Semantics of Strategy Logic

Patrick Gardy

▶ **To cite this version:**

Patrick Gardy. Semantics of Strategy Logic. Computer science. Université Paris Saclay (COmUE), 2017. English. NNT : 2017SACLN022 . tel-01561802

HAL Id: tel-01561802
https://theses.hal.science/tel-01561802

Submitted on 13 Jul 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

NNT: 2017SACLN022

Thèse de Doctorat
de l'Université Paris-Saclay
préparée à l'École Normale Supérieure
de Cachan (École normale supérieure Paris-Saclay)

École Doctorale N°580
Sciences et technologies de l'information et de la communication

Spécialité: informatique

par:
Patrick Gardy

Semantics of Strategy Logic

Thèse présentée et soutenue à Cachan, le 12 Juin 2017 devant le jury composé de :

Francesco BELARDINELLI	Maître de conférences Université Evry-Val-d'Essonne	Examinateur
Patricia BOUYER	Directrice de recherche CNRS	Co-Directrice de thèse
Thomas BRIHAYE	Chargé de cours Université de Mons	Rapporteur
Cătălin DIMA	Professeur Université Paris-Est Créteil	Examinateur
Nicolas MARKEY	Directeur de recherche CNRS	Co-directeur de thèse
Aniello MURANO	Associate Professor Università di Napoli Federico II	Rapporteur
Sophie PINCHINAT (présidente du jury)	Professeur Université Rennes I	Examinateur

Work supported by the ERC project EQualIS
done as a PhD student for the CNRS
in the Laboratoire Spécification et Vérification, UMR 8643 du CNRS

Contents

Introduction

Automated verification and model checking

Computerised and electronic systems are now part of our every day life: administration, commerce, communication, education, energy, health, media or transportation. The ever increasing number of electronic devices goes in pair with an ever increasing number of challenges. Among the most famous ones we find cryptography, privacy, bugs and design flaws. Errors in critical systems (such as in aeronautic, banking, energy, health and weaponry) have a heavy human or economic cost and every new one has made software verification more and more essential. Among hundreds of faulty softwares we highlight four:

- The Therac-25 accident was among the first failures to gain public attention. Therac-25 was a radiation therapy machine used in the 1980's to treat cancer in Canada and the USA. Between 1985 and 1987, a programming error caused multiple patients to receive massive overdose of radiation. These overdoses resulted in multiple deaths and a degradation of some patients' condition. Leveson and Turner detail their finding about the incident in [32].

- *Ariane 5* was a rocket launcher developed by the ESA (European Space Agency). During its first test flight in 1996, the rocket was prematurely terminated due to an integer-overflow error. The CNES commission charged to investigate the accident made their conclusion available [34], we quote its conclusion.

  > The failure of the Ariane 501 was caused by the complete loss of guidance and altitude information 37 seconds after start of the main engine ignition sequence (30 seconds after lift-off). This loss of information was due to specification and design errors in the software of the inertial reference system.
  >
  > The extensive reviews and tests carried out during the Ariane 5 Development Program did not include adequate analysis and testing of the inertial reference system or of the complete flight control system, which could have detected the potential failure.

  They also suggest to:

> Organise, for each item of equipment incorporating software, a specific software qualification review. The Industrial Architect shall take part in these reviews and report on complete system testing performed with the equipment.
> ...
> Review all flight software (including embedded software), and in particular : Identify all implicit assumptions made by the code and its justification documents on the values of quantities provided by the equipment. Check these assumptions against the restrictions on use of the equipment. Verify the range of values taken by any internal or communication variables in the software. Solutions to potential problems in the on-board computer software, paying particular attention to on-board computer switch over, shall be proposed by the project team and reviewed by a group of external experts, who shall report to the on-board computer Qualification Board.
> ...
> Set up a team that will prepare the procedure for qualifying software, propose stringent rules for confirming such qualification, and ascertain that specification, verification and testing of software are of a consistently high quality in the Ariane 5 program. Including external RAMS [editor's note: Reliability, Availability, Maintainability and Safety] experts is to be considered.

- In 2004 a blackout occurs on north-east regions of the north american sub-continent. A simple blackout cascaded into a massive and generalised power outage due to a software bug in the alarm system. The USA-Canada power system outage task force in its official report [22] draws the following technical recommendations.

  > Develop and deploy IT management procedures: CAs' and RCs' IT and EMS support personnel should develop procedures for the development, testing, configuration, and implementation of technology related to EMS automation systems and also define and communicate information security and performance requirements to vendors on a continuing basis
  > ...
  > Implement controls to manage system health, network monitoring, and incident management: IT and EMS support personnel should implement technical controls to detect, respond to, and recover from system and network problems. Grid operators, dispatchers, and IT and EMS support personnel should be provided the tools and training to ensure that the health of IT systems is monitored and maintained.

  The wish for better reliability can also be found in one of their institutional recommendations:

  > DOE [editor's note: U.S.A Department of Energy] should expand its

> research agenda, and consult frequently with Congress, FERC, NERC, state regulators, Canadian authorities, universities, and the industry in planning and executing this agenda.
> ...
> More investment in research is needed to improve grid reliability, with particular attention to improving the capabilities and tools for system monitoring and management. Research on reliability issues and reliability-related technologies has a large public-interest component, and government support is crucial.
> ....
> Study of air traffic control, the airline industry, and other relevant industries for practices and ideas that could reduce the vulnerability of the electricity industry and its reliability managers to human error.

The mention of the air traffic control and airline industry is not without importance. Aeronautics and space related industries have been the leading force in research on software verification for many years. The joint task force believes results developed for aeronautics may transfer to the energy sector, an opinion shared by most people in the verification community that lead to the development of quantitative verification at the turn of the new millennium.

- We finish on a more recent case: the ExoMars Schiaparelli. Schiaparelli EDM lander was a sonde in the ExoMars mission of the ESA (European Space Agency) and ROSCOSMOS (Russian Space Programm). The sonde was supposed to land in a plain of Mars on 19 October 2016. Schiaparelli however crashed on the surface of Mars due to a software malfunction. The full investigation will be conducted early 2017 but a preliminary report indicates that the Inertial Measurement Unit (IMU) worked a second longer than expected. When merged into the system, the additional informations made Schiaparelli believe its altitude was negative which then caused an early release of the parachute.

Checking for bugs and unwanted behaviour is usually done in one of three ways: testing for error (by generating an adequate battery of tests to run on the software), checking for proof of correctness (finding a formal/mathematical proof that there cannot be errors within the software's code) and exploring all the potential executions of the system. In this thesis, we will exclusively focus on the latter technique.

Checking for a proof of correct behaviour through the exploration of all potential executions can usually be decomposed in two questions: "How do I want my system to behave ?" (specification) and "Does my system behave as expected?" (model checking). This two-step method evidently requires a common framework able to handle multiple problems at once and resulted in the development of *formal verification*: methods working for classes of systems and set of properties respecting a predetermined formalism. Checking for a given property to hold on a given system can then be done by:

1. Specifying the system in a model (a formal representation of the system).

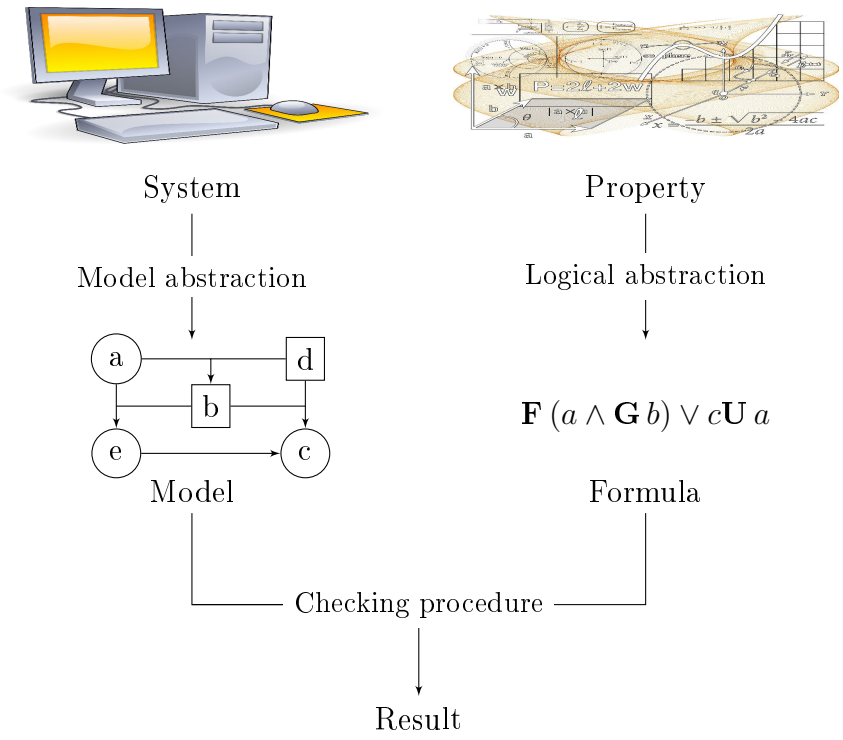


Figure 1: Model checking

2. Specifying the property in a known formalism (a condition to validate or disapprove an execution of the system).

3. Checking the property expressed adequately in the chosen formalism onto the model. We find here the usual questions of time and space efficient algorithms.

4. Transposing the answer from the formal method of the previous step to our system and property.

Figure 1 illustrates the idea. The model abstraction on the left part of the figure represents the first step, the logical abstraction on the right part mirrors the second point and the checking procedure at the bottom represents the last two steps.

Specifications for closed systems

A *closed system* is either an electronic component or a software that does not receive nor alter its environment past the initial conditions. Simple examples can be found in programs calculating standard mathematical functions: the program looks at the entry, does some calculations and returns the result. By opposition, an *open system* interacts and is influenced by its environment. Any electronic device modifying its behaviour based on the surrounding atmospheric conditions is, de facto, an open system.

The verification of closed systems has been extensively developed in the last fifty years. Numerous models have been proposed to represent closed system in a formal way: graphs, transition systems, Petri-nets. . . ; similarly, multiple formalisms have been associated with each of these models. To express complex properties, researchers often rely on *logics*: a set of properties that can all be built by following some common rules. Temporal logics extend propositional logics with modalities in order to specify some properties of executions of the programs. With temporal logics, one can express properties like the one of Figure 2 (with an adapted syntax to make the formula readable).

$$\underset{1}{\mathbf{E}}\ (\ \underset{2}{\mathbf{F}\,p}\ \underset{3}{\wedge}\ \underset{4}{\mathbf{G}\,p'}\ ) \begin{cases} \underset{1}{\text{There exists an execution}} \\ \underset{2}{\text{that will eventually satisfy } p} \\ \underset{3}{\text{and}} \\ \underset{4}{\text{that, at every step, satisfies } p'} \end{cases}$$

Figure 2: A temporal formula for closed systems.

Formulas for closed systems are to be interpreted on transition systems, i.e. directed graphs labelled with atomic propositions. Temporal logics can be decomposed in three non-disjoint families: branching time logics (see CTL [16]), linear time logics (like LTL [44]) and fixed-point modal logics (such as the μ-calculus L_μ, see [57]). The linear time logics target a single execution of the system while the branching time logics focus on multiple executions with, usually, simpler objectives. It is possible to combine both aspects (as in CTL^* [16]). The fixed-point modal logics are, as the name suggest, based on fixed-point operator.

Specification for open systems

Open systems are software or electronic devices that receive informations and interact with their environment. In closed systems, formulas express properties of paths or sets of paths (for instance, one can say that all executions of the system are safe) . With multiple agents, each agent has a specific behaviour (or strategy) and the execution is determined by the mash-up of all strategies. We can represent the environment by an agent of the system, whose behaviour can be erratic. Logics have then to integrate these agents in their syntax. We give in Figure 3 an adaptation to open systems of the formula given in the closed-system section, C represents a coalition, i.e. a set of agents.

Almost none of the models for closed systems can be used to model open systems; several formalisms have therefore been developed. The usual models for open systems are turn-based and concurrent game structures, both adapted from transition systems to integrate the possible existence of multiple agents. Similarly, many logic formalisms used

$$\underset{1}{\langle\!\langle C \rangle\!\rangle} \left(\underset{2}{\mathbf{F}\, p} \underset{3}{\wedge} \underset{4}{\mathbf{G}\, p'} \right) \begin{cases} \underset{1}{\text{There exist strategies for the agents of } C \text{ such that}} \\ \underset{1}{\text{no matter the strategies of the other agents,}} \\ \underset{2}{\text{the resulting path will eventually satisfy } p} \\ \underset{3}{\text{and}} \\ \underset{4}{\text{at every step, satisfy } p'} \end{cases}$$

Figure 3: An adaptation of the formula of Figure 2 to open systems.

to specify closed systems properties have been extended to work on open systems. Over the years, the CTL and CTL* temporal logics have been adapted respectively as ATL and ATL* and the μ-calculus L_μ have been extend in an alternating version AL_μ. (see [2]).

Multiple agents with multiple objectives

In this thesis, we move away from the traditional works on open systems where a device is responding to its environment to study multi-objective properties on multi-agent systems. An example can be found in simulation softwares: we may wish to study different simulations and check which properties they share and what differentiates them from one another. The simulations may share some common behaviours (from agents with fixed strategies) while differing in some others (agents representing the environment).

Many difficulties arise when the goal of an agent depends from the behaviour, and therefore the goals, of other agents. Most techniques developed to handle single objective specifications usually work by assuming the worst behaviour possible from the environment and finding a solution for it. This however cannot be done in multi-objective specifications as the optimal behaviour for an objective may not be optimal for another one. The question then moves on asking if an agent has a behaviour working for all its objectives while not disrupting the others.

Often the system asks for a conjunction of objectives: everyone must be able to satisfy its desires. Conjunctions are relatively easy to handle, see [4] for example where goals are made from a conjunction of a LTL and energy constraints. As we will see, the difficulty rises drastically when we move away from conjunctions to more complex boolean combinations of objectives. The way the different goals of a system limit each agent behaviour will be the heart of this thesis.

Outline

This thesis focus on a singular formalism adapted to multi-agents systems with multiple objectives: the Strategy Logic SL. We mainly work on two issues:

- algorithms for the model checking problem, i.e. algorithms that take as input a formula ϕ in SL and a multi-agent system $\mathcal{G}$, and find if ϕ holds on $\mathcal{G}$.
- the semantics of SL.

In the first chapter, we do a survey of the existing formalisms for multi-agent systems. At the end of the survey, we present SL and its main fragments (among which SL[BG]). We formally introduce the model checking problem in the second chapter. We also outline some complexity results surrounding SL, among which we find two of importance: the algorithm developed for SL model checking in [39] by Mogavero, Murano, Perelli and Vardi, and the lower bound we developed for SL[BG] model checking.

As we will see in details, strategies within SL are quantified as first-order variables before being assigned to the agents of the system. *Strategy translations* allow us to move strategies along a history ρ. Similarly, a *valuation translation* translate a set of strategies along ρ. SL semantics translate not only strategies assigned to some agents but also the strategies stored in the variables that have yet to be assigned to someone. In the third chapter, we study SL under a new notion of valuation translation where only strategies assigned to agents are shifted along ρ. We call these new translations *floating*, they give a new semantic FSL (Floating Strategy Logic) to SL. After introducing formally FSL, we highlight the differences in expressiveness with SL and study in details its model checking complexity.

The fourth chapter focuses on extending SL with quantitative constraints. There exist multiple kinds of constraints, we focus on two of them: the ones that can express equality and periodicity (we call them one-counter constraints), and the ones using upward-closed sets (energy constraints).

SL is a multi-objective logic and (as the name suggests) can handle multiple objectives at once. These objectives may spread along different paths corresponding to multiple executions of the system. The authors of [39] highlighted an interesting feature of SL: the choice of a strategy x_b on a given history ρ may depend on the choice of another strategy x_a made on another history π. We say there is a side dependency of x_b on x_a. They also wonder when and why these dependencies appear. The second part of this thesis (chapters five to seven) is devoted to this question, often referred as the *dependency problem*. Chapter five introduces the problem, maps the different dependencies and study the impacts of each dependency. Chapter six pursues this study by adding a new type of dependency to bypass the quantification order on prefixes of the current history. The seven (and last) chapter focuses on a promising fragment of SL.

Figure 4 highlights the different relations between the chapters. An arrow from chapter a to chapter b means that notions developed in chapter a are used in chapter b. In particular, the second part of the thesis (chapters five to seven) must be read linearly.

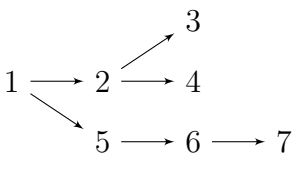


Figure 4: Reading order.

Part I

Strategy Logic

Chapter 1

State of the art

We present here a survey on different formalisms for multi-agent systems. This sets the context in which Strategy Logic (the subject of this thesis) emerged. We start with a brief introduction to temporal logics for closed systems: ω-regular automata, CTL, LTL, CTL* and L_μ. Next, we discuss the first formalisms for open systems, namely ω-regular winning conditions, ATL, ATL* and AL_μ. These logics suffer from a lack of expressiveness due to two severe drawbacks: forced commitment and forced discard. We explain each problem and detail some possible answers (GL, CATL, BSIL for the first and IATL, ATL_{sc}, QL_μ for the second). Finally, we define the Strategy Logic SL and position it in the constellation of all the formalisms for multi-agent systems.

1.1 Temporal logics for closed systems

1.1.1 ω-regular automata

We briefly discuss ω-regular automata. Though not a logic per say, ω-regular automata can also express temporal properties. There exists many kinds: Büchi, co-Büchi, parity, Rabin, Street, Muller... We however only present Büchi and parity automata, the ones used in this thesis.

Definition 1.1 (Automaton)**.**
An automaton is a tuple $\mathcal{A} := \langle S, \Sigma, \Delta, s_0 \rangle$ *where* S *is the state space,* Σ *the input alphabet,* $\Delta : S \times \Sigma \to 2^S$ *the transition function and* s_0 *the initial state.*

An automaton is complete and deterministic whenever $|\Delta(s, \xi)| = 1$ for any $s \in S$ and any $\xi \in \Sigma$. A word over Σ is a (finite or infinite) sequence $\mathfrak{w} := (w_i)_{i \leq L}$ where $L \in \mathbb{N} \cup \{\infty\}$ and $w_i \in \Sigma$. A path in an automaton $\mathcal{N}$ is a sequence $(s_i)_{i \in \mathbb{N}}$ of states. We say that a word $\mathfrak{w}$ over Σ is compatible with a path $\pi := (s_i)_{i \in \mathbb{N}}$ of an automaton $\mathcal{N}$ using Σ as its input alphabet whenever $s_{i+1} \in \Delta(s_i, w_i)$ for any $i < L$. For any path π in a automaton $\mathcal{N}$, we write $\mathsf{Inf}(\pi)$ for the set of states that appear an infinite number of times in π.

Büchi condition A Büchi automaton is a tuple $\mathcal{N} := \langle \mathcal{A}, \Omega \rangle$ made of an automaton $\mathcal{A} := \langle \mathsf{S}, \Sigma, \Delta, \mathsf{s}_0 \rangle$ and a set $\Omega \subset \mathsf{S}$ of states. A word $\mathfrak{w}$ over Σ is accepted by $\mathcal{N}$ whenever there exists a path π compatible with $\mathfrak{w}$ and

$$\mathsf{Inf}(\pi) \cap \Omega \neq \emptyset$$

Parity conditions A parity automaton $\mathcal{D}$ is also a tuple $\mathcal{N} := \langle \mathcal{A}, \Omega \rangle$ but here Ω is a mapping $\mathsf{S} \to \mathbb{N}$ assigning an integer to each state. A word $\mathfrak{w}$ over Σ is accepted by $\mathcal{D}$ whenever there exists a path π compatible with $\mathfrak{w}$ for which the maximal parity visited infinitely often is even, i.e.

$$\exists i \in 2\mathbb{N} \begin{cases} \exists s \in \mathsf{Inf}(\pi) \text{ with } \Omega(s) = i \\ \forall j > i,\ \forall s \in \mathsf{Inf}(\pi) \text{ we have } \Omega(s) \neq j \end{cases}$$

Both conditions will become useful on multiple occasions in this thesis, mainly because of Theorem 1.3 (page 20) linking LTL formulas to these automata.

1.1.2 Temporal logics on transition systems

Transition Systems

Transition systems are standard to represent the operational semantics of closed systems.

Definition 1.2 (Transition system)**.**
A transition system is a tuple $\mathcal{ST} := \langle AP, S, \hookrightarrow, \mathsf{labels}, \mathsf{s}_0 \rangle$ *where* AP *is a set of atomic propositions,* S *is a finite set of states,* $\hookrightarrow \subseteq S \times S$ *is a transition function,* $\mathsf{labels} : S \to 2^{AP}$ *is a labelling function and* $\mathsf{s}_0 \in S$ *the initial state.*

There exists many other models, transitions systems are however the standard one for checking temporal properties of potential executions. They also are sufficient for the three main closed-system temporal logics CTL, LTL and L_μ.

We model an execution of the closed system under consideration by a path in its associated transition system. A path in a transition system $\mathcal{ST} := \langle \mathsf{AP}, \mathsf{S}, \hookrightarrow, \mathsf{labels}, \mathsf{s}_0 \rangle$ is an infinite sequence $(s_i)_{i \in \mathbb{N}}$ where for any $i \in \mathbb{N}$ $s_i \in \mathsf{S}$ and $(s_i, s_{i+1}) \in \hookrightarrow$. We use the standard notations $\pi(i)$ to refer to the i^{th} element of a path π and for any $s \in \mathsf{S}$ we denote by Path_s the set of all paths in $\mathcal{ST}$ starting from s.

Branching time logic: CTL

The first formalism we introduce is the computation tree logic CTL. It was designed in 1982 by Clarke and Emerson [16] to synthetize synchronization skeletons, an abstraction of concurrent programs. Its formulas are built upon a set AP of atomic propositions by the following grammar:

$$\mathsf{CTL} \ni \phi \ ::= \ p \mid \phi \vee \phi \mid \neg\phi \mid \mathbf{EX}\,\phi \mid \mathbf{AX}\,\phi \mid \phi\,\mathbf{E}\,\mathbf{U}\,\phi \mid \phi\,\mathbf{A}\,\mathbf{U}\,\phi$$

CTL formulas are evaluated at a state s of a transition system $\mathcal{ST}$. Its satisfaction relation $\models$ is boolean, meaning that it either evaluates to true (that we write $\top$) if the formula holds on the transition system or false (written $\bot$) if it does not. We define it inductively and start with the boolean combinations:

$$\begin{array}{lcl} \mathcal{ST}, s \models \phi \vee \phi' & \Leftrightarrow & \mathcal{ST}, s \models \phi \text{ or } \mathcal{ST}, s \models \phi' \\ \mathcal{ST}, s \models \neg\phi & \Leftrightarrow & \mathcal{ST}, s \not\models \phi \end{array}$$

The $\vee$ and $\neg$ operators are the standard boolean operators from propositional logic with semantics adapted to CTL. We can retrieve the conjunction $\wedge$ with the standard translation $\phi \wedge \phi' = \neg(\neg\phi \vee \neg\phi')$. We can also define the universal true $\top$ within the logic by $\top := p \vee \neg p$ and the universal false $\bot$ by $\bot := \neg\top$. The other operators' semantics obey the following rules:

$$\begin{array}{lcl} \mathcal{ST}, s \models p & \Leftrightarrow & p \in \mathsf{labels}(s) \\ \mathcal{ST}, s \models \mathbf{EX}\,\phi & \Leftrightarrow & \exists s' \in \mathsf{S} \text{ with } (s,s') \in \hookrightarrow \text{ and } \mathcal{ST}, s' \models \phi \\ \mathcal{ST}, s \models \mathbf{AX}\,\phi & \Leftrightarrow & \forall s' \in \mathsf{S} \text{ s.t. } (s,s') \in \hookrightarrow \text{ it holds } \mathcal{ST}, s' \models \phi \\ \mathcal{ST}, s \models \phi'\,\mathbf{E}\,\mathbf{U}\,\phi & \Leftrightarrow & \exists \rho \in \mathsf{Path}_s\ \exists i \in \mathbb{N} \text{ with } \mathcal{ST}, \rho(i) \models \phi \\ & & \text{and } \forall j \leq i \text{ it holds } \mathcal{ST}, \rho(j) \models \phi' \\ \mathcal{ST}, s \models \phi'\,\mathbf{A}\,\mathbf{U}\,\phi & \Leftrightarrow & \forall \rho \in \mathsf{Path}_s\ \exists i \in \mathbb{N} \text{ with } \mathcal{ST}, \rho(i) \models \phi \\ & & \text{and } \forall j \leq i \text{ it holds } \mathcal{ST}, \rho(j) \models \phi' \end{array}$$

CTL is a branching-time logic, meaning it may express properties about multiple executions of the system. For example the formula $\mathbf{EX}\,p_1 \wedge (\top\,\mathbf{E}\,\mathbf{U}\,p_2)$ expresses the existence of a first execution that will lead in one step to a state labelled by p_1 and the existence of a second execution (potentially the same as the first one) that will eventually see a state labelled by p_2. The formula holds from s_0 on the transition system depicted on Figure 1.1.

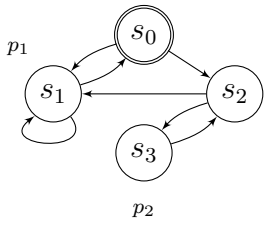


Figure 1.1: A transition system

Since any temporal modality requires a fresh quantification over paths, the logic is actually not very expressive. For example, we cannot express the existence of an execution that sees infinitely often two different atomic propositions [21]. To get more precise properties about the path within a transition system, we need another formalism.

Linear time logic: LTL

Introduced in 1977 by Pnueli [44], LTL (for linear-time temporal logic) has no branching capabilities and focuses on a single execution of the system. In compensation, it may express more complex path properties than CTL. Given a set AP of atomic propositions, its formulas obey the syntax

$$\mathsf{LTL} \ni \phi \ ::= \ p \mid \phi \vee \phi \mid \neg\phi \mid \mathbf{X}\,\phi \mid \phi\mathbf{U}\,\phi$$

Unlike CTL, LTL formulas are evaluated relatively to a path ρ and a position i of ρ. The semantics of the boolean operators are the same as in CTL, the semantics of the other operators are defined by:

$$\begin{array}{lcl} \mathcal{ST}, \rho, i \models p & \Leftrightarrow & p \in \mathsf{labels}(\rho(i)) \\ \mathcal{ST}, \rho, i \models \mathbf{X}\,\phi & \Leftrightarrow & \mathcal{ST}, \rho, i+1 \models \phi \\ \mathcal{ST}, \rho, i \models \phi' \mathbf{U}\,\phi & \Leftrightarrow & \exists i' \in \mathbb{N} \text{ with } \mathcal{ST}, \rho, i' \models \phi \\ & & \text{and } \forall j \leq i' \text{ it holds } \mathcal{ST}, \rho, j \models \phi' \end{array}$$

The **U** operator is referred to as "Until" and the **X** operator is called "Next". We will also refer to the two of them as temporal operators. From the **U** operator we can build three other temporal modalities: the "Future" operator **F**, the "Release" operator **R** and the "Global" operator **G**.

$$\mathbf{F}\,\phi := \top \mathbf{U}\,\phi \qquad \phi \mathbf{R}\,\phi' := \neg(\neg\phi \mathbf{U}\,\neg\phi') \qquad \mathbf{G}\,\phi := \neg\mathbf{F}\,\neg\phi$$

The potential nesting of temporal operators and boolean operators allows for more complex properties than the ones of CTL. For example, the formula $\mathbf{G}\,(\mathbf{F}\,p \wedge \mathbf{F}\,p')$ expresses that the path evaluated sees the atomic propositions p and p' an infinite number of times. This cannot be done in CTL [21]. There also exists a strong correlation between LTL formulas and both Büchi and parity automata[1].

Theorem 1.3 (Gerth, Peled, Vardi and Wolper [24] plus Piterman [43])**.**
For any LTL formula ϕ over a set AP of atomic propositions, there exists a Büchi automaton $\mathcal{N}_\phi$ and a deterministic parity automaton $\mathcal{D}_\phi$ over the alphabet 2^{AP} such that for any word $\mathfrak{w}$ over 2^{AP} the following are equivalent

- *$\mathfrak{w}$ (viewed as a path) satisfies ϕ*
- *$\mathfrak{w}$ is accepted by $\mathcal{N}_\phi$*
- *$\mathfrak{w}$ is accepted by $\mathcal{D}_\phi$*

Merging CTL and LTL: CTL*

We can merge LTL and CTL to get a logic capable of reasoning on multiple paths where each requirement is an LTL formula. CTL* combines the temporal operators of LTL with the paths quantifiers of CTL through the following grammar.

$$\begin{array}{rcl} \mathsf{CTL}^* \ni \phi & ::= & p \mid \phi \vee \phi \mid \neg\,\phi \mid \mathbf{E}\varphi \mid \mathbf{A}\varphi \\ \varphi & ::= & \varphi \vee \varphi \mid \neg\varphi \mid \mathbf{X}\,\varphi \mid \varphi \mathbf{U}\,\varphi \mid \phi \end{array}$$

[1] There exists many papers on the transition from LTL to Büchi automaton: [23], [49] and [60] for example.

We can see a separation within the grammar. The ϕ-type formulas derive from CTL. They are evaluated relatively to a state s of a transition system $\mathcal{ST}$ so we refer to them as *state formulas*. They obey the following semantics

$$\begin{aligned}
\mathcal{ST}, s \models p &\quad\Leftrightarrow\quad p \in \mathsf{labels}(s) \\
\mathcal{ST}, s \models \mathbf{E}\varphi &\quad\Leftrightarrow\quad \exists \rho \in \mathsf{Path}_s \text{ with } \mathcal{ST}, \rho, 0 \models \varphi \\
\mathcal{ST}, s \models \mathbf{A}\varphi &\quad\Leftrightarrow\quad \forall \rho \in \mathsf{Path}_s \text{ it holds } \mathcal{ST}, \rho, 0 \models \varphi
\end{aligned}$$

The φ-type formulas derive from LTL and are about a path property, we call them *path formulas*. They obey the following semantics:

$$\begin{aligned}
\mathcal{ST}, \rho, i \models \mathbf{X}\,\varphi &\quad\Leftrightarrow\quad \mathcal{ST}, \rho, i+1 \models \varphi \\
\mathcal{ST}, \rho, i \models \varphi' \mathbf{U}\,\varphi &\quad\Leftrightarrow\quad \exists i' \in \mathbb{N} \text{ with } \mathcal{ST}, \rho, i' \models \varphi \\
&\qquad\qquad \text{and } \forall j \leq i' \text{ it holds } \mathcal{ST}, \rho, j \models \varphi' \\
\mathcal{ST}, \rho, i \models \phi &\quad\Leftrightarrow\quad \mathcal{ST}, \rho(i) \models \phi
\end{aligned}$$

Consider the formula $\mathbf{A}(\mathbf{F}\,p_1) \wedge \mathbf{E}(\mathbf{G}\,(\mathbf{F}\,p_2))$, it translates to "All paths will eventually reach a state labelled by p_1 and there exists a path that sees p_2 an infinite number of times". We retrieve the branching aspects of CTL (with two properties about two distinct paths) and the expressiveness of LTL (with the imbrication of temporal operators).

Fix-point logics: L_μ

Other kinds of formalisms are possible. Among the most famous, we find fix-point logics and the μ-calculus. The μ-calculus (L_μ for short) extends propositional logic with "least" and "greatest" fix-point operators. L_μ formulas are build upon a set AP of atomic propositions and a set $\mathcal{V}$ of variables, they obey the following rules:

$$\mathsf{L}_\mu \ni \phi \;::=\; p \mid Z \mid \phi \vee \phi \mid \neg\phi \mid [a].\ \phi \mid \mu Z.\ \phi$$

with $p \in \mathsf{AP}$ and $Z \in \mathcal{V}$; considering a formula $\mu Z.\ \phi$, every occurrence of Z in ϕ must be positive (i.e. under an even number of negations).

The formulas in L_μ are evaluated relatively to transition systems enriched by actions, i.e. a transition system $\mathcal{ST} = \langle \mathsf{AP}, \mathsf{S}, \mathsf{Act}, \hookrightarrow, \mathsf{labels}, \mathsf{s}_0 \rangle$ with an additional set Act of new elements (the actions) and where the transition function $\hookrightarrow\colon \mathsf{S} \times \mathsf{Act} \to \mathsf{S}$ is deterministic and takes into account the actions. Unlike CTL, LTL or CTL^*, its satisfaction relation $[\![\cdot]\!]$ computes a function $[\![\cdot]\!] : \phi \to 2^{\mathsf{S}}$ returning a set of states instead of a truth value. It is defined inductively relatively to an interpretation $\chi : \mathcal{V} \to 2^{\mathsf{S}}$ of the variables in $\mathcal{V}$ by

$$\begin{aligned}
[\![p]\!]_\chi &:= \{s \in \mathsf{S} \mid p \in \mathsf{labels}(s)\} \\
[\![Z]\!]_\chi &:= \{s \in \mathsf{S} \mid s \in \chi(Z)\} \\
[\![\phi \vee \phi']\!]_\chi &:= [\![\phi]\!]_\chi \cup [\![\phi']\!]_\chi \\
[\![\neg\phi]\!]_\chi &:= \mathsf{S} \backslash [\![\phi]\!]_\chi \\
[\![[a].\ \phi]\!]_\chi &:= \{s \in \mathsf{S} \mid \forall s' \in \mathsf{S},\ \text{if } (s,a,s') \in \hookrightarrow \text{ then } s' \in [\![\phi]\!]_\chi\} \\
[\![\mu Z.\ \phi]\!]_\chi &:= \bigcap \{T \subseteq \mathsf{S} \mid [\![\phi]\!]_{\chi[Z/T]} \subseteq T\}
\end{aligned}$$

with $[Z/T]$ meaning that we substitute every Z by T.

The operator $\mu Z.\ \phi$ is the least fix-point. Likewise to the other temporal logics we can retrieve the $\wedge$ operator using $\vee$ and $\neg$. We can also get two other modalities $\nu Z.\phi$ and $\langle a\rangle\phi$ respectively dual of $\mu Z.\ \phi$ (making it the greatest fix-point) and $[a].\ \phi$.

$$\langle a\rangle\phi := \neg[a].\ \neg\phi \qquad\qquad \nu Z.\ \phi := \neg\mu Z.\ \phi[Z/\neg Z]$$

1.2 Temporal logics for open systems

As we move to open systems, we need to update both the models and the formalisms.

1.2.1 Multi-agents models

Concurrent and turn-based game structures

Concurrent[2] and turn-based games are the usual models used for open and multi-agent systems.

Definition 1.4 (Concurrent game structure)**.**
A concurrent game structure (CGS for short) is a tuple $\mathcal{G} := \langle AP, Agt, Q, Act, \Delta, labels\rangle$ *where* AP *is a non-empty and finite set of atomic proposition,* Agt *is a non-empty and finite set of agents,* Q *is a non-empty and finite set of states,* Act *is a non-empty set of actions,* $\Delta : Q \times Act^{Agt} \to Q$ *is a transition function and* $labels : Q \to 2^{AP}$ *is a labelling function.*

Intuitively, the state space Q of a CGS represents the different status of the system and the agents represent different parts of the system. The actions then express the possible interactions between the agents. Finally the labelling adds some atomic informations, in particular the bits needed to specify the good behaviour of the system.

Remark 1.5. *In some works an initial state is added to the definition of CGS. In this thesis, we deal with multiple objectives at the same time. Each objective may be evaluated at a different state from the others, hence the notion of initial state is not particularly relevant here and would only extend notations. For theses reasons, we omit it in our definition of CGS.*

Remark 1.6. *In the definition of a CGS, the action set is common to all agents. However, when building a CGS, we will need on multiple occasions to assign a given action* a_1 *to a given agent* A_1*. This can be done by giving a fixed behaviour for the interactions of* a_1 *with the other agents. For example, we can build our game such that (within the transition function) all the agents but* A_1 *interact with* a_1 *as they would with another action* a_2*. This way, an agent different from* A_1 *has no use for* a_1*, he can always play* a_2 *equivalently. We then refer to* a_1 *as an action exclusive to agent* A_1*.*

[2]The notion of concurrency in computer science usually refers to the ability to decompose a system into components whose executions can be done in any order: see for example the concurrency related problems in Petri nets [19]. While concurrent games allow a certain form of concurrency, they are more often used to model some notion of simultaneity.

A second model called turn-based game structures, can be seen as a restriction of CGS, where each state is controlled by a single agent. Turn-based game structures are simpler and more intuitive than their concurrent counter-parts but may sometimes lack expressiveness. For example they cannot convey any notion of simultaneity concerning the choices of the agents.

Definition 1.7 (Turn-based game structure)**.**
A turn-based game structure is a CGS $\mathcal{G} := \langle AP, Agt, Q, Act, \Delta, labels\rangle$ *where for each state* $q \in Q$ *there exists an agent* A *such that*

$$\forall m, m' \in Act^{Agt\setminus\{A\}}\ \forall a \in Act \qquad \Delta(q, m \cup \{A \to a\}) = \Delta(q, m' \cup \{A \to a\})$$

The agent A *is then called the owner of* q*.*

Histories, paths and strategies

To simulate an execution of the system, a pebble is placed on the state representing the initial status of the system. At time t each agent plays an action, the resulting set of actions, sometimes referred as a decision, combined with the pebble's current state determine a new state for the pebble through the transition function. The sequence of states visited by the pebble as time passes determines the evolution of the system.

Definition 1.8 (History and paths)**.**
A history in a game $\mathcal{G} := \langle AP, Agt, Q, Act, \Delta, labels\rangle$ *is a finite sequence* $(q_i)_{i\leq L}$ *(*$L \in \mathbb{N}$*) of states such that for any* $i < L$ *there exists* $d \in Act^{Agt}$ *such that* $\Delta(q_i, d) = q_{i+1}$*.*

A path in $\mathcal{G}$ *is an infinite sequence* $(q_i)_{i\in\mathbb{N}}$ *of states such that for any* $i \in \mathbb{N}$ *there exists* $d \in Act^{Agt}$ *such that* $\Delta(q_i, d) = q_{i+1}$*.*

Note that a history does not include the actions played. A path represents an execution of the system (running forever) while a history can be seen as the beginning of an execution that has yet to be completed. We denote by $\mathsf{Hist}_{\mathcal{G}}$ and $\mathsf{Path}_{\mathcal{G}}$ respectively the sets of histories and paths of $\mathcal{G}$. For any history ρ, we also write $\mathsf{lst}(\rho)$ for the last state of ρ. Finally, given either a history or a path $\rho := (q_i)_{i<L}$ (with $L \in \mathbb{N} \cup \{\infty\}$) and an integer $j < L$, we write $\rho_{\leq j}$ for the finite prefix $(q_i)_{i\leq j}$ of ρ and $\rho(j)$ for the element q_j. As is customary, the singular and generalised concatenation of histories will respectively be denoted by the symbols $\rho \cdot \rho'$ and $\Pi_{i\in I}(\rho_i)$.

We use the notion of strategy to model the different behaviour of the agents. In this thesis, strategies will be with infinite memory and deterministic, meaning no memory restriction but no randomness.

Definition 1.9 (Strategy)**.**
A strategy in a game $\mathcal{G} := \langle AP, Agt, Q, Act, \Delta, labels\rangle$ *is a (potentially partial) function* $\delta : Hist_{\mathcal{G}} \to Act$ *taking a history and returning an action.*

We write $\mathsf{Strat}_{\mathcal{G}}$ for the set of strategies on $\mathcal{G}$. A strategy is often reliant on a small subset of all the informations carried by the history in input. For example it may look

only at the last state of the history, the reactive synthesis community (another name for people working with systems reacting to their environment) call such strategies *positional* as they only depend on the position of the pebble. For reason that will become obvious in Chapter 3, strategies are defined on all histories, independently of any starting state.

We refer as a context the assignment of a strategy to each agent. Formally, a context is a function $\chi : \mathsf{Agt} \to \mathsf{Strat}_{\mathcal{G}}$. We can then define the outcome of a game.

Definition 1.10 (Outcome)**.**
Let $\mathcal{G} := \langle \mathsf{AP}, \mathsf{Agt}, \mathsf{Q}, \mathsf{Act}, \Delta, \mathsf{labels} \rangle$ be a game, q one of its states and χ a context. The outcome $\mathsf{out}(\chi, q) := (q_i)_{i \in \mathbb{N}^*}$ of χ from q is the unique path where $q_1 = q$ and $\forall i \in \mathbb{N}^*$, $q_{i+1} = \Delta(q_i, d_i)$ where $d_i \in \mathsf{Act}^{\mathsf{Agt}}$ is the function assigning an action to each agent by following χ on history $(q_{i'})_{i'<i}$.

Example

In 2005, the three french mobile operators *Orange*, *SFR* and *Bouygues-telecom* were found guilty of fixed-price fraud and respectively fined for 220, 256 and 58 millions euros. The fixed price fraud (FPF for short) consists in multiple companies agreeing to freeze their products' prices, making the concurrency (in the economic sense) disappear. In most countries, including France, this practice is illegal. We present some of the notions developed above through a naive model for the FPF applied to two of the three operators (for simplicity), for example Orange and SFR.

We model the FPF through the following CGS $\mathcal{G} := \langle \mathsf{AP}, \mathsf{Agt}, \mathsf{Q}, \mathsf{Act}, \Delta, \mathsf{labels} \rangle$ where

- The agents are the two operators Orange and SFR.
- An operator can either push a new offer on the market, represented by the *ccr* action, or stay idle, represented by the frz action. The set of actions then consists of $\mathsf{Act} := \{ccr,\ frz\}$.
- The atomic propositions represent each company policy regarding any reduction of its prices. The set AP is made of four propositions p^O_{frz}, p^S_{frz}, p^O_{ccr} and p^S_{ccr}. The proposition p^A_B represents the fact that the company A has adopted the policy B.
- The state space Q is composed of only four states, each representing a potential status of our two operators mobile market. Formally, we set $\mathsf{Q} := \{q_{\star,*} \mid \star, * \in \{frz, ccr\}\}$. The state $q_{frz,ccr}$ for example tell us that Orange is not trying to lower its prices while SFR is. Such situation is preferable for SFR and undesirable for Orange, as it will lose clients in benefits to its concurrent. Obviously, many parameters come into play before engaging new offers in the mobile market; but for all purposes in this example, we assume that at any time either of the companies may, if it wishes, lower its price.
- A state $q_{\star,*}$ (with $\star, * \in \{frz, ccr\}$) is labelled by $\{p^O_\star p^S_*\}$.

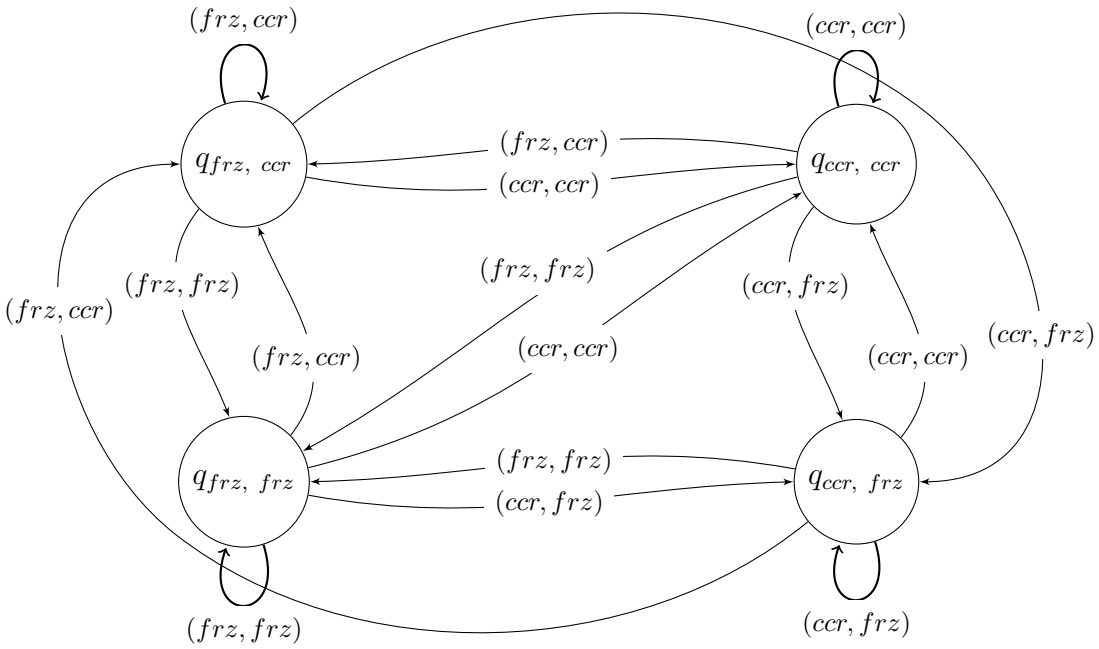
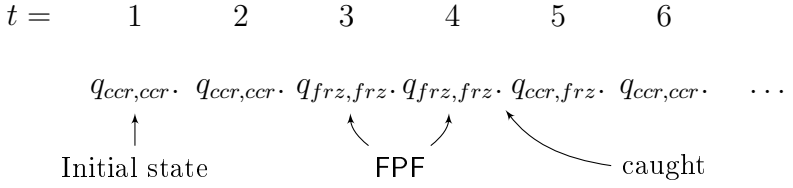


Figure 1.2: The game $\mathcal{G}$ representing the mobile operator market

- The transition function Δ is rather intuitive, it simply updates any change of mind of the operators. As often, the formal definition is tedious, we refrain from giving it and refer to Figure 1.2.

