particular challenge from appearing.

However, some limitations appear. First, the theorem 34 of composability of dynamic secure emulation only holds for a constant number of compositions, while we might be expected a polynomial number of compositions. This limitation should not be too difficult to circumvent, by using a classic hybrid argument [CCK+07]. For example, using the task-based scheduler mentioned in chapter 3, combined with the hybrid argument of [CCK+07], should directly give the result. Second, fixed polynomial runtime has been criticized in compositional simulation-based security literature [HMQU09, KTR20] and is addressed with an alternative, quite sophisticated definition. It would be interesting to show how to address this issue in a more simple LTS-based model like I/O Automata.

Conclusion

Results

We have presented Dynamic Probabilistic I/O Automata. A framework extending in a natural manner from previous I/O automata frameworks [Tut87, Seg95b, KLSV06, Mit07, AL16, CCK$^+$07] to reason on complex dynamic probabilistic systems, where features such as the notion of time and simulation-based security can be easily added. In Chapter 3, the model has been shown (see theorem 2) to be closed under parallel composition $||$. Also, inheriting from previous composability results on trace distribution precongruence $\leq_{DC}$ [Seg95b], our implementation relationship $\leq_{\varepsilon}^{S,f}$ is shown (see theorem 5) to be composable (i.e. a precongruence) for any scheduler schema S, any approximation ε and any perception function f (e.g. trace or projection on the environment). In addition, these results of closure and composabilty have been demonstrated for the timed setting in Chapter 5 (see theorems 26, 27, 29, 30) and for the secured setting in Chapter 6 (see lemma 79 and theorem 34).

Moreover, in Chapter 4, we proposed a generic proof of monotonicity of dynamic creation/destruction of automata with $\leq_0^{S_o,\mathtt{p}}$, where $\mathtt{p}$ is the function of projection on the environment and S_o is the set of schedulers that triggers actions without taking into account the triggered internal actions and the visited states of automata before their last destruction. Hence if (1) $\mathcal{A}$ implements $\mathcal{B}$ and (2) $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ only differ in the fact that $X_{\mathcal{A}}$ dynamically creates/destroys $\mathcal{A}$ instead of $\mathcal{B}$ as $X_{\mathcal{B}}$ does then (3) $X_{\mathcal{A}}$ implements $X_{\mathcal{B}}$. Formally, Theorem 23 states that:

if (1) $\mathcal{A} \leq_0^{S_o,\mathtt{p}} \mathcal{B}$ and (2) $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$, then (3) $X_{\mathcal{A}} \leq_0^{S_o,\mathtt{p}} X_{\mathcal{B}}$.

A similar theorem 24 of monotonicity has been proven for a new task-based implementation relationship $\leq_0^{ten}$, called tenacious implementation:

if (1) $\mathcal{A} \leq_0^{ten} \mathcal{B}$ and (2) $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$, then (3) $X_{\mathcal{A}} \leq_0^{ten} X_{\mathcal{B}}$.

Such a task-based framework might be more user-friendly to specify a fair and oblivious scheduler for cyber-secure applications. The generic structure of the proof should allow the application of the same reasoning to extensions of the framework with time and/or simulation-based security.

All these results allow a sound modular design of dynamic complex systems, based only on the externally visible behavior of each component.

Future Directions

Complementary Results Some questions remain open.

First, we do not know the simulation relationships necessary and/or sufficient to guarantee $\leq_0^{S_o,\mathtt{p}}$ (and even more generally to guarantee $\leq_{\varepsilon}^{S,f}$ for any scheduler schema S, perception function f and approximation ε). The similarity with $\leq_{DC}$ [Seg95a] suggests that such a simulation relationship should be similar to the probabilistic forward simulation relationship [LSV03], but in which formal sense? We stress that this question remains open even for the static setting and is linked with strong

linearizability [GHW11, HHW12], where (1) the hyperproperties preservation depends on the choice of the scheduler schema S and (2) some notions of approximate strong linearizability [AEW22] might be defined. A source of inspiration can be found in [ML07a] that introduces expanded approximate simulation for probabilistic input/output automata.

Second, we do not know if the dynamic creation/destruction of automata is monotonic with approximate implementation relationships such as $\leq_{\varepsilon}^{S_o,\mathfrak{p}}$ or $\leq_{\varepsilon}^{ten}$ (*), i.e.

if (1) $\mathcal{A} \leq_{\varepsilon}^{S_o,\mathfrak{p}} \mathcal{B}$ (resp. $\mathcal{A} \leq_{\varepsilon}^{ten} \mathcal{B}$) $\wedge$ (2) $X_{\mathcal{A}} \underset{\mathcal{A},\mathcal{B}}{\triangledown} X_{\mathcal{B}}$, do we have (3) $X_{\mathcal{A}} \leq_{0}^{S_o,\mathfrak{p}} X_{\mathcal{B}}$ (resp. $X_{\mathcal{A}} \leq_{\varepsilon}^{ten} X_{\mathcal{B}}$)?

Third, if (*) turns out to be true, can we extend it to dynamic secure-emulation. Indeed, the definition of secure-emulation implies a correspondence of one particular simulator for one particular adversary. However, adapting the structure of the proof of monotonicity of chapter 4 to secure-emulation would make appear a tree of different adversaries, which means that the dynamic simulator should find a way to "track" all these different adversaries in a polynomial time. There is nothing to say, a priori, that such an adaptation is possible.

Fourth, the possibility of adapting our monotonicity results to the timed setting also remains open. Here, we are more confident, as we do not see any conceptual differences between timed and untimed implementations. Typically, all the homomorphisms introduced in Chapter 4 can be trivially extended to timed executions, while the copy-pasting operations in Section 4.5 should be adaptable to timed executions as well. Of course, these intuitions do not constitute a proof and a rigorous proof is required here.

Finally, it would be interesting to establish the set of consistency criteria (weaker than (strong) linearizability) [Per16] whose implementation allows monotonicity of dynamic creation/destruction of automata. Here again, the generic structure of the proof of Chapter 4 might help to give a generic answer.