Our initial state (at time $t = 1$) is $q_{ccr,ccr}$, as we assume both companies to be in concurrence at the beginning of the simulation. Consider the following scenario: at time $t = 2$ through a shady meeting, representatives of both companies gather and agree to freeze their prices from the time $t = 3$. At time $t = 4$, regulatory authorities catch the subterfuge. Orange, fearful of heavy sanctions immediately issues new offers with lower prices while SFR waits until the end of $t = 5$ to do so. The whole situation can be represented by the path:

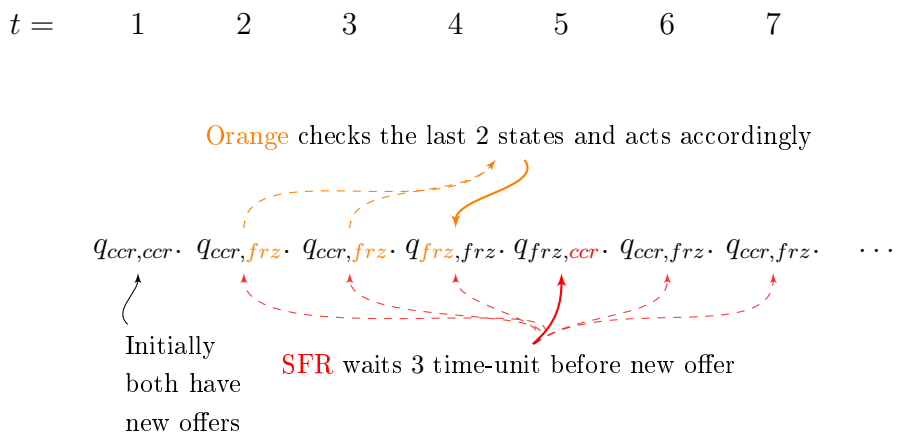


Instead of a shady meeting, Orange can build a strategy δ^O trying to obtain a tacit form of fixed-price market. Take the following behaviour: if SFR has not provided any new offer for the last two time-units then Orange postpones any plan reducing its prices; if SFR has put a new offer to attract customers in the last two time-units then Orange pushes forward a concurrent answer. This can be modeled by the following δ^O strategy: on a history $\rho := (q_i)_{i \leq L}$ of length L,

- if either $q(L-1)$ or $q(L)$ has the proposition p^S_{ccr} it means that SFR has put a low price offer and Orange must do the same so $\delta^O(\rho) = ccr$.

- if neither $q(L-1)$ nor $q(L)$ has p^S_{ccr}, Orange will try to create a tacit frozen market and postpone its new deals, so $\delta^O(\rho) = frz$.

Now, consider a second strategy δ^S for SFR that puts a new offer every 4 time-unit. The context where Orange plays along δ^O and SFR plays along δ^S will produce the following infinite outcome



1.2.2 Formalisms

ω-regular conditions

We briefly discuss how the ω-regular conditions described before can be adapted for CGS. Consider a CGS $\mathcal{G} := \langle \mathsf{AP}, \mathsf{Agt}, \mathsf{Q}, \mathsf{Act}, \Delta, \mathsf{labels} \rangle$. Likewise to words, for any history $\rho \in \mathsf{Hist}_\mathcal{G}$ we write $\mathsf{Inf}(\rho)$ for the set of states that appear an infinite number of time in ρ.

A Büchi condition over $\mathcal{G}$ is a set $\Omega \subseteq S$ of states. We then say that a coalition $C \subseteq \mathsf{Agt}$ satisfies the condition Ω from a state q_{ini} in $\mathcal{G}$ when there exist strategies for each member of C such that no matter the other agents' strategies, the resulting outcome ρ starting from q_{ini} satisfies

$$\mathsf{Inf}(\rho) \cap \Omega \neq \emptyset$$

A parity condition over a CGS $\mathcal{G}$ is a mapping $\Omega : \mathsf{Q} \to \mathbb{N}$ assigning an integer to each state. A coalition $C \subseteq \mathsf{Agt}$ can satisfy the condition Ω from a state q_{ini} when there exists strategies for each member of C such that no matter the other agents' strategies, the resulting outcome ρ starting from q_{ini} satisfies

$$\exists i \in 2\mathbb{N}. \begin{cases} \exists q \in \mathsf{Inf}(\rho) \text{ with } \Omega(q) = i \\ \forall q \in \mathsf{Inf}(\rho) \text{ we have } \Omega(q) \geq i \end{cases}$$

The alternating time temporal logics ATL and ATL*

There are three basic logics for open systems: ATL, ATL* and AL_μ (also called AMC in [2], we however rebrand it in AL_μ to pair with the symbol L_μ for the μ-calculus). We start with the easiest ones ATL and ATL*. Proposed by Alur, Henzinger and Kupferman in 2002 [2] they can be seen for all purposes as the adaptations of CTL and CTL* to open systems. ATL* formulas are build upon a set AP of atomic propositions and a set Agt of agents through the grammar

$$\begin{aligned} \mathsf{ATL}^* \ni \phi &::= p \mid \phi \vee \phi \mid \neg\phi \mid \langle\!\langle C \rangle\!\rangle \varphi \qquad \text{with } C \subseteq \mathsf{Agt} \\ \varphi &::= \varphi \vee \varphi \mid \neg\varphi \mid \mathbf{X}\,\varphi \mid \varphi \mathbf{U}\, \varphi \mid \phi \end{aligned}$$

We retrieve the next ($\mathbf{X}$) and until ($\mathbf{U}$) operators from LTL and the decomposition in state and path formulas of CTL*. Its semantics are defined similarly except for the quantifiers $\langle\!\langle C \rangle\!\rangle \varphi$. To ease the reading, and because ATL* is not at the heart of this thesis, we only give an informal definition of the quantifiers semantics.

$$\mathcal{ST}, s \models \langle\!\langle C \rangle\!\rangle \varphi \quad \Leftrightarrow \quad \begin{cases} \text{There exists a set of strategies for the agents of } C \text{ so} \\ \text{that, given any set of strategies for the agents of} \\ \mathsf{Agt} \backslash C, \text{ the resulting outcome } \rho \text{ satisfies } \mathcal{ST}, \rho \models \varphi \end{cases}$$

The logic ATL is the fragment of ATL* obtained by restricting the grammar of path formulas to $\varphi ::= \neg\phi \mid \mathbf{X}\,\phi \mid \phi \mathbf{U}\, \phi$. for example, the formula $\langle\!\langle \{A_1, A_2\} \rangle\!\rangle \mathbf{F}\, p$ is in ATL and expresses the existence of a strategy for the coalition of A_1 and A_2 that produces (no matter the other agents strategies) an outcome that will eventually see p.

AL_μ an alternating version of the μ-calculus

Alur, Henzinger and Kupferman also proposed in [2] an extension of the μ-calculus L_μ to work on CGS. They call it the alternating μ-calculus AL_μ. Also a fix-point logic, it follows the syntax of L_μ. Defined relatively to a set AP of atomic proposition, a set Agt of agents and a set $\mathcal{V}$ of variables, its syntax is dictacted by the rules below.

$$\mathsf{AL}_\mu \ni \phi ::= p \mid Z \mid \phi \vee \phi \mid \neg\phi \mid \langle\!\langle C \rangle\!\rangle \phi \mid \mu Z.\ \phi$$

The satisfaction relation $[\![\cdot]\!]$ for AL_μ formulas is still defined relatively to an interpretation χ of the variables and as a function $[\![\cdot]\!] : \phi \to 2^{\mathsf{S}}$. All operators but $\langle\!\langle C \rangle\!\rangle \phi$ have the same semantics as in L_μ. The new operator semantics are defined by

$$[\![\langle\!\langle C \rangle\!\rangle \phi]\!]_\chi := \{ s \in \mathsf{S} \mid \begin{cases} \text{there exists an action } a_A \text{ for each agent } A \in C \\ \text{ such that against any other actions } a_B \text{ of the} \\ \text{agents } B \in \mathsf{Agt} \backslash C, \text{ the state} \\ \Delta(q, (A_1 \to a_{A_1}, \dots, B_1 \to a_{B_1}, \dots)) \\ \text{ belongs in } [\![\phi]\!]_\chi \end{cases} \}$$

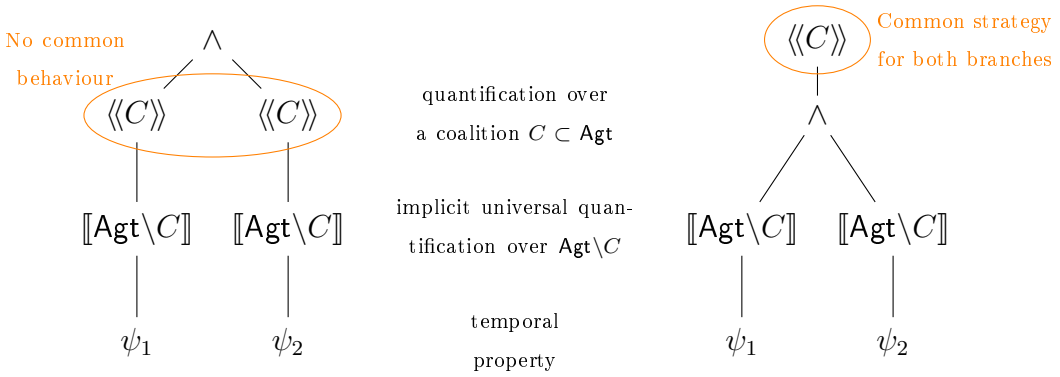


Figure 1.3: On the right an ATL formula and on the left what cannot be done neither in ω-regular conditions nor in any ATL, ATL* or AL$_\mu$.

1.3 Commitment issues in multi-agents logics

In the formalisms for closed systems, to simulate potential executions of the system we use quantifications over the potential paths in the model. These path quantifications, either implicit as in LTL or explicit as in CTL, are of two forms: "is there a path such that..." and "for all paths it holds...". In the formalism for open systems, the path quantifications are replaced by strategy quantifications ("is there a strategy for coalition C such that..."). In all the open-system formalisms defined above, after each strategy quantification there are other implicit strategy quantifications making all properties expressible in the following form

> Are there strategies (actions for AL$_\mu$) for agents $A_1, \ldots$ such that for all strategies (actions for AL$_\mu$) for agents $B_1, \ldots$ the resulting outcome satisfies some temporal property ?

This makes the branching possibilities of ATL, ATL* and AL$_\mu$ very similar to the ones of CTL, CTL* and L$_\mu$: a boolean combination of formulas, each about a given outcome. The different branches have nothing in common: an agent may act in a way in one branch and completely change its behaviour in another. This inability for different branches of a formula to share some common behaviour is what we call the commitment issue, each strategy obtained through quantification is committed to a single temporal objective. Figure 1.3 illustrates the issue: on the left we can see the possibilities of ATL and on the right a property not expressible with what we have presented so far. We present below some new formalisms to address this issue.

GL

A first way to bypass the commitment issue is to add branching between the existential and (implicit) universal quantifications, like in the right part of Figure 1.3. This was the approach chosen by Alur, Henzinger and Kupferman in [2] for their game logic GL. GL formulas are build by a three steps syntax: state formulas ϕ, tree formulas ψ and path

formulas φ.

$$\begin{array}{rcll} \mathsf{GL} \ni \phi & ::= & p \mid \neg\phi \mid \phi \vee \phi \mid \langle C\rangle\psi & \text{with } C \subseteq \mathsf{Agt} \\ \psi & ::= & \neg\psi \mid \psi \vee \psi \mid \mathbf{E}\varphi & \\ \varphi & ::= & \phi \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi & \end{array}$$

The semantics of the atomic propositions and boolean operators is similar to the ones of all the logics we have already presented. We recall that a context is a partial function associating strategies with agents. The $\langle C\rangle\psi$ operator[3] is an existential quantification for the agents in the coalition C that sets a context for tree formulas. To define explicitly a context χ in which an agent A follows a strategy δ while another agent B follows δ', we use the notation $\chi := \{A \to \delta; B \to \delta'\}$. Then, the operator $\langle C\rangle\psi$ has the following semantics:

$$\mathcal{G}, q \models \langle C\rangle\psi \quad \Leftrightarrow \quad \begin{cases} \exists \delta_1 \dots \delta_{|C|} \in \mathsf{Strat}_{\mathcal{G}} \text{ with } \mathcal{G}, q \models_{\{A_1 \to \delta_1, \dots, A_{|C|} \to \delta_{|C|}\}} \psi \\ \text{where } A_1, \dots A_{|C|} \text{ are the agents of } C \end{cases}$$

Tree formulas are defined relatively to the unwinding of the game. An *unwinding* of a game $\mathcal{G}$ from a state q is the infinite tree $\mathcal{T}_{\mathcal{G}}$ obtained with q at the root and where the children of a node are its successors. The $\mathbf{E}\varphi$ operator is an existential quantification over the paths in $\mathcal{T}_{\mathcal{G}}$.

$$\mathcal{G}, q \models \mathbf{E}\varphi \quad \Leftrightarrow \quad \exists \rho \in \mathsf{Path}_{\mathcal{T}_{\mathcal{G}}} \text{ coherent with } \chi \text{ with } \mathcal{G}, \rho \models \varphi$$

Path formulas' semantics are defined relatively to a path in the same way as in LTL.

BSIL

Another approach to the commitment problem, proposed by Wang, Huang and Yu in 2011 with their basic strategy interaction logic BSIL [63], is to add an operator $\langle +C\rangle$ to extend the current context. BSIL also decomposes its grammar in three parts: state formulas ϕ, branching formulas ψ and path formulas φ.

$$\begin{array}{rcll} \mathsf{BSIL} \ni \phi & ::= & p \mid \neg\phi \mid \phi \vee \phi \mid \langle C\rangle\psi & \text{with } C \subseteq \mathsf{Agt} \\ \psi & ::= & \neg\psi \mid \psi \vee \psi \mid \langle +C\rangle\psi \mid \varphi & \\ \varphi & ::= & \mathbf{X}\,\phi \mid \phi\mathbf{U}\,\phi & \end{array}$$

As in GL, state formulas define an initial context; however branching formulas are used to branch properties upon the current context or to refine it.

$$\begin{array}{lcl} \mathcal{G}, q \models \langle C\rangle\psi & \Leftrightarrow & \exists \delta_1 \dots \delta_{|C|} \in \mathsf{Strat}_{\mathcal{G}} \text{ with } \mathcal{G}, q \models_{\{A_1 \to \delta_1, \dots, A_{|C|} \to \delta_{|C|}\}} \psi \\ \mathcal{G}, q \models_{\chi} \langle +C\rangle\psi & \Leftrightarrow & \exists \delta_1 \dots \delta_{|C|} \in \mathsf{Strat}_{\mathcal{G}} \text{ with } \mathcal{G}, q \models_{\chi[A_1 \to \delta_1, \dots, A_{|C|} \to \delta_{|C|}]} \psi \\ \mathcal{G}, q \models_{\chi} \varphi & \Leftrightarrow & \forall \rho \in \mathsf{Path}_{\mathcal{G}} \text{ coherent with } \chi, \text{ it holds } \mathcal{G}, \rho \models \varphi \end{array}$$

[3] The symbol $\langle\langle C\rangle\rangle\psi$ is for the existential quantification of the agents in C with an additional implicit universal quantification on $\mathsf{Agt}\backslash C$. When we wish to avoid the implicit universal quantification, as in GL, we use the symbol $\langle C\rangle\psi$.

The formula on the right of Figure 1.3 can be expressed in BSIL by

$$\langle C\rangle \begin{cases} \neg\langle +\mathsf{Agt}\backslash C\rangle\neg\psi_1 \\ \wedge \\ \neg\langle +\mathsf{Agt}\backslash C\rangle\neg\psi_2 \end{cases}$$

CATL, a version of ATL with effective commitment

A different approach to the commitment problem is to treat the strategies in an explicit manner rather than through the scope of quantifications. For this, the logic CATL, developed by van der Hoek, Jamroga and Wooldridge in 2005 [59] extends ATL by adding an operator $\mathfrak{C}(\delta, A)\phi$ to assign a strategy δ to the agent A. The strategy δ is not the results of a quantification but rather has to be explicitly given by the context.

$$\mathsf{CATL} \ni \phi \ ::= \begin{cases} p \mid \phi \vee \phi \mid \neg\phi \mid \langle +C\rangle \mathbf{G}\,\phi \mid \langle +C\rangle \mathbf{X}\,\phi \mid \langle +C\rangle \mathbf{F}\,\phi \\ \langle +C\rangle \phi \mathbf{U}\,\phi \mid \langle +C\rangle \phi \mathbf{R}\,\phi \mid \mathfrak{C}(\delta, A)\phi \qquad\qquad A \in \mathsf{Agt} \end{cases}$$

At the beginning the context is empty. The $\langle +C\rangle\phi\heartsuit\phi$ and $\langle +C\rangle\diamondsuit\phi$ type formulas (for $\heartsuit \in \{\mathbf{U}, \mathbf{R}\}$ and $\diamondsuit \in \{\mathbf{X}, \mathbf{F}\,\mathbf{G}\}$) act as a concatenation of the $\langle +C\rangle$ operator from BSIL extending the current context and of LTL temporal operators ($\mathbf{F}$,$\mathbf{U}$,$\mathbf{G}$,$\mathbf{X}$ and $\mathbf{R}$). The new operator $\mathfrak{C}(\delta, A)\phi$ obeys the following semantics

$$\mathcal{G}, q \models_\chi \mathfrak{C}(\delta, A)\phi \quad \Leftrightarrow \quad \mathcal{G}, q \models_{\chi\cup\{A\to\delta\}} \psi$$

Note that unlike most logics, CATL formulas can be evaluated relatively to some strategies given as parameters. CATL algorithms then work under assumptions such as "the strategies used are positional" or "the strategies used have a memory of size λ".

1.4 Revocation issues in multi-agents logics

We have seen through the commitment issue that when working with open systems, it is useful to branch formulas in a more refined way than with closed systems. In particular we may force some shared behaviour between different simulations. A somewhat similar problem occurs linearly.

To illustrate the problem, we extend an example proposed by Agotnes, Goranko and Jamroga in [1]. Consider the CGS of Figure 1.4 with two agents A and B and an action set $\{0, 1, 2\}$ common to both agents. We wish to express the following property

> Agent A has a behaviour that always sees p, no matter what the choices of B are, but may at every moment decide to go to p'.

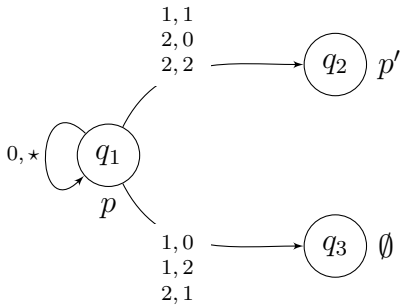


Figure 1.4: A CGS with two agents A and B, with three actions 0, 1 and 2, and where q_1 serves as initial state.

The intuitive way to express this in ATL is by the following formula:

$$\phi := \langle\langle A \rangle\rangle \ \mathbf{G}\,(p \wedge \langle\langle A \rangle\rangle \mathbf{F}\, p')$$

The property is however ambiguous: should we requantify the strategy of B when A starts to deviate towards p'? Formula ϕ works only if B is supposed to requantify its strategy, in this case there is no solution and ϕ does not hold on the CGS of Figure 1.4. If we wish to preserve the strategy B has chosen, the property should hold on the game. Indeed, the strategy of agent B is fixed and is included in the current context, therefore when A requantifies its strategy it may assume knowledge of B strategy. However ϕ does not hold on the game and therefore fails to capture the second possibility. We need a more expressive formalism than ATL. In fact none of the logics we have presented so far (ATL, ATL*, BSIL, CATL, AL$_\mu$) can manage this problem. There exist some solutions but they are all dependent on the game (which we purposely kept simple).

The problem in the example above comes from the incapability for an agent (A) to revoke its strategy without forcing other agents (B) to do the same. The incapability to requantify the strategies for some of the agents while keeping the current strategies for the other agents is what we call the revocation problem and what we investigate in this section.

ATL with irrevocable strategies

One solution to the revocation issues is to force every agent existentially quantified to keep its strategy when a nested quantification occurs. This is what Agotnes, Goranko and Jamroga did in 2007 [1] with their irrevocable version IATL of ATL. The grammar of IATL is the same as ATL:

$$\begin{aligned} \mathsf{IATL} \ni \phi \ &::= \ p \mid \phi \vee \phi \mid \neg\phi \mid \langle\langle C \rangle\rangle \varphi \\ \varphi \ &::= \ \mathbf{X}\,\phi \mid \phi \mathbf{U}\, \phi \end{aligned}$$

Formulas in IATL are evaluated relatively to a context χ initially empty (before the first quantifier) and that grows with each quantifier. Compared to ATL, only the semantics of the $\langle\langle C \rangle\rangle \varphi$ operator is modified.

$$\mathcal{G}, q \models_\chi \langle\langle C \rangle\rangle \varphi \quad \Leftrightarrow \quad \begin{cases} \exists \delta_1, \dots \delta_\lambda \in \mathsf{Strat}_\mathcal{G} \text{ such that for any strategies for the} \\ \text{agents of } \mathsf{Agt} \backslash C \text{ the resulting context with } \chi \text{ satisfies} \\ \mathcal{G}, q \models_{\chi \cup \{A_1 \to \delta_1, \dots A_\lambda \to \delta_\lambda\}} \varphi. \end{cases}$$

where $A_1, \dots A_\lambda$ are the agents of C that have yet to be assigned a strategy in χ (meaning $A_i \notin \mathsf{dom}(\chi)$). The context χ can only be refined by a quantifier.

ATL with strategy context

This approach was refined by Brihaye, Da Costa, Laroussinie and Markey in 2009 [10] where the notion of context was pushed further by explicitly adding a operator $\rangle\rangle C \langle\langle \phi$ to

revoke the strategies assigned to the agents of a coalition C. Formally, they define a logic ATL_{sc} based upon the following grammar:

$$\begin{array}{rcl} \mathsf{ATL}_{sc} \ni \phi & ::= & p \mid \phi \vee \phi \mid \neg\phi \mid \rangle\!\rangle C \langle\!\langle \phi \mid \langle\!\langle C \rangle\!\rangle \varphi \qquad \text{with } C \subseteq \mathsf{Agt} \\ \varphi & ::= & \phi \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathbf{X}\,\varphi \mid \varphi \mathbf{U}\, \varphi \end{array}$$

The semantics are somewhat similar to IATL with the exception that (like ATL and ATL*) a quantification over a coalition C relatively to a context χ also requantifies the strategies of $\mathsf{dom}(\chi) \cap C$:

$$\mathcal{G}, q \models_\chi \langle\!\langle C \rangle\!\rangle \varphi \quad \Leftrightarrow \quad \begin{cases} \exists \delta_1, \ldots \delta_{|C|} \in \mathsf{Strat}_\mathcal{G} \text{ such that for any strategy for the} \\ \text{agents of } \mathsf{Agt} \backslash (C \cup \mathsf{dom}(\chi)) \text{the resulting context with} \\ \chi \text{ satisfies } \mathcal{G}, q \models_{\chi_{|\mathsf{Agt}\backslash C} \cup \{A_1 \to \delta_1, \ldots A_{|C|} \to \delta_{|C|}\}} \varphi. \end{cases}$$

Note that the strategies from implicit universal quantifications do not override the strategies in χ. The semantics of the new operator $\rangle\!\rangle C \langle\!\langle \phi$ release the strategies assigned to C:

$$\mathcal{G}, q \models_\chi \rangle\!\rangle C \langle\!\langle \phi \quad \Leftrightarrow \quad \mathcal{G}, q \models_{\chi_{|\mathsf{Agt}\backslash C}} \phi$$

The quantified μ-calculus QL_μ

In a completely different way, the quantified μ-calculus QL_μ also solves the revocation problem. Developed in 2003 [48], Riedweg and Pinchinat idea was to add quantifications over atomic propositions to the μ-calculus. The concept of quantification over atomic propositions was initially proposed in 1983 [53] by Sistla for LTL (see also [54]). The grammar of QL_μ follows the one of L_μ with the addition of a quantification block (with boolean operators) before the L_μ formula.

$$\begin{array}{rcl} \mathsf{QL}_\mu \ni \phi & ::= & \exists p \phi \mid \neg\phi \mid \phi \vee \phi \mid \varphi \\ \mathsf{L}_\mu \ni \varphi & ::= & p \mid Z \mid \varphi \vee \varphi \mid \neg\varphi \mid [a].\ \varphi \mid \nu Z.\ \varphi \end{array}$$

QL_μ (like L_μ) formulas are interpreted on transitions systems relatively to an interpretation χ of the variables. The ϕ layer semantics are such that

$$\mathcal{TS}, s \models_\chi \exists p.\ \phi \quad \Leftrightarrow \quad \begin{cases} \text{There exists a transition system } \mathcal{TS}_p \text{ such that} \\ \mathcal{TS}_p, s \models_\chi \phi \text{ where } \mathcal{TS}_p \text{ is a transition system with} \\ \text{the same state space and transitions functions, and where} \\ p' \in \mathsf{labels}_{\mathcal{TS}}(s) \Leftrightarrow p' \in \mathsf{labels}_{\mathcal{TS}_p}(s) \text{ for any } p' \in \mathsf{AP} \backslash \{p\} \end{cases}$$

While defined over transition systems, the quantified μ-calculus can model CGS through the use of proposition quantifications. QL_μ is also a superset of QCTL* (CTL* extended with quantifications over the atomic propositions) and QCTL* has been shown in [31] to have an expressive power similar to ATL_{sc}. So, by transitivity, QL_μ is more expressive than ATL_{sc}. As said before, ATL_{sc} is an answer to both the commitment and the revocation problems so QL_μ has the same answer to these problems.

1.5 Strategy Logic

Strategy Logic (SL for short) is another formalism for expressing temporal properties on multi-agent systems. While offering another way to solve the commitment and revocation problems, SL innovates by treating the strategies as first-order variables (in the spirit of the first-order logic FO [33]).

A first version of SL was proposed in [15] by Chatterjee, Henzinger and Piterman. In this thesis, we call it CHP-SL. CHP-SL is an answer to the commitment problem similar to BSIL, with the additional possibility to nest temporal operators like LTL formulas instead of a single operator. CHP-SL was then extended by Mogavero, Murano, Perelli and Vardi [39]. We call Strategy Logic (SL) this second version. For most purposes CHP-SL is (in spirit but not formally) another form of BSIL[4]. For this reason we only give the syntax and semantics of the enhanced version.

1.5.1 Strategy translations, valuations and valuations translations

Before formally defining SL we need some new notions to manipulate strategies. As explained in Section 1.2.1, to simulate an execution of the system within a game, the agents force a pebble to move along the states and the path of the pebble describes an execution of the system. After t time units, the sequence of states visited by the pebble defines a history ρ and the strategies used by the agents make their choices at time t based on ρ. To handle with ease the evolution of a strategy along a history, we use the notion of strategy translations[5].

Definition 1.11 (Strategy translation)**.**
For a history ρ and a strategy δ both on a common game $\mathcal{G}$, we call the translation $\delta_{\overrightarrow{\rho}}$ of δ along ρ the partially defined strategy

$$\delta_{\overrightarrow{\rho}}(\rho') := \delta(\rho.\rho') \quad \textit{for any history } \rho' \textit{ starting in a successor of } \mathsf{lst}(\rho)$$

A particularity of SL is to treat strategies as first-order elements. For this, strategies are stored in variables in a way similar to the first-order logic. For the rest of this section, we fix a set $\mathcal{V}$ of variables.

Definition 1.12 (Valuation)**.**
A valuation χ over a set Agt of agents and a set $\mathcal{V}$ of variables is a partial function $\chi : \mathsf{Agt} \cup \mathcal{V} \mapsto \mathsf{Strat}_{\mathcal{G}}$.

Intuitively, the strategies stored in the variables represent the behaviours under consideration while the strategy associated with an agent represents its effective behaviour. The notion of valuation is nothing more than an extension of contexts to handle variables. We can then extend the notion of strategy translation to valuations.

[4] CHP-SL was the first one, introduced in 2007, while BSIL dated from 2011. The model checking of BSIL is however better than the one of CHP-SL, making it easier to use in practice.

[5] The strategy translation of δ along a history ρ is a similar concept to the classical one of strategy induced by another strategy in a sub-tree on games played on trees [42].

Definition 1.13 (Valuation translation)**.**
Given a game $\mathcal{G}$, a valuation χ over Agt and $\mathcal{V}$ where Agt is the set of agents of a game $\mathcal{G}$, and a history $\rho \in \mathsf{Hist}_{\mathcal{G}}$, we define the valuation translation $\chi_{\overrightarrow{\rho}}$ by

$$\forall x \in (\mathsf{Agt} \cup \mathcal{V}) \cap \mathsf{dom}(\chi) \qquad \chi_{\overrightarrow{\rho}}(x) := \chi(x)_{\overrightarrow{\rho}}$$

Sometimes, we will need to have a closer look at some specific choices. We define the notion of *move vector* to model a strategy choices at a given time.

Definition 1.14 (Move vector)**.**
A move vector over a set $D \subseteq \mathsf{Agt}$ is a partial function $m : D \mapsto \mathsf{Act}$ mapping each element of the domain to an action; if $D = \mathsf{Agt}$ we simply refer to m as a move vector and omit its domain.

When $\mathsf{Agt} \subseteq \mathsf{dom}(\chi)$, each agent has an assigned behaviour. From a game $\mathcal{G}$, one of its states q and a valuation χ with $\mathsf{Agt} \subseteq \mathsf{dom}(\chi)$, we can launch a simulation and get an outcome. We update the notion of outcome (defined page 24) to work with valuations

Definition 1.15 (Outcome)**.**
Fix a game $\mathcal{G} := \langle \mathsf{AP}, \mathsf{Agt}, Q, \mathsf{Act}, \Delta, \mathsf{labels} \rangle$, one of its states q and a valuation χ with $\mathsf{Agt} \subseteq \mathsf{dom}(\chi)$. We define the outcome $\mathsf{out}(\chi, q) := (q_i)_{i \in \mathbb{N}}$ of χ from q by the unique path where

- $q_0 = q$
- $\forall i \in \mathbb{N}$, $q_{i+1} = \Delta(q_i, d_i)$ *where $d_i \in \mathsf{Act}^{\mathsf{Agt}}$ is the move vector defined for all agent $A \in \mathsf{Agt}$ by $d_i(A) := \chi(A)((q_j)_{j \leq i})$.*

1.5.2 Strategy Logic

We are finally ready to define the logic at the heart of this manuscript. SL is built upon a set Agt of agents, a set AP of atomic propositions and a set $\mathcal{V}$ of variables. SL formulas are constructed by the following grammar:

$$\mathsf{SL} \ni \phi \ ::= \ \exists x.\phi \mid \mathsf{assign}(A, x).\phi \mid \phi \vee \phi \mid \neg\phi \mid \phi \mathbf{U}\, \phi \mid \mathbf{X}\, \phi \mid p$$

where $x \in \mathcal{V}$ is a variable, $A \in \mathsf{Agt}$ is an agent and $p \in \mathsf{AP}$ is an atomic proposition.

The operator $\mathsf{assign}(A, x)$ will be referred to as an *assignment* of x to A while we retrieve some form of strategy quantification in the $\exists x$ operator. The until ($\mathbf{U}$) and next ($\mathbf{X}$) operators are similar to their eponymous parts in LTL.

In order to define the semantics of SL we need several intermediary notions. The notation $\mathsf{free}(\phi)$ represents the set of *free agents and variables* of a formula ϕ that have yet to be associated with a strategy before ϕ can be evaluated. It is defined inductively

as follows,

$$\begin{aligned}
\mathsf{free}(p) &= \varnothing \quad \text{for all } p \in \mathsf{AP} & \mathsf{free}(\mathbf{X}\,\phi) &= \mathsf{Agt} \cup \mathsf{free}(\phi) \\
\mathsf{free}(\neg\phi) &= \mathsf{free}(\phi) & \mathsf{free}(\phi\,\mathbf{U}\,\psi) &= \mathsf{Agt} \cup \mathsf{free}(\phi) \cup \mathsf{free}(\psi) \\
\mathsf{free}(\phi \vee \psi) &= \mathsf{free}(\phi) \cup \mathsf{free}(\psi) & \mathsf{free}(\exists x.\ \phi) &= \mathsf{free}(\phi) \setminus \{x\}
\end{aligned}$$

$$\mathsf{free}(\mathsf{assign}(A, x).\ \phi) = \begin{cases} \mathsf{free}(\phi) & \text{if } A \notin \mathsf{free}(\phi) \\ (\mathsf{free}(\phi) \cup \{x\}) \setminus \{A\} & \text{otherwise} \end{cases}$$

We then say that ϕ is *closed* whenever $\mathsf{free}(\phi) = \varnothing$.

Notations. *For the sake of formality, we define the notation used for updating a valuation. For a valuation χ on a game $\mathcal{G}$, a variable x and a strategy s on $\mathcal{G}$, the notation $\chi_{[x \mapsto s]}$ represents the (unique) valuation where*

- *for any $x' \neq x$ such that $x' \in \mathsf{dom}(\chi)$, we have $\chi_{[x\mapsto s]}(x') = \chi(x')$*
- *for any $x' \neq x$ such that $x' \notin \mathsf{dom}(\chi)$, we have $x' \notin \mathsf{dom}(\chi_{[x\mapsto s]})$*
- $\chi_{[x\mapsto s]}(x) = s$

Formulas of SL are evaluated on a concurrent game $\mathcal{G} := \langle \mathsf{AP}, \mathsf{Agt}, \mathsf{Q}, \mathsf{Act}, \Delta, \mathsf{labels} \rangle$ at a state q with respect to a valuation χ where the sets of agents and atomic propositions of the formula and the game coincide.

$$\begin{aligned}
\mathcal{G}, q \models_\chi p &\quad\Leftrightarrow\quad p \in \mathsf{labels}(q) \\
\mathcal{G}, q \models_\chi \phi \vee \phi' &\quad\Leftrightarrow\quad \mathcal{G}, q \models_\chi \phi \text{ or } \mathcal{G}, q \models_\chi \phi' \\
\mathcal{G}, q \models_\chi \neg\phi &\quad\Leftrightarrow\quad \mathcal{G}, q \not\models_\chi \phi
\end{aligned}$$

If $\mathsf{free}(\phi) \setminus \{x\} \subseteq \mathsf{dom}(\chi)$, then

$$\mathcal{G}, q \models_\chi \exists x.\phi \quad\Leftrightarrow\quad \exists \delta \in \mathsf{Strat}_\mathcal{G} \text{ such that } \mathcal{G}, q \models_{\chi[x\mapsto\delta]} \phi$$

Additionally, given $A \in \mathsf{Agt}$, if $(\mathsf{free}(\phi) \setminus \{A\}) \cup \{x\} \subseteq \mathsf{dom}(\chi)$ then

$$\mathcal{G}, q \models_\chi \mathsf{assign}(A, x).\phi \quad\Leftrightarrow\quad \mathcal{G}, q \models_{\chi[A\mapsto\chi(x)]} \phi$$

If $\mathsf{Agt} \cup \mathsf{free}(\phi) \cup \mathsf{free}(\psi) \subseteq \mathsf{dom}(\chi)$ then χ produces a unique outcome π from q, i.e. $\pi := \mathsf{out}(\chi, q)$ (as explained page 34). We can translate the valuation χ based on this outcome: for any integer j, we write $\chi_{\overrightarrow{j}}$ for $\chi_{\overrightarrow{\pi_{\leq j}}}$. We then let

$$\begin{aligned}
\mathcal{G}, q \models_\chi \mathbf{X}\,\phi &\quad\Leftrightarrow\quad \mathcal{G}, \mathsf{out}(\chi, q)(1) \models_{\chi_{\overrightarrow{1}}} \phi \\
\mathcal{G}, q \models_\chi \phi \mathbf{U}\,\phi' &\quad\Leftrightarrow\quad \exists k \in \mathbb{N}.\ \mathcal{G}, \mathsf{out}(\chi, q)(k) \models_{\chi_{\overrightarrow{k}}} \phi' \text{ and} \\
&\qquad\qquad \forall j \in \mathbb{N}.\ 0 \leq j < k \Rightarrow \mathcal{G}, \mathsf{out}(\chi, q)(j) \models_{\chi_{\overrightarrow{j}}} \phi
\end{aligned}$$

As in LTL we use the abbreviations $\top$ for the universal true, $\bot$ for the universal false, $\mathbf{F}\,\phi$ for the future operator ($\top U \phi$) and $\mathbf{G}\,\phi$ for the always operator ($\neg\mathbf{F}\,\neg\phi$). We also write $\forall x.\phi$ for the universal strategy quantification ($\neg\exists x.\neg\phi$).

1.5.3 The nested and boolean goal fragments SL[NG] and SL[BG]

Like often in model checking and complexity related problems, a high expressiveness induces serious drawbacks. SL grammatically allows many things :

- the capacity to handle multiple objectives at once:

$$\forall x.\forall y.\exists x'.\ \Big(\big(\mathsf{assign}(A_1,x).\mathsf{assign}(A_2,y)\mathbf{F}\,p_1\big) \vee \big(\mathsf{assign}(A_1,x').\mathsf{assign}(A_2,y)\mathbf{F}\,p_2\big)\Big)$$

  where $\mathsf{Agt} = \{A_1, A_2\}$ and $\mathsf{AP} = \{p_1, p_2\}$

- multiple agents may share common strategies: in the following formula, the strategy z is assigned to both A_1 and A_2.

$$\forall y.\forall x.\exists z.\ \Big(\big(\mathsf{assign}(A_1,x).\mathsf{assign}(A_2,z)\mathbf{F}\,p_1\big) \vee \big(\mathsf{assign}(A_1,z).\mathsf{assign}(A_2,y)\mathbf{F}\,p_2\big)\Big)$$

- the possibility to redefine a strategy midway through the simulation:

$$\forall x.\exists y.\ \mathsf{assign}(A_1,x).\mathsf{assign}(A_2,y)\mathbf{F}\,\big(p_1 \wedge \exists x'\mathsf{assign}(A_1,x')\mathbf{F}\,p_2\big)$$

For this reason, one may wish to restrict SL to simplify both algorithms and reasoning. One way, proposed in [39], is to streamline the formulas by using the notion of *goals*. A *goal* is a sub-formula composed of an assignment followed by a temporal objective expressed with LTL operators or other (nested) goals. The idea is to create a fragment SL[NG] of SL that does not allow partial re-quantifications once the simulation has started, allowing for a clear separation of the quantifications on one side and the assignments and the temporal operators on the other side.

The SL[NG] fragment

A formula will be in SL[NG] whenever we can regroup the quantifications in blocks in such a way that any new block marks the beginning of a closed sub-formula. As a counter-example, take the formula below

$$\forall x.\exists y.\ \mathsf{assign}(A_1,x).\mathsf{assign}(A_2,y)\mathbf{F}\,\big(p_1 \wedge \exists x'\mathsf{assign}(A_1,x')\mathbf{F}\,p_2\big)$$

There are two blocks of quantifications: $\forall x.\exists y$ and $\exists x'$; the sub-formula

$$\exists x'\mathsf{assign}(A_1,x')\mathbf{F}\,p_2$$

is not closed (the agent A_2 is free) therefore the second block is partial and the overall formula will not be in SL[NG].

To simplify the grammar of SL[NG], we use the notion of *flatness*. A logic will be called *flat* when it does not allow closed sub-formulas. We first define the flat fragment SL[NG]$^\flat$ of SL[NG] before giving the complete grammar.

Remark 1.16. *More generally, for all purposes in this thesis, closed sub-formulas are seen as atomic propositions: given a game $\mathcal{G}$ and one of its state q, a closed formula ϕ either evaluate to true or false on q independently of the current valuation. We can then create a new atomic proposition p_ϕ and label the states of the game with p_ϕ whenever ϕ holds true. Model checking algorithms then may proceed inductively by solving the deepest closed sub-formula ϕ, labelling accordingly the game with a new atomic proposition p_ϕ and replacing ϕ in the main formula by p_ϕ. Members of the verification community frequently assimilate closed sub-formulas in their logics with atomic propositions, we simply continue this trend.*

Definition 1.17 (Nested goals strategy logic, SL[NG])**.**
Flat nested goal formulas are build upon the following rules:

$$\begin{aligned} \mathit{SL[NG]}^\flat \ni \phi &::= \exists x.\phi \mid \forall x.\phi \mid \xi \\ \xi &::= \xi \vee \xi \mid \xi \wedge \xi \mid \beta \\ \beta &::= \mathit{assign}(A, x).\beta \mid \varphi \\ \varphi &::= \varphi \vee \varphi \mid \neg\varphi \mid \varphi \mathbf{U} \varphi \mid \mathbf{X}\varphi \mid p \mid \beta \end{aligned}$$

where $x \in \mathcal{V}$ is a variable, $A \in \mathit{Agt}$ is an agent and $p \in \mathit{AP}$ is an atomic proposition.

SL[NG] (non flat logic) allows closed SL[NG] formulas in its grammar at the atomic proposition level (i.e. in φ-type formulas).

We can now clearly see some notion of *goal* appearing in SL[NG]: the β type sub-formulas. A closed formula of SL[NG] is then a block of quantification followed by a boolean combination of goals (with potentially some nested goals within).

The SL[BG] fragment

Recently another fragment called SL[BG] has gained importance. It further restricts SL[NG] by forbidding assignments past the start of the simulation, ensuring that each agent keeps to its initial strategy. More precisely, SL[BG] aims at creating formulas of the following form: $\wp\xi(\beta_i\varphi_i)_{i\leq n}$ where $\wp$ is a block of quantifications, ξ is a boolean combination and for all $i \leq n$, β_i is a block of assignments while φ_i is a LTL formula.

Definition 1.18 (Boolean goal strategy logic, SL[BG])**.**
The flat boolean goal fragment $\mathit{SL[BG]}^\flat$ has the grammar

$$\begin{aligned} \mathit{SL[BG]}^\flat \ni \phi &::= \exists x.\phi \mid \forall x.\phi \mid \xi \\ \xi &::= \xi \vee \xi \mid \xi \wedge \xi \mid \beta \\ \beta &::= \mathit{assign}(A, x).\beta \mid \varphi \\ \varphi &::= \varphi \vee \varphi \mid \neg\varphi \mid \varphi \mathbf{U} \varphi \mid \mathbf{X}\varphi \mid p \end{aligned}$$

The non-flat fragment SL[BG] allows closed SL[BG] formulas in its grammar at the atomic proposition level (i.e. in φ-type formulas).

Further restrictions of the boolean combination (ξ type formulas) are possible. As we will see later, a deeper look into the boolean combinations yields interesting results. Among the possible restrictions, we identify three major ones.

- SL[1G], introduced in [38], restricts SL[BG] to a unique goal. (The flat fragment of) SL[1G] is defined from the grammar of SL[BG] by skipping the ξ line. More precisely, ξ type sub-formulas must avoid any boolean operator i.e. must be of form $\xi ::= \beta$

- SL[CG], introduced in [40], is the fragment where only *conjunctions* of goals are allowed. Formally, (the flat fragment of) SL[CG] is defined from the grammar of SL[BG] with the restriction below on the ξ's type sub-formulas $\xi ::= \xi \wedge \xi \mid \beta$

- Similarly, SL[DG] only allows *disjunctions* of goals, i.e. $\xi ::= \xi \vee \xi \mid \beta$.

1.5.4 Examples

We give two examples to illustrate the expressive power of SL and of its fragments.

Nash equilibrium The first example is the existence of a pure (qualitative) Nash equilibrium (with LTL objectives $(\varphi_i)_{1\leq i\leq n}$) and was proposed in [39]. It works no matter the game and can be expressed in SL[BG] as

$$\exists x_1, \dots x_n.\ \forall y_1, \dots, y_n.\ \bigwedge_{1\leq i\leq n} \mathsf{assign}(A_j, x_j)_{j\neq i}\big(\mathsf{assign}(A_i, y_i)\varphi_i \Rightarrow \mathsf{assign}(A_i, x_i)\varphi_i\big),$$

where for the sake of readability we merge n strategy assignments into a single one. In such a formula, $(x_i)_{1\leq i\leq n}$ is the strategy profile we are looking for, and $(y_i)_{1\leq i\leq n}$ are intended to be the possible deviations of the agents. The formula states that if some agent can change his strategy and achieve his goal, then his goal is already met in the original strategy profile, thus making said strategy profile a Nash equilibrium.

Dominating strategies Consider a CGS $\mathcal{G}$ with two agents A and B. We say that a strategy δ is dominating another strategy δ' relatively to an agent A and an objective φ when there is no behaviour for B for which δ' leads to φ but δ does not.

Fix a valuation χ that stores a strategy δ in a variable x_1:

$$\chi := \{x_1 \to \delta\}$$

Then δ is dominating any other strategy for agent A relatively to φ when the following formula holds true on $\mathcal{G}$ relatively to the valuation χ

$$\forall x_2.\ \big(\exists x_3.\ \mathsf{assign}(A, x_3).\ \mathsf{assign}(B, x_2).\ \varphi\big) \Rightarrow \big(\mathsf{assign}(A, x_1).\ \mathsf{assign}(B, x_2).\ \varphi\big)$$

1.5.5 Expressiveness of SL fragments

Excluding SL[1G], all fragments allows two executions of the system to share some common behaviour. For instance, take the SL[CG] formula

$$\exists x_1.\ \forall x_2.\ \forall x_3.\ (\mathsf{assign}(A, x_1).\mathsf{assign}(B, x_2)\mathbf{F}\, p \wedge \mathsf{assign}(A, x_1).\mathsf{assign}(B, x_3)\mathbf{F}\, p')$$

The strategy stored in x_1 is common to A in both goals $(\mathsf{assign}(A, x_1).\mathsf{assign}(B, x_2)\mathbf{F}\, p$ and $\mathsf{assign}(A, x_1).\mathsf{assign}(B, x_3)\mathbf{F}\, p')$. This way, all fragments but SL[1G] offer a solution to the commitment problem. On the other hand, SL[1G] focuses on a single execution of the system; therefore, the commitment issue does not apply (does not make sense) in SL[1G].

Only the full logic SL permits partial quantification within an outcome. the fragment SL[NG] allows for the goals to change; therefore a player can change its strategy, but only among the one quantified before the start of the execution. In all the other fragments, in each execution of the system, an agent must stick to its assigned strategy (which does not prevent an agent to have two different strategies in two different executions). This way, SL provides a complete solution to the revocation issue and SL[NG] only a partial answer. The other fragments do not touch the issue.

1.6 Summary

We sum up the relations between the different logics in Figure 1.5. A solid arrow $L \to L'$ denotes that L' is at least as expressive as L, i.e. for any formula $\phi \in L$ we can find an equivalent formula $\phi' \in L'$ in the sense that

For any concurrent game structure $\mathcal{G}$ and any state q of $\mathcal{G}$ $\qquad \mathcal{G}, q \models \phi \quad \Leftrightarrow \quad \mathcal{G}, q \models \phi'$

A dashed arrow $L \dashrightarrow L'$ denotes the existence of a polynomial reduction from L to L', i.e there is a transformation $\mathcal{T}$ taking as input a formula $\phi \in L$, a model $\mathcal{M}$ adapted to L and one of its state, and returning a formula ϕ' of L', a model $\mathcal{M}'$ adapted to L' and a state q' of $\mathcal{M}'$ in polynomial time of ϕ and $\mathcal{M}$ such that

$$\mathcal{M}, q \models \phi \quad \Leftrightarrow \quad \mathcal{M}', q' \models \phi'$$

There are also a few results distinguishing the logics that do not appear on Figure 1.5. First, in [2] it was proved that GL is not more expressive than AL_μ. For example, GL cannot express that all even states along a path are labeled p without imposing some condition on the odd states while AL_μ can. The authors of [10] proved that AL_μ is not more expressive than ATL_{sc}. In particular AL_μ cannot distinguish between alternating-bisimilar models while ATL_{sc} can[6]. Combining the two, we get that AL_μ has a very distinct expressive power compared to either GL or ATL_{sc}.

[6] A bisimulation is a binary relation between transition systems that associate systems that behave similarly. We do not formally define the notion of bisimulation (nor the one of alternating-bisimulation) and refer to [36] and [55] for surveys.

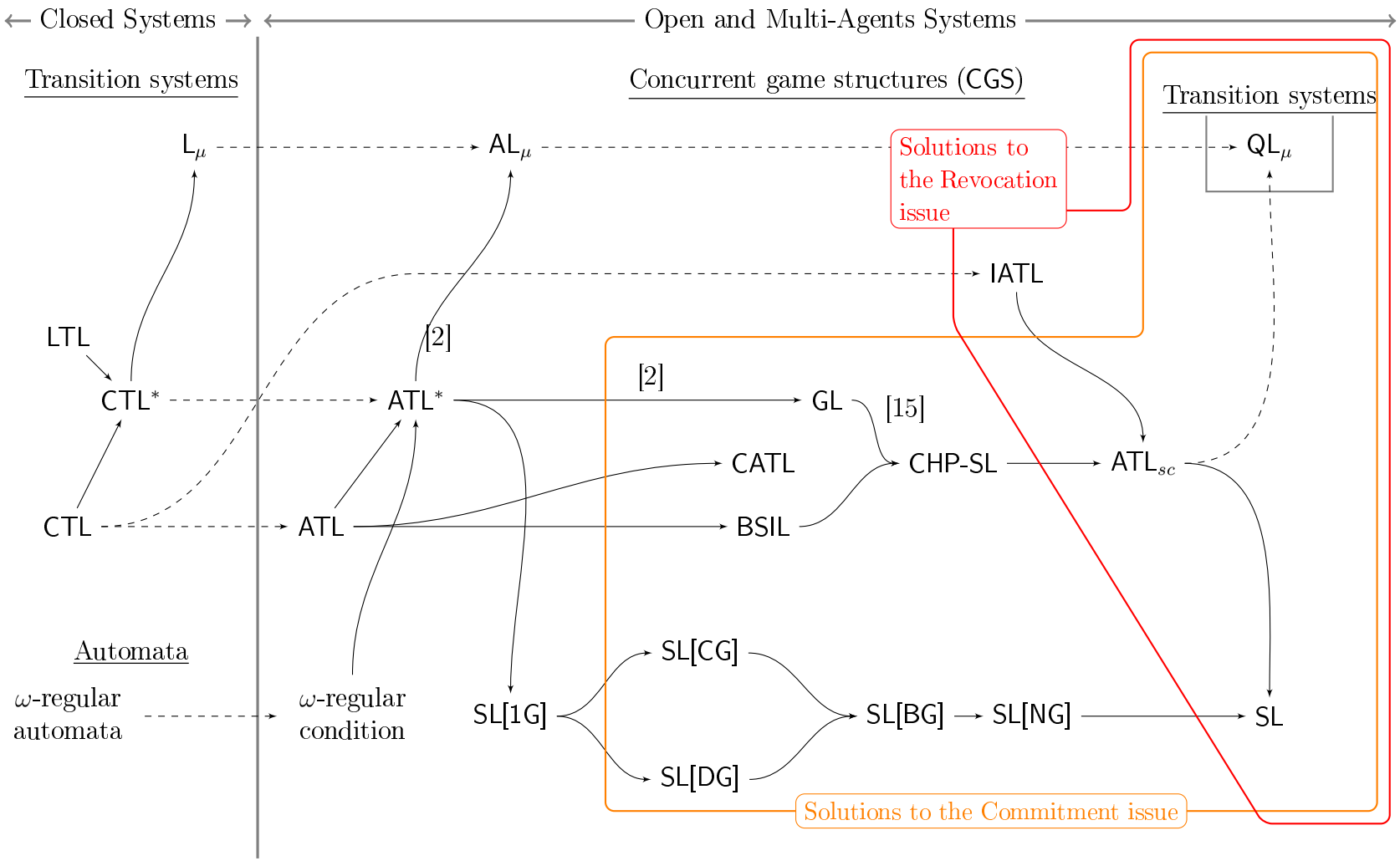


Figure 1.5: The constellation of temporal logics for multi-agent systems.

Chapter 2

Complexity of SL

In this chapter we go over the complexity results known on SL. In the theoretical framework of SL, two questions come of interest: the first one, *the satisfiability problem*, asks for an algorithm to check if there is some game on which a formula given in input is satisfiable or if the formula is structurally incoherent; the second, *the model checking* problem, asks for an algorithm taking as inputs a formula and a game that returns whether the formula holds on the game. We define formally both problems though we will mostly focus on the second one.

Definition 2.1 (Satisfiability problem)**.**
The satisfiability problem for a logic $\mathcal{L}$ ask for an algorithm that takes as input a formula $\phi \in \mathcal{L}$ and returns whether there is some CGS $\mathcal{G}$ such that ϕ holds on $\mathcal{G}$.

Definition 2.2 (Model checking problem)**.**
The model checking problem for a logic $\mathcal{L}$ asks for an algorithm that takes as input a formula $\phi \in \mathcal{L}$ and a CGS $\mathcal{G}$, and returns whether ϕ holds on $\mathcal{G}$.

We are looking for which complexity class the model checking and satisfiability problems belong. We do not define complexity classes and refer the interested reader to [3, 52]. We now focus on the model checking problem. We however recall the existing results about the satisfiability of SL, its sub-logics and a few other game-related temporal logics.

Initially developed on transition systems, LTL can easily be adapted to games: can an agent enforce that the outcome satisfies the LTL formula no matter the decisions of the other agents? Pnueli and Rosner found in 1989 [45] that LTL on games admits a *2*-EXPTIME-complete model checking. This result was applied in [2] to get the following theorem.

Theorem 2.3 (Alur, Henzinger and Kupferman [2])**.**
ATL* model checking and satisfiability are both 2-EXPTIME-complete.

Other techniques were applied for solving the SL[1G] satisfiability problem. The result is however similar, as shown by the theorem below. In particular, SL[1G] satisfiability is not more complex than ATL* or LTL (on games).

Theorem 2.4 (Mogavero, Murano, Perelli and Vardi [38])**.**
*SL[1G] satisfiability is 2-*EXPTIME*-complete.*

While there is no formal proof of the undecidability of SL[BG] satisfiability, it can easily be derived from the undecidability of the satisfiability problems for other logics. QCTL [18] is a temporal logic whose expressiveness is similar in many way to that of SL[BG] (it is also very similar to ATL_{sc} expressiveness). QCTL satisfiability can be reduce to SL[BG] satisfiability relatively easily. In [30] QCTL satisfiability was proven undecidable, we can then use the reduction from QCTL to SL[BG] to get the same undecidability for SL[BG]. Due to its high expressiveness, this result is not a surprise. As show by Theorem 2.4, we regain decidability in the SL[1G] fragment.

Theorem 2.5. *SL[BG] (and therefore SL) satisfiability problem is undecidable*

For the rest of this chapter, we focus on the model checking problem of SL and its sub-logics.

2.1 SL upper bound

In [39], Mogavero, Murano, Perelli and Vardi developed an algorithm for SL model checking.

Theorem 2.6 (Mogavero, Murano, Perelli and Vardi [39])**.**
The model checking problem for SL is in NONELEMENTARY *with respect to the size of the formula. It is in* $(k+1)$*-*EXPTIME *for SL[NG] formulas with* k *or less quantifier alternations.*

The algorithm acts in a fashion similar to the well-known decidability result for MSO on infinite binary trees (also known as S2S). The proof presented in [56] builds a Büchi tree automaton by iteration on the formula. The emptiness of the final automaton is then equivalent to the validity of the MSO formula on the infinite binary tree. For SL model checking, the proof also proceeds by building an automaton by induction on the formula and by solving the emptiness problem of the automaton at the last step. The Büchi tree automaton is however replaced by an alternating parity tree automaton. The proof is rather long, we therefore provide only a sketch of it. The curious reader may refer to the original paper for more details.

Sketch of proof.
We start with two definitions:

Definition 2.7. *A tree over the state space* Q *is a tuple* $\mathcal{T} := \langle T, E\rangle$ *where* $T \subseteq Q^*$ *is the set of nodes (with each node a finite sequence over* Q*);* $E : T \to 2^T$ *is a transition function such that for any* $t, t' \in T$*, if* $t' \in E(t)$ *then* $|t'| = |t| + 1$*;*

A Σ*-labelled tree* $\mathcal{T}$ *is a tuple* $\mathcal{T} := \langle \Sigma, T, E, \mathsf{labels}\rangle$ *where* $\langle T, E\rangle$ *is a tree,* Σ *is the input alphabet and* $\mathsf{labels} : T \to \Sigma$ *is a labelling function associating an input symbol with each node of the tree.*

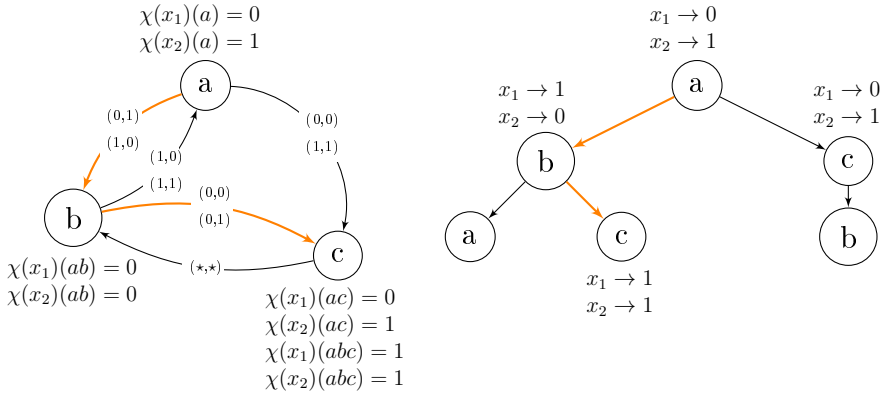


Figure 2.1: Correspondence between a valuation χ on a game $\mathcal{G}$, and a labelled unwinding of $\mathcal{G}$.

Definition 2.8. *The unwinding of a CGS* $\mathcal{G} = \langle AP, Agt, Q, Act, \Delta, labels \rangle$ *is an (unlabelled) tree* $\mathcal{T} := \langle T, E \rangle$ *where* T *is built upon the state space* Q *of* $\mathcal{G}$ *and such that for any* $t \in T$,

$$t.q' \in E(t) \text{ if and only if there exists } d \in Act^{Agt} \text{ such that } \Delta(lst(t), d) = q'$$

Consider a SL formula ϕ, a game $\mathcal{G}$, a set $\mathcal{V}$ of variables and a valuation χ over $\mathcal{G}$ and $\mathcal{V}$. We let $\mathcal{T}$ be the infinite tree representing the unwinding of $\mathcal{G}$. The valuation χ can be put in correspondence with a labelling of $\mathcal{T}$ over the alphabet $\mathsf{Act}^{\mathsf{Agt}}$, where the label of a state q in a branch t of $\mathcal{T}$ represents the action choices of $\chi(t)$ when t is viewed as a history of $\mathcal{G}$. Figure 2.1 illustrates the idea.

The model checking procedure of SL uses this idea to build an appropriate alternating parity tree automaton by a bottom-up induction on the formula. At step ϕ' (for a sub-formula ϕ' of ϕ), the algorithm produces an automaton $\mathcal{N}_{\phi'}$ based on ϕ' own sub-formulas. This automaton $\mathcal{N}_{\phi'}$ encodes the game $\mathcal{G}$ by accepting only unwindings of $\mathcal{G}$ and obeys the following property.

Proposition 2.9. *For any valuation* χ *with* $dom(\chi) = free(\phi')$, *writing* $\mathcal{T}_{dom(\chi)}$ *for the labelled tree in correspondence with* χ *(with* $Act^{dom(\chi)}$ *the set of labels), then*

$$\mathcal{G}, q_{ini} \models_\chi \phi' \quad \Leftrightarrow \quad \mathcal{T}_{dom(\chi)} \in \mathcal{L}(\mathcal{N}_{\phi'})$$

Sub-formulas made of an LTL operator are handled using standard techniques, for example the $\phi_1 \mathbf{U}\, \phi_2$ operator uses the decomposition $\phi_2 \vee (\phi_1 \wedge \mathbf{X}\,(\phi_1 \mathbf{U}\, \phi_2))$. A sub-formula $\phi' = \mathsf{assign}(A, x)$. ϕ'' starting with an assignment simply modifies the automaton $\mathcal{N}_{\phi''}$ of ϕ'' built at previous steps by updating the transition function according to $\mathsf{assign}(A, x)$. A sub-formula $\phi' = \exists x.\ \phi''$ proceeds by a projection of the $\mathcal{N}_{\phi''}$ automaton. There remains the case of the boolean operators. The $\vee$ and $\neg$ operators are handled through the standard union and intersection of automata. Finally, the $\neg$ operator is handled through complementation of parity tree automata with a standard technique (also developed in [43]). Then, at the last step of the induction we get a parity tree automaton $\mathcal{N}_\phi$

such that

$$\mathcal{G}, q_{ini} \models \phi \quad \Leftrightarrow \quad \exists \mathcal{T} \in Tree_{\emptyset} \text{ s.t } \mathcal{T} \in \mathcal{L}(\mathcal{N}_{\phi})$$

where $Tree_{\emptyset}$ is the set of unwindings of $\mathcal{G}$. The equivalences can be solved by using standard techniques to solve the emptiness of tree automata in time a^b, where a is the number of state of $\mathcal{N}_{\phi}$ and b is the number of indexes. An algorithm can be found in [29].

The algorithm over SL[NG]: The first operator of a closed formula can only be either a quantification or an atomic proposition (otherwise the formula has some free variable or free agent or is not syntactically in SL[NG]). The last possibility makes ϕ trivial and we set it aside. If ϕ starts by a universal quantification, we can solve the model checking for its negation $\neg\phi$ and reverse the result to get ϕ's model checking. We can therefore assume without loss of generality that ϕ is of form $\phi = \exists x.\ \phi'$. This existential quantification at the start allows us to stop the induction the step before ϕ and get an automaton $\mathcal{N}_{\phi'}$

$$\mathcal{G}, q_{ini} \models \phi \quad \Leftrightarrow \quad \mathcal{G}, q_{ini} \models \exists x.\ \phi' \quad \Leftrightarrow \quad \exists \mathcal{T} \in Tree_{\{x\}} \text{ s.t } \mathcal{T} \in \mathcal{L}(\mathcal{N}_{\phi'})$$

The automata built at the initial step of the induction are of size exponential in the formula and polynomial in the game. For sub-formulas starting by all but a $\neg\phi'$ operator, the automaton built at step α is of size polynomial in the automata build at steps less than α. The only step where the automaton increases in a non polynomial factor compared to the size of the automata previously built is for sub-formulas starting with the $\neg\phi'$ operator. Then, the build-up is exponential in the size of the automaton of ϕ'.

Using the $\vee$ and $\wedge$ operator we may modify any SL[NG] formula to push the negation either between quantifiers or at the atomic propositions' level. The size of the automaton for the modified formula then grows only through quantifier alternation and we retrieve the complexity result of Theorem 2.6 for SL[NG]. □

2.2 Data complexity

The data complexity of an algorithm is the complexity relative to the size of the game given as input. In the model checking procedure for SL exposed above, the size of the parity tree automaton respectively to the size of the game grows exponentially with each quantifier alternation. The data complexity is therefore a tower of exponentials of height equal to the number of quantifier alternations and final exponent a polynomial in the size of the game.

Definition 2.10.
We define the function Tower: $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *by induction. Initially* $\mathsf{Tower}(a, 0) := a$ *while* $\mathsf{Tower}(a, b+1) := 2^{\mathsf{Tower}(a,b)}$ *at the induction step. This encodes* towers of exponentials *of the form* $2^{2^{\cdot^{\cdot^{\cdot^{a}}}}}$.

To be precise, the order of magnitude of SL data complexity (for the model checking problem) is $\mathsf{Tower}(k, P(|\mathcal{G}|))$ with k the number of quantifier alternations and P a polynomial. The authors of [39] made a calculation error and claimed a data complexity

in PTIME for their SL model checking algorithm, the algorithm however provides only a NONELEMENTARY data complexity.

Theorem 2.11. *SL model checking data complexity is in* NONELEMENTARY.

With this rectification, we provide a proof of hardness for the data model checking of SL[BG] (hence also SL). We show that the data complexity of SL[BG] model checking is PH-hard(as in hard for each level of the polynomial hierarchy).

Theorem 2.12. *The data complexity of SL[BG] model checking is* PH-*hard.*

Proof. We prove that it is hard for each level of the boolean formulas hierarchy, Theorem 2.12 will follow. SL expressiveness in general is heavily linked to QCTL expressiveness and its data complexity is known to be PH-hard. We could have worked by a reduction from QCTL towards SL to prove Theorem 2.12. We instead opt for a more direct (and shorter) approach, by a reduction towards the satisfiability problem for quantified boolean formulas.

Quantified boolean formulas

Definition 2.13 (Quantified boolean formulas in conjunctive normal form)**.**
Fix an infinite set $\mathcal{V} = \{v_1, \dots\}$ of boolean variables. A quantified boolean expression in conjunctive normal form over $\mathcal{V}$ is a formula build upon the following grammar:

$$\begin{aligned} \textit{CNF-SAT} \ni \phi &::= \exists v\ \phi \mid \forall v.\ \phi \mid \zeta \quad \textit{where } v \textit{ is any variable of } \mathcal{V}. \\ \zeta &::= \zeta \wedge \zeta \mid \eta \\ \eta &::= \eta \vee \eta \mid \psi \\ \psi &::= v \mid \neg v \end{aligned}$$

We reuse standard vocabulary: ψ type formulas are called literals, η formulas are called clauses and ζ formulas are called conjunctions. Quantified boolean formulas over a set $\mathcal{V}$ of variables are evaluated over a (partial) interpretation $int : \mathcal{V} \to \{\top, \bot\}$ and are true when int evaluates ϕ to $\top$. Given a pre-existent interpretation int, the $\exists v\ \phi$ operator asks for the existence of an interpretation int' agreeing with int on any variable different from v such that int' evaluates ϕ to $\top$. The $\forall v.\ \phi$ operator asks for each interpretation int' agreeing with int on all variables different from v to be so int' evaluates ϕ to true. Over an interpretation int of domain $\mathcal{V}$, boolean operators are standard.

For any positive integer k, formulas with k quantifier alternations or less, starting by a block of existential quantifications and where all variables are quantified, form the set Σ_k^{SAT} of formulas. The satisfiability problem for a formula $\phi \in \Sigma_k^{SAT}$ asks whether ϕ evaluate to $\top$ from the empty alternation; this problem is known to be complete for Σ_k^P (the k existential level of the polynomial hierarchy).

Σ_1^P hardness of SL[BG] model checking

Lemma 2.14. *SL[BG] (and thus SL[NG] and SL) model checking with respect to the size of the game is* Σ_1^P-*hard.*

Proof. We first define a formula ϕ (independently of any Σ_1^P formula) on the sets $\mathsf{AP} := \{p_{lit}, p_{var}, ok, no\}$ of atomic propositions and $\mathsf{Agt} := \{\boxed{\wedge}, \boxed{\vee}, \boxed{lr}, \bigcirc\}$ of agents:

$$\phi := \exists x.\exists w_{\mapsto}.\exists w_{\mapsto}.\forall y.\forall z.\forall y_{\wedge}.\exists z_{\vee}. \begin{cases} \mathsf{assign}(\boxed{\wedge}, y; \boxed{\vee}, z; \boxed{lr}, w_{\mapsto}; \bigcirc, x)\mathbf{F}\, p_{lit} \\ \wedge \\ \mathsf{assign}(\boxed{\wedge}, y; \boxed{\vee}, z; \boxed{lr}, w_{\mapsto}; \bigcirc, x)\mathbf{F}\, p_{var} \\ \wedge \\ \Big(\mathsf{assign}(\boxed{\wedge}, y_{\wedge}; \boxed{\vee}, z_{\vee}; \boxed{lr}, w_{\mapsto}; \bigcirc, x)\mathbf{F}\, ok \\ \Leftrightarrow\ \mathsf{assign}(\boxed{\wedge}, y_{\wedge}; \boxed{\vee}, z_{\vee}; \boxed{lr}, w_{\mapsto}; \bigcirc, x)\mathbf{F}\, ok\Big) \end{cases}$$

Fix a boolean expression $\Phi := \exists v_1 \ldots \exists v_n.\ \bigwedge_{i\leq I} \bigvee_{j\leq J} l_{i,j}$ with I, J two sets of integers and where $l_{i,j}$ is a literal build on the variables $\{v_1, \ldots, v_n\}$. From Φ, we derive a concurrent game $\mathcal{G} := \langle \mathsf{AP}, \mathsf{Agt}, \mathsf{Q}, \mathsf{Act}, \Delta, \mathsf{labels}\rangle$ represented in Figure 2.2. AP and Agt are the same as in ϕ.

- The state space Q is composed of an initial state q_{ini}, of three states (ok, no, var_i) per variable in $\{v_1 \ldots, v_n\}$ and of one state $lit_{i,j}$ per literal $l_{i,j}$ of Φ.
- The agent $\boxed{lr}$ has two actions $\{\mapsto, \mapsto\}$. $\boxed{\wedge}$ has I actions: $1, \ldots, I$ and $\boxed{\vee}$ has J actions: $1, \ldots, J$. Finally $\bigcirc$ has two actions: a_{ok}, a_{no}.
- On the initial state q_{ini}, Δ depends on the actions of $\boxed{lr}$, $\boxed{\wedge}$ and $\boxed{\vee}$. The agent $\boxed{lr}$ decide if we go to the left (to the variable states with the action $\mapsto$) or right (to the literal states with the action $\mapsto$) part of the game as in Figure 2.2. If $\boxed{lr}$ chooses left, Δ goes from the initial state to the state representing the variable present in $l_{i,j}$; if $\boxed{lr}$ goes to the right, Δ goes to the $lit_{i,j}$ literal state.

  On a variable state var_i, the decision is done by $\bigcirc$ who can play a_{ok} to go to the ok state and a_{no} to go to the no state.
- The variable states are labelled by p_{var} while the literal states are labelled with the common proposition p_{lit}. We also label the ok states with an eponymous atomic proposition and do the same for the no states. Finally for any i, j we label the state $lit_{i,j}$ by ok if $l_{i,j}$ is a variable in Φ and by no if $l_{i,j}$ is the negation of a variable.

We now have the game and the formula. It remains to prove the correctness of the reduction, i.e.

$$\Phi \text{ evaluates to } \top \quad \Leftrightarrow \quad \mathcal{G}, q_{ini} \models \phi$$

Proof of the left-to-right implication Assume there is an interpretation int that evaluates Φ to $\top$, we must prove that $\mathcal{G}, q_{ini} \models \phi$. We define a strategy δ_{int} for $\bigcirc$ by following the choices of int: on var_i, δ_{int} plays to ok if $int(v_i) = \top$ and to no if $int(v_i) = \bot$. We also define two strategies $\delta_{\mapsto}$ and $\delta_{\mapsto}$ for $\boxed{lr}$ that play respectively the action $\mapsto$ and the action $\mapsto$. Write

$$\chi := \{x \to \delta_{int}; w_{\mapsto} \to \delta_{\mapsto}; w_{\mapsto} \to \delta_{\mapsto}\}$$

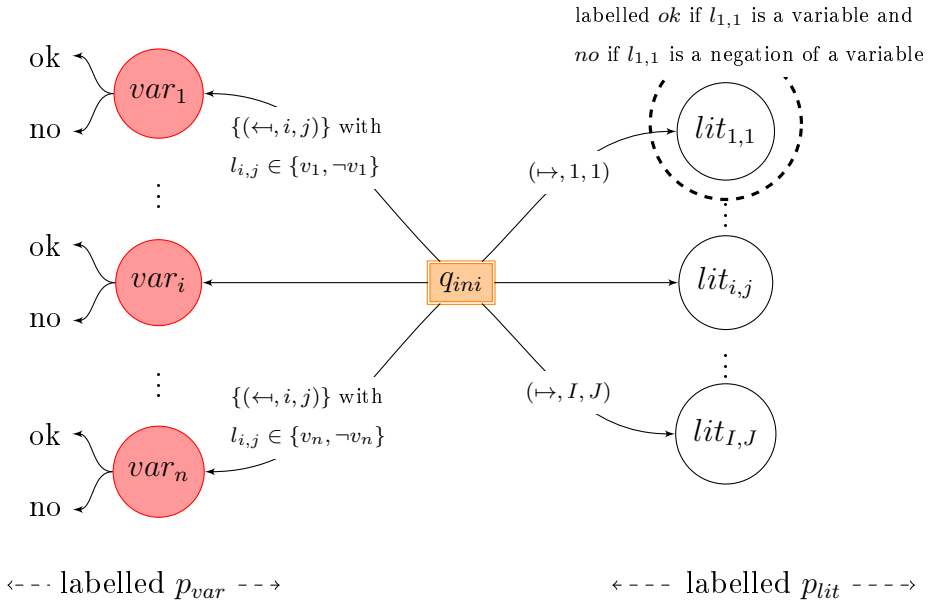


Figure 2.2: Game $\mathcal{G}$ used in lemma 2.14's proof. Not all actions appear on the figure.

Trivially, we get

$$\mathcal{G}, q_{ini} \models_\chi \forall y. \forall z. \forall y_\wedge. \exists z_\vee. \begin{cases} \mathsf{assign}(\wedge, y; \vee, z; lr, w_\mapsto; \bullet, x)\mathbf{F}\, p_{lit} \\ \wedge \\ \mathsf{assign}(\wedge, y; \vee, z; lr, w_\leftarrow; \bullet, x)\mathbf{F}\, p_{var} \end{cases} \tag{2.1}$$

Now, because int is a working interpretation for Φ, for any $i \in I$ there exists some integer $j(i) \in J$ depending on i -that we shorten to j in the following when i is clear of context- such that int evaluates $l_{i,j}$ to $\top$. Fix any strategy $\delta_\wedge$ for $\wedge$ and write i for the action played by $\delta_\wedge$ on q_{ini}. We let $\delta_\vee$ be the strategy for $\vee$ that plays $j(i)$ and write

$$\chi' := \chi \cup \{y_\wedge \to \delta_\wedge; y_\vee \to \delta_\vee\}$$

The assignment $\mathsf{assign}(\wedge, y_\wedge; \vee, z_\vee; lr, w_\mapsto; \bullet, x)$ applied to χ' produces an outcome that goes to the literal state $lit_{i,j}$ while $\mathsf{assign}(\wedge, y_\wedge; \vee, z_\vee; lr, w_\leftarrow; \bullet, x)$ applied to χ' produces an outcome to the variable state $var(i,j)$, where $var(i,j)$ is the variable present in $l_{i,j}$. This means that

- either $l_{i,j} = v(i,j)$: then $ok \in \mathsf{labels}(lit_{i,j})$. As int evaluates $l_{i,j}$ to $\top$ and $l_{i,j} = v(i,j)$, int evaluates $v(i,j)$ to $\top$ and by construction of δ_{int}, $\delta_{int}(var(i,j)) = ok$.
- or $l_{i,j} = \neg x(i,j)$: then $no \in \mathsf{labels}(lit_{i,j})$. Moreover int evaluates $l_{i,j}$ to $\top$ hence int evaluates $v(i,j)$ to $\bot$ and $\delta_{int}(var(i,j)) = no$.

In both cases, the equivalence below holds.

$$\begin{aligned} & \mathsf{assign}(\wedge, y_\wedge; \vee, z_\vee; lr, w_\leftarrow; \bullet, x).\ \mathbf{F}\, ok \\ \Leftrightarrow\ & \mathsf{assign}(\wedge, y_\wedge; \vee, z_\vee; lr, w_\mapsto; \bullet, x).\ \mathbf{F}\, ok \end{aligned}$$

Combining this equivalence with Formula (2.1), we get that if Φ is satisfiable then $\mathcal{G}, q_{ini} \models \phi$.

Proof of right-to-left implication Now, assume that there is a working strategy δ_{int} and that for any $\delta_\wedge$ we can find a working strategy $\delta_\vee$. We can deduce an interpretation int that evaluates true on Φ. To find the working literal, given $i \in I$, we assign the strategy that plays j to $y_\wedge$. We can then use the hypothesis to find a answer strategy $\delta_\vee$ and deduce from it an integer j that ensures $l_{i,j}$ to be evaluated to $\top$ by int. In the end, if $\mathcal{G}, q_{ini} \models \phi$, we can find a suitable interpretation int that evaluates Ψ to $\top$.

We get that $\mathcal{G}, q_{ini} \models \phi$ if and only if we can find a suitable interpretation int that evaluates Ψ to $\top$. This ensures the correctness of our reduction. SAT being NP -hard and the SL[BG] formula being independent from the CNF-SAT formula, the data complexity of SL[BG] model checking must also be NP-hard. □

The same idea used to prove Lemma 2.14 can be applied to extend it to the whole polynomial hierarchy.

Theorem 2.12. *The data complexity of SL[BG] model checking is* PH*-hard.*

□

2.3 A lower bound for SL[BG]

In this section, we prove that the model checking problem for SL[BG] is Tower-hard (the complexity class Tower is the class of problems of complexity bounded by a tower of exponentials, whose height is an elementary function of the input [50]). We actually prove the result for (the flat fragment) $\mathsf{SL[BG]}^\flat$, closing a question left open in [39]. The proof is an extend version of the one presented in [7].

Theorem 2.15. *SL[BG] model checking is Tower-hard.*

We prove this result by encoding the satisfiability problem for QLTL into the model checking problem for SL[BG]. QLTL is the extension of LTL with quantification over atomic propositions [53]: formulas in QLTL are of the form $\Phi = \forall p_1 \exists p_2 \ldots \forall p_{n-1} \exists p_n.\ \varphi$ where φ is in LTL. Notice that we only consider strictly alternating formulas for the sake of readability. The general case can be handled similarly. Formula $\exists p.\ \varphi$ holds true over a word $w \colon \mathbb{N} \to 2^{\mathsf{AP}}$ if there exists a word $w' \colon \mathbb{N} \to 2^{\mathsf{AP}}$ with $w'(i) \cap (\mathsf{AP} \setminus \{p\}) = w(i) \cap (\mathsf{AP} \setminus \{p\})$ and $w' \models \varphi$ for all i. Universal quantification is defined similarly. we then say that Φ is satisfiable if there is a word on which Φ holds.

As an example, consider the formula $\Phi := \forall p_1 \exists p_2\ \mathbf{G}\,(p_1 \Leftrightarrow \mathbf{X}\,p_2)$. Formula Φ is satisfiable. Indeed, consider any word $w : \mathbb{N} \to 2^{\mathsf{AP}}$ resulting of the universal quantification over p_1. Define $w' : \mathbb{N} \to 2^{\mathsf{AP}}$ such that for any $i \in \mathbb{N}$, $p_1 \in w(i)$ if and only if $p_1 \in w'(i)$ and (if $i \geq 1$) $p_2 \in w'(i)$ if and only if $p_2 \in w(i-1)$. The word w' satisfies $\mathbf{G}\,(p_1 \Leftrightarrow \mathbf{X}\,p_2)$, therefore the word w satisfies $\exists p_2\ \mathbf{G}\,(p_1 \Leftrightarrow \mathbf{X}\,p_2)$ and Φ is satisfiable.

It is well-known that satisfiability of QLTL is Tower-complete [54]. We reduce the satisfiability of QLTL into a model checking problem for a SL[BG] formula involving $n+4$ agents (where n is the number of quantifiers in the QLTL formula), and three additional quantifier alternations.

2.3.1 SL lower bound

Before developing this technical encoding, we first present an example of a reduction to SL, which already contains most of the intuitions of our reduction from QLTL to SL[BG].

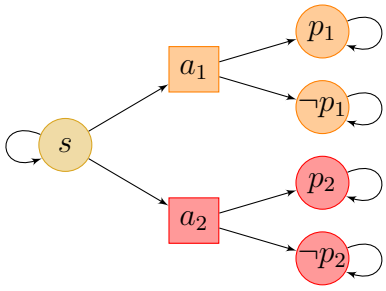


Figure 2.3: The 3-agent turn-based game for the reduction to SL model checking

Consider the QLTL formula

$$\Phi = \forall p_1.\exists p_2.\ \mathbf{G}\,(p_2 \Leftrightarrow \mathbf{X}\,p_1).$$

To solve the satisfiability problem of this formula via SL, we use the three-agent turn-based game depicted on Figure 2.3. In that game, Agent *Gold* controls the Gold state s, while Agents *Orange* and *Red* control the square states a_1 and a_2, respectively. Fix a strategy of Agent *Orange*: this strategy will be evaluated only in Orange state a_1, hence after histories of the form $s^n \cdot a_1$. Hence a strategy of Agent *Orange* can be seen as associating with each integer n a value for p_1. In other words, a strategy for Agent *Orange* defines a labeling of the time line with atomic proposition p_1. Similarly for Agent *Red* and proposition p_2.

It remains to use this correspondence for encoding our QLTL formula. We have to express that for any strategy δ_{Orange} of Agent *Orange*, there is a strategy δ_{Red} of Agent *Red* under which, at each step along the path that stays in s forever, Agent *Gold* can enforce $\mathbf{X}^2 p_2$ if and only if he can enforce $\mathbf{X}^2 p_1$ one step later. In the end, the formula reads as follows:

$$\forall x_1.\ \mathsf{assign}(\mathit{Orange}, x_1).\ \exists x_2.\ \mathsf{assign}(\mathit{Red}, x_2).\ \exists x_3.\ \mathsf{assign}(\mathit{Gold}, x_3).\ \mathbf{G} \begin{cases} s \wedge \exists x_3.\ \mathsf{assign}(\mathit{Gold}, x_3).\ \mathbf{X}^2 p_2 \\ \Leftrightarrow \\ \mathbf{X}\,(\exists x_3.\ \mathsf{assign}(\mathit{Gold}, x_3).\ \mathbf{X}^2 p_1\,) \end{cases} \tag{2.2}$$

One may notice that the above property is not in SL[BG]: for instance, the sub-formula $\exists x_3.\ \mathsf{assign}(\mathit{Gold}, x_3).\ \mathbf{X}^2 p_2$ is not closed.

2.3.2 SL[BG] lower bound

We provide a new construction, refining the ideas above, in order to reduce QLTL satisfiability to SL[BG] model checking.

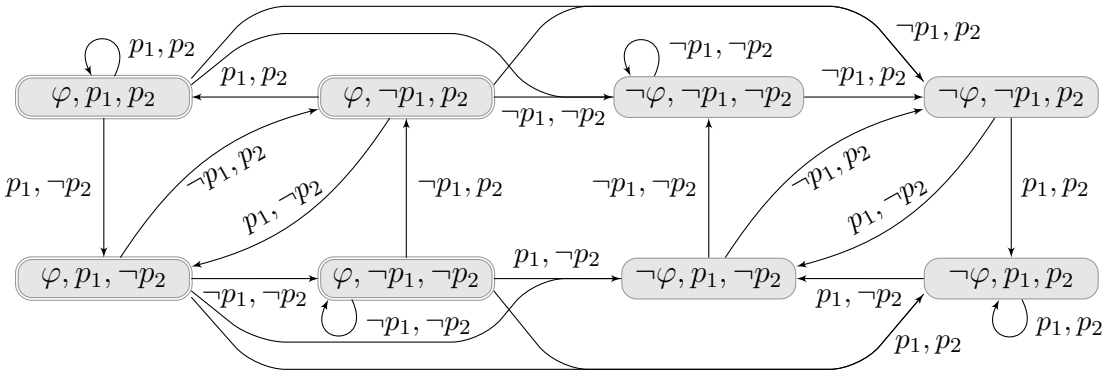


Figure 2.4: Büchi automaton for $\mathbf{G}\,(p_2 \Leftrightarrow \mathbf{X}\,p_1)$

Refining SL lower bound

In order to do so, we take another approach for encoding the LTL formula, since our technique of encoding p_2 with $\exists x_3.\, \mathsf{assign}(\mathit{Gold}, x_3).\, \mathbf{X}^2 p_2$ is not compatible with getting a formula in SL[BG]. Instead, we will use a Büchi automaton encoding the formula; another agent, say Agent *Oak*, will be in charge of selecting states of the Büchi automaton at each step. Using the same trick as above in the game structure on the left of Figure 2.5, a strategy for Agent *Oak* can be seen as a mapping from $\mathbb{N}$ to states of the Büchi automaton. Our formula will ensure that this sequence of states is in accordance with the atomic propositions selected by the square agents in states a_i, and that it forms an accepting run of the Büchi automaton.

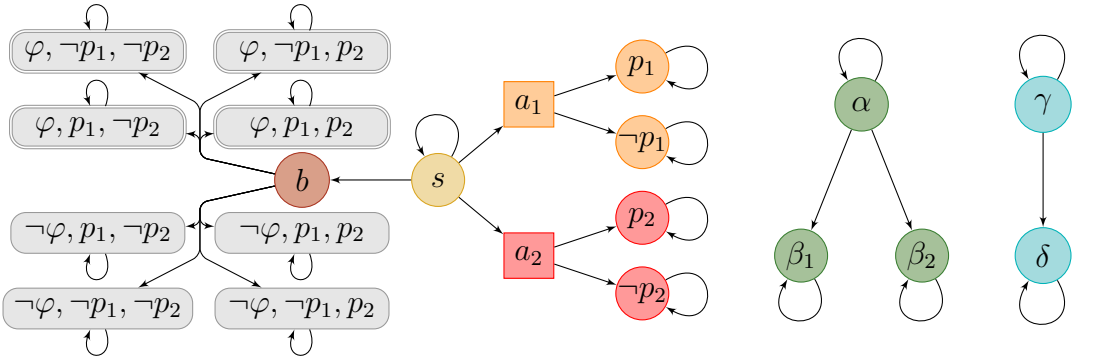


Figure 2.5: The concurrent game for the reduction to SL[BG] model checking.

For our example, an eight-state Büchi automaton associated with the (LTL part of the) QLTL formula is depicted on Figure 2.4. Notice that smaller automata exist for this property (for instance, the four states on the right could be merged into a single one), but for technical reasons in our construction, we require that each state of the Büchi automaton corresponds to a single valuation of the atomic propositions, hence the number of states must be a multiple of $2^{|\mathsf{AP}|}$. Accordingly, we augment our game structure of Figure 2.3 with eight extra states, as depicted on the left of Figure 2.5. Again, a strategy of Agent *Oak* (controlling state b) defines a sequence of states of the Büchi automaton.

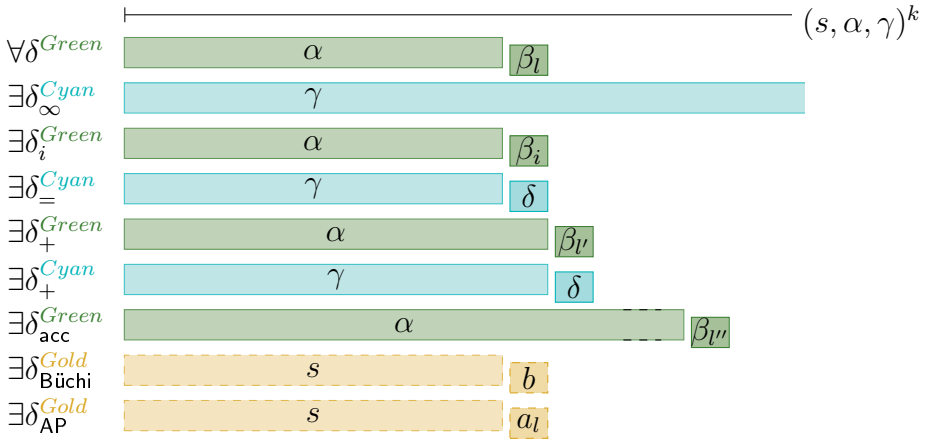


Figure 2.6: Visualization of the strategies selected by Ψ_{aux} on history $(s, \alpha, \gamma)^k$.

It then remains to "synchronise" the run of the Büchi automaton with the valuations of the atomic propositions, selected by the agents controlling the square states. This is achieved by taking the product of the game we just built with two extra one-agent structures, as depicted on the right of Figure 2.5. The product gives rise to a concurrent game, where one transition is taken simultaneously in the main structure and in the OliveGreen and BlueGreen structures. In this product, as long as Agent *Gold* remains in s and Agent *Green* remains in α, a strategy of Agent *Cyan* (controlling state γ) either remains in γ forever, or it can be characterised by a value $n \in \mathbb{N}$. Similarly, as long as Agent *Gold* remains in s and Agent *Cyan* remains in γ, a strategy of Agent *Green* (controlling state α) either loops forever in α, or can be uniquely characterised by a pair (k, p_l), where k is the number of times the loop over α is taken before entering state β_l corresponding to $p_l \in \mathsf{AP}$.

Our construction can then be divided in two steps:

- first, with any strategy of Agent *Green* (characterised by (k, p_l) for the interesting cases), we associate auxiliary strategies of Agents *Gold*, *Green* and *Cyan* satisfying certain properties, that can be enforced by an SL[BG] formula Ψ_{aux}. Figure 2.6 should help visualising the associated strategies. In particular, strategies δ^{Cyan}_{+}, δ^{Gold}_{+} and δ^{Green}_{+} characterise position $k+1$ (which will be useful for checking transitions of the Büchi automaton), while $\delta^{Gold}_{\mathsf{Büchi}}$ and $\delta^{Gold}_{\mathsf{AP}}$ are Agent *Gold* strategies that go either to the Büchi part or to the proposition part of the main part of the game.

- Then, using those strategies, we write another SL[BG] formula to enforce that the transitions of the Büchi automaton are correctly applied, following the valuations of the atomic propositions selected in the square states, and that an accepting state is visited infinitely many times.

The construction of the game structure $\mathcal{G}_\Phi$ depicted on Figure 2.5 is readily extended to any number of atomic propositions, and to any Büchi automaton. We now explain how we build our SL[BG] formula replacing Formula (2.2), and ensuring correctness of our reduction.

Specifying auxiliary strategies

We begin with simple constructions to select specific strategies, independently of the choice of the valuations for the atomic propositions. In the rest of this proof, we name strategy variables after the name of the agent who will be assigned that strategy; for instance, δ_{∞}^{Cyan} (that we use below) is the strategy of Agent $Cyan$ that always plays to γ. Our assignment operator will thus take only a strategy as argument, as the associated agent will be clear from the name of the strategy. Also, agents with no assigned strategies may have any strategy.