Verification of concrete dynamic distributed protocols In future work, we would like to give formal and modular proofs of concrete distributed algorithms that manipulate notions of dynamicity, randomization, partial synchrony, and cryptography. Such a venture should be faced in a gradual manner. First, we could exhibit non-randomized simulation relationships [LV95] for non-randomized dynamic protocols [DW93, MRT$^+$05, BBKR09, BBR11, BN11, KLW11, BBS11, IRS$^+$13, KLW13, KW19, AKSW20, AKSW22] in a DIOA-based formalism. Second, we could exhibit probabilistic (forward) simulation relationships [LSV03] for randomized dynamic protocols [BKLW12]. Third, such an investigation could be extended to approximate simulations [ML07a] for randomized dynamic protocols (in the symbolic model [DY83]) with a non-zero probability of failure, such as Algorand [CGMV18] or a dynamic scalable Byzantine reliable broadcast, typically obtained from the combination of [GKM$^+$19] and [GKK$^+$20]. Finally, the symbolic model could be replaced by a computational model to obtain a composable simulation-based security analysis [CCK$^+$07]. This would allow the composable analysis of some sophisticated tasks such as dynamic secret sharing [BDLO15, ZZM$^+$19, GKM$^+$20] or side-chains protocols that instantiates new blockchains on-the-fly on the top of the main blockchain [RPG19].

Parallelly, the Dynamic Automata framework gives a simple way to model mobile Byzantine processes [Gar94, BDNPB16, BPPb$^+$16, Del17]. Each process X_i is a PCA with a unique underlying PSIOA $\mathcal{A}_i$ representing the program executed by the process, that additionally admits at each state a countable set of input actions of the form $corrupt_{\mathcal{B}_i}$, such that triggering $corrupt_{\mathcal{B}_i}$ both creates $\mathcal{B}_i$ and destroys $\mathcal{A}_i$ in X_i at the same time. In addition, we require that every malicious program $\mathcal{B}_i$ additionally admits at each state the input $repair_i$ such that triggering $repair_i$ both creates $\mathcal{A}_i$ and destroys $\mathcal{B}_i$ in X_i at the same time. Analyzing protocols resilient to mobile Byzantine processes with this framework is an interesting direction of research.

Sybil-Attack resistance mechanisms Another direction would be the investigation of relationships between Sybil-Attack resistance mechanisms for group membership in the permissionless model [LPR21]. For example, what can be the exact guarantees provided by a Proof-of-Work [GKO$^+$20] mechanism versus a Proof-of-Stake [SW21], versus a Proof-of-Elapsed-time [PK21], versus a Proof-of-Space [TZDC21], etc. Can we compose them in an intuitive sense? Do we need hybrid automata [Mit07] to model properly the physical assumptions?

Also, we can identify different approaches to solving distributed tasks in the permissionless model. A first approach is to ensure a kind of agreement and a Sybil-Attack resistance at once [Nak08], while a second approach separates these two sub-tasks: (1) first, using a proof-of-X mechanism to elect a committee of size $n = o(m)$ among an enormous set of m processes that will then (2) implements a classic Byzantine distributed task (e.g. consensus) on behalf of the other anonymous nodes [PS17]. This second technique allows scalability against a slight concession in terms of security. It seems easier to reason on distributed tasks in the permissionless model if these two different sub-problems are solved independently. Maybe the first achievement holds additional properties that can be used by the second sub-protocol to improve its efficiency. The different pairings between (1) and (2) should be investigated.

Simulation-based security Clearly, Universal-Composable (UC) framework [Can01, Can20] is the most widely adopted model to analyze cryptographic systems with composable simulation-based security. Both UC and alternative models [HS15, KTR20] proposed to overcome some weaknesses of UC are sequential, i.e. pure-non determinism is resolved implicitly via a "master machine" that will select the next machine to activate. It has been shown that non-sequential and sequential scheduling can lead to different notions of security [CCLP07, Yon10]. Hence, a clarification of the potential results of equivalency between the two models would be welcome. In which situations can we simulate UC (or IITM [KTR20] or GNUC [HS15]) with Automata and vice-versa? The same kind of clarifications would be appreciated for timed systems. The frameworks based on sequential Interactive Turing Machines (ITMs) need extra efforts to handle timing assumptions [KMTZ13], while the notion of trajectory [KLSV06, CCK$^+$08] seems more user-friendly. Hence, what are the relationships between these timed frameworks? Finally, what would be the translation of the non-fixed polynomial-runtime [HMQU09] to the non-sequential scheduling?

Final word In this thesis, we provided new tools and new results to reason in a modular and rigorous way about dynamic probabilistic distributed systems. We believe it is a relevant approach to face the vertiginous complexity of modern systems. Such an undertaking should improve the understanding, verification, and synthesis of current and forthcoming real complex systems.

Appendices

Appendix A

Second Part of the thesis: Byzantine Distributed Tasks

A.1. Overview

The second part of the thesis addresses the implementation of Byzantine distributed tasks. Here, we briefly present the associated contributions.

Decision task A *decision task* is a distributed input-output problem between n processes interacting by message-passing, in which each process starts with its *input value* and eventually produces its *output value*. In this second part of the thesis, we are particularly interested in *non-synchronous* protocols: protocols that operate in an environment without (permanent) timely communication. A protocol is said to be a t_0-resilient implementation of a distributed task if it meets the specification of the task despite the presence of $t \leq t_0$ Byzantine processes that collude and perform arbitrary (Byzantine) failures. Unfortunately, for interesting tasks, it is impossible to design a t_0-resilient algorithm that avoids safety violations in the presence of a coalition of $t \geq n - 2t_0$ Byzantine processes, even in the authenticated setting. This comes from a classic partition argument (see for example the proof of theorem 4.4 of [DNS88]).

Accountability By contrast, only recently did the community discover that some of these distributed protocols can be made *accountable* [HKD07, HK09] by ensuring that correct processes irrevocably detect some faulty processes responsible for any safety violation. This realization is particularly surprising (and positive) given that accountability is a powerful tool to mitigate safety violations in distributed protocols. Indeed, exposing crimes and introducing punishments naturally incentivize exemplarity. A protocol is said to be a t_0-resilient accountable implementation of a distributed tasks $\mathcal{D}$ if (1) it is a t_0-resilient accountable implementation of $\mathcal{D}$ and (2) in case of a safety violation, every correct process eventually outputs an irrefutable proof of misbehavior against at least $n - 2t_0$ Byzantine processes.

Consensus Among these tasks, consensus is a fundamental problem of distributed computing which enables processes to agree on a common value despite Byzantine failures. The Byzantine consensus problem is defined among n processes, out of which some processes can behave arbitrarily; processes that follow their protocol are *correct*, whereas processes that do not are *faulty*.

Each correct process initially *proposes* its value and may eventually produce an irrevocable *decision* on some value. The Byzantine consensus problem is characterized by the following properties:

- *Agreement:* No two correct processes decide different values.
- *Validity:* If all correct processes propose the same value v, then no correct process decides a value $v' \neq v$.

– *Termination:* Every correct process eventually decides.