We consider the following formula, which we denote Ψ_{aux} in the sequel, in which we write β as a shorthand for $\bigvee_{p_i \in \mathsf{AP}} \beta_i$:

$$
\begin{aligned}
&\forall \delta^{Green}.\ \exists \delta_{\infty}^{Cyan}.\ \exists (\delta_i^{Green})_{p_i \in \mathsf{AP}}.\ \exists \delta_{=}^{Cyan}.\ \exists \delta_{+}^{Green}.\ \exists \delta_{+}^{Cyan}.\ \exists \delta_{\mathsf{acc}}^{Green}.\ \exists \delta_{\mathsf{Büchi}}^{Gold}.\\
&\exists \delta_{\mathsf{AP}}^{Gold}.\ \exists \delta_{+}^{Gold}.\ \exists \delta_{acc}^{Gold}.\ \forall \delta_{\forall}^{Gold}.\ \forall \delta_{\forall}^{Green}.\\
&\qquad \mathsf{assign}(\delta_{\forall}^{Gold}, \delta_{\forall}^{Green}, \delta_{\infty}^{Cyan}).\ \mathbf{G}\,(\neg\delta) && (\varphi_1)\\
&\wedge\\
&\qquad \mathsf{assign}(\delta_{\mathsf{Büchi}}^{Gold}, \delta^{Green}, \delta_{\infty}^{Cyan}).\ \left[\mathbf{G}\,(\alpha \wedge s)\ \vee\ (\alpha \wedge s)\,\mathbf{U}\,(\beta \wedge b)\right] && (\varphi_2)\\
&\wedge\\
&\qquad \mathsf{assign}(\delta_{+}^{Gold}, \delta_{+}^{Green}, \delta_{\infty}^{Cyan}).\ \left[\mathbf{G}\,(\alpha \wedge s)\ \vee\ (\alpha \wedge s)\,\mathbf{U}\,(\beta \wedge b)\right] && (\varphi_3)\\
&\wedge\\
&\qquad \mathsf{assign}(\delta_{acc}^{Gold}, \delta_{\mathsf{acc}}^{Green}, \delta_{\infty}^{Cyan}).\ \left[\mathbf{G}\,(\alpha \wedge s)\ \vee\ (\alpha \wedge s)\,\mathbf{U}\,(\beta \wedge b)\right] && (\varphi_4)\\
&\wedge\\
&\qquad \mathsf{assign}(\delta_{\mathsf{AP}}^{Gold}, \delta^{Green}, \delta_{\infty}^{Cyan}).\ \left[\mathbf{G}\,(\alpha \wedge s)\ \vee\ (\alpha \wedge s)\,\mathbf{U}\,(\bigvee_{p_i \in \mathsf{AP}} \beta_i \wedge a_i)\right] && (\varphi_5)\\
&\wedge\\
&\qquad \mathsf{assign}(\delta_{\mathsf{Büchi}}^{Gold}, \delta^{Green}, \delta_{=}^{Cyan}).\ \mathbf{G}\,(\beta \Leftrightarrow \delta) && (\varphi_6)\\
&\wedge\\
&\qquad \bigwedge\nolimits_{p_i \in \mathsf{AP}} \left[\mathsf{assign}(\delta_{\mathsf{Büchi}}^{Gold}, \delta_i^{Green}, \delta_{=}^{Cyan}).\ \mathbf{G}\,(\beta_i \Leftrightarrow \delta) \wedge \bigwedge\nolimits_{p_j \neq p_i} \mathbf{G}\,\neg\beta_j\right] && (\varphi_7)\\
&\wedge\\
&\qquad \mathsf{assign}(\delta_{\mathsf{Büchi}}^{Gold}, \delta^{Green}, \delta_{\infty}^{Cyan}).\ \mathbf{F}\,\beta \Rightarrow \begin{cases} \mathsf{assign}(\delta_{+}^{Gold}, \delta_{+}^{Green}, \delta_{\infty}^{Cyan}).\ \mathbf{F}\,\beta \\ \wedge \\ \mathsf{assign}(\delta_{\mathsf{Büchi}}^{Gold}, \delta_{+}^{Green}, \delta_{=}^{Cyan}).\ \mathbf{F}\,(\delta \wedge \neg\beta) \end{cases} && (\varphi_8)\\
&\wedge\\
&\qquad \mathsf{assign}(\delta_{\mathsf{Büchi}}^{Gold}, \delta^{Green}, \delta_{\infty}^{Cyan}).\ \mathbf{F}\,\beta \Rightarrow \begin{cases} \mathsf{assign}(\delta_{acc}^{Gold}, \delta_{\mathsf{acc}}^{Green}, \delta_{\infty}^{Cyan}).\ \mathbf{F}\,\beta \\ \wedge \\ \mathsf{assign}(\delta_{\mathsf{Büchi}}^{Gold}, \delta_{\mathsf{acc}}^{Green}, \delta_{=}^{Cyan}).\ \mathbf{F}\,(\delta \wedge \neg\beta) \end{cases} && (\varphi_9)\\
&\wedge\\
&\qquad \mathsf{assign}(\delta_{+}^{Gold}, \delta_{+}^{Green}, \delta_{+}^{Cyan}).\ \mathbf{G}\,(\neg s \Leftrightarrow \beta \Leftrightarrow \delta) && (\varphi_{10})\\
&\wedge\\
&\qquad \Big(\left[\mathsf{assign}(\delta_{\mathsf{Büchi}}^{Gold}, \delta_{\forall}^{Green}, \delta_{=}^{Cyan}).\ \mathbf{F}\,(\delta \wedge \alpha)\right] \Rightarrow\\
&\qquad\qquad \left[\mathsf{assign}(\delta_{+}^{Gold}, \delta_{\forall}^{Green}, \delta_{+}^{Cyan}).\ \alpha\,\mathbf{U}\,\delta\right]\Big) && (\varphi_{11})
\end{aligned}
$$

We now explain how this formula holds true in the product game of Figure 2.5,

whichever strategies are assigned to the square agents and to Agent *Oak*. Actually, it only associates, with each strategy δ^{Green} (quantified universally at the beginning of the formula) a set of strategies that "synchronize" with δ^{Green}:

(i) strategy δ^{Cyan}_{∞} always plays to γ, whatever the history of the game (Formula (φ_1));

(ii) strategy $\delta^{Gold}_{\mathsf{Büchi}}$ must be such that, for all ρ of the form $(s,\alpha,\gamma)^j$, it holds $\delta^{Gold}_{\mathsf{Büchi}}(\rho) = s$ as long as $\delta^{Green}(\rho) = \alpha$, and $\delta^{Gold}_{\mathsf{Büchi}}(\rho) = b$ when (if ever) $\delta^{Green}(\rho) \neq \alpha$ (Formula (φ_2));

(iii) with similar ideas, strategy δ^{Gold}_{+} must be such that, for all ρ of the form $(s,\alpha,\gamma)^j$, it holds $\delta^{Gold}_{+}(\rho) = s$ as long as $\delta^{Green}_{+}(\rho) = \alpha$, and $\delta^{Gold}_{+}(\rho) = b$ when (if ever) $\delta^{Green}_{+}(\rho) \neq \alpha$ (Formula (φ_3));

(iv) also, strategy δ^{Gold}_{acc} must be such that, for all ρ of the form $(s,\alpha,\gamma)^j$, it holds $\delta^{Gold}_{acc}(\rho) = s$ as long as $\delta^{Green}_{\mathsf{acc}}(\rho) = \alpha$, and $\delta^{Gold}_{acc}(\rho) = b$ when (if ever) $\delta^{Green}_{\mathsf{acc}}(\rho) \neq \alpha$ (Formula (φ_4));

(v) similarly, strategy $\delta^{Gold}_{\mathsf{AP}}(\rho) = s$ when $\delta^{Green}(\rho) = \alpha$, and $\delta^{Gold}_{\mathsf{AP}}(\rho) = a_i$ when (if ever) $\delta^{Green}(\rho) = \beta_i$ (Formula (φ_5));

(vi) similarly, strategy $\delta^{Cyan}_{=}(\rho) = \gamma$ as long as $\delta^{Green}(\rho) = \alpha$, and $\delta^{Cyan}_{=}(\rho) = \delta$ when (if ever) $\delta^{Green}(\rho) = \beta_i$ for some i (Formula (φ_6) combined with item (ii));

(vii) using similar ideas, it must be the case that $\delta^{Green}_{i}(\rho) = \alpha$ as long as $\delta^{Cyan}_{=}(\rho) = \gamma$, and $\delta^{Green}_{i}(\rho) = \beta_i$ when (if ever) $\delta^{Cyan}_{=}(\rho) = \delta$ (Formula (φ_7)). Combining this with item (vi), it must be the case that $\delta^{Green}_{i}(\rho) = \alpha$ as long as $\delta^{Green}(\rho) = \gamma$, and $\delta^{Green}_{i}(\rho) = \beta_i$ when (if ever) $\delta^{Green}(\rho) \neq \alpha$;

(viii) if $\delta^{Green}(\rho) = \beta_i$ for some $\rho = (s,\alpha,\gamma)^j$ and for some i, then $\delta^{Green}_{+}(\rho') = \beta_l$ for some $\rho' = (s,\alpha,\gamma)^k$ and some l. Moreover, the last part of Formula (φ_8) imposes that $k > j$;

(ix) strategy $\delta^{Green}_{\mathsf{acc}}$ satisfies the same condition as above (possibly for a different value of k) (Formula (φ_9));

(x) strategy $\delta^{Cyan}_{+}(\rho) = \gamma$ as long as $\delta^{Green}_{+}(\rho) = \alpha$, and $\delta^{Cyan}_{+}(\rho) = \delta$ when (if ever) $\delta^{Green}_{+}(\rho) = \beta_i$ for some i (Formula (φ_{10})). Similarly, strategy $\delta^{Gold}_{+}(\rho) = s$ as long as $\delta^{Green}_{+}(\rho) = \alpha$, and $\delta^{Gold}_{+}(\rho) = \neg\alpha$ when (if ever) $\delta^{Green}_{+}(\rho) = \beta_i$ for some i;

(xi) finally, Formula (φ_{11}) imposes that δ^{Cyan}_{+} plays δ (for the first time) exactly one step after δ^{Green} has played β (for the first time). This also imposes the same property for the first time at which δ^{Green}_{+} plays β_l. By item (viii) we know that δ^{Cyan}_{+} plays δ (for the first time) after δ^{Green}. Assume δ^{Cyan}_{+} plays to δ on $(s,\alpha,\gamma)^k$ for the first time and δ^{Green} plays to a β_i on $(s,\alpha,\gamma)^j$. Take for $\delta^{Green}_{\forall}$ the strategy playing to β_i for the first time after $(s,\alpha,\gamma)^{j+1}$, then the first formula holds. If $k \neq j+1$ the second formula does not hold and neither does Formula (φ_{11}).

Figure 2.6 page 51 summarises the constraints imposed by the formulas above on the selected strategies.

Enforcing correct transitions

We now focus on the Büchi automaton simulation. We look for a strategy of Agent *Oak* that will mimic the run of the Büchi automaton, following the valuation of the atomic propositions selected by the square agents A_1 to A_n. We also require that the run of the Büchi automaton be accepting.

The formula Ψ enforcing these constraints is as follows [1], with Q the state space of the Büchi automaton and q_{ini} its initial state :

$$\forall \delta^{A_1}.\ \exists \delta^{A_2}.\ \ldots \forall \delta^{A_{n-1}}.\ \exists \delta^{A_n}.\ \exists \delta^{Oak}.\ \mathsf{assign}(\delta^{A_1}, \delta^{A_2}, \ldots, \delta^{A_{n-1}}, \delta^{A_n}, \delta^{Oak}, \delta^{Cyan}_{\infty}).\ \Psi_{\mathsf{aux}}$$

$$\bigwedge_{p_i,p_j \in \mathsf{AP}} \bigwedge_{q \in Q} (\mathsf{assign}(\delta^{Gold}_{\mathsf{Büchi}}, \delta^{Green}_{i})\mathbf{F}\, q) \Leftrightarrow (\mathsf{assign}(\delta^{Gold}_{\mathsf{Büchi}}, \delta^{Green}_{j})\mathbf{F}\, q) \qquad (\varphi_{12})$$

$$\wedge\ \mathsf{assign}(\delta^{Gold}_{\mathsf{Büchi}}, \delta^{Green})(\mathbf{X}\, b \Rightarrow \mathbf{X}^{\,2} q_{ini}) \qquad (\varphi_{13})$$

$$\wedge\ \wedge_{q \in Q}\, \mathsf{assign}(\delta^{Gold}_{\mathsf{Büchi}}, \delta^{Green}).\ \mathbf{F}\, q \Rightarrow \bigvee_{q' \in \mathsf{succ}(q)} \mathsf{assign}(\delta^{Gold}_{+}, \delta^{Green}_{+}).\ \mathbf{F}\, q' \qquad (\varphi_{14})$$

$$\wedge\ \mathsf{assign}(\delta^{Gold}_{acc}, \delta^{Green}_{\mathsf{acc}}).\ \bigvee_{q \in accept(Q)} \mathbf{F}\, q \qquad (\varphi_{15})$$

$$\wedge\ \wedge_{p_i \in \mathsf{AP}} \Big(\big(\mathsf{assign}(\delta^{Gold}_{\mathsf{AP}}, \delta^{Green}).\ \mathbf{F}\, \neg p_i\big) \Rightarrow \big(\mathsf{assign}(\delta^{Gold}_{\mathsf{Büchi}}, \delta^{Green}).\ \bigvee_{q \in Q | p_i \notin \mathsf{labels} q} \mathbf{F}\, q\big)\Big) \qquad (\varphi_{16})$$

$$\wedge\ \bigwedge_{p_i \in \mathsf{AP}} \Big(\big(\mathsf{assign}(\delta^{Gold}_{\mathsf{AP}}, \delta^{Green}).\ \mathbf{F}\, p_i\big) \Rightarrow \big(\mathsf{assign}(\delta^{Gold}_{\mathsf{Büchi}}, \delta^{Green}).\ \bigvee_{q \in Q | p_i \in \mathsf{labels} q} \mathbf{F}\, q\big)\Big) \qquad (\varphi_{17})$$

We now analyze formula Ψ:

(xii) Formula (φ_{12}) requires that strategy δ^{Oak} returns the same move after any history of the form $(s, \alpha, \gamma)^k (b, \beta_i, \gamma)$, whichever β_i has been selected by δ^{Green};

(xiii) Formula (φ_{13}) ensures the initial state in the run of the Büchi automaton. The universal quantification of δ^{Green} force δ^{Oak} to preventively chose the appropriate state while items (i), (ii) and (xii) ensure that the outcome is of the correct shape according to our encoding, namely $(s, \alpha, \gamma)^j (b, \beta_i, \gamma)$ for some i;

(xiv) Items (iii), (x) and (xi) ensure that the outcome of δ^{Gold}_{+}, δ^{Green}_{+} and δ^{Cyan}_{∞} loops one more time on (s, α, γ) than the outcome of $\delta^{Gold}_{\mathsf{Büchi}}$, δ^{Green} and δ^{Cyan}_{∞}. Formula (φ_{14})

[1] We notice that Ψ is not syntactically in SL[BG], as some assignments appear before quantifications in Ψ_{aux}. However, quantifiers in Ψ_{aux} could be moved before the assignments of Ψ.

then additionally requires that two consecutive states of the run of the Büchi automaton indeed correspond to a transition.

(xv) Similarly, items (iv) and (ix) ensure that the outcome of $\delta_{acc}^{Gold}\,\delta_{\mathsf{acc}}^{Green}$ and δ_{∞}^{Cyan} loops more time on (s,α,γ) than the outcome of $\delta_{\mathsf{Büchi}}^{Gold}$, δ^{Green} and δ_{∞}^{Cyan}. Formula (φ_{15}) then states that for any position (selected by δ^{Green}), there exists a later position (given by $\delta_{\mathsf{acc}}^{Green}$) at which the run of the Büchi automaton visits an accepting state.

(xvi) Items (ii) and (v) ensure that $\delta_{\mathsf{AP}}^{Gold}$ and $\delta_{\mathsf{Büchi}}^{Gold}$ loop the same number of time on (s,α,γ). Formulas (φ_{16}) and (φ_{17}) then constrain the state of the Büchi automaton to correspond to the valuation of the atomic propositions selected. Because of the universal quantification over δ^{Green}, this property will be enforced at all positions and for all atomic propositions;

Correctness of the reduction

It remains to prove the correctness of the construction. For this we establish a correspondence between words over 2^{AP} and valuations of domain $\{\delta^{A_1},\dots,\delta^{A_n}\}$. The correspondence follows the idea described at the beginning of the proof: a word $w\colon \mathbb{N}\to 2^{\mathsf{AP}}$ is in correspondence with the valuation χ_w where

$$\forall k\in\mathbb{N}\ \forall 1\leq i\leq n \qquad p_i\in w(k) \quad\Leftrightarrow\quad \chi_w\big(\delta^{A_i}\big((s,\alpha,\gamma)^k.(a_i,\beta_i,\gamma)\big)\big)=p_i$$

In the following we write Ψ_{noAP} for the sub-formula corresponding to the part of Ψ without the quantifiers coding the atomic propositions $\forall\delta^{A_1}.\ \exists\delta^{A_2}.\ \dots\forall\delta^{A_{n-1}}.\ \exists\delta^{A_n}$. We also denote Φ_{LTL} the LTL part of the QLTL formula Φ. The lemmas below state the correctness of the construction in two steps. Their proofs, which can be found in the annex (page 58), are based on the correspondence described above.

Lemma 2.16. *A word w satisfies Φ_{LTL} if and only if $\mathcal{G},(s,\alpha,\gamma)\models_{\chi_w}\Psi_{noAP}$ with χ_w the valuation corresponding to w.*

Lemma 2.17. *Formula Φ is satisfiable if and only if Formula Ψ holds true in state (s,α,γ) of the game $\mathcal{G}_\Phi$.*

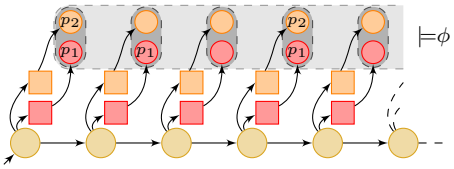
The Lemma 2.16 handles the correctness of the construction based on the correspondence described above between words and valuations. In a standard fashion, the Lemma 2.17 extends Lemma 2.16 by taking care of the quantifications over the atomic propositions. It also proves the correctness of our reduction and concludes the proof of Theorem 2.15.

Closing remarks

SL[BG] and several other fragments were defined in [40, 41] with the aim of getting more tractable fragments of SL. In particular, the authors advocate for the restriction to *behavioural strategies*: this forbids strategies that prescribe actions depending of what other

strategies would prescribe later on, or after different histories. Non-behavioural strategies are thus claimed to have limited interest in practice; moreover, they are suspected of being responsible for the non-elementary complexity of SL model checking. Our hardness result strengthens the latter claim, as SL[BG] is known for not having behavioural strategies. In the second part of the thesis, we will study behavioural strategies and more generally dependencies in SL[BG].

We had to rely on a Büchi automaton instead of directly using the original LTL formula directly in the SL[BG] formula. This is because we need to evaluate the formula not on a real path of our game structure, but on a sequence of "unions" of states. The figure on the right represents this situation for the game structure of Figure 2.3: the path on which the LTL formula is evaluated is given by the orange and red circle states, which define the valuations for p_1 and p_2.



2.4 Conclusion

<table>
<tr><th colspan="2" rowspan="2"></th><th colspan="2">Model Checking</th><th rowspan="2">Satisfiability</th></tr>
<tr><th>Formula</th><th>Data</th></tr>
<tr><td rowspan="2">SL</td><td>Upper Bound</td><td colspan="2">NONELEMENTARY [39]</td><td rowspan="3">UNDECIDABLE</td></tr>
<tr><td>Lower Bound</td><td>TOWER</td><td>PH-hard</td></tr>
<tr><td rowspan="2">SL[NG] and SL[BG]</td><td>Upper Bound</td><td colspan="2">In $(k+1)$-EXPTIME for k quantifier alternations [39]</td></tr>
<tr><td>Lower Bound</td><td>TOWER</td><td>PH-hard</td><td></td></tr>
<tr><td rowspan="2">SL[1G]</td><td>Upper Bound</td><td colspan="3" rowspan="2">2-EXPTIME-complete [38]
(PTIME-complete for the model checking data complexity)</td></tr>
<tr><td>Lower Bound</td></tr>
</table>

Table 2.1: Complexities of SL and its fragments.

We can see on Table 2.1 that SL[NG] and SL[BG] model checking complexities are high while the satisfiabilities are outright undecidable. The TOWER lower bound we developed in Section 2.3.2 for SL[BG] proves that we cannot do much better. It would be interesting to close the gap in the data complexity between the NONELEMENTARY upper bound and the PH lower bound. The SL model checking algorithm proceeds by a bottom up induction using the state space of the game in the basic case; there may exists a potential solution in a similar algorithm that proceed without using the game in the basic case.

To regain a decent complexity (if we may call 2-EXPTIME decent), the only solution

is to heavily restrict the logic. Satisfiability of SL[1G], the fragment with a unique objective, is *2*-EXPTIME-complete. While never proved formally, the technics developed by Mogavero, Murano, Perelli and Vardi in [38] for the satisfiablity problem may be adapted to get a *2*-EXPTIME algorithm for the model checking problem. We will also provide our own algorithm in Chapter 5.

2.A Annex

Lemma 2.16. *A word w satisfies Φ_{LTL} if and only if $\mathcal{G}, (s, \alpha, \gamma) \models_{\chi_w} \Psi_{noAP}$ with χ_w the valuation corresponding to w.*

Proof. Write $w := (w_j)_{j\in\mathbb{N}}$ for the word $w : \mathbb{N} \to 2^{AP}$, $\mathcal{N}$ for the Büchi automaton associate with Φ_{LTL} and $\Delta_{\mathcal{N}}$ for its transition function.

Proof of left-to-right implication Assume that $w \in \mathcal{L}(\mathcal{N})$, then there exists a path π in $\mathcal{N}$ witnesses that w is accepted by $\mathcal{N}$, i.e. $\pi(1)$ is the initial state of $\mathcal{N}$, $\pi \in \mathcal{L}(\mathcal{N})$ and

$$\forall j \in \mathbb{N} \qquad \Delta_{\mathcal{N}}(\pi(j), w(j)) = \pi(j+1) \tag{2.3}$$

We define a strategy δ^{Oak} for Agent *Oak*. For any k, we set

$$\delta^{Oak}((s, \alpha, \gamma)^j) := \pi(j)$$

We can then follow the choices explained in Section 2.3.2 to complete χ_w into a complete valuation χ that satisfies the goals of Ψ_{Aux}. It remains to check that χ satisfies the Formulas (φ_{12}) to (φ_{17}). Formula (φ_{12}) holds de facto by the choices made in Ψ_{Aux}. π is, by definition, a proper path of $\mathcal{N}$ so by items (xiii) and (xiv), Formulas (φ_{13}) and (φ_{14}) must hold. Similarly, as π is accepting and because of item (xv), Formula φ_{15} must also hold. Finally, by construction of δ^{Oak} and as explained in item (xvi), Formulas (φ_{16}) and (φ_{17}) must be true.

Proof of right-to-left implication For the converse implication, we assume that $w \notin \mathcal{L}(\mathcal{N})$, and towards a contradiction we further assume that

$$\mathcal{G}, (s, \alpha, \gamma) \models_{\chi_w} \Psi_{noAP} \tag{2.4}$$

Fix a strategy δ^{Oak} for Agent *Oak* and let χ be a complete valuation resulting from the interplay between the quantifications that extend χ_w and use δ^{Oak}. First, as described in Section 2.3.2, the behaviour of all strategies existentially quantified in Ψ_{Aux} is imposed by the goals within Ψ_{Aux} and the universally quantified strategies within Ψ_{Aux} (namely δ^{Green}, $\delta^{Gold}_{\forall}$ and $\delta^{Green}_{\forall}$). If the strategies of χ do not follow the rules imposed by Ψ_{Aux} and $\chi(\delta^{Green})$, $\chi(\delta^{Gold}_{\forall})$ and $\chi(\delta^{Green}_{\forall})$ then there must exist some goal in Ψ_{Aux} that does not hold. This means that Ψ_{noAP} does not hold either and we get a contradiction with Formula (2.4). So for the rest of the right-to-left implication, we assume that χ is working for the goals of Ψ_{Aux}.

Similarly to the left-to-right implication, the strategy δ^{Oak} describes a sequence of state $(\pi_j)_{j\in\mathbb{N}}$ in $\mathcal{N}$ through the following equality

$$\delta^{Oak}((s, \alpha, \gamma)^j) = \pi(j)$$

Items (xiii) and (xiv) ensure that π is a proper path, starting in the initial state q_{ini} of $\mathcal{N}$. Items (xv) makes sure π is accepted by $\mathcal{N}$. Finally, item (xvi) establishes a

correspondence between the word w and the path π. Combining items (xiii) to (xvi), we get that w is accepted by $\mathcal{N}$ which contradicts the hypothesis we made at the start of the implication. So the hypothesis $w \notin \mathcal{L}(\mathcal{N})$ and the Formula (2.4) are contradictory and the right to left implication must hold. □

Lemma 2.17. *Formula Φ is satisfiable if and only if Formula Ψ holds true in state (s, α, γ) of the game $\mathcal{G}_\Phi$.*

Proof. We proceed by induction on the number of atomic propositions. For this proof we write Ψ_i for the formula consisting of Ψ without the quantifications $\forall \delta^{A_1}, \dots Q_{i-1}\delta^{A_{i-1}}$ and Φ_i for the sub-formula of Φ that withdraws the quantifications $\forall p_1, \dots, Q_{i-1}p_{i-1}$. The induction aims to prove that

$$w \text{ satisfies } \Phi_i \quad \Leftrightarrow \quad \mathcal{G}_\Phi, (s, \alpha, \gamma) \models_{\chi_w} \Psi_i \tag{2.5}$$

where $w : \mathbb{N} \to 2^{p_1,\dots,p_{i-1}}$ is a word, χ_w is the valuation associate with w as described in Section 2.3.2, and with i ranging from n to 0.

The initial case ($i = n$, $\Phi_i = \Phi_{\mathsf{LTL}}$ and $\Psi_i = \Psi_{no\mathsf{AP}}$) has already be done by Lemma 2.16 and only the induction case remains. For the i^{th} step, by induction hypothesis Formula (2.5) holds for $i + 1$.Without loss of generality we assume that $Q_i = \exists$, the case $Q_i = \forall$ is similar. First, consider a word w and assume that w satisfies Φ_i. Then there exists a function $w_i : \mathbb{N} \to \{p_i, \emptyset\}$ such that $w' := w \cup w_i$ satisfies Φ_{i-1}. Then by induction hypothesis, $\mathcal{G}_\Phi, (s, \alpha, \gamma) \models_{\chi_{w'}} \Psi_i$. This implies that there exists a strategy $(\chi_{w'}(\delta^{A_i}))$ that extends χ_w in χ' and such that $\mathcal{G}_\Phi, (s, \alpha, \gamma) \models_{\chi'} \Psi_i$. We get the left-to-right implication of the i^{th} step of the induction. The right-to-left implication works similarly, and we use the working strategy δ^{A_i} to create a word $w_i : \mathbb{N} \to \{p_i, \emptyset\}$ handling the atomic proposition p_i. Having both implications, we deduce that the equivalence in Formula (2.5) holds. We can then conclude the induction. The case with $i = 0$ gives us the lemma. □

Chapter 3

Floating strategy logic

Consider a network of several clients, who may ask a central server for access to a shared resource. One (or several) user(s) can turn the clients on and off, and when turned on, each client then requests access to the resource. The server then has two objectives: one is to enforce that no two clients access the resource at the same time, whatever the clients do; the second property is that the clients must have a strategy that each of them can apply when turned on, and that ensures access to the resource (by collaborating with the server). This, in SL (with adapted syntax to make the formula readable), would be written

$$\exists \delta_{\mathsf{server}}.\ \exists \delta_{\mathsf{client}}.\ \text{ if server applies } \delta_{\mathsf{server}} \text{ then } \big[(\ \mathsf{always}\ \text{mutual exclusion}) \\ \wedge\ (\ \mathsf{always}\ (\ \text{if client applies } \delta_{\mathsf{client}} \text{ then } \mathsf{eventually}\ \text{access }).)\big]$$

Intuitively, when assigned, the strategy δ_{client} should start on an empty history as it has no knowledge of requests made by the other users to the server; however in SL the history of a strategy is preserved from the moment of its quantification. SL is limited in its expressiveness by its semantics forcing strategies to retain the history from the moment they were quantified.

The importance of this semantic choice has been under-considered in all SL related papers. We can indeed propose a second semantics making another choice, where the client has no knowledge about the history prior to being assigned a strategy. Both semantics are interesting but they model different phenomenons; as we will see the semantics also has an heavy impact on the model checking problem.

We propose a framework to handle this issue and study its algorithmic possibilities and limitations. We start in Section 3.1 by changing the definition of valuation translation and reworking SL semantics into a new logic FSL to take into account the aforementioned semantical subtlety. We then focus on the model checking problem. In Section 3.2 we show that FSL[NG] (the nested goal fragment of FSL) is undecidable while in Section 3.3 we highlight a decidable fragment FR-FSL[NG] whose expressiveness is between SL[BG] and FSL[NG].

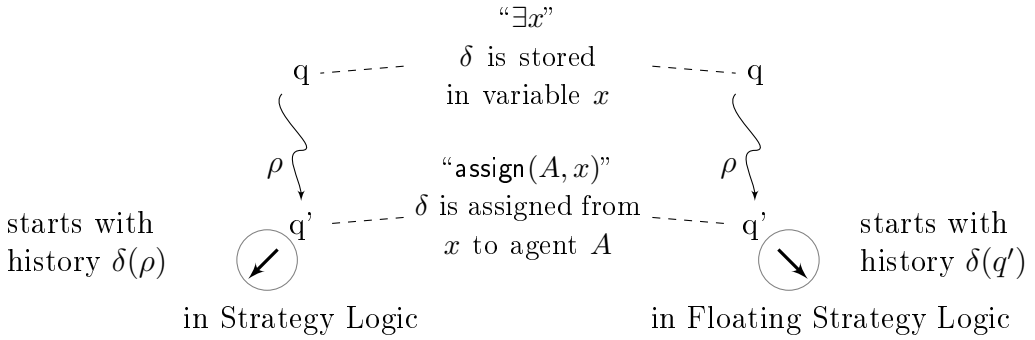


Figure 3.1: Strategy knowledge about history in Strategy Logic and Floating Strategy Logic.

3.1 Definition

One way to solve the issue described above is to translate[1] only the strategies of the agents but not the ones of the variables. This way we ensure that when an agent is assigned a strategy stored in a variable midway through the simulation, the said strategy has no knowledge of the history. Figure 3.1 illustrates the concept.

Definition 3.1 (Floating valuation translations)**.**
Given a valuation χ over a game $\mathcal{G}$, a set of variables $\mathcal{V}$ and a history ρ in $\mathcal{G}$, we define the floating valuation translation $\chi_{\underrightarrow{\rho}}$ by

$$\forall x \in \mathit{Agt} \cap \mathit{dom}(\chi),\ \chi_{\underrightarrow{\rho}}(x) := \chi(x)_{\overrightarrow{\rho}}$$
$$\forall x \in \mathcal{V} \cap \mathit{dom}(\chi),\ \chi_{\underrightarrow{\rho}}(x) := \chi(x)$$

The floating strategy logic FSL obey the same grammar as SL:

$$\mathsf{FSL} \ni \phi \ ::= \ \exists x.\phi \mid \mathsf{assign}(A, x).\phi \mid \phi \vee \phi \mid \neg\phi \mid \phi \mathbf{U}\, \phi \mid \mathbf{X}\, \phi \mid p$$

but the semantics of the temporal operators and quantifications are modified.

Firstly, we want the temporal operators to use floating translations. Fix a valuation χ with $\mathsf{Agt} \cup \mathsf{free}(\phi) \cup \mathsf{free}(\phi') \subseteq \mathsf{dom}(\chi)$; as before χ produces a unique outcome ρ. We write $\chi_{\underrightarrow{j}}$ for $\chi_{\underrightarrow{\rho_{\leq j}}}$ and let

$$\mathcal{G}, q \models_\chi \mathbf{X}\, \phi \quad \Leftrightarrow \quad \mathcal{G}, q \models_{\chi_{\underrightarrow{1}}} \phi$$
$$\mathcal{G}, q \models_\chi \phi' \mathbf{U}\, \phi \quad \Leftrightarrow \quad \exists k \in \mathbb{N}.\ \begin{cases} \mathcal{G}, q \models_{\chi_{\underrightarrow{k}}} \\ \forall j \in \mathbb{N}.\ 0 \leq j < k \Rightarrow \mathcal{G}, q \models_{\chi_{\underrightarrow{j}}} \phi' \end{cases}$$

Secondly, a strategy is not bound to its history and therefore not bound to its initial state: it may be quantified on a state q_0 and applied at state q_1, and thus must be

[1] "Translate" as in strategy translation, see Section 1.5.1.

defined on suffixes of q_0 but also on suffixes of q_1. To maintain a rigorous definition of FSL's semantics, we adapt the semantics of quantifications to add the initial state onto the history[2]. If $\mathsf{free}(\phi) \setminus \{x\} \subseteq \mathsf{dom}(\chi)$, then

$$\mathcal{G}, q \models_\chi \exists x.\phi \quad \Leftrightarrow \quad \exists \delta \in \mathsf{Strat}_\mathcal{G} \text{ such that } \mathcal{G}, q \models_{\chi[x \mapsto \delta_{\overrightarrow{q}}]} \phi$$

We call FSL[NG] the fragment of FSL using the same grammar as SL[NG], we recall the grammar of the flat fragment:

$$\begin{aligned} \mathsf{FSL[NG]}^\flat \ni \phi &::= \exists x.\phi \mid \forall x.\phi \mid \psi \\ \xi &::= \xi \vee \xi \mid \xi \wedge \xi \mid \beta \\ \beta &::= \mathsf{assign}(A, x).\beta \mid \varphi \\ \varphi &::= \varphi \vee \varphi \mid \neg\varphi \mid \varphi \mathbf{U}\, \varphi \mid \mathbf{X}\, \varphi \mid p \mid \beta \end{aligned}$$

The two semantics (with usual valuation translation or with floating valuation translation) coincide in SL[BG]. We recall that SL[BG] formulas are made by a series of quantifications followed by a boolean combination of goals (assignment followed by LTL operators) and that there is no assignment possible after the first temporal operator outside of closed sub-formulas. This means that when applying a formula $\phi \in \mathsf{SL[BG]}$ to a game $\mathcal{G}$ from a state q_{ini}, the strategies used in a SL[BG] formula are all assigned on q_{ini}. The strategies stored in the variables are never assigned outside q_{ini} and whether we translate them (as the usual valuation translation) or not (as in the floating valuation translation) does not matter, see Figure 3.2.

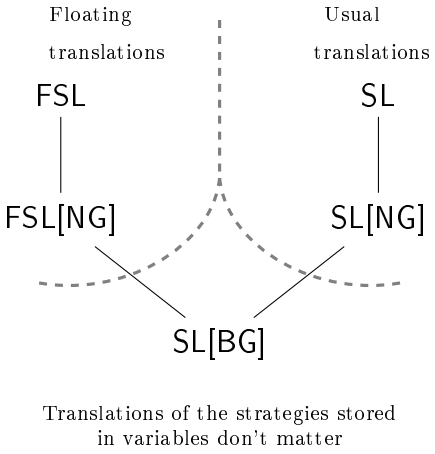


Figure 3.2: The two semantics.

Client/Server interaction. We return to the example given in the introduction of this chapter. Consider the formula (still with an adapted syntax to make the formula readable)

$$\begin{aligned} \exists \delta_{\mathsf{server}}.\ \exists \delta_{\mathsf{client}}.\ \text{ if server applies } \delta_{\mathsf{server}} \text{ then } \big[&(\text{ always mutual exclusion}) \\ \wedge\ &(\text{ always } (\text{ if client applies } \delta_{\mathsf{client}} \text{ then eventually access }).)\big] \end{aligned}$$

but within the semantics of FSL. When the client connects to the server, i.e. when δ_{client} is applied, he has no knowledge of the previous actions of the server (as δ_{client} has the current state for history). FSL's semantics therefore seem an appropriate answer to the problem exposed in the introduction to this chapter.

[2]This is a technical detail. A strategy usually starts on the empty history ε and is defined (at a meta level) relatively to an initial state. We just make the initial state appear clearly in the history. In FSL strategies are never evaluated on ε.

3.2 Undecidability of FSL[NG]

The high expressive power of these new semantics implies an undecidable model checking of FSL. In [8], we proved FSL model checking to be undecidable Here, we improve the proof to work in FSL[NG] and for formulas with a single quantifier alternation.

Theorem 3.2. *The model checking problem for closed FSL[NG] formulas is undecidable.*

Proof. We do a reduction from the halting problem for deterministic two-counter automata, known to be undecidable to FSL[NG] model checking.

Deterministic two-counter automata

Definition 3.3. *A deterministic two-counter automaton is a tuple* $\mathcal{M} = \langle S, E, s_0, s_h\rangle$ *where* S *is the state space,* s_0 *is an initial state of* $Q_{\mathcal{M}}$ *and* $s_h \in Q_{\mathcal{M}}$ *is a halting state.* $E : S \to \{c_1, c_2\} \times \{S \cup S \times S\}$ *is the transition function. Transitions of form* $E(s) = (c, s')$ *increment the counter* c *and go to* s'*, while transitions of form* $E(s) = (c, s', s'')$ *either go to* s' *if the counter* c *equals* 0 *or decrement* c *and go to* s'' *if* $c > 0$*.*

A configuration of $\mathcal{M}$ is a triple $(s, c_1, c_2) \in \mathsf{S} \times \mathbb{N} \times \mathbb{N}$, with the initial configuration being $(\mathsf{s}_0, 0, 0)$. A history is then a finite sequence of configurations that follows the rules of E, while a path is an infinite sequence, also following the rules of E.

The halting problem for two-counter automata takes as input an automaton $\mathcal{M}$ and asks if there exists a path in $\mathcal{M}$ that eventually reaches the halting state.

Theorem 3.4. *(Minsky [37])*
The halting problem for two-counter automata is undecidable.

The reduction

Let $\mathcal{M} = \langle \mathsf{S}, \mathsf{E}, \mathsf{s}_0, \mathsf{s}_h\rangle$ be a two-counter automaton. We start by building a turn-based game $\mathcal{G} := \langle \mathsf{AP}, \mathsf{Agt}, \mathsf{Q}, \mathsf{Act}, \Delta, \mathsf{labels}\rangle$ before explaining the idea behind our reduction.

- There are two agents in Agt: Decider (represented below in the ● states) and Checker (represented in the ■ states).
- The set of atomic propositions is $\mathsf{AP} := \{p_{main}, p_{halt}\} \cup \{p_s \mid s \in \mathsf{S}\} \cup \bigcup_{j\in\{1,2\}} \{p^j_\perp, p^j_●, p^j_■\}$.
- The state space is $\mathsf{Q} := \bigcup_{s\in\mathsf{S}} \{s_■, s_●\} \cup \bigcup_{j\in\{1,2\}} \{g^j_●, g^j_■, g^j_\perp\}$, with two states per element of S (we will call them the main states) and one gadget per counter, each having three states.
- For each $s \in S$, there is a transition from $s_■$ to $s_●$; from $s_■$ to $g^j_●$ for any $j \in [1, 2]$; and from $s_●$ to $s'_■$ whenever $E(s) = (\mathsf{c}_j, s')$ or $E(s) = (\mathsf{c}_j, s', s'')$ for some $j \in \{1, 2\}$ and some $s'' \in \mathsf{S}$.

  We also add transitions inside the gadgets: from $g^j_●$ to $g^j_■$, from $g^j_■$ to $g^j_●$, from $g^j_●$ to $g^j_\perp$, from $g^j_■$ to $g^j_\perp$ and a loop over $g^j_\perp$ for any $j \in [1, 2]$. See Figure 3.3

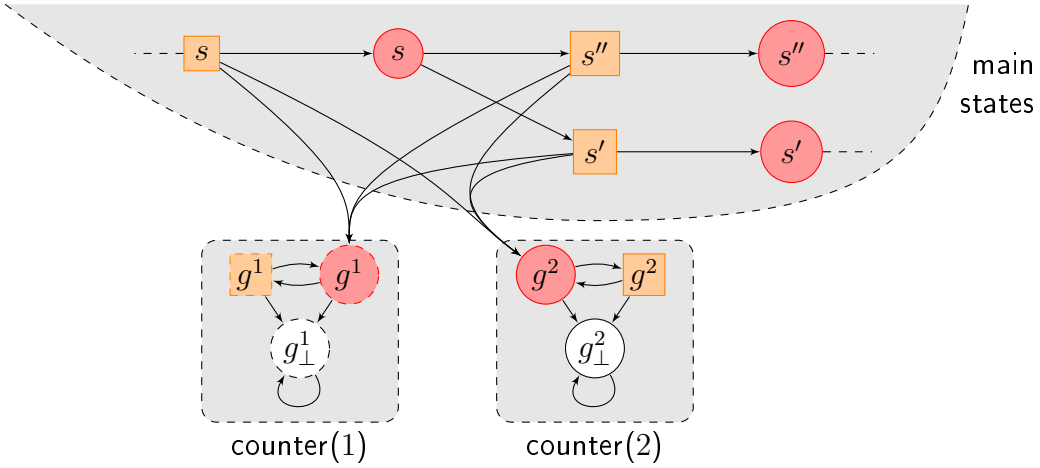


Figure 3.3: The game $\mathcal{G}$.

- We label each main square state $s_\square$ (for $s \in \mathsf{S}$) with $p_s^\square$ and each main state with p_{main}. Moreover both copies ($\square$ and $\bigcirc$) of the halting state s_h of $\mathcal{M}$ are labelled with p_{halt}. Finally, we label the three states of each gadget with an eponymous proposition.
- We also identify an initial state for $\mathcal{G}$: q_{ini} is the $\square$ copy of the initial state s_0 of $\mathcal{M}$.

The intuition is for Decider ($\bigcirc$) to recreate an accepting path of $\mathcal{M}$ in the "main part" of $\mathcal{G}$ and for Checker ($\square$) to check for any error. The main difficulty lies in the counters updates: for this task we will "store" a counter into a strategy and check it later on to ensure proper incrementation/decrementation. More precisely given a path $\rho = \rho^1.\rho^2\ldots$ in $\mathcal{M}$ we associate a strategy $\delta^\bigcirc_{main}$ for Decider that will try to create the path $\pi := \rho^1_\square.\rho^1_\bigcirc\,\rho^2_\square.\rho^2_\bigcirc\ \ldots$ and encode the first (resp. second) counter at step j through the number of time it loops on $g^1_\bigcirc.g^1_\square$ (resp. $g^2_\bigcirc.g^2_\square$) after history $\rho^1_\square.\rho^1_\bigcirc\ldots\rho^j_\square.g^1_\bigcirc$ (resp. $\rho^1_\square.\rho^1_\bigcirc\ldots\rho^j_\square.g^2_\bigcirc$). For this encoding to work, we need to ensure that $\delta^\bigcirc_{main}$ satisfies the initial conditions on the counter, deals correctly with the zero-test edges and updates the counters correctly.

The formula

Before going into details, we give the main formula so the reader can see the order in the quantifications.

$$\phi := \exists x^\bigcirc_{main}.\exists x^\square_{\infty}.\exists x^\square_{\to g^1}.\exists x^\square_{\to g^2}.\exists x^{\bigcirc,\square}_{loop}.\forall x^\square_{Count}.\begin{cases}\psi_{code}\wedge\psi_{init}\wedge\psi_{zero}\\ \wedge\psi_{Accept}\wedge\psi_{Update}\end{cases}$$

Each sub-formula serves a specific purpose and we provide the definitions below with some comments before proving the correctness of the reduction. Within the name of each

variable we make obvious the intended agent. The x_{loop} variable will be assigned to both agents, as allowed by FSL syntax. It simplifies the formula and avoids adding another variable, the result however still holds if we forbid strategy sharing between two agents. To shorten the formulas, we abbreviate "Decider" to "Dec" and "Checker" to "Che".

Preliminary work

We start by some preliminary work. The formula ψ_{code} ensures that any realisation of $\exists x_{\infty}^{\square}$ must always play to the main states and that any realisation of $\exists x_{\to g^j}^{\square}$ from the $s_{\square}$ states must play to $g_{\bigcirc}^j$. Formally, $\psi_{code} := \psi_{code}^1 \wedge \psi_{code}^2 \wedge \psi_{code}^3$ with

$$\psi_{code}^1 := \mathsf{assign}(Dec, x_{main}^{\bigcirc}; Che, x_{\infty}^{\square}).\mathbf{G}\, p_{main}$$
$$\psi_{code}^2 := \mathsf{assign}(Dec, x_{main}^{\bigcirc}; Che, x_{\infty}^{\square}).\mathbf{G}\,\big[(\bigvee_{s\in \mathsf{S}} p_s^{\square}) \Rightarrow \mathsf{assign}(Che, x_{\to g^1}^{\square})\mathbf{X}\, g_{\bigcirc}^1\big]$$
$$\psi_{code}^3 := \mathsf{assign}(Dec, x_{main}^{\bigcirc}; Che, x_{\infty}^{\square}).\mathbf{G}\,\big[(\bigvee_{s\in \mathsf{S}} p_s^{\square}) \Rightarrow \mathsf{assign}(Che, x_{\to g^2}^{\square})\mathbf{X}\, g_{\bigcirc}^2\big]$$

Trivially, we have

Proposition 3.5. *Given four strategies $\delta_{main}^{\bigcirc}$, $\delta_{\infty}^{\square}$, $\delta_{\to g^1}^{\square}$ and $\delta_{\to g^2}^{\square}$, we have that*

$$\mathcal{G}, q_{ini} \models_{\{x_{main}^{\bigcirc}\to\delta_{main}^{\bigcirc};\ x_{\infty}^{\square}\to\delta_{\infty}^{\square};\ x_{\to g^1}^{\square}\to\delta_{\to g^1}^{\square};\ x_{\to g^2}^{\square}\to\delta_{\to g^2}^{\square}\}} \bigwedge_{j\in[1;3]} \psi_{code}^j$$

if and only if the outcome of $\delta_{main}^{\bigcirc}$ and $\delta_{\infty}^{\square}$ follows the main states and for $j \in \{1,2\}$ and any $s_{\square}$ state ($s \in S$), $\delta_{\to g^j}^{\square}$ on history $s_{\square}$ plays to g^j.

Consider a strategy $\delta_{main}^{\bigcirc}$ intended for the variable $x_{main}^{\bigcirc}$. It defines a (unique) sequence $(s^i, c_1^i, c_2^i)_{i\in\mathbb{N}}$ of $(\mathsf{S}\times\mathbb{N}\cup\{\infty\}\times\mathbb{N}\cup\{\infty\})^{\mathbb{N}}$ where $s^i = \delta_{main}^{\bigcirc}(s_{\square}^1.s_{\bigcirc}^1\ldots s_{\bigcirc}^{i-1})$ and for $j\in\{1,2\}$, c_j^i is the number of time $\delta_{main}^{\bigcirc}$ loops on $g_{\bigcirc}^j.g_{\square}^j$ after history $s_{\square}^1.s_{\bigcirc}^1\ldots s_{\square}^i.g_{\bigcirc}^j$. We use the sequence to encode the counter values.

The initial conditions

We define the formula ψ_{init} below to ensure that both counters are initialised to zero.

$$\psi_{init} := \bigwedge_{j\in\{1,2\}} \mathsf{assign}(Dec, x_{main}^{\bigcirc}; Che, x_{\to g^j}^{\square}).\mathbf{X}^{\,2} g_{\perp}^j$$

Proposition 3.6. *Fix four strategies $\delta_{main}^{\bigcirc}$, $\delta_{\infty}^{\square}$, $\delta_{\to g^1}^{\square}$ and $\delta_{\to g^2}^{\square}$ such that they satisfy $\bigwedge_{j\in[1;3]}\psi_{code}^j$. Write $(s^i, c_1^i, c_2^i)_{i\in\mathbb{N}}$ for the sequence associated with $\delta_{main}^{\bigcirc}$ and $\delta_{\infty}^{\square}$ in $(S\times\mathbb{N}\cup\{\infty\}\times\mathbb{N}\cup\{\infty\})^{\mathbb{N}}$. Then*

$$\mathcal{G}, q_{ini} \models_{\{x_{main}^{\bigcirc}\to\delta_{main}^{\bigcirc};\ x_{\infty}^{\square}\to\delta_{\infty}^{\square};\ x_{\to g^1}^{\square}\to\delta_{\to g^1}^{\square};\ x_{\to g^2}^{\square}\to\delta_{\to g^2}^{\square}\}} \psi_{init} \quad\Leftrightarrow\quad c_1^1 = c_2^1 = 0.$$

Proof. Assume

$$\mathcal{G}, q_{ini} \models_{\{x^{\bullet}_{main}\to\delta^{\bullet}_{main};\ x^{\square}_{\infty}\to\delta^{\square}_{\infty};\ x^{\square}_{\to g^1}\to\delta^{\square}_{\to g^1};\ x^{\square}_{\to g^2}\to\delta^{\square}_{\to g^2}\}} \psi_{init} \tag{3.1}$$

Because of Proposition 3.5, the outcome of $\{x^{\bullet}_{main} \to \delta^{\bullet}_{main};\ x^{\square}_{\to g^1} \to \delta^{\square}_{\to g^1}\}$ starts as $q_{ini}.g^1_{\bullet}$, and because of Formula (3.1), it continues as $q_{ini}.g^1_{\bullet}.g^1_{\perp}$ and therefore $c^1_1 = 0$. The same idea can be applied to show that $c^1_2 = 0$. The other direction is similar: if Formula (3.1) does not hold, then $c^1_1 \neq 0$ or $c^1_2 \neq 0$. □

The zero tests

We define the formula ψ_{zero} below. Its role is to ensures that whenever the outcome of $\{Dec, x^{\bullet}_{main};\ Che, x^{\square}_{\infty}\}$ takes a zero-test edge on counter 1 (resp. 2) between step i and $i+1$, $\delta^{\bullet}_{main}$ encodes at step i the value 0 on the gadget 1(resp. 2). The fact that the counter encoded at step $i+1$ is 0 will be ensured in the incrementation/decrementation step later on.

$$\psi_{zero} := \bigwedge_{\substack{j,s,s' \text{ s.t.} \\ j\in\{1,2\} \\ E(s)=(\mathsf{c}_j,s',s'')}} \mathsf{assign}(Dec, x^{\bullet}_{main}; Che, x^{\square}_{\infty}).\ \mathbf{G} \begin{cases} (p^{\square}_{s} \wedge \mathbf{X}^{2} p^{\square}_{s'}) \Rightarrow \\ \mathsf{assign}(Che, x^{\square}_{\to g^j})(\mathbf{X}\, g^j_{\bullet} \wedge \mathbf{X}^{2} g^j_{\perp}) \end{cases}$$

Proposition 3.7. *Fix four strategies $\delta^{\bullet}_{main}$, $\delta^{\square}_{\infty}$, $\delta^{\square}_{\to g^1}$ and $\delta^{\square}_{\to g^2}$ such that they satisfy $\bigwedge_{j\in[1;3]} \psi^j_{code}$. Write $(s^i, c^i_1, c^i_2)_{i\in\mathbb{N}}$ for its associated sequence in $(S\times\mathbb{N}\cup\{\infty\}\times\mathbb{N}\cup\{\infty\})^{\mathbb{N}}$. Then*

$$\begin{aligned} &\mathcal{G}, q_{ini} \models_{\{x^{\bullet}_{main}\to\delta^{\bullet}_{main};\ x^{\square}_{\infty}\to\delta^{\square}_{\infty};\ x^{\square}_{\to g^1}\to\delta^{\square}_{\to g^1};\ x^{\square}_{\to g^2}\to\delta^{\square}_{\to g^2}\}} \psi_{zero} \\ \Leftrightarrow\ &\forall i \in \mathbb{N}\ \forall j \in \{1,2\}\ \forall s'' \in S \qquad E(s^i) = (\mathsf{c}_j, s^{i+1}, s'') \Rightarrow c^i_j = 0 \end{aligned}$$

Proof. Assume that

$$\mathcal{G}, q_{ini} \models_{\{x^{\bullet}_{main}\to\delta^{\bullet}_{main};\ x^{\square}_{\infty}\to\delta^{\square}_{\infty};\ x^{\square}_{\to g^1}\to\delta^{\square}_{\to g^1};\ x^{\square}_{\to g^2}\to\delta^{\square}_{\to g^2}\}} \psi_{zero} \tag{3.2}$$

and that there exists $s'' \in \mathsf{S}$, $i \in \mathbb{N}$ and $j \in \{1,2\}$ with $\mathsf{E}(s^i) = (\mathsf{c}_j, s^{i+1}, s'')$. After $2i$ steps, the outcome of $\{\delta^{\bullet}_{main};\ \delta^{\square}_{\infty}\}$ will satisfy $(p^{\square}_{s} \wedge \mathbf{X}^{2} p^{\square}_{s'})$. This means, by Formula (3.2), that at step $2i$, it must also satisfy $\mathsf{assign}(Che, x^{\square}_{\to g^j})(\mathbf{X}\, g^j_{\bullet} \wedge \mathbf{X}^{2} g^j_{\perp})$ which implies that $c^i_j = 0$ by definition of c^i_j.

The converse implication is done similarly by assuming that Formula (3.2) does not hold and deducing some i, j and s'' such that $\mathsf{E}(s^i) = (\mathsf{c}_j, s^{i+1}, s'')$ and $c^i_j \neq 0$. □

The path encoded reaches the halting state in $\mathcal{M}$

We set

$$\psi_{Accept} := \mathsf{assign}(Dec, x^{\bullet}_{main}; Che, x^{\square}_{\infty}).\mathbf{F}\, p_{halt}$$

The proof of the following proposition is straightforward thus omitted.

Proposition 3.8. *Fix two strategies $\delta^{\bullet}_{main}$ and $\delta^{\square}_{\infty}$, and write $(s^i, c^i_1, c^i_2)_{i\in\mathbb{N}}$ for the associated sequence of $\delta^{\bullet}_{main}$ and $\delta^{\square}_{\infty}$.*

$$\mathcal{G}, q_{ini} \models_{\{x^{\bullet}_{main}\to\delta^{\bullet}_{main};\ x^{\square}_{\infty}\to\delta^{\square}_{\infty}\}} \psi_{Accept} \quad\Leftrightarrow\quad \exists i \in \mathbb{N} \quad p_{halt} \in \mathit{labels}(s^i)$$

Storing counter values within strategies

To check the counter updates, we store them within a strategy using the specificity of FSL[NG]. First, for $j \in \{1, 2\}$ we define the following open formula

$$\psi^j_{Egal} := \mathsf{assign}(Che, x^{\square}_{\to g^j}).\mathbf{X}\,\mathsf{assign}(Che, x^{\square}_{Count}).\mathbf{F} \begin{cases} g^j_{\bullet} \\ \wedge \mathbf{X}\, g^j_{\perp} \\ \wedge \mathsf{assign}(Dec, x^{\bullet,\square}_{loop}) \begin{cases} \mathbf{X}\, \neg g^j_{\perp} \\ \wedge \mathbf{X}^2 g^j_{\perp} \end{cases} \end{cases}$$

We can compare the number of times two strategies $\delta^{\bullet}_{main}$ and $\delta^{\square}_{Count}$, intended respectively for the variables $x^{\bullet}_{main}$ and $x^{\square}_{Count}$, loop respectively on $g^j_{\bullet}$ and $g^j_{\square}$.

Proposition 3.9. *Fix $i \in \mathbb{N}$, $j \in \{1, 2\}$ and a history ρ that travels through the main states and finishes in a $p^{\square}_s$ states. Fix any four strategies $\delta^{\bullet}_{main}$, $\delta^{\square}_{\infty}$, $\delta^{\square}_{Count}$ and $\delta^{\bullet,\square}_{loop}$ where $\delta^{\bullet,\square}_{loop}$ always plays from $g^j_{\bullet}$ to $g^j_{\square}$. Write $(s^i, c^i_1, c^i_2)_{i\in\mathbb{N}}$ for the associated sequence of $\delta^{\bullet}_{main}$ and $\delta^{\square}_{\infty}$. Let*

$$\chi := \{Dec \to \delta^{\bullet}_{main};\ Che \to \delta^{\square}_{\infty};\ x^{\square}_{Count} \to \delta^{\square}_{Count};\ x^{\bullet,\square}_{loop} \to \delta^{\bullet,\square}_{loop}\}$$

Then

$$\mathcal{G}, g^j_{\bullet} \models_{\chi_{\underrightarrow{\rho}}} \psi^j_{Egal} \quad\Leftrightarrow\quad \delta^{\square}_{Count} \text{ loops } c^{|\rho|}_j \text{ times on } g^j_{\square}$$

Proof. Assume that $\delta^{\square}_{Count}$ loops $c^{|\rho|}_j$ times on $g^j_{\square}$. Then, reusing the $\chi_{\underrightarrow{\rho}}$ notation for the floating valuation translation, the outcome of $\{Dec \to \delta^{\bullet}_{Count};\ Che \to (\delta^{\square}_{main})_{\rho.g^j_{\bullet}\longrightarrow}\}$ from $g^j_{\bullet}$ will loop $c^{|\rho|}_j$ times on $g^j_{\bullet}.g^j_{\square}$ before seeing $g^j_{\bullet}$ then $g^j_{\perp}$ (as $\delta^{\bullet}_{main}$ stops looping after $c^{|\rho|}_j$ times). Hence

$$\mathcal{G}, g^j_{\bullet} \models_{\chi_{\underrightarrow{\rho}}} \mathsf{assign}(Che, x^{\square}_{\to g^j}).\mathbf{X}\left(\mathsf{assign}(Che, x^{\square}_{Count}).\mathbf{F}\,(g^j_{\bullet} \wedge \mathbf{X}\, g^j_{\perp})\right)$$

Moreover, if we assign $\delta_{loop}^{\bullet,\blacksquare}$ to $x_{loop}^{\bullet,\blacksquare}$ after $|\rho|$ steps then, because $\delta_{Count}^{\blacksquare}$ is also done looping and keeps its history, $\delta_{Count}^{\blacksquare}$ will send the outcome to $g_{\perp}^{j}$. Hence

$$\mathcal{G}, g_{\bullet}^{j} \models_{\chi_{\underrightarrow{\rho}}} \psi_{Egal}^{j}$$

and the $\Leftarrow$ implication of Proposition 3.9 holds. The other way is similar and Proposition 3.9 must hold. □

Test for incrementation and decrementation

Using formulas similar to ψ_{Egal}^{j}, we can test if the strategy stored in $x_{Count}^{\blacksquare}$ loops one more or one less time than the one stored in $x_{main}^{\bullet}$.

$$\psi_{Inc}^{j} := \mathsf{assign}(Che, x_{\rightarrow g^j}^{\blacksquare}).\ \mathbf{X}\,\mathsf{assign}(Che, x_{Count}^{\blacksquare}).\ \mathbf{F} \begin{cases} g_{\bullet}^{j} \wedge \mathbf{X}\, g_{\perp}^{j} \\ \wedge \mathsf{assign}(Dec, x_{loop}^{\bullet,\blacksquare}).\ \begin{cases} \mathbf{X}^{3} \neg g_{\perp}^{j} \\ \wedge\ \mathbf{X}^{4} g_{\perp}^{j} \end{cases} \end{cases}$$

$$\psi_{Dec}^{j} := \mathsf{assign}(Che, x_{\rightarrow g^j}^{\blacksquare}).\ \mathbf{X}\,\mathsf{assign}(Che, x_{Count}^{\blacksquare}).\ \mathbf{F} \begin{cases} g_{\blacksquare}^{j} \wedge \mathbf{X}\, g_{\perp}^{j} \\ \wedge \mathsf{assign}(Che, x_{loop}^{\bullet,\blacksquare}).\ \begin{cases} \mathbf{X}^{2} \neg g_{\perp}^{j} \\ \wedge\ \mathbf{X}^{3} g_{\perp}^{j} \end{cases} \end{cases}$$

Similarly to Proposition 3.9, we get

Proposition 3.10. *Fix $i \in \mathbb{N}$, $j \in \{1,2\}$ and a history ρ that travels through the main states and finishes in a $p_s^{\blacksquare}$ states. Fix any four strategies $\delta_{main}^{\bullet}$, $\delta_{\infty}^{\blacksquare}$, $\delta_{Count}^{\blacksquare}$ and $\delta_{loop}^{\bullet,\blacksquare}$ where $\delta_{loop}^{\bullet,\blacksquare}$ always plays from $g_{\bullet}^{j}$ to $g_{\blacksquare}^{j}$ and from $g_{\blacksquare}^{j}$ to $g_{\bullet}^{j}$. Write $(s^i, c_1^i, c_2^i)_{i\in\mathbb{N}}$ for the associated sequence of $\delta_{main}^{\bullet}$ and*

$$\chi := \{Dec \to \delta_{main}^{\bullet};\ Che \to \delta_{\infty}^{\blacksquare};\ x_{Count}^{\blacksquare} \to \delta_{Count}^{\blacksquare};\ x_{loop}^{\bullet,\blacksquare} \to \delta_{loop}^{\bullet,\blacksquare}\}$$

Then

$$\mathcal{G}, g_{\bullet}^{j} \models_{\chi_{\underrightarrow{\rho}}} \psi_{Inc}^{j} \quad \Leftrightarrow \quad \delta_{Count}^{\blacksquare} \text{ loops } c_j^{|\rho|} + 1 \text{ times on } g_{\blacksquare}^{j}$$
$$\mathcal{G}, g_{\bullet}^{j} \models_{\chi_{\underrightarrow{\rho}}} \psi_{Dec}^{j} \quad \Leftrightarrow \quad \delta_{Count}^{\blacksquare} \text{ loops } c_j^{|\rho|} - 1 \text{ times on } g_{\blacksquare}^{j}$$

Updating counter values

Using the work done above, we can now specify the formula encoding proper counters updates.

$$\psi_{Update} := \bigwedge_{j\in\{1,2\}} \mathsf{assign}(Dec, x_{main}^{\bullet}; Che, x_{\infty}^{\blacksquare}).\mathbf{G}\,(\psi_{=} \wedge \psi_{+} \wedge \psi_{-})$$

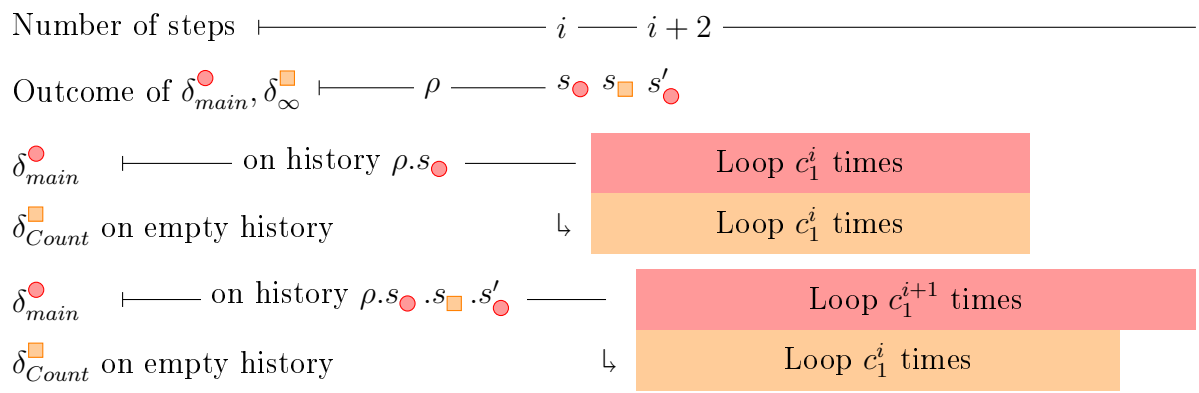


Figure 3.4: Testing for an incrementation of the first counter at step i.

where

$$\psi_+ := \bigwedge_{s,s' \mid \mathsf{E}(s)=(\mathsf{c}_j,s')} (p^{\square}_s \wedge \mathbf{X}^2 p^{\square}_{s'}) \Rightarrow \psi^j_{Egal} \Rightarrow \mathbf{X}^2 \psi^j_{Inc}$$

$$\psi_= := \bigwedge_{s,s' \mid \mathsf{E}(s)=(\mathsf{c}_j,s')} (p^{\square}_s \wedge \mathbf{X}^2 p^{\square}_{s'}) \Rightarrow \psi^j_{Egal} \Rightarrow \mathbf{X}^2 \psi^j_{Egal}$$

$$\psi_- := \bigwedge_{s,s' \mid \mathsf{E}(s)=(\mathsf{c}_j,s')} (p^{\square}_s \wedge \mathbf{X}^2 p^{\square}_{s'}) \Rightarrow \psi^j_{Egal} \Rightarrow \mathbf{X}^2 \psi^j_{Dec}$$

Figure 3.4 illustrates the idea. $\delta^{\square}_{Count}$ loops the same number of time when assigned after history $\rho.s_{\bullet}$ and after history $\rho.s_{\bullet}.s_{\square}.s'_{\bullet}$. This is due to the floating semantics of FSL and is not true in SL. The strategy $\delta^{\square}_{Count}$ can then be used to compare the number of times $\delta^{\bullet}_{main}$ loops after history $\rho.s_{\bullet}$ and after history $\rho.s_{\bullet}.s_{\square}.s'_{\bullet}$. This comparison allows us to check for proper update of a counter values.

Correctness of the reduction

This terminates the definitions of the intermediary formulas. Using the propositions provided in each part, we establish a correspondence between the existence of a working strategy $\delta^{\bullet}_{main}$ in $\mathcal{G}$ and the existence of an accepting path in $\mathcal{M}$. This is done through the following proposition.

Proposition 3.11. *There is an accepting path in $\mathcal{M}$ if and only if $\mathcal{G}, q_{ini} \models \phi$*

Proof. Assume there is an accepting path $(q^i, c^i_1, c^i_2)_{i\in\mathbb{N}}$. Let $\delta^{\bullet}_{main}$ be a strategy with associated sequence $(q^i, c^i_1, c^i_2)_{i\in\mathbb{N}}$ (intended for $x^{\bullet}_{main}$). Let also $\delta^{\square}_{\infty}$, $\delta^{\square}_{\to g^1}$, $\delta^{\square}_{\to g^2}$ and $\delta^{\bullet,\square}_{loop}$ be four strategies such that

$$\mathcal{G}, q_{ini} \models_\chi \psi_{code}$$

where

$$\chi := \{x^{\bullet}_{main} \to \delta^{\bullet}_{main};\ x^{\square}_{\infty} \to \delta^{\square}_{\infty};\ x^{\square}_{\to g^1} \to \delta^{\square}_{\to g^1};\ x^{\square}_{\to g^2} \to \delta^{\square}_{\to g^2};\ x^{\bullet,\square}_{loop} \to \delta^{\bullet,\square}_{loop}\}$$

Then, because $(q^i, c_1^i, c_2^i)_{i\in\mathbb{N}}$ is an accepting path and because of Propositions 3.5, 3.6, 3.7, 3.8, 3.9 and 3.10, we have that

$$\mathcal{G}, q_{ini} \models_\chi \psi_{init} \wedge \psi_{zero} \wedge \psi_{Accept} \wedge \psi_{Update}$$

For the reverse implication, assume there is no accepting path and fix any five strategies $\delta^{\bullet}_{main}$, $\delta^{\square}_{\infty}$, $\delta^{\square}_{\to g^1}$, $\delta^{\square}_{\to g^2}$ and $\delta^{\bullet,\square}_{loop}$. Write

$$\chi := \{x^{\bullet}_{main} \to \delta^{\bullet}_{main};\ x^{\square}_{\infty} \to \delta^{\square}_{\infty};\ x^{\square}_{\to g^1} \to \delta^{\square}_{\to g^1};\ x^{\square}_{\to g^2} \to \delta^{\square}_{\to g^2};\ x^{\bullet,\square}_{loop} \to \delta^{\bullet,\square}_{loop}\}$$

If $\mathcal{G}, q_{ini} \not\models_\chi \psi_{Code}$, trivially $\mathcal{G}, q_{ini} \not\models_\chi \psi_{Code} \wedge \psi_{init} \wedge \psi_{zero} \wedge \psi_{Accept} \wedge \psi_{Update}$. So we assume $\mathcal{G}, q_{ini} \models_\chi \psi_{Code}$.

Let $(q^i, c_1^i, c_2^i)_{i\in\mathbb{N}}$ be the sequence associated with $\delta^{\bullet}_{main}$. By hypothesis, $(q^i, c_1^i, c_2^i)_{i\in\mathbb{N}}$ cannot be an accepting path. There are two possibilities:

- First possibility, it is a path but it does not reach an accepting state. In this case, by Proposition 3.8, $\mathcal{G}, q_{ini} \not\models_\chi \psi_{Accept}$.
- Second possibility, $(q^i, c_1^i, c_2^i)_{i\in\mathbb{N}}$ does not correspond to a valid path.
  - If it fails a zero test then by Proposition 3.7 we have that $\mathcal{G}, q_{ini} \not\models_\chi \psi_{zero}$.
  - If it does not have the proper initial values for the counters, then by Proposition 3.6 we have that $\mathcal{G}, q_{ini} \not\models_\chi \psi_{init}$.
  - If it fails to update one of the counters, then by Propositions 3.9 and 3.10 we have that $\mathcal{G}, q_{ini} \not\models_\chi \psi_{Update}$.

In any case, $\delta^{\bullet}_{main}$ is not an adequate strategy and therefore $\mathcal{G}, q_{ini} \models \phi$. □

This last proposition proves the correctness of our reduction and therefore gives us Theorem 3.2. □

3.3 Decidability of FR-FSL[NG]

The undecidability of FSL[NG] model checking comes from the ability to compare strategy choices on multiple histories. One way to gain decidability on a fragment of FSL is to force a reassignment of all agents. This ensures that comparison of strategies can only be done on a common history: as we reassign all the agents at every reassignment, any two agents at any moment in the game and on any branch of the formula will always share a common history. Figure 3.5 shows the differences. With FSL we can correlate the choices of a strategy on the orange history with the choice on the red history by checking the outcome they produces in combination with an annex strategy on a fixed history (represented by the gold arrow on fixed history q). In SL the comparison of two strategies happens on the same history hence we cannot compare the choices made by a strategy on the orange and red histories. In this section, we define a fragment FR-FSL[NG] between FSL[NG] and FSL[BG] (which we recall coincides with SL[BG]) and prove that its model checking is decidable.

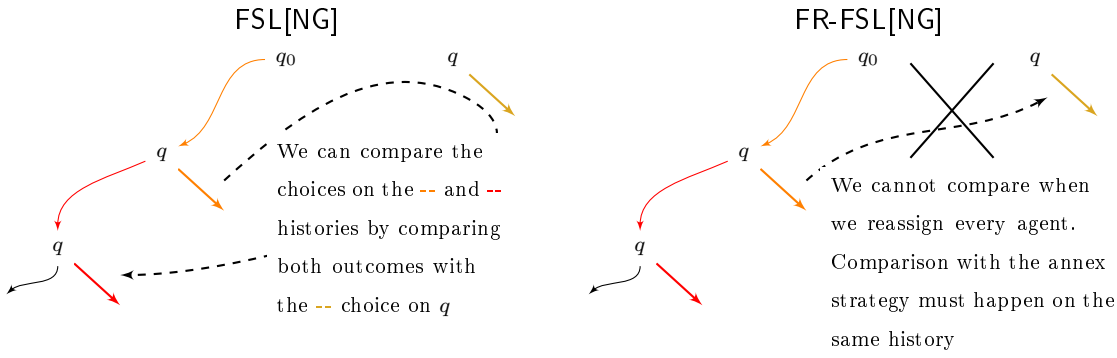


Figure 3.5: The differences between FSL[NG] (on the left) and FR-FSL[NG] (on the right).

Definition 3.12 (FSL[NG] with full reassignments: FR-FSL[NG])**.**
A flat FR-FSL[NG] formula over a set of agents $\mathsf{Agt} := \{A_1, \dots A_\lambda\}$ *and a set of variables* $\mathcal{V} := \{x_1, \dots\}$ *obey the grammar*

$$\begin{aligned} \textit{FR-FSL[NG]}^\flat \ni \phi &::= \exists x.\phi \mid \forall x.\phi \mid \xi \\ \xi &::= \xi \vee \xi \mid \xi \wedge \xi \mid \beta \\ \beta &::= \mathsf{assign}(A_1, x_1; \dots; A_\lambda, x_\lambda).\varphi \\ \varphi &::= \varphi \vee \varphi \mid \neg\varphi \mid \varphi \mathbf{U}\, \varphi \mid \mathbf{X}\, \varphi \mid p \mid \beta \end{aligned}$$

where $x, x_1, \dots, x_\lambda$ *are variables in* $\mathcal{V}$*.*

For the sake of readability and because any assignment must reassign all the agents, we regroup the assign() operators to work on all agents at once. As usual, the non flat fragment allows closed formulas at the same level as atomic propositions. The semantics follow the ones of FSL and FSL[NG].

Consider the two closed formulas below where A and B are two agents:

$$\begin{aligned} \phi_1 :=& \exists x_1.\forall x_2.\exists x_3.\ \mathsf{assign}(A, x_1; B, x_2).\big(\mathbf{F}\, p \wedge \mathsf{assign}(B, x_3)\mathbf{G}\, \neg p\big) \\ \phi_2 :=& \exists x_1.\forall x_2.\exists x_3.\ \mathsf{assign}(A, x_1; B, x_2).\big(\mathbf{F}\, p \wedge \mathsf{assign}(A, x_1; B, x_3).\mathbf{G}\, \neg p\big) \end{aligned}$$

In ϕ_1, we can see two goals: a standard goal $\mathsf{assign}(A, x_1; B, x_2).\big(\mathbf{F}\, p \wedge \mathsf{assign}(B, x_3).\mathbf{G}\, \neg p\big)$ and a nested goal $\mathsf{assign}(B, x_3).\mathbf{G}\, \neg p$ within the standard one. Formula ϕ_1 is not in FR-FSL[NG] because $\mathsf{assign}(B, x_3).\mathbf{G}\, \neg p$ does not reassign the strategy of agent A. The strategies of agents A and B are running on different histories. On the other hand, ϕ_2 is in FR-FSL[NG], the nested goal $\mathsf{assign}(A, x_1; B, x_3).\mathbf{G}\, \neg p$ reassigns every agent and their histories run on the same history. This common history makes all the difference.

Theorem 3.13. *The model checking problem for closed formulas of (the flat fragment)* FR-FSL[NG]$^\flat$ *with* k *alternations is in* $(k+2)$-EXPTIME.

Proof. The proof consists in a reduction to SL[BG] model checking, known to be decidable (see Chapter 2). The idea is to lift the floating fully reassigned nested goals into the

game in order to get something belonging to SL[BG]. We can then check the new formula through SL[BG] model checking algorithm. The central idea at the heart of FR-FSL[NG] can be expressed through the following lemma.

Lemma 3.14. *Fix a game $\mathcal{H}$, a goal ψ in FR-FSL[NG] (hence starting by a full reassignment) and a valuation χ in $\mathcal{H}$ with $\mathsf{free}(\psi) \subseteq \mathsf{dom}(\chi)$. Then, for any history π in $\mathcal{H}$,*

$$\mathcal{H}, \mathsf{lst}(\pi) \models_{\chi_{\overrightarrow{\pi}}} \psi \quad \Leftrightarrow \quad \mathcal{H}, \mathsf{lst}(\pi) \models_{\chi} \psi$$

The proof follows from the semantics of FSL where all agents are reassigned a new strategy on empty history, and from the fact that the strategies used by the agents at step i all share a common history. The lemma does not hold for nested goals of FSL[NG], where there may be two strategies assigned to two different agents that progress based on different histories (a first strategy assigned to an agent at some previous point with a non-empty history and a newly assigned strategy that has no history).

Consider a game $\mathcal{G} := \langle \mathsf{AP}, \mathsf{Agt}, \mathsf{Q}, \mathsf{Act}, \Delta, \mathsf{labels} \rangle$ and an initial state q_{ini} of Q. Consider also a FR-FSL[NG]$^\flat$ formula $\phi := \wp\xi(\psi_{d,i})_{d\in[1;D]\ i\in[1;\lambda_d]}$ with $\wp$ a quantification prefix, with ξ a boolean combination of goals where every nested goal is fully reassigned, with $D \in \mathbb{N}$ the maximal depth[3] of the nested goals of ϕ, with λ_d the number of goals at depth d, and with $\psi_{\{d,1\}}, \ldots, \psi_{\{d,\lambda_d\}}$ the list of goals at depth d. For example in the formula below on two agents A and B:

$$\exists x_1.\ \forall x_2.\ \exists x_3.\ \forall x_4 \begin{cases} \mathsf{assign}(A, x_1; B, x_2).\begin{cases} \mathbf{F}\, p_1 \wedge \\ \mathsf{assign}(A, x_3; B, x_4).\begin{cases} \mathbf{F}\, p_2 \vee \\ \mathsf{assign}(A, x_1; B, x_2).\mathbf{F}\, p_1 \end{cases} \end{cases} \\ \mathsf{assign}(A, x_1; B, x_4).\mathbf{G}\, \neg p_3 \end{cases}$$

the maximal depth of the nested goals is three; each sub-formula starting by an assignment is a new goal. At depth three there is a unique goal $\mathsf{assign}(A, x_1; B, x_2)\mathbf{F}\, p_1$. At depth two there is also a unique goal $\mathsf{assign}(A, x_3; B, x_4)\mathbf{F}\, p_2 \vee (\mathsf{assign}(A, x_1; B, x_2)\mathbf{F}\, p_1)$. At depth one there are two goals:

$$\mathsf{assign}(A, x_1; B, x_2).\Big(\mathbf{F}\, p_1 \wedge \big(\mathsf{assign}(A, x_3; B, x_4).(\mathbf{F}\, p_2 \vee \mathsf{assign}(A, x_1; B, x_2).\mathbf{F}\, p_1)\big)\Big)$$
$$\mathsf{assign}(A, x_1; B, x_4).\mathbf{G}\, \neg p_3$$

Building the game

We start by defining a game $\mathcal{H} := \langle \mathsf{AP}_{\mathcal{H}}, \mathsf{Agt}_{\mathcal{H}}, \mathsf{Q}_{\mathcal{H}}, \mathsf{Act}_{\mathcal{H}}, \Delta_{\mathcal{H}}, \mathsf{labels}_{\mathcal{H}} \rangle$ as follows

- $\mathsf{Agt}_{\mathcal{H}} := \mathsf{Agt} \cup \{Dec\}$. The agents of $\mathcal{H}$ are the agents of $\mathcal{G}$ plus a Decider agent Dec.

[3]The depth of reassignment of a goal ψ is the number of (full) reassignments containing ψ. A goal ψ appearing just after the quantification block is at depth 1 while a nested goal ψ' within ψ is at depth 2.

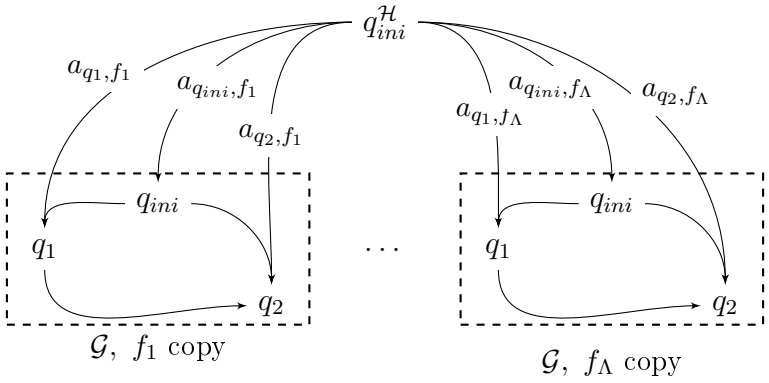


Figure 3.6: The game $\mathcal{H}$ where $\Lambda = \mathsf{Q} \times \{(d,i) \mid d \leq D,\ i \leq \lambda_d\}$

- $\mathsf{Q}_{\mathcal{H}} := \{q^{\mathcal{H}}_{ini}\} \cup \{q^f \mid q \in \mathsf{Q},\ f \in \{0,1\}^{\mathsf{Q}\times\{(d,i)|d\leq D,\ i\leq\lambda_d\}}\}$: the state space is made of one initial state $q^{\mathcal{H}}_{ini}$ and of $\{0,1\}^{\mathsf{Q}\times\{(d,i)|d\leq D,\ i\leq\lambda_d\}}$ copies of Q.
- $\mathsf{Act}_{\mathcal{H}} := \mathsf{Act} \cup \{a_{q,f} \mid q \in \mathsf{Q},\ f \in \{0,1\}^{\mathsf{Q}\times\{(d,i)|d\leq D,\ i\leq\lambda_d\}}\}$. The actions of $\mathcal{G}$ (Act) are accessible to any non-decider agent. We also add an action for each couple $(q,f) \in \mathsf{Q} \times \{0,1\}^{\mathsf{Q}\times\{(d,i)|d\leq D,\ i\leq\lambda_d\}}$, all exclusive to the decider agent.
- $\Delta_{\mathcal{H}} :=$: In $q^{\mathcal{H}}_{ini}$, only Dec plays and only the $a_{q,f}$'s actions are activated. If Decider plays $a_{q,f}$ the game moves to the q state within the f copy of Q. Within a copy of Q, the $a_{q,f}$ actions are deactivated and $\Delta_{\mathcal{H}}$ follows Δ (the transition function of $\mathcal{G}$).
- $\mathsf{labels}_{\mathcal{H}}$: For any $f \in \{0,1\}^{\mathsf{Q}\times\{(d,i)|d\leq D,\ i\leq\lambda_d\}}$ and any state q^f of the f copy, we label q^f as q in $\mathcal{G}$ and add the labels p_f and p_q. For example, given $q \in \mathsf{Q}$ and $f \in \{0,1\}^{\mathsf{Q}\times\{(d,i)|d\leq D,\ i\leq\lambda_d\}}$, we have $\mathsf{labels}_{\mathcal{H}}(q^f) = \mathsf{labels}(q) \cup \{p_f, p_q\}$.

Figure 3.6 illustrates the construction. It remains to specify the formula we use and to prove the correctness of our reduction.

Specifying the formula

The main idea behind our reduction is to transfer the validity of the nested goal to the choice of a copy of Q. Then, using Lemma 3.14, we can enforce the correct choice with an $\mathsf{SL[BG]}$ formula.

The formula $\phi^{\mathcal{H}}$ of our reduction is defined by

$$\phi^{\mathcal{H}} := \wp.\ \exists x^{Dec}_{Copy}.\Pi_{q\in\mathsf{Q}}(\exists x^{Dec}_{\to q}) \begin{cases} \mathsf{assign}(Dec, x^{Dec}_{Copy}).\mathbf{X} \begin{cases} \xi\Big[\bigvee_{f|f(d,i)=1} p_f/\psi_{d,i}\Big]_{2\leq d\leq D,\ i\leq\lambda_d} \\ \wedge\ p_{q_{ini}} \end{cases} \\ \wedge \bigwedge_{\jmath\in[1;2]} \psi^{\jmath}_{code} \end{cases}$$

where $\Pi_{q\in\mathsf{Q}}(\exists x^{Dec}_{\to q})$ represents a sequence of existential quantifiers, one per state of Q. The sub-formula $\bigwedge_{\jmath\in[1;2]} \psi^{\jmath}_{code}$ transfers the validity of the nested goal through the choice

of which copy of Q the strategy of x^{Dec}_{Copy} must go. We define them formally below, where β is any assignment present in ϕ. By the nature of $\mathcal{H}$, which assignment β is chosen does not matter, it is just a technicality to assign a strategy to each agent.

$$\psi^1_{code} := \begin{cases} \bigwedge_{q\in\mathsf{Q}} \mathsf{assign}(Dec, x^{Dec}_{\to q}).\mathbf{X}\, p_q \\ \wedge \bigwedge_{f\in\{0,1\}^{\mathsf{Q}\times\{(d,i)|d\le D,\ i\le\lambda_d\}}} \begin{cases} \mathsf{assign}(Dec, x^{Dec}_{Copy}).\beta\mathbf{X}\, p_f \\ \Rightarrow \mathsf{assign}(Dec, x^{Dec}_{\to q}).\beta\mathbf{X}\, p_f \end{cases} \end{cases}$$

$$\psi^2_{code} := \bigwedge_{(q,d,i)|q\in\mathsf{Q},\ d\le D,\ i\le\lambda_d} \begin{cases} \mathsf{assign}(Dec, x^{Dec}_{\to q}).\mathbf{X}\, \psi_{d,i} \\ \Leftrightarrow \\ \mathsf{assign}(Dec, x^{Dec}_{Copy}).\mathbf{X} \bigvee_{f|f(q,d,i)=1} p_f \end{cases}$$

Correctness of the reduction

We start by a purely technical proposition whose proof is a consequence of the definition of ψ^1_{code}.

Proposition 3.15. *Let χ be a valuation defined on the variables in $\phi^{\mathcal{H}}$. Then the following are equivalent*

- *for any $q \in \mathsf{Q}$, $\chi(x^{Dec}_{\to q})(q^{\mathcal{H}}_{ini})$ plays to q in the f copy of Q and $\chi(x^{Dec}_{Copy})(q^{\mathcal{H}}_{ini})$ plays to the copy of q_{ini} in the f copy of Q.*
- $\mathcal{G}, q^{\mathcal{H}}_{ini} \models_\chi \psi^1_{code}$

The strategy stored in x^{Dec}_{Copy} must choose a copy of the game. Which copy it chooses defines a function f taking as input a state q and the indexes d and i representing a goal, and returning a truth value.

We write $\mathsf{Var}(\wp)$ for the set of variables appearing in $\wp$. For a history ρ in $\mathcal{G}$, we define ρ^f as the history in $\mathcal{H}$ where $\rho^f(i)$ is the copy of the $\rho(i)$ state in the f copy of Q. For a valuation $\mathfrak{X}$ in $\mathcal{G}$ of domain $\mathsf{dom}(\mathfrak{X}) = \mathsf{Var}(\wp)$, we define a valuation $\mathfrak{X}^{\mathcal{H}}$ of domain $\mathsf{dom}(\mathfrak{X}^{\mathcal{H}}) = \mathsf{Var}(\wp)$ in $\mathcal{H}$ and where for any $x \in \mathsf{Var}(\wp)$ and any history ρ in $\mathcal{G}$

$$\mathfrak{X}^{\mathcal{H}}(x)(q^{\mathcal{H}}_{ini}.\rho^f) := \mathfrak{X}(x)(\rho)$$

Trivially, $\mathfrak{X}^{\mathcal{H}}$ is a properly defined valuation. We can then prove the main step in the correctness of our reduction.

Proposition 3.16. *Fix a valuation $\mathfrak{X}$ in $\mathcal{G}$ of domain $\mathsf{dom}(\mathfrak{X}) = \mathsf{Var}(\wp)$. Then we have $\mathcal{G}, q_{ini} \models_{\mathfrak{X}} \xi(\psi_{d,i})_{d\in[1;D]\ i\in[1;\lambda_d]}$ if and only if*

$$\mathcal{H}, q^{\mathcal{H}}_{ini} \models_{\mathfrak{X}^{\mathcal{H}}} \exists x^{Dec}_{Copy}.\Pi_{q\in\mathsf{Q}}(\exists x^{Dec}_{\to q}) \begin{cases} \mathsf{assign}(Dec, x^{Dec}_{Copy}).\ \mathbf{X} \begin{cases} \xi\Big[\bigvee_{f|f(d,i)=1} p_f/\psi_{d,i}\Big]^{2\le d\le D}_{1\le i\le\lambda_d} \\ \wedge\ p_{q_{ini}} \end{cases} \\ \wedge \bigwedge_{j\in[1;2]} \psi^j_{code} \end{cases}$$

Proof. Assume $\mathcal{G}, q_{ini} \models_{\mathfrak{X}} \xi(\psi_{d,i})_{d\in[1;D]\ i\in[1;\lambda_d]}$. We deduce from $\mathfrak{X}$ a function $f \in \{0,1\}^{\mathsf{Q}\times\{(d,i)|d\leq D,\ i\leq\lambda_d\}}$ such that

$$f(q,d,i) = 1 \quad \Leftrightarrow \quad \mathcal{G}, q \models_{\mathfrak{X}} \psi_{d,i} \tag{3.3}$$

We define δ^{Dec}_{Copy} to play $a_{q_{ini},f}$, i.e. to go to the copy of q_{ini} in the f copy of Q. We also set $\delta^{Dec}_{\to q}$ to play $a_{q,f}$ for any $q \in \mathsf{Q}$. Finally we write χ for the valuation

$$\chi \ := \ \mathfrak{X} \cup \{x^{Dec}_{Copy} \to \delta^{Dec}_{Copy}\} \cup \bigcup_{q\in\mathsf{Q}} \{x^{Dec}_{\to q} \to \delta^{Dec}_{\to q}\}$$

Our choices plus Proposition 3.15 gives

$$\mathcal{H}, q^{\mathcal{H}}_{ini} \models_{\chi} \psi^1_{code} \tag{3.4}$$

Moreover, again by construction of $\mathfrak{X}^{\mathcal{H}}$ and definition of χ,

$$\begin{aligned} \mathcal{H}, q^{\mathcal{H}}_{ini} \models_{\chi} \mathsf{assign}(Dec, x^{Dec}_{\to q})\mathbf{X}\,\psi_{d,i} \quad &\Leftrightarrow \quad \mathcal{G}, q \models_{\mathfrak{X}} \psi_{d,i} \\ &\Leftrightarrow \quad f(q,d,i) = 1 \qquad \text{(by Formula 3.3)} \end{aligned}$$

so

$$\mathcal{H}, q^{\mathcal{H}}_{ini} \models_{\chi} \psi^2_{code} \tag{3.5}$$

Finally, the hypothesis $\mathcal{G}, q_{ini} \models_{\mathfrak{X}} \xi(\psi_{d,i})_{d\in[1;D]\ i\in[1;\lambda_d]}$ gives

$$\mathcal{H}, q^{\mathcal{H}}_{ini} \models_{\chi} \mathsf{assign}(Dec, x^{Dec}_{Copy}).\ \mathbf{X}\, p_{q_{ini}} \wedge \xi\Big[\bigvee_{f|f(d,i)=1} p_f/\psi_{d,i}\Big]_{2\leq d\leq D,\ i\leq\lambda_d} \tag{3.6}$$

The combination of Formulas (3.4), (3.5) and (3.6) gives the left-to-right way of Proposition 3.16.

Conversely, the right-to-left implication is somewhat similar. Assume the right-hand side of the equivalence to hold and let $\mathfrak{X}$ be any valuation resulting from the interplay within $\wp$. Write χ for a working valuation that extends $\mathfrak{X}$ by giving strategies to x^{Dec}_{Choice} and the $x^{Dec}_{\to q}$'s variables. Write f for the copy chosen by $\chi(x^{Dec}_{Choice})(q^{\mathcal{H}}_{ini})$. Because of ψ^1_{code} and Proposition 3.15, the choices of $x^{Dec}_{\to q}$ are set to play to q in the f copy of Q. Then because ψ_2 and ψ_3 hold,

$$\mathcal{H}, q^{\mathcal{H}}_{ini} \models_{\chi} \mathsf{assign}(Dec, x^{Dec}_{\to q})\mathbf{X}\,\psi_{d,i} \quad \Leftrightarrow \quad f(q,d,i) = 1$$

Now, by definition of $\mathfrak{X}^{\mathcal{H}}$,

$$\mathcal{G}, q \models_{\chi} \psi_{d,i} \quad \Leftrightarrow \quad f(q,d,i) = 1$$

hence

$$\mathcal{G}, q \models_{\chi} \psi_{d,i} \quad \Leftrightarrow \quad \mathcal{H}, q^{\mathcal{H}}_{ini} \models_{\chi} \mathsf{assign}(Dec, x^{Dec}_{\to q})\mathbf{X}\,\psi_{d,i}$$

Combining this with the hypothesis

$$\mathcal{H}, q^{\mathcal{H}}_{ini} \models_{\chi} \mathsf{assign}(Dec, x^{Dec}_{Copy}).\ \mathbf{X}\,(p_{q_{ini}} \wedge \xi\Big[\bigvee_{f|f(d,i)=1} p_f/\psi_{d,i}\Big]_{2\leq d\leq D,\ i\leq\lambda_d})$$

we get that $\mathcal{G}, q_{ini} \models_{\mathfrak{X}} \xi(\psi_{d,i})_{d\in[1;D]\ i\in[1;\lambda_d]}$. □

It remains to handle the quantifications within $\wp$. We will proceed by induction but first we need some notations. We write $\wp := (Q_i x_i)_{1 \leq i \leq l}$ where $Q_i \in \{\exists, \forall\}$ and, for any $j \leq l$, we define $\phi_{\geq j}$ as the sub-formulas of ϕ without $Q_1 x_1 \dots Q_{j-1} x_{j-1}$.