This problem is one of the most important problems in distributed computing since it is equivalent to (1) the problem of atomicity, which provides the illusion that a set of distributed machines acts as (emulates) a unique machine, equivalent itself to (2) problem of state machine replication, where several nodes can observe the same execution of a state machine, even if this machine is emulated by geographically separated processes.

Contributions This part of the thesis is the result of a collaboration with the Distributed Computing Lab (DCL) of EPFL, the Concurrent Systems Research Group (CSRG) of the University of Sydney, and the Department of Computer Science: Algorithms and Theory of NUS Computing, under the respective supervision of Professor Rachid Guerraoui, Professor Vincent Gramoli and Professor Seth Gilbert. This cooperation led to the following 4 publications:

– Polygraph: Accountable Byzantine Agreement. P. Civit, S. Gilbert and V. Gramoli. 41st IEEE International Conference on Distributed Computing Systems (ICDCS), 2021.

– As easy as ABC: Optimal (A)ccountable (B)yzantine (C)onsensus is easy! P. Civit, S. Gilbert, V. Gramoli, R. Guerraoui, and J. Komatovic. 2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2022.
(Best Paper Award)

– Crime and Punishment in Distributed Byzantine Decision Tasks. P. Civit, S. Gilbert, V. Gramoli, R. Guerraoui, J. Komatovic and A. Seredinschi. 42nd IEEE International Conference on Distributed Computing Systems (ICDCS), 2022.
(Best Paper Award)

– Byzantine Consensus is $\Theta(n^2)$: The Dolev-Reischuk Bound is Tight even in Partial Synchrony! P. Civit, M. A. Dzulfikar, S. Gilbert, V. Gramoli, R. Guerraoui, J. Komatovic and M. Vidigueira. 36th International Symposium on Distributed Computing (DISC), 2022.
(Best Student Paper Award)

A.2. Polygraph: Accountable Byzantine Agreement.

In this paper, we present Polygraph [CGG20, CGG21], the first accountable consensus protocol (with optimal $\lceil \frac{n}{3} \rceil - 1$-resiliency). The paper also shows why a large class of "leader-based" Byzantine consensus protocols cannot be made accountable by a naive transformation. We deployed blockchains based on Polygraph on up to $n = 80$ c4.xlarge AWS virtual machines, located in 5 availability zones on two continents: Frankfurt, Ireland, London, North California, and North Virginia.

A.3. As easy as ABC: Optimal (A)ccountable (B)yzantine (C)onsensus is easy!

In this paper [CGG+22a], we present the following contributions:

– We introduce $\mathcal{ABC}$: a simple yet efficient transformation of any Byzantine consensus protocol to an accountable one. $\mathcal{ABC}$ introduces an overhead of only two all-to-all communication rounds and $O(n^2)$ additional bits in executions with up to t_0 faults (i.e., in the common case).

– We define accountability complexity, a complexity metric representing the number of accountability-specific messages that correct processes must send. Furthermore, we prove a tight lower bound. In particular, we show that any accountable Byzantine consensus protocol incurs cubic accountability complexity. Moreover, we illustrate that the bound is tight by applying the $\mathcal{ABC}$ transformation to any Byzantine consensus protocol.

– We demonstrate that, when applied to an optimal Byzantine consensus protocol, $\mathcal{ABC}$ constructs an accountable Byzantine consensus protocol that is (1) optimal with respect to the communication complexity in solving consensus whenever consensus is solvable, and (2) optimal with respect to the accountability complexity in obtaining accountability whenever disagreement occurs.

– We generalize $\mathcal{ABC}$ to other distributed computing problems besides the classic consensus problem. We characterize a class of agreement tasks, including reliable and consistent broadcast, that $\mathcal{ABC}$ renders accountable.

A.4. Crime and Punishment in Distributed Byzantine Decision Tasks.

In this paper [CGG+22b], we propose the first generic transformation, called τ_{scr}, of any *non-synchronous* distributed protocol solving a decision task into its accountable version. Our τ_{scr} transformation is built upon the well-studied simulation of crash failures on top of Byzantine failures and increases the communication complexity by a quadratic multiplicative factor in the worst case.

First, we show that one must be able to detect *commission faults* – faults that occur once a faulty process invalidly sends a message – in order to achieve accountability in a non-synchronous setting. Indeed, we prove that (1) every irrevocable detection must be based on a detected commission fault (otherwise, a correct process can falsely be detected), and (2) (luckily for accountability!) whenever safety is violated, "enough" processes have committed commission faults.

Furthermore, we separate all commission faults into (1) *equivocation faults*, faults associated with an act of claiming conflicting statements, and (2) *evasion faults*, faults that occur once a faulty process sends a message which cannot be sent given the previously received messages. Then, we illustrate that detecting equivocation faults is easier in non-synchronous settings than detecting evasion faults, concluding that equivocation faults are preferable means of violating safety in non-synchronous distributed protocols.

Finally, we observe that the approach exploited by the well-studied simulation of crash failures on top of Byzantine failures can be modified to ensure that evasion faults are *masked* (i.e., their effect is eliminated), thus allowing *only* equivocation faults to violate safety.

Such a simulation is achieved using the secure broadcast primitive [MMR00]: (1) each originally sent message is secure-broadcast, and (2) no secure-delivered message "affects" the receiver before a correct causal past of the message has been established. Hence, no message that is a product of an evasion fault influences a correct process (even if the system is entirely corrupted), implying that any safety violation is necessarily the consequence of some equivocation faults. We base the τ_{scr} transformation on the aforementioned approach based on the secure broadcast. Due to the complexity of the secure broadcast primitive, our transformation increases the communication and message complexities of the original protocol by an $O(n^2)$ multiplicative factor.

A.5. Byzantine Consensus is $\Theta(n^2)$: The Dolev-Reischuk Bound is Tight even in Partial Synchrony!

The Dolev-Reischuk bound says that any deterministic Byzantine consensus protocol has (at least) quadratic communication complexity in the worst case. While it has been shown that the bound is tight in synchronous environments, it was still unknown whether a consensus protocol with quadratic communication complexity can be obtained in partial synchrony, where a bound on time of message exists but holds only eventually. Until now, the most efficient known solutions for Byzantine consensus in partially synchronous settings had cubic communication complexity (e.g., HotStuff, binary DBFT).

This paper [CDG+22] closes the existing gap by introducing SQuad, a partially synchronous Byzantine consensus protocol with quadratic worst-case communication complexity. In addition, SQuad is optimally-resilient and achieves linear worst-case latency complexity. The key technical contribution underlying SQuad lies in the way we solve *view synchronization*, the problem of bringing all correct processes to the same view with a correct leader for sufficiently long. Concretely, we present RareSync, a view synchronization protocol with quadratic communication complexity and linear latency complexity, which we utilize in order to obtain SQuad.