Proposition 3.17. *For any $1 \leq j \leq l+1$ and any valuation $\mathfrak{X}_{<j}$ over $\mathsf{dom}(\mathfrak{X}_{<j}) := \{x_1, \dots, x_{j-1}\}$, we have*

$$\mathcal{G}, q_{ini} \models_{\mathfrak{X}_{<j}} \phi_{\geq j} \quad \Leftrightarrow \quad \mathcal{H}, q^{\mathcal{H}}_{ini} \models_{\mathfrak{X}^{\mathcal{H}}_{<j}} \phi^{\mathcal{H}}_{\geq j}$$

Proof. We proceed by induction on j. The initial step ($j = l+1$) was done in Proposition 3.16. So only the induction step ($1 \leq j \leq l$) remains. We assume that by induction we have

$$\mathcal{G}, q_{ini} \models_{\mathfrak{X}_{<j+1}} \phi_{\geq j+1} \quad \Leftrightarrow \quad \mathcal{H}, q^{\mathcal{H}}_{ini} \models_{\mathfrak{X}^{\mathcal{H}}_{<j+1}} \phi^{\mathcal{H}}_{\geq j+1} \tag{3.7}$$

and aim to prove that $\mathcal{G}, q_{ini} \models_{\mathfrak{X}_{<j}} \phi_{\geq j} \quad \Leftrightarrow \quad \mathcal{H}, q^{\mathcal{H}}_{ini} \models_{\mathfrak{X}^{\mathcal{H}}_{<j}} \phi^{\mathcal{H}}_{\geq j}$.

We treat the case where $Q_j = \exists$, the case with $Q_j = \forall$ is similar. For the left-to-right implication, assume that $\mathcal{G}, q_{ini} \models_{\mathfrak{X}_{<j}} \phi_{\geq j}$; then there is δ_j such that $\mathcal{G}, q_{ini} \models_{\mathfrak{X}_{<j} \cup \{x_j \to \delta_j\}} \phi_{\geq j+1}$ and therefore, by induction hypothesis –Formula (3.7)–, we get

$$\mathcal{H}, q^{\mathcal{H}}_{ini} \models_{\mathfrak{X}^{\mathcal{H}}_{<j} \cup \{x_j \to \delta^{\mathcal{H}}_j\}} \phi^{\mathcal{H}}_{\geq j}$$

with $\delta^{\mathcal{H}}_j(x)(q^{\mathcal{H}}_{ini}.\rho_f) := \delta^{\mathcal{H}}_j(x)(\rho)$ for any history ρ in $\mathcal{G}$. We can then infer

$$\mathcal{H}, q^{\mathcal{H}}_{ini} \models_{\mathfrak{X}^{\mathcal{H}}_{<j}} \exists x_j.\ \phi^{\mathcal{H}}_{\geq j+1}$$
$$\text{and} \quad \mathcal{H}, q^{\mathcal{H}}_{ini} \models_{\mathfrak{X}^{\mathcal{H}}_{<j}} \phi^{\mathcal{H}}_{\geq j}$$

The right to left way is similar hence omitted. In the end we get the induction step and, by induction principle, we can conclude the proof of Proposition 3.17. □

We can then deduce the correctness of our reduction (expressed by the proposition below) from Proposition 3.17 applied to $j = 1$.

Proposition 3.18.

$$\mathcal{G}, q_{ini} \models \phi \quad \Leftrightarrow \quad \mathcal{H}, q^{\mathcal{H}}_{ini} \models \phi^{\mathcal{H}}$$

We recall that the algorithm proposed in Chapter 2 for SL[BG] model checking works in time $(k+1)$-EXPTIME for formulas with k alternations (see Theorem 2.6 page 42). The size of the SL[BG] formula $\phi^{\mathcal{H}}$ is exponential in the size of $\mathcal{G}$ (the original game), in the maximal depth of ϕ and in the number of nested goals in ϕ; the others parameters are polynomial in the size of ϕ. We can therefore solve the model checking of FR-FSL[NG]$^\flat$ formulas with k alternations in $(k+2)$-EXPTIME, one exponential more than SL[BG]. □

We recall that SL[BG] is a fragment of FR-FSL[NG]. So, with the hardness proofs developed for SL[BG] in Sections 2.2 and 2.3, we cannot hope for a fundamentally better complexity. At best we can reduce it by an exponential so that FR-FSL[NG] upper bound matches the one known for SL[BG]. Note that Theorem 3.13 can be trivially extend to the non flat fragment by using a labelling algorithm (a la CTL) and applying Theorem 3.13 in an inductive fashion. Hence

Theorem 3.19. *The model checking of FR-FSL[NG] with formula of k alternations is in $(k+2)$-EXPTIME.*

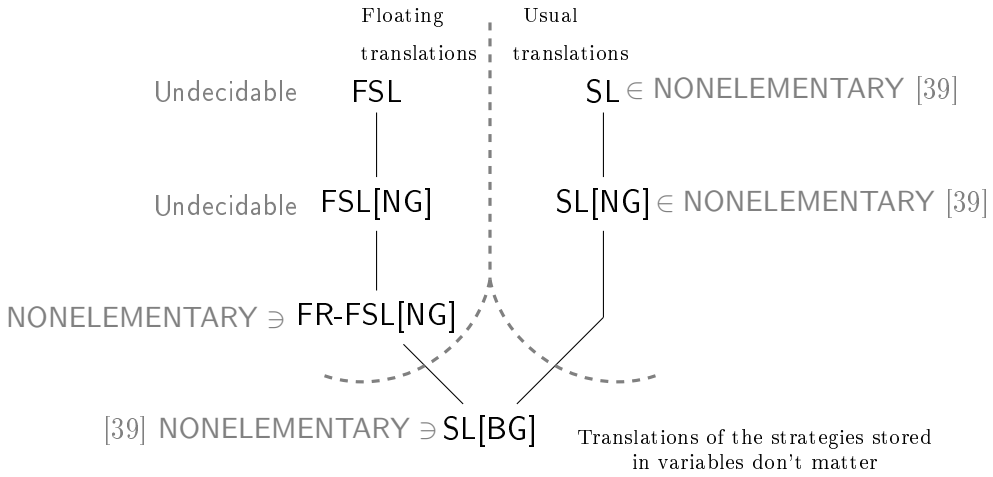


Figure 3.7: Inclusion graph with the different semantic choices and the model checking complexity.

3.4 Conclusion

Overall view: In this chapter we have highlighted an important subtlety of SL semantics related to the notion of *valuation translation*. We proposed in Section 3.1 a new variant FSL of SL that tackles the aforementioned problem. FSL has the same grammar as SL, only the notions of valuation translations and strategies are adapted. FSL and SL coincide on their common fragment SL[BG]. We then proved in Section 3.2 that FSL[NG] (the equivalent of SL[NG] within the framework of FSL) admits an undecidable model checking. As shown in Section 3.3, decidability can be regained when forcing the reassignment to be total; for this we defined a fragment FR-FSL[NG] of FSL. Figure 3.7 sums up our work.

On a more refined level: The undecidability of FSL[NG] holds for the set of formulas with one quantifier-alternation and for games with only two agents. Outside of FR-FSL[NG] decidability, regaining decidability takes us away from the framework defined by SL and related logics. For example we may look at formulas without quantifier alternation but this essentially takes us away from closed systems. Indeed, while working with one quantifier-alternation formulas does not technically take us away from multi-agent CGS, we can easily draw connections with existing results for closed systems and from automata theory. We believe the model checking to be decidable though we do not give a formal proof.

A second possibility is to consider the number of agents allowed to reassign their strategies. In the undecidable FSL and FSL[NG], any subset of Agt may reassign its strategies; in the decidable FR-FSL[NG], all agents are forced to reassign. In the proof of undecidability for FSL[NG] model checking, both agents reassign their strategies, it may be interesting to know about the decidability status of FSL where only a single agent is allowed to do so. The essence of the undecidability proof explained at the beginning

of Section 3.3 and on Figure 3.5 (page 72) is still possible if a single agent is allowed reassignment. However, the gadget used to prove SL[NG] undecidability in Section 3.2 needs both agents to reassign their strategies: to check for proper incrementation of the counters, the agent *Dec* needs to be reassigned while for the decrementation, it is the agent *Che* who needs reassignment. Torn between these two arguments, we do not conjecture one way or the other on the decidability of FSL[NG] where a single agent is allowed to reassign its strategies.

Chapter 4

Quantitative Strategy Logic

Many computerised systems have internal quantitative aspects: systems managing energy grids, computer softwares with loops... The standard (qualitative) formalisms usually fail to model properly such systems. The verification community therefore developed adaptations of existing formalisms to handle quantitative aspects. So far, all our results have been on CGS, hence qualitative. We take a look at what can be done to adapt SL and its sub-logics to express quantitative constraints.

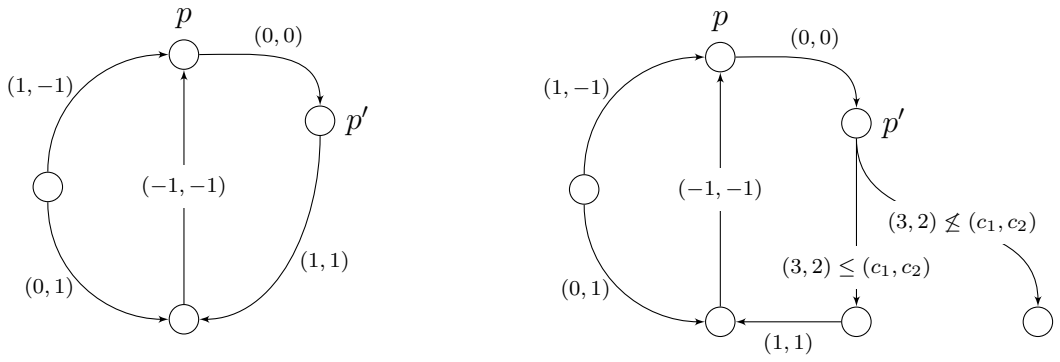
Quantitative verification of open systems is a well-studied domain with many branches. Among the most famous ones, we find energy games [13, 14], mean-payoff games [64, 14], alternating-VASS [46], counter-games [5], pushdown-games [62], timed games [6] and hybrid games [27]. Each one corresponds to a type of quantitative systems, to which we often add other conditions (imperfect information, probabilistic edges...). Often an algorithm working for one model fails in another. For this reason there is a large (too large, some may say) number of algorithms, each adapted to a different model representing a different type of quantitative open systems.

In this chapter, we focus on two of them: energy games and counter games. For each model, we give a brief introduction, update the model, add adequate constraints to the logic, discuss the choices we made and provide some complexity results.

4.1 With counters

Counters are omnipresent in programming, even without considering mathematical formulas. Indeed, every loop is handled through a counter (as explained in [5]) and it is therefore not surprising to find a large literature on systems with counters [5, 51]. The usual framework consists in adding weights on the transitions of the model (be it a graph or an automaton). Counter constraints are then appended either on the transition of the model (as with zero-test edges in counter automata) or in the formalism for the properties (as in weighted MSO, see [20]). Figure 4.1 shows the idea with a LTL formula and a two-counter constraint $(3, 2) \leq (c_1, c_2)$. Here we adopt the second approach, note

nevertheless that in SL both are equivalent[1].



$\mathbf{F}\, p \wedge \mathbf{G}\left(p' \Rightarrow (3,2) \leq (c_1, c_2)\right)$ $\qquad$ $\mathbf{F}\, p$

Figure 4.1: The two ways to work with quantitative constraints: on the left, constraints are on the formula and on the right, constraints are on the game.

Many things are known for closed systems with counters. However, when working with open systems, there exist fewer results and many questions remain open. We highlight the three most important results for our work. First, reachability in counter games with two agents is undecidable; this is a consequence of the undecidability of two-counter machines [37]. Second, as shown by Serre in [51], reachability, Büchi and parity conditions are PSPACE-complete in one-counter games. Finally, ATL* model checking was shown to be *2*-EXPTIME-complete in [61].

We first adapt our framework by defining weighted concurrent game structures and by extending SL with counter constraints. Then, in Section 4.1.2 we prove a periodicity result for the satisfaction relation. Finally we study in Section 4.1.3 the expressiveness of SL with counter constraints through a correspondence with MSO theory on ordinals.

4.1.1 Adding counter constraints to SL

Adding weights to CGS

Definition 4.1 (Weighted Concurrent Game Structure with weights: WCGS)**.**
An n-dimensional Weighted Concurrent Game Structure (n-WCGS) is a tuple $\mathcal{G} := \langle AP, Agt, Q, Act, \Delta, Weights, labels\rangle$ where AP, Agt, Q, Act, Δ, and $labels$ represents the same notions as in (qualitative) CGS, and where $Weights : Q \times Act^{Agt} \to \{-1, 0, 1\}^n$ is a function that assigns n integers (called weights) to each transition of Δ.

A Weighted Concurrent Game Structure (WCGS) is an n-WCGS for some strictly positive integer n.

[1] In the sens of a polynomial reduction from the framework with constraints on the transitions of the model toward the one with constraints onto the logic formalism, and vice-versa. We do not provide a proof however.

To simplify the proofs, we chose to use only 0, -1 and 1 weights, we could have also chosen weights in $\mathbb{Z}$. Choosing $\mathbb{Z}$ and encoding the weights in binary may however affect the complexity by an exponential factor. Haase, Kreutzer, Ouaknine and Worrell studied this question in [26] for one-counter automata.

Beside CGS, we also need to adapt the notions we used in previous chapters. For an n-WCGS structure $\mathcal{G}$, a *configuration* is a pair $(q, c) \in \mathsf{Q} \times \mathbb{N}^n$ of a state and a tuple of n non-negative integers. A *history* (resp. *path*) is a sequence $(q_i, c_i)_{i<L}$ of configurations where $L \in \mathbb{N}$ (resp. $L = \infty$), for any $i \leq L$ (resp. $i \in \mathbb{N}$) $c_i \in \mathbb{N}^n$ and such that for any $i < L - 1$ (resp. $i \in \mathbb{N}$) $q_{i+1} = \Delta(q_i, m_i)$, and $c_{i+1} = c_i + \mathsf{Weights}(q_i, m_i)$ for some $m_i \in \mathsf{Act}^{\mathsf{Agt}}$. All other notions are defined similarly to the ones of Chapter 1, using the new notions of histories and paths.

Remark 4.2. *For the sake of tradition, we ask for all weights to stay above* 0. *Although there may exists results that are impacted by such restriction [47], this is not the case of the results presented in this thesis. In particular, in this case, the periodicity result (Theorem 4.7) of Section 4.1.2 still holds and one can prove a similar theorem for negative values.*

Adding constraints to SL

In conjunction with the addition of weights on CGS, we extend SL by allowing it to express numerical constraints on the weights.

Definition 4.3 (Counter Constraints)**.**
A counter constraint on n weights is a subset S of $\mathbb{N}^n$ made of a (finite) union of periodic sets of integers: sets of the form $a + b \cdot \mathbb{N}^n$ for a fixed initial threshold vector a and a period vector b, both in $\mathbb{N}^n$.

This definition of constraints allows us to express standard counter constraints like $cnt \leq (4, 5, 2)$ (by taking $b = (0, \ldots, 0)$) but also more advanced constraints[2].

Definition 4.4 (Strategy logic with counter constraints: cSL)**.**
The logic cSL is built upon a number n of weights, a set Agt of agents, a set AP of atomic propositions and a set $\mathcal{V}$ of variables. Its formulas are structured by the following grammar:

$$\mathsf{cSL} \in \phi \ ::= \ \exists x.\phi \mid \mathsf{assign}(A, x).\phi \mid \phi \vee \phi \mid \neg\phi \mid \phi \mathbf{U}\, \phi \mid \mathbf{X}\, \phi \mid p \mid \mathsf{cnt} \in S$$

with $x \in \mathcal{V}$ a variable, $A \in \mathsf{Agt}$ an agent and $p \in \mathsf{AP}$ an atomic proposition, and where S is a counter constraint on n weights.

The notions of free agents and variables for cSL are the same as in SL, with the addition of the condition

$$\mathsf{free}(\mathsf{cnt} \in \mathsf{S}) = \varnothing \quad \text{for all counter constraint } \mathsf{S}$$

[2]The notion of period used here (numerical constraints on the weights) is to be dissociated with the one of Section 4.1.2 (related to the potential initial values of the initial configuration). There is some degree of connection between the two but the two notions are clearly distinct.

Formulas of cSL based on n weights are evaluated on an n-WCGS structure $\mathcal{G}$, at a configuration (q, c) of $\mathcal{G}$ and relatively to a valuation χ. The semantics of all but the $\mathsf{cnt} \in \mathsf{S}$ operator is the same as in SL, using the state q and the updated notions of histories and paths. The $\mathsf{cnt} \in \mathsf{S}$ operator has the following semantic rule:

$$\mathcal{G}, (q, c) \models_\chi \mathsf{cnt} \in \mathsf{S} \quad \Leftrightarrow \quad c \in \mathsf{S}$$

Example 4.5. *An example of 2-*WCGS *with an unique player A can be found on the left of Figure 4.1. Let ϕ be the following* cSL *formula:*

$$\phi := \exists x.\ \mathsf{assign}(A, x)\mathbf{F}\, p \wedge \mathbf{G}\left(p' \Rightarrow (3, 2) \leq (c_1, c_2)\right)$$

On can see that starting from the node at the bottom, ϕ does not hold on the game. Indeed, while a p labelled state p can be eventually reached, the path must pass through the p' labelled state and its counter will be 0. Hence the second conjunct will not be satisfied and ϕ will not hold.

The model checking problem for cSL is defined similarly to the one for SL (see Chapter 2 for a formal definition). Since Minsky's result [37], it is well known that reachability in two-counter machines is undecidable and therefore so is the reachability problem in n-WCGS where $n \geq 2$ (see [9] for more results on reachability in counter games that keep weights above 0).

Theorem 4.6. *cSL model checking over* WCGS *is undecidable.*

As a consequence, cSL cannot have a decidable model checking without restricting the number of weights. On the other hand, the undecidability result fails when automata have only a single counter. For this reason, we focus on games with a unique weight. We call a 1-WCGS any concurrent game structure with one weight and 1cSL the fragment of cSL built using a single weight. Finally, we extend the definition of SL and all its fragments with counter constraints on a single weight. In particular, the (flat) fragment $\mathsf{1cSL[BG]}^\flat$ analogue of $\mathsf{SL[BG]}^\flat$ follows the grammar:

$$\begin{aligned}
\mathsf{1cSL[BG]}^\flat \ni \phi &::= \exists x.\phi \mid \forall x.\phi \mid \xi \\
\xi &::= \xi \vee \xi \mid \xi \wedge \xi \mid \beta \\
\beta &::= \mathsf{assign}(A, x).\beta \mid \varphi \\
\varphi &::= \varphi \vee \varphi \mid \neg\varphi \mid \varphi \mathbf{U}\, \varphi \mid \mathbf{X}\, \varphi \mid p \mid \mathsf{cnt} \in \mathsf{S}
\end{aligned}$$

As usual, the full fragment 1cSL[BG] is obtained by allowing closed 1cSL[BG] formulas at the level of the atomic propositions. We call 1cLTL the set of formulas of type φ in the grammar of $\mathsf{1cSL[BG]}^\flat$.

4.1.2 Periodicity of 1cSL[BG]

In this section, we prove a periodicity property for 1cSL[BG]. We recall that Tower is the function $\mathsf{Tower}\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ taking two entries a, b and returning the integer equals to a tower of exponentials of height b and value a, see page 44 for the formal definition.

Theorem 4.7. *Let $\mathcal{G}$ be a 1-WCGS, and ϕ be a 1cSL[BG] formula. Then there exists a threshold $\Delta \geq 0$ and a period $\Lambda \geq 0$ for the truth value of ϕ over $\mathcal{G}$. That is, for every configuration (q, c) of $\mathcal{G}$ with $c \geq \Delta$, for every $k \in \mathbb{N}$, $\mathcal{G}, (q, c) \models \phi$ if and only if $\mathcal{G}, (q, c + k \cdot \Lambda) \models \phi$.*

Furthermore the order of magnitude for $\Delta + \Lambda$ is bounded by

$$\mathit{Tower}\Big(\max_{\theta \in \mathit{SubForm}(\phi)} n_\theta, \max_{\theta \in \mathit{SubForm}(\phi)} k_\theta + 1\Big)^{|Q| \cdot 2^{2^{|\phi|}}}$$

where Q is the state space of $\mathcal{G}$, $\mathit{SubForm}(\phi)$ is the set of 1cSL[BG] sub-formulas of ϕ, k_θ is the number of quantifier alternations in θ, and n_θ is the number of different assignments used in θ.

A result similar to the one of Theorem 4.7 should be mentioned: Göller and Lohrey showed in [25] that CTL admits exponential thresholds and periods, where the exponent is made of the left imbrication of the $\mathsf{E} \cdot \mathbf{U} \cdot$ operators. The verification community usually focuses on complexity results but rarely on combinatorial ones, therefore periodic properties are unusual[3].

Proof. We first prove this property for the flat fragment $\mathsf{1cSL[BG]}^\flat$, and then extend it to the full 1cSL[BG]. To keep the proof readable we moved the proof of some intermediary results to the annex (page 106) and replace it with a sketch. The rest of this section is devoted to the proof of Theorem 4.7.

The flat fragment $\mathsf{1cSL[BG]}^\flat$

We fix a 1-WCGS $\mathcal{G}$ and a formula $\phi := Q_1 x_1 \dots Q_k x_k.\ \xi(\beta_i \varphi_i)_{1 \leq i \leq n}$ in $\mathsf{1cSL[BG]}^\flat$, where for every $1 \leq j \leq k$, we have $Q_j \in \{\exists, \forall\}$ (assuming quantifiers strictly alternate), ξ is a Boolean formula over n atoms, and for every $1 \leq i \leq n$, β_i is a complete assignment for the agents' strategies, and φ_i is a 1cLTL formula. We write M for the maximal constant appearing in one of the finite sets describing a counter constraint S appearing in ϕ.

For every $1 \leq i \leq n$, we let $\mathcal{D}_i$ be a deterministic (one counter) parity automaton that recognises formula φ_i. This is the standard LTL to deterministic parity automata construction of Theorem 1.3 (page 20) in which quantitative constraints are seen as atoms. A run of $\mathcal{G}$ is read in a standard way, with the additional condition that quantitative constraints labelling a state should be satisfied by the counter value when the state is traversed (a state can be labelled by a constraint $\mathsf{cnt} \in \mathsf{S}$, with S arbitrarily complex—it does not impact the description of the automaton).

The proof proceeds by showing that, above some threshold, the truth value of ϕ is periodic w.r.t. counter values. To prove this, we define an equivalence relation over

[3] The verification community solves the model checking of quantitative temporal logics almost exclusively by a reduction to the emptiness of a given automata. The periodicity properties are then hidden within the algorithm solving the emptiness of the automata. On a personal note, I believe that working exclusively with automata techniques is not sufficient to fully understand the underlying behaviour of the quantitative logics. While the proof of Theorem 4.7 uses of automata, their roles are reduced to transforming the LTL goals in parity conditions.

counter values that generates identical strategic possibilities (in a sense that will be made clear later on).

Definition of an equivalence relation Fix a configuration $\gamma = (q, c)$ in $\mathcal{G}$, pick for every $1 \leq i \leq n$ a state d_i in the automaton $\mathcal{D}_i$, and define the tuple $D = (d_1, \dots, d_n)$. For every valuation χ_k for variables $\{x_1, \dots, x_k\}$, we define the level-0 identifier $\mathsf{Id}_{\chi_k}(\gamma, D)$ as:

$$\mathsf{Id}_{\chi_k}(\gamma, D) := \left\{ i \;\middle|\; 1 \leq i \leq n \text{ and } \mathsf{out}(\beta_i(\chi_k), \gamma) \text{ is accepted by } \mathcal{D}_i \text{ from } d_i \right\}$$

where $\beta_i(\chi_k)$ is the valuation obtained by assigning a strategy from χ_k to each agent in Agt following β_i.

Assuming we have defined level-$(k-j+1)$ identifiers $\mathsf{Id}_{\chi_{j+1}}(\gamma, D)$ for every partial valuation χ_{j+1} for variables $\{x_1, \dots, x_{j+1}\}$, we define the level-$(k-j)$ identifier $\mathsf{Id}_{\chi_j}(\gamma, D)$ for every partial valuation χ_j for variables $\{x_1, \dots, x_j\}$ as follows:

$$\mathsf{Id}_{\chi_j}(\gamma, D) := \left\{ \mathsf{Id}_{\chi_{j+1}}(\gamma, D) \;\middle|\; \chi_{j+1} \text{ is a valuation for } \{x_1, \dots, x_{j+1}\} \text{ that extends } \chi_j \right\}.$$

There is a unique level-k identifier for every configuration $\gamma = (q, c)$ and every D, which corresponds to the empty valuation. It somehow contains full information about what kinds of strategies can be used in the game (this is a hierarchical information set, which contains all level-j identifiers for $j < k$).

We will first give a characterization of the construction of identifiers, which will help understand how it can be used; we then count how many values the level-k identifier can take, from which our period will be derived.

Characterization We inductively define the following boolean property:

$$\mathbb{P}_0^{q,D}(\chi_k, \chi'_k)(c, c') : \quad \text{(truth value of) } \mathsf{Id}_{\chi_k}((q, c), D) = \mathsf{Id}_{\chi'_k}((q, c'), D)$$

and for every $0 \leq j < k$,

$$\mathbb{P}_{k-j}^{q,D}(\chi_j, \chi'_j)(c, c') : \quad \begin{cases} \forall v_{j+1}.\ \exists v'_{j+1}.\ \mathbb{P}_{k-j-1}^{q,D}(\chi_j \cup \{v_{j+1}\}, \chi'_j \cup \{v'_{j+1}\})(c, c') & \text{and} \\ \forall v'_{j+1}.\ \exists v_{j+1}.\ \mathbb{P}_{k-j-1}^{q,D}(\chi_j \cup \{v_{j+1}\}, \chi'_j \cup \{v'_{j+1}\})(c, c') \end{cases}$$

This property reads a bit like an alternating equivalence between c and c'. It allows to characterize equivalent configurations w.r.t. the identifier predicate.

Lemma 4.8. *Fix some $0 \leq j \leq k$ and some partial valuations χ_j and χ'_j for variables $\{x_1, \dots, x_j\}$ from (q, c) and (q, c'), respectively. The following two properties are equivalent:*

- $\mathsf{Id}_{\chi_j}((q, c), D) = \mathsf{Id}_{\chi'_j}((q, c'), D)$
- $\mathbb{P}_{k-j}^{q,D}(\chi_j, \chi'_j)(c, c')$

Proof. We show the equivalence by induction on $0 \leq j \leq k$, starting from $j = k$, where the equivalence precisely corresponds to the definition.

Assume $\mathsf{Id}_{\chi_j}((q,c),D) = \mathsf{Id}_{\chi'_j}((q,c'),D)$ for some $0 \leq j < k$. By definition, it means that for every extended valuation χ_{j+1} of χ_j, there exists an extended valuation χ'_{j+1} of χ'_j (and conversely) such that $\mathsf{Id}_{\chi_{j+1}}((q,c),D) = \mathsf{Id}_{\chi'_{j+1}}((q,c'),D)$. By induction hypothesis, it holds $\mathbb{P}^{q,D}_{k-j-1}(\chi_{j+1},\chi'_{j+1})(c,c')$. Since this holds for all appropriate quantifications, we get that $\mathbb{P}^{q,D}_{k-j}(\chi_j,\chi'_j)(c,c')$ holds as well. The converse is similar. □

Let P be the least common multiple of all the periods appearing in periodic constraints used in formula ϕ. We define the following equivalence on counter values:

$$c \sim c' \quad \text{if and only if} \quad c = c' \bmod \mathsf{P} \text{ and } \forall D.\ \forall q.\ \mathsf{Id}_\emptyset((q,c),D) = \mathsf{Id}_\emptyset((q,c'),D).$$

Combinatorics. Given a configuration (q,c) and a tuple D, the number of possible values for the level-0 identifier is $\mathsf{Tower}(n,1)$, and for the level-j identifier it is $\mathsf{Tower}(n,j+1)$. Hence, the number $\mathsf{ind}_\sim$ of equivalence classes of the relation $\sim$ satisfies

$$\mathsf{ind}_\sim \leq \mathsf{P} \cdot (\mathsf{Tower}(n,k+1))^{\left(|\mathsf{Q}|\cdot\Pi_{1\leq i\leq n}\, 2^{2^{|\varphi_i|}}\right)} \leq \mathsf{P} \cdot (\mathsf{Tower}(n,k+1))^{\left(|\mathsf{Q}|\cdot 2^{2^{|\phi|}}\right)}$$

with $|\mathsf{Q}|$ the number of states in $\mathcal{G}$. We let $\overline{\mathsf{M}} = \mathsf{M} + \mathsf{ind}_\sim + 1$. By the pigeon-hole principle, there must exist $\mathsf{M} < \Delta < \Delta' \leq \overline{\mathsf{M}}$ such that $\Delta \sim \Delta'$.

Periodicity property We define $\Lambda = \Delta' - \Delta$, and now prove that Λ is a period for ϕ for counter values larger than or equal to Δ. Assume that $\gamma = (q,c)$ is a configuration such that $c \geq \Delta$, and define $\gamma' = (q, c+\Lambda)$ (note that $c + \Lambda \geq \Delta'$). We show that $\mathcal{G},\gamma \models \phi$ if and only if $\mathcal{G},\gamma' \models \phi$.

Notations. *For the rest of this proof, we fix the following notations:*

1. *if ρ is a run starting with counter value $a > c$, then either the counter always remains strictly above c along ρ (in which case we say that ρ is fully above c), or it eventually hits value c, and we define $\rho_{\searrow c}$ for the smallest prefix of ρ such that $\mathsf{lst}(\rho_{\searrow c})$ has counter value c;*

2. *let ρ be a run that is fully above M, and let c be the least counter value appearing in ρ. For every $\nu \geq \mathsf{M} - c$, we write $\mathsf{Shift}_\nu(\rho)$ for the run ρ' obtained from ρ by shifting the counter value by ν. It is a well-defined run since the counter values along ρ' are also all above M hence above 0.*

3. *if D is a tuple of states of the deterministic automata $(\mathcal{D}_i)_{1\leq i\leq n}$, and if ρ is a finite run of $\mathcal{G}$ that is fully above M, then we write $D_{+\rho}$ for the image of D after reading ρ.*

We first show an easy result:

Lemma 4.9. *Let ρ be a finite run, and $\rho' = \mathsf{Shift}_{+\Lambda}(\rho)$. Let D be a tuple of states of the automata $(\mathcal{D}_i)_{1\leq i\leq n}$. Then, ρ is fully above Δ iff ρ' is fully above Δ'. In case both conditions are wrong, it holds that $\rho'_{\searrow\Delta'} = \mathsf{Shift}_{+\Lambda}(\rho_{\searrow\Delta})$. Furthermore, in the first case, $D_{+\rho} = D_{+\rho'}$ whereas in the second case, $D_{+\rho_{\searrow\Delta}} = D_{+\rho'_{\searrow\Delta'}}$.*

Proof. The two first properties are obvious by definition of $\mathsf{Shift}_{+\Lambda}$ (since $\Delta' = \Delta + \Lambda$).

Since $\Delta > \mathsf{M}$, all counter values along both histories are larger than M, and hence, two corresponding configurations along ρ and ρ' satisfy the same non-modulo counter constraints. The period Λ is a multiple of P, the lcm of all the periods, hence two corresponding configurations along ρ and ρ' also satisfy the same modulo constraints. Finally, all atomic propositions are equivalently satisfied at two corresponding positions along ρ and ρ'. Fix $1 \leq i \leq n$. Since $\mathcal{D}_i$ is deterministic, using the above arguments, we get the last results. □

Let $0 \leq j \leq k$. We assume that χ_j and χ'_j are two valuations for $\{x_1, \ldots, x_j\}$, and D is a tuple of states of the $\mathcal{D}_i$'s. We write $\mathbb{R}^{D,j}_{(\gamma,\gamma')}(\chi_j, \chi'_j)$ if the following property holds for any run ρ from γ:

(i) if ρ is fully above Δ (or equivalently, if $\rho' = \mathsf{Shift}_{+\Lambda}(\rho)$, which starts from γ', is fully above Δ'), then for every $1 \leq g \leq j$, $\chi_j(x_g)(\rho) = \chi'_j(x_g)(\rho')$;

(ii) if ρ is not fully above Δ (equivalently, if $\rho' = \mathsf{Shift}_{+\Lambda}(\rho)$ is not fully above Δ'), then we decompose ρ (resp. ρ') w.r.t. Δ (resp. Δ') and write $\rho = \rho_{\searrow\Delta} \cdot \overline{\rho}$ and $\rho' = \rho'_{\searrow\Delta'} \cdot \overline{\rho}'$. Then:

$$\mathsf{Id}_{\chi_j \overrightarrow{\rho_{\searrow\Delta}}}(\mathsf{lst}(\rho_{\searrow\Delta}), \widetilde{D}) = \mathsf{Id}_{\chi'_j \overrightarrow{\rho'_{\searrow\Delta'}}}(\mathsf{lst}(\rho'_{\searrow\Delta'}), \widetilde{D})$$

with $\widetilde{D} = D_{+\rho_{\searrow\Delta}} = D_{+\rho'_{\searrow\Delta'}}$. Recall that $\chi_j \overrightarrow{\rho_{\searrow\Delta}}$ shifts all strategies in valuation χ_j after the prefix $\rho_{\searrow\Delta}$ (that is, χ_j is the strategy such that $\chi_j \overrightarrow{\rho_{\searrow\Delta}}(\pi) = \chi_j(\rho_{\searrow\Delta} \cdot \pi)$ for every π).

We can see in the $\mathbb{R}$ property an extension of the $\mathbb{P}$ property. We can then get a result similar to Lemma 4.8 for $\mathbb{R}$.

Lemma 4.10. *Fix $0 \leq j < k$, and assume that $\mathbb{R}^{D,j}_{(\gamma,\gamma')}(\chi_j, \chi'_j)$ holds true. Then:*

1. *for every strategy v for x_{j+1} from γ, one can build a strategy $\mathcal{T}(v)$ for x_{j+1} from γ' such that $\mathbb{R}^{D,j+1}_{(\gamma,\gamma')}(\chi_j \cup \{v\}, \chi'_j \cup \{\mathcal{T}(v)\})$ holds true;*

2. *for every strategy v' for x_{j+1} from γ', one can build a strategy $\mathcal{T}^{-1}(v')$ for x_{j+1} from γ such that $\mathbb{R}^{D,j+1}_{(\gamma,\gamma')}(\chi_j \cup \{\mathcal{T}^{-1}(v')\}, \chi'_j \cup \{v'\})$ holds true.*

Sketch of proof. The idea is the following: either we are in case (i), in which case identical (but shifted) strategies can be applied; or we are in case (ii), in which case identical (but shifted) strategies can be applied until counter value Δ (resp. Δ') is hit, then the equality of identifiers allows to apply equivalent strategies. The construction is illustrated in Figure 4.2. □

We use this lemma to transfer a proof that $\gamma \models_\emptyset \phi$ to a proof that $\gamma' \models_\emptyset \phi$. We decompose the proof of this equivalence into two lemmas:

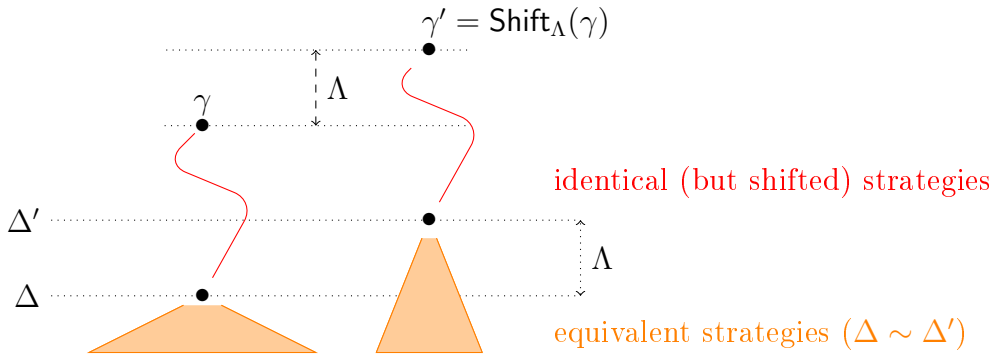


Figure 4.2: Construction in Lemma 4.10 (case (ii))

Lemma 4.11. *Fix D^0 for the tuple of initial states of the $\mathcal{D}_i$'s. Assume that $\mathbb{R}^{D^0,k}_{(\gamma,\gamma')}(\chi,\chi')$ holds (for full valuations χ and χ'). Let $1 \leq i \leq n$, and write $\rho = \mathsf{out}(\beta_i(\chi),\gamma)$ and $\rho' = \mathsf{out}(\beta_i(\chi'),\gamma')$. Then $\rho \models \varphi_i$ if and only if $\rho' \models \varphi_i$. In particular, $\gamma \models_\chi \xi(\beta_i\varphi_i)_{1\leq i\leq n}$ if and only if $\gamma' \models_{\chi'} \xi(\beta_i\varphi_i)_{1\leq i\leq n}$.*

Sketch of proof. As long as runs are above Δ (resp. Δ') they visit states that satisfy exactly the same atomic properties (atomic propositions and counter constraints), hence they progress in each $\mathcal{D}_i$ along the same run. When value Δ (resp. Δ') is hit, they are generated by strategies that have the same level-0 identifiers, which precisely means they are equivalently accepted by each $\mathcal{D}_i$. Hence both outcomes satisfy the same formulas φ_i under the valuation $\beta_i(\chi)$ (resp. $\beta_i(\chi')$). □

We finally show the following lemma. The proof proceeds by induction on the valuation, and by noticing that the hypothesis $\Delta \sim \Delta'$ precisely implies the induction property at level 0 (i.e $\mathbb{R}^{D^0,0}_{(\gamma,\gamma')}(\emptyset,\emptyset)$).

Lemma 4.12. *$\gamma \models_\emptyset \phi$ if and only if $\gamma' \models_\emptyset \phi$.*

This allows us to conclude with the following corollary:

Corollary 4.13. *Λ is a period for the satisfiability of ϕ for configurations with counter values larger than or equal to Δ.*

Furthermore, $\Delta + \Lambda$ is bounded by $M + P \cdot (\mathsf{Tower}(n, k+1))^{|Q|\cdot\prod_{1\leq i\leq} 2^{2^{|\phi|}}} + 1$.

Remark 4.14. *Note that the above proof of existence of a period, though effective (a period can be computed by computing the truth of identifier predicates), does not allow for an algorithm to decide the model checking problem. One possible idea to lift that periodicity result to an effective algorithm would be to bound the counter values; however things are not so easy. In Figure 4.2, equivalent strategies from Δ and Δ' might generate runs with (later on) counter values larger than Δ or Δ' (despite the filled representations staying below the thresholds, counter values of equivalent strategies are not bounded). The decidability status of 1cSL[BG]† (and of 1cSL[BG]) model checking remains open.*

Extension to 1cSL[BG]

We explain how we can extend the previous periodicity analysis to the full logic 1cSL[BG]. Let ϕ be a fixed formula of 1cSL[BG]

$$\phi := Q_1 x_1 \dots Q_k x_k \cdot \xi(\beta_i \varphi_i)_{1 \leq i \leq n}$$

with the same notations than the ones at the beginning of the previous subsection, but now φ_i can use closed formulas of 1cSL[BG] as sub-formulas.

Let Ψ_ϕ be the set of closed sub-formulas of 1cSL[BG] that appear directly under the scope of some φ_i. We will replace sub-formulas of Ψ_ϕ by other formulas involving only (new) atomic propositions and counter constraints. Pick $\psi \in \Psi_\phi$. Let Δ_ψ and Λ_ψ be the threshold and the period mentioned in Corollary 4.13 for ψ. For every location q of the game, the set of counter values c such that $(q, c) \models \psi$ can be written as the union S_q^ψ of a finite (non periodic) set for the values smaller than Δ_ψ and of a periodic set Λ_ψ for the values above Δ_ψ. Note that we know such a set exists, even though there is (for now) no effective procedure to express it. The size of S_q^ψ is 1 (we do not take into account the complexity of writing the precise sets used in the constraint). Expand the set of atomic propositions AP with an extra atomic proposition for each location, say p_q for location q, which holds only at location q. For every $\psi \in \Psi_\phi$, replace that occurrence of ψ in ϕ by formula $\bigwedge_{q \in L} p_q \to (\mathsf{cnt} \in \mathsf{S}_q^\psi)$. This defines formula ϕ', which is now a $\text{1cSL[BG]}^\flat$ formula, and holds equivalently (w.r.t. ϕ) from every configuration of $\mathcal{G}$. The size of ϕ' is that of ϕ. We apply the result of the previous subsection and get a proof of periodicity of the satisfaction relation for ϕ', hence for ϕ.

It remains to compute bounds on the overall period Λ_ϕ and threshold Δ_ϕ. The modulo constraints in ϕ' involve periods Λ_ψ ($\psi \in \Psi_\phi$), and the constants used are bounded by Δ_ψ. So the maximal constant $\mathsf{M}_{\phi'}$ appearing in ϕ' is bounded by $\max(\max_{\psi \in \Psi}(\Delta_\psi), \mathsf{M}_\phi)$ where M_ϕ is the maximal constant used in ϕ, and the l.c.m $\mathsf{P}_{\phi'}$ of the periods appearing in ϕ' is the l.c.m. of the periods used in ϕ (call it P_ϕ) and of the Λ_ψ's (for $\psi \in \Psi_\phi$): hence $\mathsf{P}_{\phi'} \leq \mathsf{P}_\phi \cdot \max_{\psi \in \Psi_\phi}(\Lambda_\psi)^{|\phi|}$ Hence for formula ϕ', we get

$$\Delta_{\phi'} + \Lambda_{\phi'} \quad \leq \quad \mathsf{M}_{\phi'} + \mathsf{P}_{\phi'} \cdot \mathsf{Tower}(n_\phi, k_\phi + 1)^{|\mathsf{Q}| \cdot 2^{2^{|\phi'|}}} + 1$$

We infer the following order of magnitude for $\Delta_\phi + \Lambda_\phi$, where $\omega_{\Psi_\phi} = \max_{\psi \in \Psi_\phi} \omega_\psi$:

$$\begin{aligned} \omega_\phi &\approx \omega_{\Psi_\phi} + \mathsf{M}_\phi^{|\phi|} \cdot (\max_{\psi \in \Psi_\phi} \Lambda_\psi)^{|\phi|} \cdot \mathsf{Tower}(n_\phi, k_\phi + 1)^{|\mathsf{Q}| \cdot 2^{2^{|\phi|}}} \\ &\approx \mathsf{M}_\phi^{|\phi|} \cdot \omega_{\Psi_\phi}^{|\phi|} \cdot \mathsf{Tower}(n_\phi, k_\phi + 1)^{|\mathsf{Q}| \cdot 2^{2^{|\phi|}}} \end{aligned}$$

Using notations of Theorem 4.7, the order of magnitude can therefore be bounded by

$$\mathsf{Tower}\Big(\max_{\theta \in \mathsf{SubForm}(\phi)} n_\theta, \max_{\theta \in \mathsf{SubForm}(\phi)} k_\theta + 1 \Big)^{|\mathsf{Q}| \cdot 2^{2^{|\phi|}}}.$$

□

Remark 4.15. *Note that this proof is non-constructive, even for the period and the threshold, since it relies on the model checking of sub-formulas, which we do not know how to do. We can nevertheless effectively compute a threshold and a period by taking the l.c.m. of all the integers up to the bound over the period and threshold given in this proof. The model checking problem of 1cSL[BG] (and its effectivity) remains an important open problem.*

4.1.3 An application of 1cSL[BG]: MSO$(\omega^\omega, <)$

As said before, while we have a periodicity property, we cannot derive a model checking algorithm for 1cSL[BG]. In this section, we assume that such an algorithm exists and works in "reasonable" time and we use this to improve the validity problem of MSO over $(\omega^\omega, <)$. This problem is shown to be decidable in [12] using automata over linear orderings, but this requires complementing those automata, which has doubly-exponential complexity Hence globally the validity problem of MSO over $(\omega^\omega, <)$ can be decided in time Tower$(|\phi|, 2k + c)$, where ϕ is the MSO formula, k is the number of alternations in the quantifications of ϕ and c is a positive constant. Other algorithms are known for MSO over $(\omega^i, <)$, running in time Tower$(|\phi|, k + c)$, see [11] for more details. However, as explained in [11], the technique they employ, based on tree automata, cannot extend to $(\omega^\omega, <)$, since it would contradict the fact that any tree-automatic ordinal is less than ω^{ω^ω}.