List of Figures

1.1. A representation of 2 automata U and V and their composition at 2 different states with 2 different signatures . . . 8
1.2. A tree of possible executions for a PSIOA $\mathcal{A}$. . . . 9
1.3. An environment $\mathcal{E}$, which is nothing more than a PSIOA compatible with both $\mathcal{A}$ and $\mathcal{B}$, tries to distinguish $\mathcal{A}$ from $\mathcal{B}$. . . . 10
1.4. PCA execution fragment . . . 11
1.5. The respective executions tree of two automata $\mathcal{A}$ and $\mathcal{B}$ are linked by an executions-matching. . . . 16
1.6. PCA deprived of a sub-PSIOA . . . 17
1.7. Simpleton Wrapper . . . 18
1.8. Reconstruction . . . 18
1.9. homomorphism-based-proof . . . 19
1.10. The necessity of a scheduler in S_o. . . . 20

2.1. the variations of the I/O Automata framework . . . 37

3.1. Trivial homomorphism between preserving distribution and distribution on states of automata of its family support. . . . 46
3.2. An intrinsic transition . . . 47
3.3. A PCA life cycle. . . . 48
3.4. Static Non-probabilistic Byzantine Reliable Broadcast . . . 51
3.5. Specification of Byzantine Reliable Broadcast . . . 52
3.6. Static Non-probabilistic Byzantine Reliable Broadcast with Byzantine failures . . . 53
3.7. A simple solution from Imbs and Raynal to Byzantine Reliable Broadcast . . . 53
3.8. Local Protocol: Process P_i following the solution proposed by Imbs and Raynal. . . . 54
3.9. Static Network . . . 54
3.10. Static Probabilistic Consensus without failure . . . 55
3.11. Static Probabilistic Consensus with Byzantine failures . . . 56
3.12. BenOr Consensus . . . 57
3.13. Dynamic Byzantine Reliable Broadcast Real without-failure. . . . 58
3.14. Specification of Dynamic Reliable Broadcast without failure: DRB_{ideal} . . . 59
3.15. Local protocol without failure: Process P_i . . . 59
3.16. Manager . . . 59
3.17. Dynamic Set of Processes (DPIOA) . . . 60
3.18. Network . . . 60
3.19. Dynamic Byzantine Reliable Broadcast with Byzantine failures . . . 61
3.20. Dynamic Byzantine Reliable Broadcast with Byzantine failures . . . 62
3.21. Dynamic Byzantine Reliable Broadcast Specification: DBRB_{ideal} . . . 63
3.22. Relay . . . 64
3.23. Non-deterministic execution requires a scheduler . . . 76
3.24. $\lhd_{\mathcal{AB}}$ corresponding-configuration . . . 83

4.1. Executions-matching . . . 94

4.2. State projection . . . 109
4.3. total probability law for family transition projection . . . 110
4.4. total probability law for preserving configuration . . . 111
4.5. intrinsic transition projection . . . 112
4.6. constructive definition of PCA projection . . . 115
4.7. Projection on PCA (part 1/2, the part 2/2 is in figure 4.8): the original PCA X . . . 116
4.8. Projection on PCA (part 2/2, the part 1/2 is in figure 4.7): the PCA $Y = X \setminus \{T\}$ . . . 116
4.9. Simpleton wrapper . . . 119
4.10. Reconstruction of a PCA via $\mathbf{Z} = (X, X \setminus \{V\})$ . . . 120
4.11. $\lhd_{\mathcal{AB}}$ corresponding-configuration . . . 131
4.12. Signature is not fixed . . . 155
4.13. An execution with clones . . . 156
4.14. An execution with clones: The perspective of sub-automaton $\mathcal{A}$ . . . 157
4.15. homomorphism between PSIOA and PCA . . . 157
4.16. non-deterministic execution requires a (task ?) scheduler . . . 160
4.17. $\mathcal{AB}$-environment-corresponding schedules are also $X_{\mathcal{A}}X_{\mathcal{B}}$-environment-corresponding schedules. . . . 175

Bibliography

[ADGR05] Emmanuelle Anceaume, Xavier Défago, Maria Gradinariu, and Matthieu Roy. Towards a Theory of Self-organization. In *OPODIS*, volume 3974, pages 191–205, 2005.

[ADPBR10] Emmanuelle Anceaume, Xavier Défago, Maria Potop-Butucaru, and Matthieu Roy. A framework for proving the self-organization of dynamic systems. *CoRR*, abs/1011.2, 2010.

[AE19] Hagit Attiya and Constantin Enea. Putting strong linearizability in context: Preserving hyperproperties in programs that use concurrent objects. *Leibniz International Proceedings in Informatics, LIPIcs*, 146, 2019.

[AEW21] Hagit Attiya, Constantin Enea, and Jennifer Welch. Impossibility of Strongly-Linearizable Message-Passing Objects via Simulation by Single-Writer Registers. In *35th International Symposium on Distributed Computing, DISC 2021*, pages 7:1–7:18, Freiburg, Germany (Virtual Conference), 2021. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[AEW22] Hagit Attiya, Constantin Enea, and Jennifer L. Welch. Blunting an Adversary Against Randomized Concurrent Programs with Linearizable Implementations. In *PODC'22: ACM Symposium on Principles of Distributed Computing*, pages 209–219, Salerno, Italy, 2022. Association for Computing Machinery.

[AJ05] James Aspnes and Collin Jackson. Exposing computationally-challenged Byzantine impostors. *Department of Computer Science, Yale University, New Haven, CT, Tech. Rep*, 2005.

[AKSW20] Hagit Attiya, Sweta Kumari, Archit Somani, and Jennifer L. Welch. Store-Collect in the Presence of Continuous Churn with Application to Snapshots and Lattice Agreement. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12514 LNCS:1–15, 2020.

[AKSW22] Hagit Attiya, Sweta Kumari, Archit Somani, and Jennifer L. Welch. Store-collect in the presence of continuous churn with application to snapshots and lattice agreement. *Information and Computation*, 285:104869, 2022.

[AL92] Martín Abadi and Leslie Lamport. An old-fashioned recipe for real time. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 600 LNCS(September):1–27, 1992.

[AL16] Paul C. Attie and Nancy A. Lynch. Dynamic input/output automata: A formal and compositional model for dynamic systems. *Information and Computation*, 249:28–75, 2016.