We propose a reduction from the validity problem of MSO over $(\omega^\omega, <)$ toward the model checking problem of 1cSL[BG] over 1-WCGS. This implies that any algorithm for 1cSL[BG]'s model checking working in time Tower$(|\phi|, k+c)$ where ϕ is an SL[BG] formula, k is the number of alternations of quantifications in ϕ and c is any constant, will improve the complexity of the validity problem of MSO over $(\omega^\omega, <)$. The periodicity property of Section 4.1.2 makes the existence of such algorithm for 1cSL[BG] very likely and we have good hope of finding it in the near future.

A note on ordinals

We assume basic knowledge about ordinals and refer to [28] for further details. An *ordinal* is a well-ordered set. It is either 0 (or $\emptyset$), or the successor of an ordinal α, which we write $\alpha + 1$, or a limit ordinal. The first limit ordinal, denoted ω, is identified with the set of natural numbers. For any two ordinals α and β, it holds that $\alpha < \beta$ if and only if $\alpha \in \beta$. Also, any ordinal α is equal to $\{\beta \mid \beta < \alpha\}$. In the following, we will be interested in ω^ω, and we will write equivalently $\alpha \in \omega^\omega$ or $\alpha < \omega^\omega$.

The *Cantor normal form* of ordinals w.r.t. ω [28] allows to (abusively)[4] write any ordinal $\emptyset < \alpha < \omega^\omega$ in a unique way as a sum

$$\alpha = \omega^p n_p + \omega^{p-1} n_{p-1} + \cdots + \omega^1 n_1 + n_0$$

where p and all the n_i's are non-negative integers, and n_p is positive. By extension, we write $\emptyset = 0$, (that is, $p = 0$ and $n_0 = 0$). In the above writing, we call p the degree

[4] The standard normal form would remove terms where $n_i = 0$.

of α and write $\mathsf{deg}(\alpha) = p$. We then write $\mathsf{cnf}(\alpha)$ for the tuple $(n_{\mathsf{deg}(\alpha)}, \ldots, n_0)$ associated with this writing.

This writing does not allow for easy computation of the addition of two ordinals (which we do not need in the sequel), but it allows to very easily compare two ordinals. Indeed, given $\alpha, \beta \in \omega^\omega$, the following holds: $\alpha < \beta$ if and only if one of the two following conditions hold:

- $\mathsf{deg}(\alpha) < \mathsf{deg}(\beta)$, or
- $\mathsf{deg}(\alpha) = \mathsf{deg}(\beta)$ and $\mathsf{cnf}(\alpha) <_{lex} \mathsf{cnf}(\beta)$ where $<_{lex}$ is the lexicographic order, with standard order over the integers.

MSO over ordinals

The logic MSO is defined inductively as follows:

$$\mathsf{MSO} \ni \Phi \ ::= \ (x < y) \mid X(x) \mid \neg\Phi \mid \Phi \vee \Phi \mid \exists X \cdot \Phi \mid \exists x \cdot \Phi$$

where x, y are first-order variables (ranging over elements) and X is a second-order variable (ranging over sets of elements). Let Φ be an MSO formula. A variable x (resp. X) is free in Φ whenever it is not under the scope of a corresponding quantification. The formula is closed whenever it has no free variable. It is then also called a *sentence*.

Let $\Phi(X_1, \ldots, X_k, x_1, \ldots, x_l)$ be an MSO formula with free second-order variables $X_1, \ldots, X_k$ and free first-order variables $x_1, \ldots, x_l$. In our context, it is interpreted over a tuple $(P_1, \ldots, P_k, a_1, \ldots a_l)$, where $P_i \subseteq \omega^\omega$ and $a_j \in \omega^\omega$, with the standard inductive semantics:

- $(\alpha_1, \alpha_2) \models (x_1 < x_2)$ if and only if $\alpha_1 < \alpha_2$;
- $(P, \alpha) \models X(x)$ if and only if $\alpha \in P$;
- $(P_1, \ldots, P_k, a_1, \ldots a_l) \models \exists x \cdot \Phi(X_1, \ldots, X_k, x_1, \ldots, x_l, x)$ if and only if there is $a \in \omega^\omega$ such that $(P_1, \ldots, P_k, a_1, \ldots a_l, a) \models \Phi(X_1, \ldots, X_k, x_1, \ldots, x_l, x)$;
- $(P_1, \ldots, P_k, a_1, \ldots a_l) \models \exists X \cdot \Phi(X_1, \ldots, X_k, X, x_1, \ldots, x_l)$ if and only if there is $P \subseteq \omega^\omega$ such that $(P_1, \ldots, P_k, P, a_1, \ldots a_l, a) \models \Phi(X_1, \ldots, X_k, X, x_1, \ldots, x_l)$;
- Boolean combinations have their standard semantics.

The validity problem

Given a sentence Φ in MSO, the validity problem asks whether Φ is true over $(\omega^\omega, <)$. We recall that there exists an algorithm for the validity of MSO over $(\omega^\omega, <)$ working in time $\mathsf{Tower}(|\phi|, 2k + c)$ where ϕ is the MSO formula, k is the alternation height of the formula and c is a constant; it was shown using automata over linear orderings [12].

Theorem 4.16. *Assume that there exists an algorithm for 1cSL[BG] model checking problem working in time* Tower$(|\phi|, k+c)$ *for a formula* ϕ *with* k *alternations of quantifications, for some fixed constant* c*. Then the logic MSO over* $(\omega^\omega, <)$ *can be decided in time* Tower$(|\Phi|, k+c)$ *for a formula* Φ *with alternation-height* k *and where* c *is the constant of the 1cSL[BG]'s algorithm.*

Proof. The proof uses a reduction to 1cSL[BG] over a fixed game. Many elements are similar to Theorem 2.15 (page 48). Let

$$\Phi := Q_1 X_1 \dots Q_h X_h.\ Q_{h+1} x_{h+1} \dots Q_{h+l} x_{h+l}.\ \Psi(X_1, \dots, X_h, x_{h+1}, \dots, x_{h+l})$$

be a sentence of MSO in prenex normal form. For every i, we have $Q_i \in \{\exists, \forall\}$. The X_i are second-order variables, whereas the x_i are first-order variables; the X_i and the x_i are the only free variables of Ψ.

The proof consists in building a fixed concurrent game $\mathcal{G}$ (independent of the formula Φ) and a formula ϕ such that $\mathcal{G} \models \phi$ if and only if Φ is true over ω^ω.

The game $\mathcal{G}$ has three agents *Decider* (Dec), *Follower* (Fol) and *SecondOrder* (SO). It is the concurrent product of the two graphs depicted in Figure 4.3. That is, a state of $\mathcal{G}$ is a pair (x, y) with $x \in \{a, b, c, D, \mathsf{no}, \mathsf{yes}\}$ and $y \in \{a', b', c', d'\}$, and there is a concurrent move from (x, y) to (x', y') whenever there is a move from x to x' in the left arena and a move from y to y' in the right arena; actions are given to the agent to which belongs the corresponding state. For instance, there is a concurrent move from (a, a') to (b, d') whose first action belongs to Agent Dec and second action to Agent Fol. The weights in $\mathcal{G}$ is taken from the left-hand side arena.

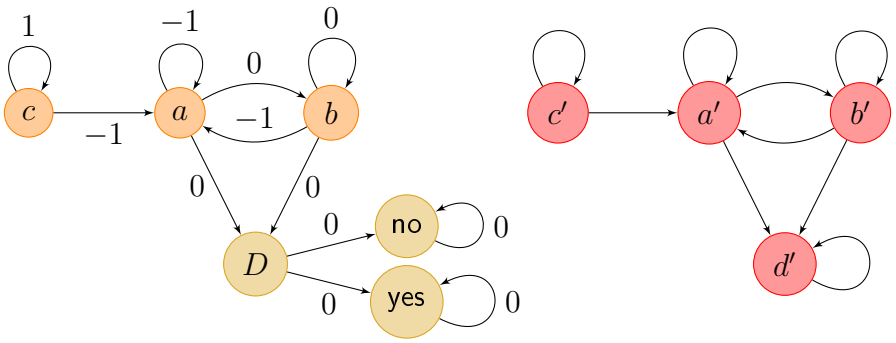


Figure 4.3: Game $\mathcal{G}$ is the concurrent product of these two arenas

An ordinal $\alpha \in \omega^\omega$ will be encoded by the finite outcome

$$\rho_\alpha = (c, c')^{p+2}(a, a')(b, b')^{n_p}(a, a')(b, b')^{n_{p-1}} \dots (a, a')(b, b')^{n_0}(D, d')$$

where $p = \mathsf{deg}(\alpha)$ and $\mathsf{cnf}(\alpha) = (n_p, \dots, n_0)$. The convention for encoding $\emptyset$ is to use the finite outcome $\rho_\emptyset = (c, c')^2(a, a')(D, d')$. Starting from weight 0, we realize that, along ρ_α, the weight is $p+1$ just before leaving (c, c'), and is back to 0 when reaching (D, d'). An infinite outcome ρ is said to encode an ordinal α whenever ρ_α is a prefix of ρ (independently of what happens after ρ_α).

We define the winning condition Ω_{ord} as follows: a run in $\mathcal{G}$ is in Ω_{ord} if and only if

1. it starts in (c, c') with weight 0
2. it visits (c, c') at least twice
3. it visits only states in $\{(a, a'), (b, b'), (c, c')\}$ until reaching (D, d')
4. when it reaches (D, d'), the weight is 0
5. if the weight when reaching (a, a') for the first time is positive, then it should visit (b, b') immediately after.

This winning condition can be expressed in 1cLTL as follows:

$$\psi_\Omega := \begin{cases} (c,c') \wedge \mathsf{cnt} \in \{0\} \wedge \mathbf{X}\,(c,c') \\ \wedge\ \big((a,a') \vee (b,b') \vee (c,c')\big)\ \mathbf{U}\ \big((D,d') \wedge \mathsf{cnt} \in \{0\}\big) \\ \wedge\ (c,c')\ \mathbf{U}\ \Big((a,a') \wedge \big(\mathsf{cnt} \in \{0\} \vee \mathbf{X}\,(b,b')\big)\Big) \end{cases}$$

Lemma 4.17. *Let $\sigma = (\sigma^{Dec}, \sigma^{Fol}, \sigma^{SO})$ be a valuation giving a strategy to each of the three agents, and ρ be its unique outcome. Then ρ belongs to Ω_{ord} (or equivalently satisfies ψ_Ω) if and only if it encodes an ordinal α. The value α is independent of the strategy σ^{SO}, and we then say that $(\sigma^{Dec}, \sigma^{Fol})$ encodes ordinal α.*[5]

Proof. Let ρ be the infinite outcome of the valuation $(\sigma^{\text{Dec}}, \sigma^{\text{Fol}}, \sigma^{\text{SO}})$ from (c, c').

- Assume $\rho \models \psi_\Omega$. Then, just looking at the discrete part of formula ψ_Ω, we get that

$$\rho \in (c,c')^2 (c,c')^* (a,a') \Big((a,a') + (b,b')\Big)^* (D,d') \Big((\mathsf{no},d')^\omega + (\mathsf{yes},d')^\omega\Big)$$

  Now, if $p+2$ is the number of visits to (c, c') in the beginning of ρ, the weight before the first visit to (a, a') is $p + 1$, and then, the number of visits to (a, a') is $p + 1$ since the accumulated weight must be 0 when reaching (D, d'). So we can write:

$$\rho = (c,c')^{p+2} (a,a')(b,b')^{n_p} (a,a')(b,b')^{n_{p-1}} (a,a') \dots (a,a')(b,b')^{n_0} (D',d)\rho'$$

  for some suffix $\rho' \in \{(\mathsf{no}, d')^\omega, (\mathsf{yes}, d')^\omega\}$ and integers $n_p, \dots, n_0$. Furthermore, if $p > 0$ then, by the fifth point in ψ_Ω's definition, $n_p > 0$. If $n_p > 0$, then it implies that ρ encodes the (unique) ordinal $\emptyset < \alpha < \omega^\omega$ with $\mathsf{deg}(\alpha) = p$ and $\mathsf{cnf}(\alpha) = (n_p, \dots, n_0)$. If $n_p = 0$ (in which case $p = 0$ as well), ρ encodes $\emptyset$.

  Note that the strategy σ^{SO} has no impact on the above prefix of ρ, and hence on the ordinal being encoded.

- Assume ρ encodes an ordinal α. Then formula ψ_Ω obviously holds along ρ.

□

[5] Somehow those two strategies agree on "playing α".

From now on, we will write $(\sigma_\alpha^{\text{Dec}}, \sigma_\alpha^{\text{Fol}})$ for a pair of strategies that encodes ordinal α. All the results will be independent of the choice of these strategies.

We now show how we compare two ordinals by playing the strategies of the two different ordinals, as follows:

Lemma 4.18. *Fix two ordinals $\alpha, \beta \in \omega^\omega$, and let σ^{SO} be a strategy of agent SO. Let ρ be the outcome of $(\sigma_\alpha^{Dec}, \sigma_\beta^{Fol}, \sigma^{SO})$ from (c, c') and define ψ_1, ψ_2 and ψ_3 the following LTL formulas*

$$\psi_1 := (\bigvee_{i \in \{a,b,c\}} (i, i')) \,\mathbf{U}\, (a, b') \qquad \psi_2 := (c, c') \,\mathbf{U}\, (a, a') \qquad \psi_3 := (c, c') \,\mathbf{U}\, (a, c')$$

- *Assume $\mathsf{deg}(\alpha) = \mathsf{deg}(\beta)$. Then $\alpha < \beta$ if and only if ρ satisfies ψ_1;*
- *Furthermore $\mathsf{deg}(\alpha) = \mathsf{deg}(\beta)$ if and only if ρ satisfies ψ_2.*
- *Similarly, $\mathsf{deg}(\alpha) < \mathsf{deg}(\beta)$ if and only if ρ satisfies ψ_3*
- *As a consequence, $\alpha < \beta$ if and only if ρ satisfies $\psi_< := \psi_3 \vee (\psi_2 \wedge \psi_1)$.*

Proof. Since $(\sigma_\alpha^{\text{Dec}}, \sigma_\alpha^{\text{Fol}})$ (resp. $(\sigma_\beta^{\text{Dec}}, \sigma_\beta^{\text{Fol}})$) encodes α (resp. β), it holds that the outcome of $(\sigma_\alpha^{\text{Dec}}, \sigma_\alpha^{\text{Fol}}, \sigma^{\text{SO}})$ (resp. $(\sigma_\beta^{\text{Dec}}, \sigma_\beta^{\text{Fol}}, \sigma^{\text{SO}})$) satisfies ψ_Ω.

Now, if $\mathsf{deg}(\alpha) = \mathsf{deg}(\beta) \stackrel{\text{def}}{=} \delta$, then the two strategies together generate $(c, c')^{\delta+1}(a, a')$. Now, as long as $\mathsf{cnf}(\alpha)$ and $\mathsf{cnf}(\beta)$ agree, they generate the corresponding expected outcome. The first time they disagree, it will lead either to (a, b') (in case $\mathsf{cnf}(\alpha) < \mathsf{cnf}(\beta)$) or to (a', b). This paragraph allows to infer the various properties of lemma 4.18. □

We now explain how we encode sets of ordinals (for second-order variables in the logic). It will be the role of Agent SO, which should play from state D to yes (resp. no) whenever the ordinal played so far is (resp. is not) in the encoded set. Let $A \subseteq \omega^\omega$ be a set of ordinals and σ_A^{SO} be a strategy of Agent SO. We say that this strategy encodes A if the following conditions hold: $\sigma_A^{\text{SO}}(\rho_\alpha) = \mathsf{yes}$ if $\alpha \in A$ and $\sigma_A^{\text{SO}}(\rho_\alpha) = \mathsf{no}$ if $\alpha \notin A$ (the value of $\sigma_A^{\text{SO}}(\rho)$ when $\rho \notin \{\rho_\alpha \mid \alpha \in \omega^\omega\}$ is irrelevant). There is not a unique encoding of every second-order set, but a family of such encodings. Nevertheless, any choice that satisfies the above will be correct. The following lemma is then straightforward:

Lemma 4.19. *Let $A \subseteq \omega^\omega$, and σ_A^{SO} be a corresponding strategy for Agent SO. Let $\alpha \in \omega^\omega$ and ρ be the outcome of $(\sigma_\alpha^{Dec}, \sigma_\alpha^{Fol}, \sigma^{SO})$. Then, $\alpha \in A$ if and only if ρ satisfies the formula $\psi_{SO} \stackrel{\text{def}}{=} \mathbf{F}\, yes$.*

We now explain how we transform the MSO formula into a 1cSL[BG] formula

$$\Phi = Q_1 X_1 \dots Q_h X_h.\ Q_{h+1} x_{h+1} \dots Q_{h+l} x_{h+l}.\ \Psi(X_1, \dots, X_h, x_{h+1}, \dots, x_{h+l})$$

We first focus on the block of quantifiers, and then on the quantifier-free formula Ψ. We define the transformation $\mathcal{T}$ as follows:

- $\mathcal{T}(\exists X_i) = \exists \tau_i^{\text{SO}}$ (it will later be assigned to Agent SO)

- $\mathcal{T}(\forall X_i) = \forall \tau_i^{\mathrm{SO}}$ (it will later be assigned to Agent SO)
- $\mathcal{T}(\exists x_i) = \exists \tau_i^{\mathrm{Dec}}\ \exists \overline{\tau}_i^{\mathrm{Fol}}$ (strategy τ_i^{Dec} will later be assigned to Agent Dec and strategy $\overline{\tau}_i^{\mathrm{Fol}}$ to Agent Fol)
- $\mathcal{T}(\forall x_i) = \forall \tau_i^{\mathrm{Dec}}\ \forall \overline{\tau}_i^{\mathrm{Fol}}$ (strategy τ_i^{Dec} will later be assigned to Agent Dec and strategy $\overline{\tau}_i^{\mathrm{Fol}}$ to Agent Fol)
- $\mathcal{T}(Qv\ Qv') = \mathcal{T}(Qv)\ \mathcal{T}(Q'v')$ where Q, Q' are existential or universal quantifications, and v, v' are first-order or second-order variables.

We will need to check that strategies τ_i^{Dec} and $\overline{\tau}_i^{\mathrm{Fol}}$ do actually encode ordinals, and that τ_i^{Dec} and $\overline{\tau}_i^{\mathrm{Fol}}$ agree when quantifying existentially. We will also need to restrict the universal quantifications blocks such as $\forall \tau_i^{\mathrm{Dec}}\ \forall \overline{\tau}_i^{\mathrm{Fol}}$ in $\mathcal{T}$ to actually encode a common ordinal. For that, we define the two following formulas, assuming that $I_\exists$ (resp $I_\forall$) is the subset of indices i in $\{1, \dots, l\}$ such that variable x_i is existentially (resp. universally) quantified in Φ:

$$\kappa_\star = \bigwedge_{i \in I_\star} \mathsf{assign}(\mathrm{Dec}, \tau_i^{\mathrm{Dec}}; \mathrm{Fol}, \overline{\tau}_i^{\mathrm{Fol}})\ \psi_\Omega \qquad (\text{with } \star \in \{\exists, \forall\}).$$

We now define the transformation $\mathcal{T}$ to the quantifier-free formula $\Psi(X_1, \dots, X_h, x_1, \dots, x_l)$. We proceed inductively on the structure of the formula as follows:

- $\mathcal{T}(\Psi_1 \vee \Psi_2) = \mathcal{T}(\Psi_1) \vee \mathcal{T}(\Psi_2)$
- $\mathcal{T}(\neg \Psi) = \neg \mathcal{T}(\Psi)$
- $\mathcal{T}(X_i(x_j)) = \mathsf{assign}(\mathrm{Dec}, \tau_j^{\mathrm{Dec}}; \mathrm{Fol}, \overline{\tau}_j^{\mathrm{Fol}}; \mathrm{SO}, \tau_i^{\mathrm{SO}})\ \psi_{\mathrm{SO}}$
- $\mathcal{T}(x_i < x_j) = \mathsf{assign}(\mathrm{Dec}, \tau_i^{\mathrm{Dec}}; \mathrm{Fol}, \overline{\tau}_j^{\mathrm{Fol}})\ \psi_<$

Finally, we define $\mathcal{T}(\Phi)$ as follows:

$$\mathcal{T}(\Phi) := \mathcal{T}(Q_1 X_1 \dots Q_h X_h Q_{h+1} x_{h+1} \dots Q_{h+l} x_{h+l}) \begin{cases} \kappa_\exists \wedge \\ \kappa_\forall \Rightarrow \mathcal{T}(\Psi(X_1, \dots, X_h, x_{h+1}, \dots, x_{h+}l)) \end{cases}$$

Lemma 4.20. *Let $((c, c'), 0)$ be the initial configuration of $\mathcal{G}$. The sentence Φ is true if and only if $\mathcal{G}, ((c, c'), 0) \models \mathcal{T}(\Phi)$.*

Proof. In this proof, for the sake of readability, instead of writing $\mathcal{G}, ((c, c'), 0) \models_\chi \theta$ (as given by the semantics of the logic), we will assume implicitly $\mathcal{G}$ and $((c, c'), 0)$, and simply write $\chi \models \theta$.

Let $A_1, \dots, A_h$ and $a_{h+1}, \dots, a_{h+l}$ be realisations for the second-order and first-order variables (over ω^ω). We first notice, applying Lemma 4.17, that:

$$(\sigma_{A_1}^{\mathrm{SO}}, \dots, \sigma_{A_h}^{\mathrm{SO}}, \sigma_{a_{h+1}}^{\mathrm{Dec}}, \sigma_{a_{h+1}}^{\mathrm{Fol}}, \dots, \sigma_{a_{h+l}}^{\mathrm{Dec}}, \sigma_{a_{h+l}}^{\mathrm{Fol}}) \models \kappa_\exists \wedge \kappa_\forall$$

Also, it follows from Lemmas 4.18 and 4.19 that:

$$\begin{array}{l}(A_1,\ldots,A_h,a_{h+1},\ldots,a_{h+l}) \models \Psi \text{ iff}\\ (\sigma^{\mathrm{SO}}_{A_1},\ldots,\sigma^{\mathrm{SO}}_{A_h},\sigma^{\mathrm{Dec}}_{a_{h+1}},\sigma^{\mathrm{Fol}}_{a_{h+1}},\ldots,\sigma^{\mathrm{Dec}}_{a_{h+l}},\sigma^{\mathrm{Fol}}_{a_{h+l}}) \models \mathcal{T}(\Psi(X_1,\ldots,X_h,x_{h+1},\ldots,x_{h+l}))\end{array} \quad (4.1)$$

For a subset $I' \subseteq \{h+1,\ldots,h+l\}$, we write $\kappa^{I'}_{\star}$ ($\star \in \{\exists,\forall\}$) for the formula

$$\kappa^{I'}_{\star} = \bigwedge_{i \in I_\star \cap I'} \mathsf{assign}(\mathrm{Dec},\tau^{\mathrm{Dec}}_i;\mathrm{Fol},\overline{\tau}^{\mathrm{Fol}}_i)\ \psi_\Omega$$

Notations. *To simplify reading, we will use the following notations:*

- $(A_{[1,h]},a_{[h+1,h+j]})$ *for the realisation profile* $(A_1,\ldots,A_h,a_{h+1},\ldots,a_{h+j})$
- $(\sigma^{SO}_{A_{[1,h]}},\sigma^{Dec,Fol}_{a_{[h+1,h+j]}})$ *for the valuation* $(\sigma^{SO}_{A_1},\ldots,\sigma^{SO}_{A_h},\sigma^{Dec}_{a_{h+1}},\sigma^{Fol}_{a_{h+1}},\ldots\sigma^{Dec}_{a_{h+j}},\sigma^{Fol}_{a_{h+j}})$
- $Q_{[h+j+1,h+l]}x_{[h+j+1,h+l]}$ *for the quantification* $Q_{h+j+1}x_{h+j+1}\ldots Q_{h+l}x_{h+l}$

We will now show by a downward induction on j ($0 \leq j \leq l$) that for all realisations $A_1,\ldots,A_h$ for the second-order variables and $a_{h+1},\ldots,a_{h+j}$ for the j first first-order variables, the following holds:

$$\begin{array}{l}(A_{[1,h]},a_{[h+1,h+j]}) \models Q_{h+j+1}x_{h+j+1}\ldots Q_{h+l}x_{h+l}\ \Psi \text{ iff}\\ (\sigma^{\mathrm{SO}}_{A_{[1,h]}},\sigma^{\mathrm{Dec,Fol}}_{a_{[h+1,h+j]}}) \models \mathcal{T}(Q_{h+j+1}x_{h+j+1}\ldots Q_{h+l}x_{h+l}) \begin{cases}\kappa^{\geq h+j+1}_{\exists}\wedge\\ \kappa^{\geq h+j+1}_{\forall} \Rightarrow \mathcal{T}(\Psi)\end{cases}\end{array} \quad (4.2)$$

The first part of the proof shows the case $j = l$ through Formula (4.1). We fix $0 < j \leq l$ and we assume the condition expressed in Formula (4.2) to hold for j. We will show it for $j-1$. We distinguish between two cases:

- Case $Q_{h+j} = \exists$. We first assume that the formula below holds

  $$(A_{[1,h]},a_{[h+1,h+j-1]}) \models \exists x_{h+j}Q_{[h+j+1,h+l]}x_{[h+j+1,h+l]}\ \Psi$$

  Let a_{h+j} be a realisation of x_{h+j} such that

  $$(A_{[1,h]},a_{[h+1,h+j-1]},a_{h+j}) \models Q_{[h+j+1,h+l]}x_{[h+j+1,h+l]}\ \Psi$$

  By induction hypothesis, this implies:

  $$(\sigma^{\mathrm{SO}}_{A_{[1,h]}},\sigma^{\mathrm{Dec,Fol}}_{a_{[h+1,h+j]}}) \models \mathcal{T}(Q_{[h+j+1,h+l]}x_{[h+j+1,h+l]})\ \left(\kappa^{\geq h+j+1}_{\exists} \wedge \left(\kappa^{\geq h+j+1}_{\forall} \Rightarrow \mathcal{T}(\Psi)\right)\right)$$

  Hence, by assigning $\sigma^{\mathrm{Dec}}_{a_{h+j}}$ (resp. $\sigma^{\mathrm{Fol}}_{a_{h+j}}$) to $\tau^{\mathrm{Dec}}_{h+j}$ (resp. $\overline{\tau}^{\mathrm{Fol}}_{h+j}$), we get:

  $$(\sigma^{\mathrm{SO}}_{A_{[1,h]}},\sigma^{\mathrm{Dec,Fol}}_{a_{[h+1,h+j-1]}}) \models \exists\tau^{\mathrm{Dec}}_{h+j}\exists\overline{\tau}^{\mathrm{Fol}}_{h+j}\mathcal{T}(Q_{[h+j+1,h+l]}x_{[h+j+1,h+l]}) \begin{cases}\kappa^{\geq h+j}_{\exists}\wedge\\ \kappa^{\geq h+j}_{\forall} \Rightarrow \mathcal{T}(\Psi)\end{cases}$$

since those strategies properly encode an ordinal a_{h+j} (hence the part of $\kappa_\exists$ corresponding to x_{h+j} holds true under that assignment).

Conversely, assume that

$$(\sigma^{\text{SO}}_{A_{[1,h]}}, \sigma^{\text{Dec,Fol}}_{a_{[h+1,h+j-1]}}) \models \exists \tau^{\text{Dec}}_{h+j} \exists \overline{\tau}^{\text{Fol}}_{h+j} \mathcal{T}(Q_{[h+j+1,h+l]} x_{[h+j+1,h+l]}) \begin{cases} \kappa_\exists^{\geq h+j} \wedge \\ \kappa_\forall^{\geq h+j} \end{cases} \Rightarrow \mathcal{T}(\Psi)$$

There exist assignments η^{Dec} and η^{Fol} for τ^{Dec}_{h+j} and $\overline{\tau}^{\text{Fol}}_{h+j}$ such that:

$$(\sigma^{\text{SO}}_{A_{[1,h]}}, \sigma^{\text{Dec,Fol}}_{a_{[h+1,h+j-1]}}, \eta^{\text{Dec}}, \eta^{\text{Fol}}) \models \mathcal{T}(Q_{[h+j+1,h+l]} x_{[h+j+1,h+l]}) \begin{cases} \kappa_\exists^{\geq h+j} \wedge \\ \kappa_\forall^{\geq h+j} \end{cases} \Rightarrow \mathcal{T}(\Psi)$$

Now, due to formula $\kappa_\exists^{\geq h+j}$ (Lemma 4.17), it holds that there is an ordinal a_{h+j} such that η^{Dec} and η^{Fol} encode a_{h+j}. We can then rewrite:

$$(\sigma^{\text{SO}}_{A_{[1,h]}}, \sigma^{\text{Dec,Fol}}_{a_{[h+1,h+j]}}) \models \mathcal{T}(Q_{[h+j+1,h+l]} x_{[h+j+1,h+l]}) \left(\kappa_\exists^{\geq h+j+1} \wedge \left(\kappa_\forall^{\geq h+j+1} \Rightarrow \mathcal{T}(\Psi) \right) \right)$$

By induction hypothesis we get:

$$(A_{[1,h]}, a_{[h+1,h+j]}) \models Q_{[h+j+1,h+l]} x_{[h+j+1,h+l]} \ \Psi$$

Hence:

$$(A_{[1,h]}, a_{[h+1,h+j-1]}) \models \exists x_{h+j} Q_{[h+j+1,h+l]} x_{[h+j+1,h+l]} \ \Psi$$

- Case $Q_{h+l} = \forall$. First assume that

$$(A_{[1,h]}, a_{[h+1,h+j-1]}) \models \forall x_{h+j} Q_{[h+j+1,h+l]} x_{[h+j+1,h+l]} \ \Psi \tag{4.3}$$

and toward a contradiction, assume that

$$(\sigma^{\text{SO}}_{A_{[1,h]}}, \sigma^{\text{Dec,Fol}}_{a_{[h+1,h+j-1]}}) \not\models \forall \tau^{\text{Dec}}_{h+j} \forall \overline{\tau}^{\text{Fol}}_{h+j} \mathcal{T}(Q_{[h+j+1,h+l]} x_{[h+j+1,h+l]}) \begin{cases} \kappa_\exists^{\geq h+j} \wedge \\ \kappa_\forall^{\geq h+j} \end{cases} \Rightarrow \mathcal{T}(\Psi)$$

This means there exists realisations η^{Dec} and $\overline{\eta}^{\text{Fol}}$ for τ^{Dec}_{h+j} and $\overline{\tau}^{\text{Fol}}_{h+j}$ such that

$$(\sigma^{\text{SO}}_{A_{[1,h]}}, \sigma^{\text{Dec,Fol}}_{a_{[h+1,h+j-1]}}, \eta^{\text{Dec}}, \overline{\eta}^{\text{Fol}}) \not\models \mathcal{T}(Q_{[h+j+1,h+l]} x_{[h+j+1,h+l]}) \begin{cases} \kappa_\exists^{\geq h+j} \wedge \\ \kappa_\forall^{\geq h+j} \end{cases} \Rightarrow \mathcal{T}(\Psi)$$

We distinguish between two cases:

  - Case $(\eta^{\text{Dec}}, \overline{\eta}^{\text{Fol}}) \models \kappa_\forall^{\{h+j\}}$. This pair of strategy encodes an ordinal a_{h+j}. Due to assumption (4.3), it holds that:

$$(A_{[1,h]}, a_{[h+1,h+j-1]}, a_{h+j}) \models Q_{[h+j+1,h+l]} x_{[h+j+1,h+l]} \ \Psi$$

By induction hypothesis, we get:

$$(\sigma^{\mathrm{SO}}_{A_{[1,h]}}, \sigma^{\mathrm{Dec,Fol}}_{a_{[h+1,h+j]}}) \models \mathcal{T}(Q_{[h+j+1,h+l]}x_{[h+j+1,h+l]}) \left(\kappa_{\exists}^{\geq h+j+1} \wedge \left(\kappa_{\forall}^{\geq h+j+1} \Rightarrow \mathcal{T}(\Psi)\right)\right)$$

which implies

$$(\sigma^{\mathrm{SO}}_{A_{[1,h]}}, \sigma^{\mathrm{Dec,Fol}}_{a_{[h+1,h+j-1]}}, \eta^{\mathrm{Dec}}, \overline{\eta}^{\mathrm{Fol}}) \models \mathcal{T}(Q_{[h+j+1,h+l]}x_{[h+j+1,h+l]}) \begin{cases} \kappa_{\exists}^{\geq h+j} \wedge \\ \kappa_{\forall}^{\geq h+j} \end{cases} \Rightarrow \mathcal{T}(\Psi)$$

Contradiction.

– Case $(\eta^{\mathrm{Dec}}, \overline{\eta}^{\mathrm{Fol}}) \not\models \kappa_{\forall}^{\{h+j\}}$. This implies that:

$$(\sigma^{\mathrm{SO}}_{A_{[1,h]}}, \sigma^{\mathrm{Dec,Fol}}_{a_{[h+1,h+j-1]}}, \eta^{\mathrm{Dec}}, \overline{\eta}^{\mathrm{Fol}}) \not\models \mathcal{T}(Q_{[h+j+1,h+l]}x_{[h+j+1,h+l]})\ (\kappa_{\exists}^{\geq h+j})$$

which is meaningless since we can always find encodings of ordinals.

We conclude that the assumption was wrong, hence:

$$(\sigma^{\mathrm{SO}}_{A_{[1,h]}}, \sigma^{\mathrm{Dec,Fol}}_{a_{[h+1,h+j-1]}}) \models \forall\tau^{\mathrm{Dec}}_{h+j} \forall\overline{\tau}^{\mathrm{Fol}}_{h+j} \mathcal{T}(Q_{[h+j+1,h+l]}x_{[h+j+1,h+l]}) \begin{cases} \kappa_{\exists}^{\geq h+j} \wedge \\ \kappa_{\forall}^{\geq h+j} \end{cases} \Rightarrow \mathcal{T}(\Psi)$$

Assume now that

$$(\sigma^{\mathrm{SO}}_{A_{[1,h]}}, \sigma^{\mathrm{Dec,Fol}}_{a_{[h+1,h+j-1]}}) \models \forall\tau^{\mathrm{Dec}}_{h+j} \forall\overline{\tau}^{\mathrm{Fol}}_{h+j} \mathcal{T}(Q_{[h+j+1,h+l]}x_{[h+j+1,h+l]}) \begin{cases} \kappa_{\exists}^{\geq h+j} \wedge \\ \kappa_{\forall}^{\geq h+j} \end{cases} \Rightarrow \mathcal{T}(\Psi)$$

This holds in particular when $\tau^{\mathrm{Dec}}_{h+j}$ and $\overline{\tau}^{\mathrm{Fol}}_{h+j}$ encode an arbitrary ordinal, which allows to conclude by induction hypothesis.

Getting rid of second-order quantifications is made using a bijection between strategies of Agent SO and assignments of second-order variables (Lemma 4.19). This concludes the proof. □

By this reduction, we can deduce a procedure for the $\mathsf{MSO}(\omega^\omega, <)$ validity problem from an algorithm for 1cSL[BG]'s model checking. The size of formula $\mathcal{T}(\Phi)$ is linear in the size of Φ, and the number of alternations is identical. Therefore, an algorithm that works in time $\mathsf{Tower}(|\phi|, k+c)$ for a 1cSL[BG] formula ϕ with k alternations of quantifications (and c a fixed constant) will give a procedure in $\mathsf{Tower}(|\Phi|, k+c)$ for a MSO formula Φ with k alternations of quantifications. This concludes the proof of Theorem 4.16. □

Remark 4.21. *The current construction does not extend to ordinals higher than ω^ω since there is no Cantor normal form based on ω for larger ordinals.*

4.2 With energies

In Minsky's proof of undecidability for reachability in two-counter machines, two aspects appear essential: having at least two counters and being able to test equality of each counter with 0. To get around the undecidability, we have two main possibilities: working with a single weight or simplifying the constraints which are allowed. In Section 4.1 we studied the first option, we now focus on the second one. Energy constraints are a weaker form of quantitative assertions, the idea is similar to counter constraints as it works on weighted games but limits the constraints to upward-closed sets and their complements[6].

In contrast with counters, reachability (or objectives characterised through ω-regular conditions) is decidable in multi-dimensional energy games. In this section, we propose an adaptation of SL with energy assertions and prove its multi-dimensional model checking to be undecidable.

4.2.1 Adding energy constraints to SL[BG]

We reuse WCGS (see definition in Section 4.1.1) when working with energy assertions; in particular, we keep a definition of path that forbid strictly negative weights[7]. We start by defining energy constraints formally. For this, we recall that an upward-closed set S of $\mathbb{N}^n$ is a set such that if $(s_1, \dots, s_n) \in \mathsf{S}$ then $(s_1+i_1, \dots, s_n+i_n) \in \mathsf{S}$ for any $(i_1, \dots, i_n) \in \mathbb{N}^n$.

Definition 4.22 (Energy Constraints)**.**
An energy constraint on n weights is an upward-closed subset S of $\mathbb{N}^n$.

By taking $\mathsf{S} = \mathbb{N}^n$ we retrieve the constraint that all weights should be non-negative. We can now update SL with the new constraints.

Definition 4.23 (Strategy logic with energies: eSL)**.**
The logic eSL is built upon a dimension n, a set Agt of agents, a set AP of atomic propositions and a set $\mathcal{V}$ of variables. Its formulas are to be evaluated on a n-WCGS using Agt and AP respectively as their set of agents and atomic propositions. The eSL formulas are constructed using the following grammar:

$$\mathsf{eSL} \in \phi \ ::= \ \exists x.\phi \mid \mathsf{assign}(A, x).\phi \mid \phi \vee \phi \mid \neg\phi \mid \phi \mathbf{U}\, \phi \mid \mathbf{X}\, \phi \mid p \mid \mathsf{cnt} \in \mathsf{S}$$

with S an energy constraint, p an atomic proposition, x a variable of $\mathcal{V}$ and A an agent of Agt.

We recall the semantics of the $\mathsf{cnt} \in \mathsf{S}$ operator, which is unchanged from the one used for cSL (SL with counters constraints):

$$\mathcal{G}, (q, c) \models_\chi \mathsf{cnt} \in \mathsf{S} \quad \Leftrightarrow \quad c \in \mathsf{S}$$

[6]Many of the works in the literature ([4, 14, 13]) use constraints forbidding any of the weights to become negative. We can however be more liberal and ask for the weights to belong to upward-closed sets or complements of upward-closed sets without loss of generality. We can easily establish polynomial reductions from one framework to the other.

[7]However, similarly to the previous sections, this has no impact on the results developed below.

Remark 4.24. *We draw a parallel between energy constraints, eSL[BG] and energy games in the sense that we aim to keep all the energies above a given threshold in all our goals. This type of constraints were the first ones studied in energy games. Research around energy games has since then moved towards more complex questions (parameters, cost minimisation...). The parallel with energy games however stops there and it is unclear whether the more advanced questions make sense when coupled with SL.*

We also define the (flat) analogue $\mathsf{eSL[BG]}^\flat$ of $\mathsf{SL[BG]}^\flat$ by

$$\begin{aligned}\mathsf{eSL[BG]}^\flat \ni \phi &::= \exists x.\phi \mid \forall x.\phi \mid \xi\\ \xi &::= \xi \wedge \xi \mid \xi \vee \xi \mid \beta\\ \beta &::= \mathsf{assign}(A,x).\beta \mid \varphi\\ \varphi &::= \varphi \vee \varphi \mid \neg\varphi \mid \varphi \mathbf{U} \varphi \mid \mathbf{X}\varphi \mid p \mid \mathsf{cnt} \in \mathsf{S}\end{aligned}$$

We get $\mathsf{eSL[CG]}^\flat$, the enriched version of $\mathsf{SL[CG]}^\flat$, by taking the grammar of eSL[BG] and restraining ξ-type formulas to $\xi ::= \xi \wedge \xi \mid \beta$. As always, the full fragments allow closed formulas at the atomic propositions' level.

4.2.2 Model checking of eSL[BG]

As reachability is decidable in multi-dimensional energy games (see [14] for an algorithm), we hoped that eSL[BG] would be decidable. The theorem below proves that we were wrong and shows that the conjunctive fragment eSL[CG] is already sufficient to get undecidability.

Theorem 4.25. *The eSL[CG] model checking problem over 2-WCGS is undecidable*

Proof. The proof consists in a reduction from the halting problem for deterministic two-counter automata. We recall the definition below and remind the reader that the halting problem for two-counter automata is undecidable (Minsky [37]).

Definition 3.3. *A deterministic two-counter automaton is a tuple $\mathcal{M} = \langle S, E, s_0, s_h\rangle$ where S is the state space, s_0 is an initial state of $Q_\mathcal{M}$ and $s_h \in Q_\mathcal{M}$ is a halting state. $E : S \to \{c_1, c_2\} \times \{S \cup S \times S\}$ is the transition function. Transitions of form $E(s) = (c, s')$ increment the counter c and go to s', while transitions of form $E(s) = (c, s', s'')$ either go to s' if the counter c equals 0 or decrement c and go to s'' if $c > 0$.*

Fix a two-counter automaton $\mathcal{M} := \langle \mathsf{S}, \mathsf{E}, \mathsf{s}_0, \mathsf{s}_h\rangle$ with two counters c_1 and c_2. Without loss of generality, we assume $\mathcal{M}$ to have no self loop.

Building the game

We start by defining a 2-WCGS $\mathcal{G} := \langle \mathsf{AP}, \mathsf{Agt}, \mathsf{Q}, \mathsf{Act}, \Delta, \mathsf{labels}\rangle$. Figure 4.4 illustrates the construction. Formally, we set

- There are two agents *Decider* (Dec) and *Checker* (Che) in Agt.

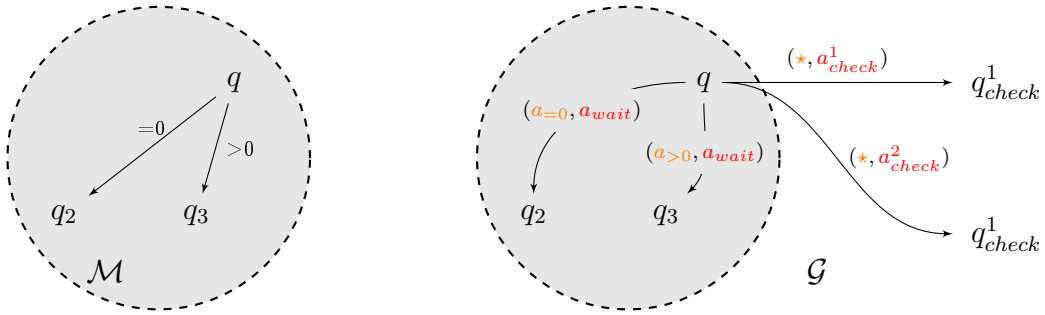


Figure 4.4: The two-counter automaton $\mathcal{M}$ on the left and the concurrent game $\mathcal{G}$ on the right.

- The state space Q is the union of S plus 2 new states: q^1_{check} and q^2_{check}.
- The actions set Act is made of five actions: $a_{>0}$, $a_{=0}$, a_{wait}, a^1_{check} and a^2_{check}. The first two actions belong to Agent Dec and the last three to Agent Che.
- There is no transition from q^1_{Check} and q^2_{Check}. For $q \in \mathsf{S}$ and a move vector $m : \mathsf{Act}^{\mathsf{Agt}}$, $\Delta(q, m)$ is defined as follows:
  - if Che plays a^1_{check}, resp. a^2_{check}, then $\Delta(q, m) := q^1_{check}$, resp. $\Delta(q, m) := q^2_{check}$.
  - if Che plays a_{wait} then there is a unique transition leaving q (when viewing q as a state of $\mathcal{M}$)
    - if it is of the form $\mathsf{E}(q) = (\mathsf{c}, q')$, meaning it is an incrementing edge on counter c, then we set $\Delta(q, m) := q'$.
    - if it is of the form $\mathsf{E}(q) = (\mathsf{c}, q', q'')$, meaning it is a zero test or decrementation on counter c, then either Dec plays $a_{=0}$ and we set $\Delta(q, m) := q'$ or Dec plays $a_{>0}$ then we set $\Delta(q, m) := q''$.
- The weights of the transitions in $\mathcal{G}$ follow the weights of $\mathcal{M}$. We also create a weight $(-1, 0)$, resp. $(0, -1)$, on the edges going to q^1_{check}, resp. q^2_{check}.
- We label the states of $\mathcal{G}$ that are accepting in $\mathcal{M}$ by an atomic proposition p_{acc}. q^1_{check} and q^2_{check} are respectively labelled by p^1_{check} and p^2_{check}. We label each state of S by an eponymous proposition, for example a state q will be labelled p^q. For any counter c and any transition of the form $\mathsf{E}(q) = (\mathsf{c}, q')$, we label both q and q' by $p_{\mathsf{E}(q)=(\mathsf{c},q')}$. Similarly, for any transition of the form $\mathsf{E}(q) = (\mathsf{c}, q', q'')$, we label q, q' and q'' by $p_{\mathsf{E}(q)=(\mathsf{c},q',q'')}$.
- We use the notation q_{ini} to talk about the starting state s_0 of $\mathcal{M}$ in the context of $\mathcal{G}$[8].

[8]This is mostly a question of terminology: do we see the states of $\mathcal{M}$ in $\mathcal{G}$ as copies or as the same entity? To avoid confusion and to make clear which structure we are working on, we choose the first option and use the q_{ini} notation. For the partisans of the second approach, for all purposes $\mathsf{s}_0 = q_{ini}$.