[ALL+06] Myla Archer, Hong Ping Lim, Nancy Lynch, Sayan Mitra, and Shinya Umeno. Specifying and proving properties of timed I/O automata in the TIOA toolkit. *Proceedings - Fourth ACM and IEEE International Conference on Formal Methods and Models for Co-Design, MEMOCODE'06*, pages 129–138, 2006.

[ALL+08] Myla Archer, Hongping Lim, Nancy Lynch, Sayan Mitra, and Shinya Umeno. Specifying and proving properties of timed I/O automata using Tempo. *Design Automation for Embedded Systems*, 12(1-2):139–170, 2008.

[AR00] Martin Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1872 LNCS(August):3–22, 2000.

[AS85] Bowen Alpern and Fred B. Schneider. Defining liveness. *Information Processing Letters*, 21(4):181–185, 1985.

[BBKR09] Roberto Baldoni, Silvia Bonomi, Anne-marie Kermarrec, and Michel Raynal. Implementing a Register in a Dynamic Distributed System. In *29th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 639—-647, 2009.

[BBR11] Roberto Baldoni, Silvia Bonomi, and Michel Raynal. K-Bounded Set Objects in Eventually Synchronous Distributed Systems With Churn and Continuous Accesses. *ACM International Conference Proceeding Series*, pages 99–104, 2011.

[BBS11] Roberto Baldoni, Silvia Bonomi, and Amir Soltani Nezhad. Validity bound of regular registers with churn and byzantine processes. *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing*, pages 225–226, 2011.

[BDLO15] Joshua Baron, Karim El Defrawy, Joshua Lampkins, and Rafail Ostrovsky. Communication-optimal proactive secret sharing for dynamic groups. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9092:23–41, 2015.

[BDNPB16] François Bonnet, Xavier Défago, Thanh Dang Nguyen, and Maria Potop-Butucaru. Tight bound on mobile Byzantine Agreement. *Theoretical Computer Science*, 609:361–373, 2016.

[BGL02] Andrej Bogdanov, Stephen J. Garland, and Nancy A. Lynch. Mechanical translation of i/o automaton specifications into first-order logic. In *Formal Techniques for Networked and Distributed Systems (FORTE)*, volume 2529, pages 364–368, 2002.

[BKLW12] Silvia Bonomi, Andreas Klappenecker, Hyunyoung Lee, and Jennifer L. Welch. Stochastic modeling of dynamic distributed systems with crash recovery and its application to atomic registers. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7702 LNCS:76–90, 2012.

[BN11] Silvia Bonomi and Amir Soltani Nezhad. Multi-writer regular registers in dynamic distributed systems with byzantine failures. *ACM International Conference Proceeding Series*, pages 8–12, 2011.

[BNS22] Rachel Bricker, Mikhail Nesterenko, and Gokarna Sharma. Blockchain in Dynamic Networks. In *Stabilization, Safety, and Security of Distributed Systems - 24th International Symposium, SSS*, pages 114–129, Clermont-Ferrand, France, 2022. Springer.

[BO83] Michael Ben-Or. Another Advantage of Free Choice: Completely Asynchronous Agreement Protocols. *Proceedings of the Second Annual Symposium on Principles of Distributed Computing*, pages 27–30, 1983.

[BPPb+16] Silvia Bonomi, Antonella Pozzo, Maria Potop-butucaru, Sébastien Tixeuil, Silvia Bonomi, Antonella Pozzo, Maria Potop-butucaru, Sébastien Tixeuil, Optimal Mobile, Silvia Bonomi, Antonella Del Pozzo, and Maria Potop-butucaru. Optimal Mobile Byzantine Fault Tolerant Distributed Storage. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 269—-278, 2016.

[BPW03] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A composable cryptographic library with nested operations. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 220–230, 2003.

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. *Annual Symposium on Foundations of Computer Science - Proceedings*, pages 136–145, 2001.

[Can20] R. Canetti. Universally composable security. *J. ACM*, 67:28:1–28:94, 2020.

[CCK+06a] Ran Canetti, Ling Cheung, Dilsun Kaynar, Moses Liskov, Nancy Lynch, Olivier Pereira, and Roberto Segala. Task-Structured Probabilistic {I/O} Automata. 2006.

[CCK+06b] Ran Canetti, Ling Cheung, Dilsun Kaynar, Moses Liskov, Nancy Lynch, Olivier Pereira, and Roberto Segala. Time-bounded task-PIOAs: A framework for analyzing security protocols. *Distributed Computing, 20th International Symposium (DISC)*, 4167:238–253, 2006.

[CCK+07] Ran Canetti, Ling Cheung, Dilsun Kaynar, Nancy Lynch, and Olivier Pereira. Compositional security for task-PIOAs. *Proceedings - IEEE Computer Security Foundations Symposium*, pages 125–139, 2007.

[CCK+08] Ran Canetti, Ling Cheung, Dilsun Kaynar, Nancy Lynch, and Olivier Pereira. Modeling computational security in long-lived systems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5201 LNCS:114–130, 2008.

[CCK+18] Ran Canetti, Ling Cheung, Dilsun Kaynar, Moses Liskov, Nancy Lynch, Olivier Pereira, and Roberto Segala. Task-structured probabilistic I/O automata. *Journal of Computer and System Sciences*, 94:63–97, 2018.

[CCLP07] Ran Canetti, Ling Cheung, Nancy Lynch, and Olivier Pereira. On the Role of Scheduling in Simulation-Based Security. *IACR Cryptol. ePrint Arch.*, page 102, 2007.

[CDG+22] Pierre Civit, Muhammad Ayaz Dzulfikar, Seth Gilbert, Vincent Gramoli, Rachid Guerraoui, Jovan Komatovic, and Manuel Vidigueira. Byzantine Consensus is $\Theta(n^2)$: The Dolev-Reischuk Bound is Tight even in Partial Synchrony! In *36th International Symposium on Distributed Computing (DISC)*, 2022.

[CGG20] Pierre Civit, Seth Gilbert, and Vincent Gramoli. Brief announcement: Polygraph: Accountable Byzantine Agreement. In *34th International Symposium on Distributed Computing (DISC)*, volume 179, pages 3–5, 2020.

[CGG21] Pierre Civit, Seth Gilbert, and Vincent Gramoli. Polygraph: Accountable Byzantine Agreement 1. In *41st IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2021.

[CGG+22a] Pierre Civit, Seth Gilbert, Vincent Gramoli, Rachid Guerraoui, and Jovan Komatovic. As easy as ABC: Optimal (A)ccountable (B)yzantine (C)onsensus is easy! In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2022.

[CGG+22b] Pierre Civit, Seth Gilbert, Vincent Gramoli, Rachid Guerraoui, Jovan Komatovic, and Adi Seredinschi. Crime and Punishment in Distributed Byzantine Decision Tasks. In *42nd IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2022.

[CGMV18] Jing Chen, Sergey Gorbunov, Silvio Micali, and Georgios Vlachos. ALGORAND AGREEMENT: Super Fast and Partition Resilient Byzantine Agreement. *Cryptology ePrint Archive*, 377:1–10, 2018.

[CH11] Ran Canetti and Jonathan Herzog. Universally composable symbolic security analysis. *Journal of Cryptology*, 24(1):83–147, 2011.

[CKS20] Shir Cohen, Idit Keidar, and Alexander Spiegelman. Not a COINcidence: Sub-Quadratic Asynchronous Byzantine Agreement WHP. In *34th International Symposium on Distributed Computing (DISC)*, pages 25:1—-25:17, 2020.

[CM88] K. Mani Chandy and Jayadev Misra. *Parallel Program Design: A Foundation*. Addison-Wesley Pub, 1988.

[CMP07] Ling Cheung, Sayan Mitra, and Olivier Pereira. Verifying Statistical Zero Knowledge with Approximate Implementations. *IACR Cryptology ePrint Archive*, 2007:195, 2007.

[CS08] Michael R. Clarkson and Fred B. Schneider. Hyperproperties. *Proceedings - IEEE Computer Security Foundations Symposium*, 0424422:51–65, 2008.

[CSS10] Salvatore Campana, Luca Spalazzi, and Francesco Spegni. Dynamic networks of timed automata for collaborative systems: A network monitoring case study. *2010 International Symposium on Collaborative Technologies and Systems, CTS 2010*, pages 113–122, 2010.

[CW05] Véronique Cortier and Bogdan Warinschi. Computationally sound, automated proofs for security protocols. In *Programming Languages and Systems, 14th European Symposium on Programming, ESOP*, volume 3444, pages 157–171, 2005.

[Del17] Antonella Del Pozzo. *Building distributed computing abstractions in the presence of mobile byzantine failures*. PhD thesis, Pierre and Marie Curie University, Paris, 2017.

[DGHK05] L. Davis, R. Gamble, M. Hepner, and M. Kelkar. Toward formalizing service integration glue code. In *Proceedings - 2005 IEEE International Conference onServices Computing, SCC 2005*, volume I, pages 165–172, 2005.

[DGV06] Ajoy Kumar Datta, Maria Gradinariu, and Antonino Virgillito. Deterministic delta-Connected Overlay for Peer-to-Peer Networks. In *Ninth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, pages 159—-168, 2006.

[DNS88] Cynthia Dwork, Lynch Nancy, and Larry Stockmeyer. Concensus in presence of Partial Synchrony. *Journal of the ACM (JACM)*, 35(2):288–323, 1988.

[DS95] Rocco De Nicola and Roberto Segala. A process algebraic view of input/output automata. *Theoretical Computer Science*, 138(2):391–423, 1995.

[Dud04] R. M. Dudley. *Real Analysis and Probability*. Cambridge Studies in Advanced Mathematics, 2004.

[DW93] Shlomi Dolev and Jennifer L. Welch. Crash resilient communication in dynamic networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 725 LNCS(September 1993):129–144, 1993.

[DY83] Danny Dolev and Andrew C. Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[DZ22] Sisi Duan and Haibin Zhang. Foundations of Dynamic BFT. *Proceedings - IEEE Symposium on Security and Privacy*, 2022-May:1317–1334, 2022.

[FHN+11] Jasmin Fisher, Thomas A. Henzinger, Dejan Nickovic, Nir Piterman, Anmol V. Singh, and Moshe Y. Vardi. Dynamic reactive modules. In *Concurrency Theory - 22nd International Conference (CONCUR)*, volume 6901, pages 404—-418. Springer, 2011.

[FLP85] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of Distributed Consensus with One Faulty Process. *Journal of the Association for Computing Machinery,*, 32(2):374–382, 1985.

[Gar94] Juan A. Garay. Reaching (And maintaining) agreement in the presence of mobile faults. In *Distributed Algorithms, 8th International Workshop, WDAG '94*, volume 857, pages 253–264, Terschelling, 1994. Springer.

[GHW11] Wojciech Golab, Lisa Higham, and Philipp Woelfel. Linearizable implementations do not suffice for randomized distributed computation. *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 373–382, 2011.

[GKK+20] Rachid Guerraoui, Jovan Komatovic, Petr Kuznetsov, Yvonne-Anne Pignolet, Dragos-Adrian Seredinschi, and Andrei Tonkikh. Dynamic Byzantine Reliable Broadcast. In *Opodis*, 2020.

[GKM+19] Rachid Guerraoui, Petr Kuznetsov, Matteo Monti, Matej Pavlovic, and Dragos-Adrian Seredinschi. Scalable Byzantine Reliable Broadcast (Extended Version). *Leibniz International Proceedings in Informatics, LIPIcs*, 146, aug 2019.

[GKM+20] Vipul Goyal, Abhiram Kothapalli, Elisaweta Masserova, Bryan Parno, and Yifan Song. Storing and Retrieving Secrets on a Blockchain. *IACR Cryptology ePrint Archive*, (2020/504):1–62, 2020.

[GKO+20] Juan Garay, Aggelos Kiayias, Rafail M. Ostrovsky, Giorgos Panagiotakos, and Vassilis Zikas. Resource-Restricted Cryptography: Revisiting MPC Bounds in the Proof-of-Work Era. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12106 LNCS:129–158, 2020.

[GL00] Stephen Garland and Nancy Lynch. Using I/O Automata for Developing Distributed Systems. In *Foundations of Component Based Systems*, pages 285–312. Cambridge University Press, 2000.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, pages 270–299, 1984.

[GMR85] Shafi Goldwasser, Silvio Micali, and Chales Rackoff. The knowledge complexity of interactive proof-systems. 1985.

[HHW12] Maryam Helmi, Lisa Higham, and Philipp Woelfel. Strongly linearizable implementations: Possibilities and impossibilities. *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing*, pages 385–393, 2012.

[HK09] Andreas Haeberlen and Petr Kuznetsov. The Fault Detection Problem. *OPODIS*, (December 2009), 2009.

[HKD07] Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. PeerReview: Practical accountability for distributed systems. *SOSP'07 - Proceedings of 21st ACM SIGOPS Symposium on Operating Systems Principles*, pages 175–188, 2007.

[HMQU09] Dennis Hofheinz, Jörn Müller-Quade, and Dominique Unruh. Polynomial runtime in simulatability definitions. *Journal of Computer Security*, 17(5):703–735, 2009.

[Hoa85] Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[HS15] Dennis Hofheinz and Victor Shoup. GNUC: A New Universal Composability Framework. *Journal of Cryptology*, 28(3):423–508, 2015.

[IR15] Damien Imbs and Michel Raynal. Simple and Efficient Reliable Broadcast in the Presence of Byzantine Processes. pages 1–7, 2015.

[IRS+13] Rebecca Ingram, Tsvetomira Radeva, Patrick Shields, Saira Viqar, Jennifer E. Walter, and Jennifer L. Welch. *A leader election algorithm for dynamic networks with causal clocks*, volume 26. 2013.

[JMMS10] Aaron D. Jaggard, Catherine Meadows, Michael Mislove, and Roberto Segala. Reasoning about probabilistic security using task-PIOAs. *Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security - Joint Workshop, ARSPA-WITS*, 6186:2–22, 2010.

[Kap09] Tatjana Kapus. Using mobile TLA as a logic for dynamic I/O automata. *IEICE Transactions on Information and Systems*, E92-D(8):1515–1522, 2009.

[KDMR08] Ralf Küsters, Anupam Datta, John C. Mitchell, and Ajith Ramanathan. On the relationships between notions of simulation-based security. *Journal of Cryptology*, 21(4):492–546, 2008.

[KLSV03] Dilsun K. Kaynar, Nancy Lynch, Roberto Segala, and Frits Vaandrager. Timed I/O automata: A mathematical framework for modeling and analyzing real-time systems. *Proceedings - Real-Time Systems Symposium*, pages 166–177, 2003.

[KLSV06] Dilsun K. Kaynar, Nancy Lynch, Roberto Segala, and Frits Vaandrager. *The theory of timed I/O automata*, volume 1. Morgan {\&} Claypool Publishers, 2006.

[KLW11] Andreas Klappenecker, Hyunyoung Lee, and Jennifer L. Welch. Quorum-based dynamic regular registers in systems with churn. *ACM International Conference Proceeding Series*, pages 3–7, 2011.

[KLW13] Andreas Klappenecker, Hyunyoung Lee, and Jennifer L. Welch. Dynamic regular registers in systems with churn. *Theoretical Computer Science*, 512:84–97, 2013.

[KMTZ13] Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Universally composable synchronous computation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7785 LNCS:477–498, 2013.

[KPH15] Ilham W. Kurnia and Arnd Poetzsch-Heffter. Modeling actor systems using dynamic I/O automata. *Perspectives of Systems Informatics - Ershov Memorial Conference*, 9609:186–202, 2015.

[KSPL06] Fabian Kratz, Oleg Sokolsky, George J. Pappas, and Insup Lee. R-Charon, a modeling language for reconfigurable hybrid systems. *Hybrid Systems: Computation and Control, 9th International Workshop (HSCC)*, 3927:392–406, 2006.

[KT20] Petr Kuznetsov and Andrei Tonkikh. *Asynchronous reconfiguration with byzantine failures*, volume 179. Association for Computing Machinery, 2020.

[KTR20] Ralf Küsters, Max Tuengerthal, and Daniel Rausch. The IITM Model: A Simple and Expressive Model for Universal Composability. *Journal of Cryptology*, 33(4):1461–1584, 2020.

[Kur15] Ilham W. Kurnia. *An Automata-Theoretic Approach to Open Actor System Verification*. PhD thesis, 2015.

[KW19] Saptaparni Kumar and Jennifer L. Welch. Byzantine-Tolerant Register in a System with Continuous Churn. *CoRR*, abs/1910.0, 2019.

[Lam] Leslie Lamport. Some Thoughts on Specification. https://lamport.azurewebsites.net/tla/notes/92-05-05.txt.

[Lam93] Leslie Lamport. Verification and specification of concurrent programs. In J. W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg, editors, *A Decade of Concurrency, Reflections and Perspectives*, volume 803, pages 347–374. Springer, 1993.

[Lam94] Leslie Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, 1994.

[Lin17] Yehuda Lindell. *How to simulate it – A tutorial on the simulation proof technique*. Springer International Publishing, 2017.

[LKLM05] Hongping Lim, Dilsun K. Kaynar, Nancy Lynch, and Sayan Mitra. Translating Timed I/O Automata Specifications for Theorem Proving in PVS. In *Formal Modeling and Analysis of Timed Systems, Third International Conference, (FORMATS)*, volume 3829, pages 17—-31, 2005.

[LPR21] Andrew Lewis-Pye and Tim Roughgarden. *Byzantine Generals in the Permissionless Setting*, volume abs/2101.0. Association for Computing Machinery, 2021.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[LSV03] Nancy Lynch, Roberto Segala, and Frits Vaandrager. Compositionality for Probabilistic Automata. In *CONCUR 2003 - Concurrency Theory, 14th International Conference*, volume 2761, pages 204–222, Marseille, France, 2003. Springer.

[LSVW95] Nancy Lynch, Roberto Segala, Frits Vaandrager, and H B Weinberg. Hybrid I / O Automata. In *Hybrid Systems III: Verification and Control, Proceedings of the (DIMACS/SYCON) Workshop on Verification and Control of Hybrid Systems*, pages 496—-510, 1995.

[LT87] Nancy A. Lynch and Mark R. Tuttle. Hierarchical correctness proofs for distributed algorithms. *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing*, (August):137–151, 1987.

[LTWW22] Jing Li, Yu Tianming, Ye Wang, and Roger Wattenhofer. Dynamic Byzantine Broadcast in Asynchronous Message-Passing Systems. *IEEE Access*, 10:91372—-91384, 2022.

[LV95] Nancy A. Lynch and Frits W. Vaandrager. Forward and Backward Simulations, I. Untimed Systems. *Information and Computation*, 121(2):214–233, 1995.

[Mil80] Robin Milner. *A Calculus of Communicating Systems*. Springer-Verlag Berlin, 1980.

[Mit07] Sayan Mitra. *A Verification Framework for Hybrid Systems*. PhD thesis, 2007.

[ML07a] Sayan Mitra and Nancy Lynch. Proving Approximate Implementations for Probabilistic I/O Automata. *Electronic Notes in Theoretical Computer Science*, 174(8):71–93, 2007.

[ML07b] Sayan Mitra and Nancy Lynch. Trace-based semantics for probabilistic timed I/O automata. *Hybrid Systems: Computation and Control, 10th International Workshop (HSCC)*, 4416:718–722, 2007.

[MMR00] Dahlia Malkhi, Michael Merritt, and Ohad Rodeh. Secure reliable multicast protocols in a WAN. *Distributed Computing*, 13(1):19–28, 2000.

[MRT+05] Achour Mostefaoui, Michel Raynal, Corentin Travers, Stacy Patterson, Divyakant Agrawal, and Amr El Abbadi. From static distributed systems to dynamic systems. *Proceedings of the IEEE Symposium on Reliable Distributed Systems*, pages 109–118, 2005.

[Mül98] Olaf Müller. *A Verification environment for I/O Automata Based on Formalized Meta-theory.* PhD thesis, Technischen Universität München, 1998.

[MW05] Daniele Micciancio and Bogdan Warinschi. Soudness of formal encryption in the presence of active adversaries. In *Theory of Cryptography, First Theory of Cryptography Conference (TCC)*, volume 2951, pages 133—-151, 2005.

[Nak08] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf, 2008.

[Per16] Matthieu Perrin. *Spécification des objets partagés dans les systèmes répartis sans-attente. (Specification of shared objects in wait-free distributed systems).* PhD thesis, University of Nantes, 2016.

[PK21] Amitangshu Pal and Krishna Kant. DC-PoET: Proof-of-Elapsed-Time Consensus with Distributed Coordination for Blockchain Networks. *2021 IFIP Networking Conference, IFIP Networking 2021*, 2021.

[Pla10] André Platzer. Quantified differential dynamic logic for distributed hybrid systems. *Computer Science Logic, 24th International Workshop, (CSL) 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27,2010. Proceedings*, 6247 LNCS:469–483, 2010.

[Pla11] André Platzer. Quantified differential invariants. *HSCC'11 - Proceedings of the 2011 ACM/SIGBED Hybrid Systems: Computation and Control*, pages 63–72, 2011.

[Pla12] André Platzer. Dynamic Logics of Dynamical Systems. *CoRR*, abs/1205.4, 2012.

[PS17] Rafael Pass and Elaine Shi. Hybrid Consensus : Efficient Consensus in the. In *31st International Symposium on Distributed Computing, DISC*, number 39, pages 1–16, Vienna, Austria, 2017. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

[PSL80] M. Pease, R. Shostak, and L. Lamport. Reaching Agreement in the Presence of Faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.

[PW00] Birgit Pfitzmann and Michael Waidner. Composition and Integrity Preservation of Secure Reactive Systems. In *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS)*, pages 245–254, 2000.

[RKC22] Daniel Rausch, Ralf Küsters, and Céline Chevalier. Embedding the UC Model into the IITM Model. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 13276 LNCS:242–272, 2022.

[RPG19] Alejandro Ranchal-Pedrosa and Vincent Gramoli. Platypus: a Partially Synchronous Offchain Protocol for Blockchains. *18th IEEE International Symposium on Network Computing and Applications, (NCA)*, jul 2019.

[SADADJ05] Muthian Sivathanu, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, and Somesh Jha. A logic of file systems. In *FAST 2005 - 4th USENIX Conference on File and Storage Technologies*, pages 1–15, 2005.

[Seg93] Roberto Segala. Quiescence, fairness, testing, and the notion of implementation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 715 LNCS:324–338, 1993.

[Seg95a] Roberto Segala. A compositional trace-based semantics for probabilistic automata. *CONCUR 95: Concurrency Theory, 6th International Conference*, 962:234–248, 1995.

[Seg95b] Roberto Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Massachusettes Institute of technology, 1995.

[SK17] Alexander Spiegelman and Idit Keidar. On liveness of dynamic storage. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10641 LNCS:356–376, 2017.

[SKM17] Alexander Spiegelman, Idit Keidar, and Dahlia Malkhi. Dynamic reconfiguration: Abstraction and optimal asynchronous solution. *Leibniz International Proceedings in Informatics, LIPIcs*, 91:1–27, 2017.

[SW21] Jakub Sliwinski and Roger Wattenhofer. Asynchronous Proof-of-Stake. In *Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 194—-208, 2021.

[Ter22] Benjamin Terner. Permissionless Consensus in the Resource Model. In *Financial Cryptography and Data Security - 26th International Conference, FC*, volume 13411, pages 577–593, Grenada, 2022. Springer.

[Tut87] Mark R. Tuttle. *Hierarchical correctness proofs for distributed algorithms*. PhD thesis, Massachusetts Institute of Technology, 1987.

[TZDC21] Shuyang Tang, Jilai Zheng, Yao Deng, and Qinxiang Cao. Resisting newborn attacks via shared Proof-of-Space. *Journal of Parallel and Distributed Computing*, 150:85–95, 2021.

[Vaa91] Frits W. Vaandrager. On the relationship between process algebra and input/output automata. *Proceedings - Symposium on Logic in Computer Science*, pages 387–398, 1991.

[WSS97] Sue Hwey Wu, Scott A. Smolka, and Eugene W. Stark. Composition and behaviors of probabilistic I/O automata. *Theoretical Computer Science*, 176:1—-38, 1997.

[YKO07] Kazuki Yoneyama, Yuichi Kokubun, and Kazuo Ohta. A Security Analysis on Diffie-Hellman Key Exchange against Adaptive Adversaries using Task-Structured PIOA. *Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA)*, pages 131–148, 2007.

[Yon10] Kazuki Yoneya. Indifferentiable Security Reconsidered: Role of Scheduling. In Mike Burmester, Gene Tsudik, Spyros S. Magliveras, and Ilic Ivana, editors, *Information Security - 13th International Conference, {ISC}*, pages 430–444. Springer, 2010.

[Yon18] Kazuki Yoneyama. Formal modeling of random oracle programmability and verification of signature unforgeability using task-PIOAs. *International Journal of Information Security*, 17(1):43–66, 2018.

[YSSY15] Ryo Yanase, Tatsunori Sakai, Makoto Sakai, and Satoshi Yamane. Formal verification of dynamically reconfigurable systems. In *2015 IEEE 4th Global Conference on Consumer Electronics, GCCE 2015*, pages 71–75, 2015.

[Zim03] Daniel M. Zimmerman. A UNITY based formalism for dynamic distributed systems. *Proceedings - International Parallel and Distributed Processing Symposium (IPDPS)*, 2003.

[ZZM+19] Fan Zhang, Yupeng Zhang, Sai Krishna Deepak Maram, Lun Wang, Ari Juels, Andrew Low, and Dawn Song. ChURP: Dynamic-committee proactive secret sharing. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 2369–2386, 2019.