Building the formula

It remains to specify an eSL[CG] formula to complete the reduction. We give the final formula so the reader can get a general idea before proving the correctness of the reduction.

$$\phi \;:=\; \exists x_1.\, \forall x_2.\, \psi_1 \wedge \psi_2 \wedge \psi_3$$

where

$$\psi_1 := \mathsf{assign}(\mathrm{Dec}, x_1; \mathrm{Che}, x_2).\ \mathbf{G}\,(\neg p^1_{check} \wedge \neg p^2_{check}) \Rightarrow \mathbf{F}\, p_{acc}$$

$$\psi_2 := \mathsf{assign}(\mathrm{Dec}, x_1; \mathrm{Che}, x_2).\ \mathbf{G} \bigwedge_{q,q',q''\in \mathsf{S}} \begin{cases} p^q \wedge p_{\mathsf{E}(q)=(\mathsf{c}_1,q',q'')} \\ \wedge\ \mathbf{X}\,(p^{q''} \wedge p_{\mathsf{E}(q)=(\mathsf{c}_1,q',q'')}) \\ \wedge\ \mathbf{X}^2 p^1_{check} \end{cases} \Rightarrow \mathsf{cnt} \in \mathbb{N}\backslash\{0\} \times \mathbb{N}$$
$$\begin{cases} p^q \wedge p_{\mathsf{E}(q)=(\mathsf{c}_2,q',q'')} \\ \wedge\ \mathbf{X}\,(p^{q''} \wedge p_{\mathsf{E}(q)=(\mathsf{c}_2,q',q'')}) \\ \wedge\ \mathbf{X}^2 p^2_{check} \end{cases} \Rightarrow \mathsf{cnt} \in \mathbb{N} \times \mathbb{N}\backslash\{0\}$$

$$\psi_3 := \mathsf{assign}(\mathrm{Dec}, x_1; \mathrm{Che}, x_2).\ \begin{cases} \neg\mathbf{F} \begin{cases} p^q \wedge p_{\mathsf{E}(q)=(\mathsf{c}_1,q',q'')} \\ \wedge\ \mathbf{X}\,(p^{q'} \wedge p_{\mathsf{E}(q)=(\mathsf{c}_1,q',q'')}) \\ \wedge\ \mathbf{X}^2 p^1_{check} \wedge\ \mathsf{cnt} \in \mathbb{N}\backslash\{0\} \times \mathbb{N} \end{cases} \\ \wedge \\ \neg\mathbf{F} \begin{cases} p^q \wedge p_{\mathsf{E}(q)=(\mathsf{c}_2,q',q'')} \\ \wedge\ \mathbf{X}\,(p^{q'} \wedge p_{\mathsf{E}(q)=(\mathsf{c}_2,q',q'')}) \\ \wedge\ \mathbf{X}^2 p^2_{check} \wedge\ \mathsf{cnt} \in \mathbb{N} \times \mathbb{N}\backslash\{0\} \end{cases} \end{cases}$$

Correctness of the reduction

Consider a strategy δ^{Dec} for Dec. We associate with δ^{Dec} a sequence $\rho := (q_i, c^i_1, c^i_2)_{i\in\mathbb{N}}$ with $c^i_1, c^i_2 \in \mathbb{Z}$ of configurations of $\mathcal{G}$ defined by $q_{i+1} = \delta^{\mathrm{Dec}}(\rho_{\leq i})$ and where the values c^{i+1}_1 and c^{i+1}_2 are updated accordingly to the weights on the transition from q^i to q^{i+1} (due to the determinism of $\mathcal{M}$, there can only be one transition from q^i to q^{i+1}). Note that δ^{Dec} does not have access to the two new states, it only decides between the two choices offered by the branching (that corresponds to a zero test in $\mathcal{M}$). This implies that the sequence ρ always stays in the copy of $\mathcal{M}$, therefore $\delta^{\mathrm{Dec}}(\rho_{\leq i})$ is always defined and ρ is properly defined. Note that it may be the case that δ^{Dec} does not follow zero tests. In particular, if a value c^i_1 (or c^i_2) is in $\mathbb{Z} - \mathbb{N}$ then ρ is not a (well-defined) path.

Because ρ never reaches q^1_{check} and q^2_{check}, we can see it as a sequence of configurations in $\mathcal{M}$. The proposition below characterises when ρ can be seen as a proper path in $\mathcal{M}$.

Proposition 4.26. *The sequence ρ is the unique path in $\mathcal{M}$ if and only if for any strategy δ^{Che}, it holds $\{Dec \to \delta^{Dec};\ Che \to \delta^{Che}\}$ satisfies ψ_2 and ψ_3*

Proof. First, assume that ρ is the proper path in $\mathcal{M}$, meaning ρ takes the zero-test edges appropriately:

$$\forall j \in \{1,2\}\ \forall i \in \mathbb{N} \qquad \Big[\mathsf{E}(q_i) = (\mathsf{c}_j, q_{i+1}, q'') \text{ for some } q'' \in \mathsf{S}\Big] \Rightarrow c_i = 0 \tag{4.4}$$

$$\forall j \in \{1,2\}\ \forall i \in \mathbb{N} \qquad \Big[\mathsf{E}(q_i) = (\mathsf{c}_j, q', q_{i+1}) \text{ for some } q' \in \mathsf{S}\Big] \Rightarrow c_i > 0 \tag{4.5}$$

Then, for any strategy δ^{Che} and writing $\chi := \{x_1 \to \delta^{\text{Dec}}; x_2 \to \delta^{\text{Che}}\}$, it holds

$$\mathcal{G}, (q_{ini}, (0,0)) \models_\chi \psi_2 \wedge \psi_3 \tag{4.6}$$

Indeed, consider an integer i where a zero test $\mathsf{E}(q_i) = (\mathsf{c}^j, q_{i+1}, q'')$ occurs on counter j. If δ^{Che} plays to p^j_{check} after $i+1$ steps, because the zero edges are taken accordingly, we must have $c^j_i = 0$ and therefore η can only reach p^j_{check} at step $i+2$ with a null value in the j counter and formula ψ_3 must hold at step i. Also, $\mathbf{X}\,(p^{q''} \wedge p_{\mathsf{E}(q)=(\mathsf{c}_2,q',q'')})$ is not verified and ψ_2 trivially holds, hence Formula (4.6) is satisfied at step i. If $\rho(i+1) = q''$, we can apply the inverse reasoning and Formula (4.6) still holds at step i.

Now, assume that ρ is not a proper path of $\mathcal{M}$. As said before, ρ always stays within the copy of $\mathcal{M}$ and never reaches q^1_{check} and q^2_{check}. So, the first components of each element of ρ form a sequence of states in $\mathcal{M}$. The only way for ρ not to be a path is to wrongly manage the counter, i.e. an error in the decisions of δ^{Dec}. We treat the case where ρ went by the non-zero first counter transition with a null first counter, the inverse case and the ones with the second counter are similar. There is $i_0 \in \mathbb{N}$ such that $\mathsf{E}(q_{i_0}) = (\mathsf{c}_1, q', q_{i_0+1})$ for some $q' \in \mathsf{S}$ and with $c^1_i(\mathsf{c}_1) = 0$. The path ρ is of the form (considering the first counter)

$$(q_1,1,1)\ldots(q_{i_0},1,c^2_{i_0}).(q_{i_0+1},0,\star).(q^1_{Check},0,\star)\ldots$$

so $\mathcal{G}, (q_{ini},0,0) \not\models_\chi \psi_3$ and $\mathcal{G}, (q_{ini},0,0) \not\models_\chi \psi_2 \wedge \psi_3$. This combined with the previous paragraph gives us Proposition 4.26. □

It remains to characterise when ρ is accepting, which is done in the proposition below. The proof is straightforward hence omitted.

Proposition 4.27. *Fix two strategies δ^{Dec} and δ^{Che} such that, writing $\chi := \{x_1 \to \delta^{Dec};\ x_2 \to \delta^{Che}\}$, it holds*

$$\mathcal{G}, (q_{ini}, (0,0)) \models_\chi \psi_2 \wedge \psi_3 \wedge \mathit{assign}(Dec, x_1; Che, x_2).\ \mathbf{G}\,(\neg p^1_{check} \wedge \neg p^2_{check})$$

Write ρ for the sequence of configurations associated with δ^{Dec}, then ρ sees an accepting state if and only if $\mathcal{G}, (q_{ini}, (0,0)) \models_\chi \psi_1$.

From there, by combining Propositions 4.26 and 4.27, we get that

$$\text{The unique path in } \mathcal{M} \text{ is accepting} \quad \Leftrightarrow \quad \begin{cases} \text{There exists a strategy } \delta^{\text{Dec}} \text{ such that} \\ \mathcal{G}, (q_{ini}, (0,0)) \models_{\{x_1 \to \delta^{\text{Dec}}\}} \forall x_2.\ \psi_1 \wedge \psi_2 \wedge \psi_3 \end{cases}$$

which concludes the proof of Theorem 4.25. □

Remark 4.28. *The game used in Theorem 4.25 is concurrent. The same result holds on weighted turn-based games. The proof would just decompose sequentially the choices of Che and Dec by adding some extra states.*

Theorem 4.25 does not rely on complex numerical constraints: each constraint asks for one of the energies to be positive. This is enough to get an undecidable model checking problem for eSL[CG]. The result holding for WCGS with only two agents shows that there is little hope to regain decidability without restraining ourselves to a unique weight or going for WCGS with one agent (which brings us back to closed systems). For the model checking over WCGS with a single weight, we believe eSL[BG] to be decidable. Energy constraints are (by definition) strictly less expressive than counter constraints and, as said in Section 4.1, we believe 1cSL[BG] (the development of SL[BG] with counter constraints on WCGS with a single weight) to be decidable.

One possibility is to force energy constraints to appear under the scope of an even number of negation, still allowing constraints on upward-closed sets but forbidding their negations. Without the complements of upward-closed sets, we can check (using the technique developed to prove Theorem 4.25) that configurations of the form $(q, 0, c_2)$ and $(q, c_1, 0)$ take the zero-test edges properly. We however cannot enforce proper behaviour for configurations of other forms without the complements of upward-closed sets. What happens to the undecidability of eSL[CG] is then unclear and worth investigating.

4.3 Conclusion

Like most temporal logics for multi-agents systems, SL can be enriched with quantitative constraints. These constraints can be of many kinds. We have proposed two versions: one with counter constraints (cSL) and the other with energy constraints (eSL).

The model checking of cSL on WCGS with more than two counters is trivially undecidable. We tried in Section 4.1.2 to get a decidability result for cSL on 1-WCGS (WCGS with a single counter) but failed to design an algorithm. We however managed to prove a periodicity property for the satisfaction relation. This gives us hope to find in the near future a working algorithm.

We have also shown in Section 4.1.3 a potential application of cSL on 1-WCGS by forging a correspondence with MSO on ordinals. While the lack of algorithm for cSL makes this correspondence only theoretical, it highlights the large expressive power of cSL, even on 1-WCGS: without much difficulty, cSL can encode well-ordered sets, and especially ω^ω which cannot be encoded through tree automata.

With the decidability of reachability in multi-dimensional energy games, we had hope that the eSL[BG] model checking problem would be decidable. Our aspirations however died in Section 4.2.2 when we found eSL[CG] (the conjunctive fragment of eSL[BG]) to be undecidable. It seems that not much can be done for eSL and its expressiveness is too powerful to ever regain decidability. The only three options would be to restrict ourselves to a single objective (some eSL[1G] type of logic), to restrict ourselves to a single weight or to force energy constraints to be under a even number of negation (i.e. forbidding the complements of upward-closed sets).

4.A Annex

We give the full proof of the intermediary results (Lemma 4.10, Lemma 4.11 and Lemma 4.12) used to prove the periodicity theorem of Section 4.1.2 (page 85) which we recall below

Theorem 4.7. *Let $\mathcal{G}$ be a 1-WCGS, and ϕ be a 1cSL[BG] formula. Then there exists a threshold $\Delta \geq 0$ and a period $\Lambda \geq 0$ for the truth value of ϕ over $\mathcal{G}$. That is, for every configuration (q, c) of $\mathcal{G}$ with $c \geq \Delta$, for every $k \in \mathbb{N}$, $\mathcal{G}, (q, c) \models \phi$ if and only if $\mathcal{G}, (q, c + k \cdot \Lambda) \models \phi$.*

Furthermore the order of magnitude for $\Delta + \Lambda$ is bounded by

$$\mathit{Tower}\Big(\max_{\theta \in \mathit{SubForm}(\phi)} n_\theta, \max_{\theta \in \mathit{SubForm}(\phi)} k_\theta + 1\Big)^{|Q| \cdot 2^{2^{|\phi|}}}$$

where Q is the state space of $\mathcal{G}$, $\mathit{SubForm}(\phi)$ is the set of 1cSL[BG] sub-formulas of ϕ, k_θ is the number of quantifier alternations in θ, and n_θ is the number of different assignments used in θ.

Lemma 4.10. *Fix $0 \leq j < k$, and assume that $\mathbb{R}^{D,j}_{(\gamma,\gamma')}(\chi_j, \chi'_j)$ holds true. Then:*

1. *for every strategy v for x_{j+1} from γ, one can build a strategy $\mathcal{T}(v)$ for x_{j+1} from γ' such that $\mathbb{R}^{D,j+1}_{(\gamma,\gamma')}(\chi_j \cup \{v\}, \chi'_j \cup \{\mathcal{T}(v)\})$ holds true;*
2. *for every strategy v' for x_{j+1} from γ', one can build a strategy $\mathcal{T}^{-1}(v')$ for x_{j+1} from γ such that $\mathbb{R}^{D,j+1}_{(\gamma,\gamma')}(\chi_j \cup \{\mathcal{T}^{-1}(v')\}, \chi'_j \cup \{v'\})$ holds true.*

Proof. We prove the first property. The second property is proven similarly by replacing $\mathsf{Shift}_{-\Lambda}$ with $\mathsf{Shift}_{+\Lambda}$.

We fix a new strategy v for variable x_{j+1} from γ. We define the lifted strategy $\mathcal{T}(v)$ for variable x_{j+1} (which we will add to valuation χ'_j) from γ' as follows:

- for every (finite) ρ' from γ' that is a prefix along which the counter is always larger than Δ', we define $\mathcal{T}(v)(\rho') = v(\rho)$, where $\rho = \mathsf{Shift}_{-\Lambda}(\rho')$ (note that in that case, the counter is always larger than Δ along ρ, and ρ starts at configuration γ), so this is well-defined;
- if ρ' hits value Δ', then decompose ρ' w.r.t. Δ' as $\rho'_{\searrow\Delta'} \cdot \overline{\rho}'$. Similarly, decompose $\rho = \mathsf{Shift}_{-\Lambda}(\rho')$ w.r.t. Δ, yielding $\rho = \rho_{\searrow\Delta} \cdot \overline{\rho}$. It is not difficult to see that $\rho_{\searrow\Delta} = \mathsf{Shift}_{-\Lambda}(\rho'_{\searrow\Delta'})$. By hypothesis, it holds

  $$\mathsf{Id}_{\chi_j \xrightarrow{\rho_{\searrow\Delta}}}(\mathsf{lst}(\rho_{\searrow\Delta}), \widetilde{D}) = \mathsf{Id}_{\chi'_j \xrightarrow{\rho'_{\searrow\Delta'}}}(\mathsf{lst}(\rho'_{\searrow\Delta'}), \widetilde{D}))$$

  (with $\widetilde{D} = D_{+\rho_{\searrow\Delta}} = D_{+\rho'_{\searrow\Delta'}}$).

  By Lemma 4.8 we find a strategy v' from $\mathsf{lst}(\rho'_{\searrow\Delta'})$ for variable x_{j+1} such that

  $$\mathsf{Id}_{(\chi_j \cup \{v\}) \xrightarrow{\rho_{\searrow\Delta}}}(\mathsf{lst}(\rho_{\searrow\Delta}), \widetilde{D}) = \mathsf{Id}_{\chi'_j \xrightarrow{\rho'_{\searrow\Delta'}} \cup \{v'\}}(\mathsf{lst}(\rho'_{\searrow\Delta'}), \widetilde{D}).$$

  We then define $\mathcal{T}(v)(\rho') = v'(\overline{\rho}')$. The property $\mathbb{R}^{D,j+1}_{(\gamma,\gamma')}(\chi_j \cup \{v\}, \chi'_j \cup \{\mathcal{T}(v)\})$ holds.

□

The construction made in the above lemma is illustrated in Figure 4.5.

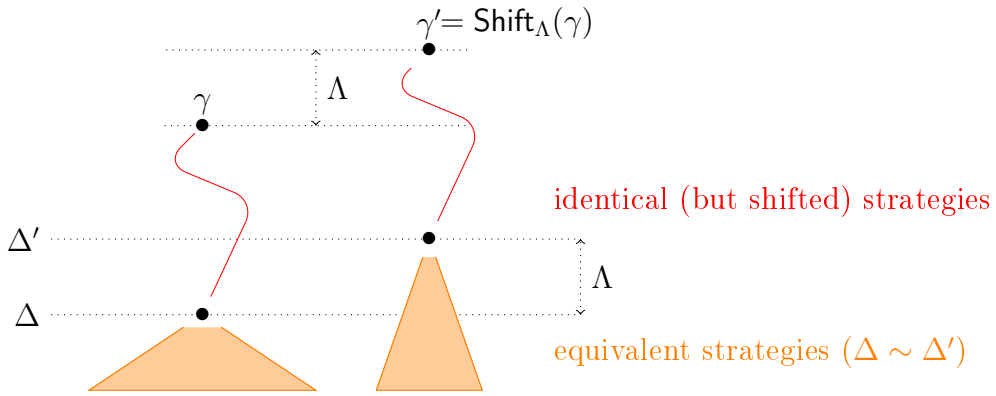


Figure 4.5: Construction in Lemma 4.10 (case (ii))

Lemma 4.11. *Fix D^0 for the tuple of initial states of the $\mathcal{D}_i$'s. Assume that $\mathbb{R}^{D^0,k}_{(\gamma,\gamma')}(\chi,\chi')$ holds (for full valuations χ and χ'). Let $1 \leq i \leq n$, and write $\rho = \mathsf{out}(\beta_i(\chi),\gamma)$ and $\rho' = \mathsf{out}(\beta_i(\chi'),\gamma')$. Then $\rho \models \varphi_i$ if and only if $\rho' \models \varphi_i$. In particular, $\gamma \models_\chi \xi(\beta_i\varphi_i)_{1\leq i\leq n}$ if and only if $\gamma' \models_{\chi'} \xi(\beta_i\varphi_i)_{1\leq i\leq n}$.*

Proof. We distinguish between two cases:

- Assume that ρ is fully above Δ. By definition of property $\mathbb{R}^{D^0,k}_{(\gamma,\gamma')}(\chi,\chi')$, it holds that $\rho' = \mathsf{Shift}_{+\Lambda}(\rho)$. Applying Lemma 4.9 to all prefixes of ρ and ρ' (which are all above Δ, resp. Δ'), we get that they follow the same paths in all automata $\mathcal{D}_i$'s, hence for each $1 \leq i \leq n$, $\rho \models \psi_i$ iff $\rho' \models \psi_i$.

- Assume that ρ is not fully above Δ. Then, by definition of property $\mathbb{R}^{D^0,k}_{(\gamma,\gamma')}(\chi,\chi')$, ρ' is not fully above either, and $\rho'_{\searrow\Delta'} = \mathsf{Shift}_{+\Lambda}(\rho_{\searrow\Delta})$. Also,

$$\mathsf{Id}_{\chi_{\overrightarrow{\rho_{\searrow\Delta}}}}(\mathsf{lst}(\rho_{\searrow\Delta}), D) = \mathsf{Id}_{\chi'_{\overrightarrow{\rho'_{\searrow\Delta'}}}}(\mathsf{lst}(\rho'_{\searrow\Delta'}), D)$$

  with $D = D^0_{+\rho_{\searrow\Delta}} = D^0_{+\rho'_{\searrow\Delta'}}$ (by Lemma 4.9).

  There exists some location $\hat{q}$ such that $\mathsf{lst}(\rho_{\searrow\Delta}) = (\hat{q},\Delta)$ and $\mathsf{lst}(\rho'_{\searrow\Delta'}) = (\hat{q},\Delta')$, and by definition of the id, this means that for every $1 \leq i \leq n$, the two following properties are equivalent:

  - $\overline{\rho} = \mathsf{out}\big(\beta_i(\chi_{\overrightarrow{\rho_{\searrow\Delta}}}), (\hat{q},\Delta)\big)$ is accepted by $\mathcal{D}_i$ from d_i
  - $\overline{\rho}' = \mathsf{out}\big(\beta'_i(\chi'_{\overrightarrow{\rho'_{\searrow\Delta'}}}), (\hat{q},\Delta')\big)$ is accepted by $\mathcal{D}_i$ from d_i

  We conclude by noticing that $\rho = \rho_{\searrow\Delta} \cdot \overline{\rho}$ and $\rho' = \rho'_{\searrow\Delta'} \cdot \overline{\rho}'$, which are then equivalently accepted or rejected by each of the $\mathcal{D}_i$.

□

Lemma 4.12. *$\gamma \models_\emptyset \phi$ if and only if $\gamma' \models_\emptyset \phi$.*

Proof. For every $0 \leq j \leq k$, we write $\phi_j = Q_{j+1}x_{j+1} \dots Q_l x_l \; \xi(\beta_i \psi_i)_{1 \leq i \leq n}$.

We show by induction that if χ_j and χ'_j are valuations such that $\mathbb{R}^{D^0,j}_{(\gamma,\gamma')}(\chi_j, \chi'_j)$ holds, then $\gamma \models_{\chi_j} \phi_j$ if and only if $\gamma' \models_{\chi'_j} \phi_j$.

This holds for full valuations χ_k and χ'_k by applying Lemma 4.11. Assume it holds at rank $j+1$ with $1 \leq j < k$; we show it for j. Assume χ_j and χ'_j are valuations such that $\mathbb{R}^{D^0,j}_{(\gamma,\gamma')}(\chi_j, \chi'_j)$ holds and $\gamma \models_{\chi_j} \phi_j$. We distinguish two cases:

- **Case $Q_{j+1} = \exists$.** Pick a strategy v_{j+1} for variable x_{j+1} such that $\gamma \models_{\chi_j \cup \{v_{j+1}\}} \phi_{j+1}$. Applying Lemma 4.10, choose strategy v'_{j+1} such that $\mathbb{R}^{D^0,j+1}_{(\gamma,\gamma')}(\chi_j \cup \{v_{j+1}\}, \chi'_j \cup \{v'_{j+1}\})$ holds. Applying the induction hypothesis, we deduce that $\gamma' \models_{\chi'_j \cup \{v'_{j+1}\}} \phi_{j+1}$.

- **Case $Q_{j+1} = \forall$.** Pick a strategy v'_{j+1} for variable x_{j+1} from γ'. Applying Lemma 4.10, choose strategy v_{j+1} such that $\mathbb{R}^{D^0,j+1}_{(\gamma,\gamma')}(\chi_j \cup \{v_{j+1}\}, \chi'_j \cup \{v'_{j+1}\})$ holds. Applying the induction hypothesis, we deduce that $\gamma' \models_{\chi'_j \cup \{v'_{j+1}\}} \phi_{j+1}$ iff $\gamma \models_{\chi_j \cup \{v_{j+1}\}} \phi_{j+1}$. The last relation is valid, hence $\gamma' \models_{\chi'_j \cup \{v'_{j+1}\}} \phi_{j+1}$.

We conclude the proof by noticing that $\mathbb{R}^{D^0,0}_{(\gamma,\gamma')}(\emptyset, \emptyset)$ holds since $\Delta \sim \Delta'$. □

Part II

The dependency problem in SL[BG]

Chapter 5

Introduction to the dependency problem

In the quantifiers' semantics, the strategy assigned to a variable x on a history ρ depends on all the other strategies quantified before it *in their entirety*. These dependencies are rather unusual, make the semantic of the quantifiers counter-intuitive and prevent us from making local decisions. The example below illustrates the issue.

Fix two agents $\square$ and $\bigcirc$, two atomic propositions p_1 and p_2 and four variables $x_A^{\square}$, $x_B^{\square}$, $y_A^{\bigcirc}$ and $y_B^{\bigcirc}$. Consider the game $\mathcal{G}$ on Figure 5.1. We define a SL[BG] formula ϕ_0 below.

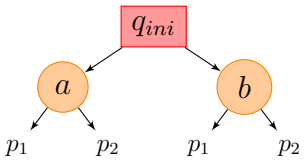


Figure 5.1: A game $\mathcal{G}$.

$$\phi_0 := \forall x_A^{\square}.\forall y_A^{\bigcirc}.\exists x_B^{\square}.\exists y_B^{\bigcirc}.\underbrace{\begin{cases} (\mathsf{assign}(\square, x_A^{\square}; \bigcirc, y_A^{\bigcirc})\mathbf{F}\, p_1 \\ \Leftrightarrow\ \mathsf{assign}(\square, x_B^{\square}; \bigcirc, y_B^{\bigcirc})\mathbf{F}\, p_2) \\ \wedge \\ \mathsf{assign}(\square, x_B^{\square}; \bigcirc, y_B^{\bigcirc})\mathbf{F}\, b \end{cases}}_{\phi_1}$$

One can see that using the intuitive semantic given in Chapter 1, ϕ_0 holds on $\mathcal{G}$ from q_{ini}. For example, using the name of a variable for the strategy assigned to it, if $x_A^{\square}(q_{ini}) = a$ and $y_A^{\bigcirc}(q_{ini}.a) = p_2$ then we can set $x_B^{\square}(q_{ini}) = b$ and $y_B^{\bigcirc}(q_{ini}.b) = p_1$. The outcome of $\{x_A^{\square}, y_A^{\bigcirc}\}$ does not see p_1 so the first goal is not satisfied and the outcome of $\{x_B^{\square}, y_B^{\bigcirc}\}$ eventually sees b but not p_2 so the second goal is not satisfied but the third is. This makes the equivalence between the first two goals hold as well as the third goal, satisfying ϕ_1. Now, assume $x_A^{\square}(q_{ini}) = a$ and $y_A^{\bigcirc}(q_{ini}.a) = p_1$, we can take $x_B^{\square}(q_{ini}) = b$ and $y_B^{\bigcirc}(q_{ini}.b) = p_2$. Unlike previously, all three goals are satisfied and both the equivalence and the conjunction hold, making ϕ_1 true. A similar reasoning can be done when $x_A^{\square}(q_{ini}) = b$. In the end, the overall formula ϕ_0 is satisfied by $\mathcal{G}$.

In the reasoning above, one can notice that the choice of $y_B^{\bigcirc}$ on the history $q_{ini}.b$ depends on what $y_A^{\bigcirc}$ decided to play on $q_{ini}.a$. In particular, the choice of $y_B^{\bigcirc}$ on a history

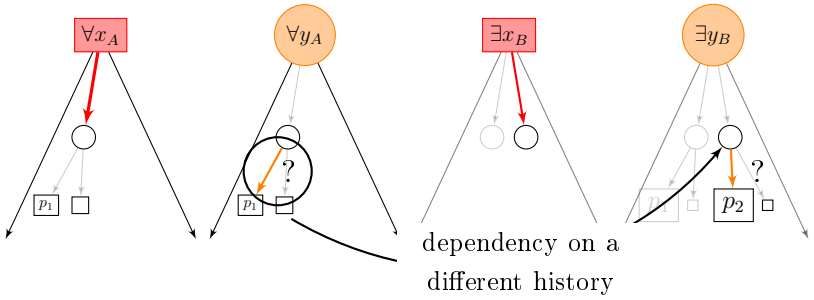


Figure 5.2: Dependencies from the existential variable y_B on the universal variable y_A.

depends on the choice of y_A on another history. Figure 5.2 illustrates the situation: there is an unnatural influence of y_A on y_B. Indeed, in some practical cases, the strategy y_A can be only partially revealed and we may therefore wish not only that y_B exists but also to build it using a limited amount of information from y_A. Finding when these dependencies appear is what Mogavero, Murano, Perelli and Vardi refer as finding elementary witnesses in [39].

Consider a formula $\phi := Q_1 x_1, \dots Q_l x_l.\ \xi(\beta_j \varphi_j)_{1 \leq j \leq n}$. A strategy δ stored in a variable x_j is said local when it depends only on the current history and the choices of the variables quantified before x_j on said history. The question of elementary witnesses formally asks which SL[BG] formulas can be solved using only strategies with local choices for the existentially quantified variables[1]. In other words, we aim to find formulas where the situation illustrated on Figure 5.2 does not happen.

In this chapter, we give a layout of the situation, propose a framework (adapted from [39]) and highlight a few results. As we will see, the notion of elementary witness developed in [39] is not sufficient for a thorough study of the problem. In later chapters, we will study the *dependency problem* in a broader sense.

5.1 A partial classification of the dependencies

We start by formally defining the notions at the heart of the dependency problem. Given a history $\rho = (q_i)_{i \leq L}$ where $L \in \mathbb{N}$, we recall that a prefix of ρ is a history of the form $\rho' := (q_i)_{i \leq L'}$ where $L' < L$. We also recall that an extension of ρ is any history of the form $\rho'' := (q_i)_{i < L''}$ where $L'' > L$. We write $\mathsf{Pref}_{\leq \rho}$ for the set of all prefixes of ρ including ρ and $\mathsf{Pref}_{< \rho}$ for the set of strict prefixes of ρ. We also regroup all the other histories within the notion of counter-factual history:

Definition 5.1 (Counter-factual history)**.**
Given two different histories ρ and π, we say that π is counter-factual of ρ whenever π is neither a prefix nor an extension of ρ.

[1] Initially introduced in [39], the authors formulate the problem in a different manner through the use of mathematical objects. We will explain and refine their idea in Section 5.2 later on.

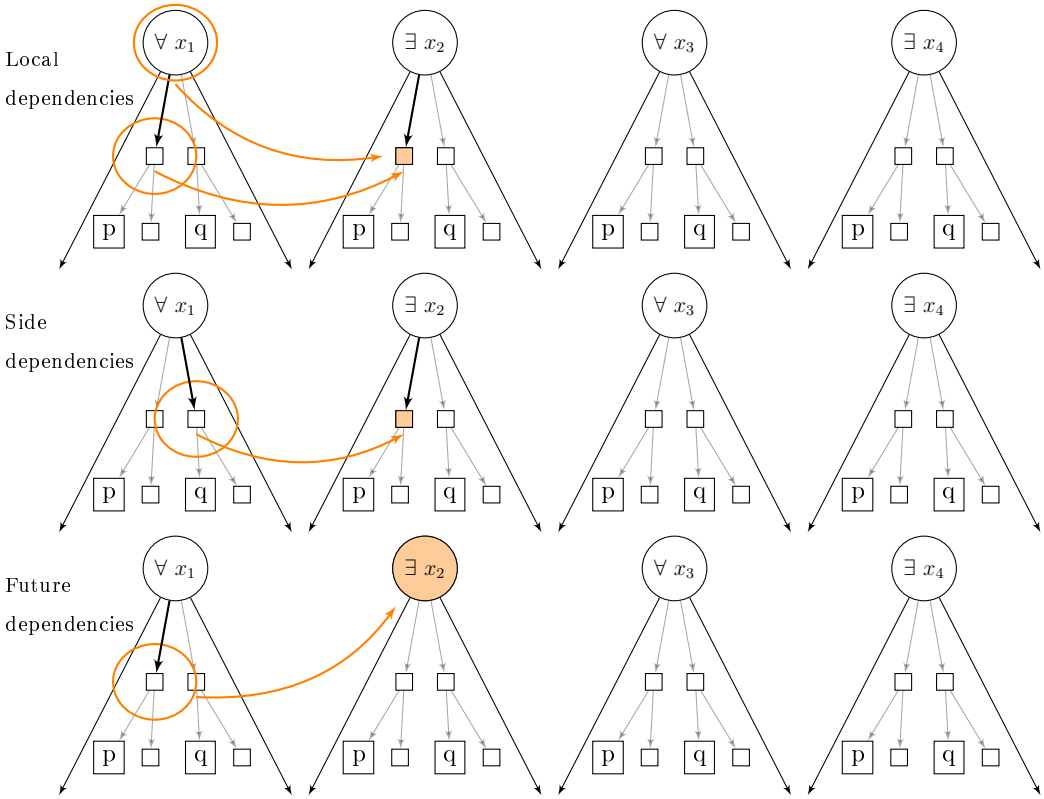


Figure 5.4: The three kinds of dependencies

We highlight three kinds of possible dependencies and explore their usage to model open and multi-agents systems. More kinds are possible, we restrict ourselves to the ones we believe are the most interesting.

The first kind is called *local dependency*: it happens when a strategy y on a history ρ requires knowledge of the choices from strategies quantified before y on prefixes of ρ or on ρ itself. Local dependencies are rather common with CGS, they order who has knowledge of what within a state or a history. For example, consider the game of Figure 5.3 with two actions 0 and 1, and two agents A and B. From q_{ini}, A aims to reach a p-labelled state while B tries to avoid it. The agent who announces his strategy first loses: if A talks first, B may play the same action as A, forcing a transition towards the state not labelled by p. On the other hand if B is the first notifying his choice, A can choose the opposite action, ensuring that the transition results in the p-labelled state. Which agent (or in SL[BG] which strategy) discloses his choice first can be modelled by choosing an appropriate quantification order, thus creating local dependencies. Note however that, due to the quantifier alternation, SL[BG] is not the best logic to deal with simultaneous decision makings.

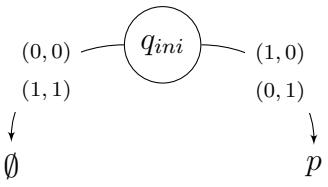


Figure 5.3: A game $\mathcal{G}$ with two agents and two actions (0 and 1)

The second kind of dependency is the one described in the introduction of this chapter: a strategy on a given history that depends on choices made by previously quantified strategies on counter-factual histories. We call it *side dependency*. Such dependencies may represent some knowledge from one component behaviour (in counter-factual plays) that is accessible to another component (the play under consideration). They may also represent some weak form of concurrency (a side thread can be executed before the current one and therefore we can assume some knowledge about how such thread went), but there exist better models for this. In most multi-agents systems, these dependencies cannot happen and therefore these systems cannot be modelled through CGS and SL[BG] formulas. Therefore, the main interest in studying these dependencies is to know when they appear to expand the class of practical problems that can effectively be solved through SL[BG] model checking. It may also give some information about the intrinsic difficulty of certain formulas.

The third kind of dependency is when an existential strategy y on history π depends on the choice made by a previously quantified strategy x on an extension $\pi.\pi'$ of π, we call it *future dependency*. Future dependencies, like local ones, represent some knowledge accessible to an agent at a given time, here about future decision. They may appear when the system we aim to model has some intrinsic limitations, or when there are some formal and predetermined protocols interspersed in the system. More often than not and unlike local influences, future dependencies are a burden preventing many problems to be solved through SL[BG] model checking procedure. Unfortunately we will see that they appear frequently. These three kinds are illustrated in Figure 5.4.

5.2 Formal framework

A framework proposed to investigate this issue is to represent the existential choices through mathematical objects that we call *maps*. We rediscover and extend the concept of *dependence maps* introduced in [38, 39] into a more general framework adapted to the three kinds of dependencies we highlighted before. The idea is similar to the Skolemisation of first-order formulas [17]: representing the existentially quantified variables as functions of the universally quantified variables. The notion of "there exists a strategy such that for all strategy..., it holds ξ" where ξ is a boolean combination of goals is transformed in "there exists a map satisfying ξ". We can then apply adequate restrictions on the maps to model the dependencies.

Choosing to work with dependence maps allows us to treat the strategy quantifications as a bloc, in a somewhat global fashion. Other frameworks are possible and we could have worked with one similar to what we used in Chapter 1 to define SL; however such framework conveys the idea that strategies are individual elements and does not simplify the notations.

Treating the dependency problem for the full SL[BG] greatly complexifies definitions; we therefore focus exclusively on (the flat fragment) $\mathsf{SL[BG]}^\flat$ and recall its grammar below. To ease the reading of Chapters 5 to 7, we remove the flat symbol.

$$\begin{aligned}\mathsf{SL[BG]} \ni \phi &::= \exists x.\phi \mid \forall x.\phi \mid \xi\\ \xi &::= \xi \vee \xi \mid \xi \wedge \xi \mid \beta\\ \beta &::= \mathsf{assign}(A,x).\beta \mid \varphi\\ \varphi &::= \varphi \vee \varphi \mid \neg\varphi \mid \varphi \mathbf{U}\, \varphi \mid \mathbf{X}\, \varphi \mid p\end{aligned}$$

As we have seen with the definition in Chapter 1, all SL[BG] formulas have their quantifications grouped together at the beginning and can be written in the following form[2]

$$\phi := (Q_i x_i)_{i\leq l}\ \xi\big(\beta_j\varphi_j\big)_{j\leq n}$$

where for any $i \leq l$, $Q_i \in \{\exists, \forall\}$ and $x_i \in \mathcal{V}$ (with $x_i \neq x_{i'}$ for any $i \neq i'$), ξ is a boolean combination, for any $j \leq n$, β_j is a sequence of assignments and φ_j is an LTL formula over AP. We write $\wp := (Q_i x_i)_{1\leq i\leq l}$ and, to simplify the notations, we assume without loss of generality that $\mathcal{V} = \{x_i \mid 1 \leq i \leq l\}$. We also write $\mathcal{V}^\forall := \{x_i \mid Q_i = \forall\}$ and $\mathcal{V}^\exists := \{x_i \mid Q_i = \exists\}$ respectively for the universally and existentially quantified variables of $\wp$.

5.2.1 Maps

We refer to a function θ as a $\wp$-*map* (or simply map when $\wp$ is clear of context) if it is of form

$$\theta : (\mathsf{Hist} \to \mathsf{Act})^{\mathcal{V}^\forall} \to (\mathsf{Hist} \to \mathsf{Act})^{\mathcal{V}} \quad \text{or equivalently} \quad \theta : (\mathcal{V}^\forall \to \mathsf{Strat}) \to (\mathcal{V} \to \mathsf{Strat})$$

and satisfies $\theta(w)(x_i)(\rho) = w(x_i)(\rho)$ for any $w : (\mathsf{Hist} \to \mathsf{Act})^{\mathcal{V}^\forall}$, any universally quantified variable $x_i \in \mathcal{V}^\forall$ and any history ρ. A map therefore defines the strategies existentially quantified in function of the universally quantified strategies.

We can forcibly remove certain influences by applying adequate restrictions on the maps. Local dependencies are allowed in all cases: the use of CGS makes it mandatory to handle who has which information within a state and, as explained in Section 5.1, not much can be done in CGS without them. We define two parameters: S, F respectively for the *side* and *future* dependencies. We can then draw four sub-types of maps by choosing parameters among S and F: $\{\mathcal{M}(\spadesuit, \heartsuit) \mid \spadesuit \in \{\emptyset, S\},\ \heartsuit \in \{\emptyset, F\}\}$ where a $\mathcal{M}(\spadesuit, \heartsuit)$ map has the additional restriction:

$$\left.\begin{array}{l}\forall \rho \in \mathsf{Hist},\ \forall x_i \in \mathcal{V}\\ \forall w_1, w_2 : (\mathcal{V}^\forall \to \mathsf{Strat})\end{array}\right\} \Big(\mathcal{C}(Local) \wedge \mathcal{C}(\spadesuit) \wedge \mathcal{C}(\heartsuit)\Big) \Rightarrow \Big(\theta(w_1)(x_i)(\rho) = \theta(w_2)(x_i)(\rho)\Big)$$

with

[2] Without loss of generality, we assume the variables to be used in at most a single quantification. We further assume them to be ordered: x_1 is quantified before x_2 which is itself quantified before x_3 and so on.

- $\mathcal{C}(\emptyset)$: empty condition (always satisfied)
- $\mathcal{C}(Local)$: w_1 and w_2 coincide on $\mathcal{V}^\forall \cap [x_1; x_{i-1}]$ and on ρ, i.e.
$$\forall y \in \mathcal{V}^\forall \cap [x_1; x_{i-1}].\ \forall \mu \leq \rho \quad w_1(y)(\mu) = w_2(y)(\mu)$$
- $\mathcal{C}(S)$: w_1 and w_2 coincide on $\mathcal{V}^\forall \cap [x_1; x_{i-1}]$ and on side histories of ρ (side), i.e.
$$\forall y \in \mathcal{V}^\forall \cap [x_1; x_{i-1}].\ \forall \mu \text{ counter-factual of } \rho \quad w_1(y)(\mu) = w_2(y)(\mu)$$
- $\mathcal{C}(F)$: w_1 and w_2 coincide on $\mathcal{V}^\forall \cap [x_1; x_{i-1}]$ and on extensions of ρ (future), i.e.
$$\forall y \in \mathcal{V}^\forall \cap [x_1; x_{i-1}].\ \forall \mu \text{ extension of } \rho \quad w_1(y)(\mu) = w_2(y)(\mu)$$

The case of $\mathcal{M}(S, F)$ maps can be simplified by asking that for any $x_i \in \mathcal{V}$ and any two valuations $w_1, w_2 \colon (\mathcal{V}^\forall \to \mathsf{Strat})$ that coincide on variables $(x_{i'})_{i'<i}$, it holds that $\theta(w_1)(x_i) = \theta(w_2)(x_i)$. The semantics of Chapter 1 then corresponds to the existence of a $\mathcal{M}(S, F)$ map.

The framework may seem technical but is rather intuitive when referring to Figure 5.4. By definition, a $\mathcal{M}(S, \emptyset)$ or $\mathcal{M}(\emptyset, F)$ map will also be of type $\mathcal{M}(S, F)$ and a cardinality argument suffices to show that these inclusions are strict[3].

Example 5.2. *A $\mathcal{M}(S, F)$ map θ for a formula $\forall x_1.\exists x_2.\forall x_3.\exists x_4.$ ζ is a function that maps pairs of strategies (for universally-quantified variables x_1 and x_3) to pairs of strategies (for existentially-quantified variables x_2 and x_4) and that respects the order of quantifiers: here x_2 is only allowed to depend on x_1, but may depend on its entirety.*

Example 5.3. *A $\mathcal{M}(\emptyset, \emptyset)$ map θ for $\forall x_1.\exists x_2.\forall x_3.\exists x_4.$ ζ can be seen as a function making local choices. The strategy stored in x_2 on a history π may only depend on the value of x_1 on π and its prefixes. Similarly, the choice of x_4 on π may only depend on the values of x_1 and x_3 on π and its prefixes.*

5.2.2 Satisfaction relations

We can then define four satisfaction relations $(\models^{\mathcal{M}(\spadesuit,\heartsuit)})_{\spadesuit \in \{\emptyset, S\},\ \heartsuit \in \{\emptyset, F\}}$ using the concept of maps. First, consider a map θ and a function $w \colon (\mathcal{V}^\forall \to \mathsf{Strat})$. The image $\theta(w)$ of θ by w defines a valuation. In the following, when writing $\models_{\theta(w)}$, we simply mean the satisfaction relation $(\models_\chi)$ of Chapter 1 on valuation $\theta(w)$. To define the four new satisfaction relations, we treat the quantifications as blocks and given a (closed) SL[BG] formula $(Q_i x_i)_{i \leq l}\ \xi\big(\beta_j \varphi_j\big)_{j \leq n}$, we set

$$\mathcal{G}, q \models^{\mathcal{M}(\spadesuit,\heartsuit)} (Q_i x_i)_{i \leq l}\ \xi\big(\beta_j \varphi_j\big)_{j \leq n} \quad \Leftrightarrow \quad \begin{cases} \exists \theta \in \mathcal{M}(\spadesuit, \heartsuit) \\ \forall w \colon (\mathcal{V}^\forall \to \mathsf{Strat}) \\ \mathcal{G}, q \models_{\theta(w)} \xi\big(\beta_j \varphi_j\big)_{j \leq n} \end{cases}$$

For a fixed game $\mathcal{G}$ and a SL[BG] formula $\phi := (Q_i x_i)_{i \leq l}\ \xi\big(\beta_j \varphi_j\big)_{j \leq n}$, a $\mathcal{M}(\spadesuit, \heartsuit)$ map θ relative to $(Q_i x_i)_{i \leq l}$ for some given parameters $\spadesuit, \heartsuit$ is called a $\mathcal{M}(\spadesuit, \heartsuit)$ *witness* of ϕ

[3]Equivalently, one can build a $\mathcal{M}(S, F)$ map that is not $\mathcal{M}(S, \emptyset)$.

holding true on $\mathcal{G}$ whenever

$$\forall w\colon (\mathcal{V}^{\forall} \to \mathsf{Strat}) \qquad \mathcal{G}, q \models_{\theta(w)} \xi\big(\beta_j\varphi_j\big)_{j\leq n}$$

The satisfaction relation $\models^{\mathcal{M}(S,F)}$ then redefines the satisfaction relation $\models$ of Chapter 1 in terms of maps. This equivalence between $\models^{\mathcal{M}(S,F)}$ and $\models$ corresponds to the work done by Mogavero, Murano, Perelli and Vardi in [39] (though we formalise it slightly differently).

Remark 5.4. *It is important to notice that only the quantifier operators have multiple possible semantics. The semantics of the assignments, of the boolean and of temporal operators are unchanged.*

5.2.3 Can a formula and its syntactic negation both hold on a game ?

Notations. *We write* $\overline{Q_i} := \forall$ *if* $Q_i = \exists$ *and* $\overline{Q_i} := \exists$ *if* $Q_i = \forall$. *Given a SL[BG] formula* $\phi = (Q_i x_i)_{i\leq l}\ \xi(\beta_j\varphi_j)_{j\leq n}$, *we then define* $\neg\phi := (\overline{Q_i}x_i)_{i\leq l}\ \neg\xi(\beta_j\varphi_j)_{j\leq n}$.

The use of maps to define the four relations ($\models^{\mathcal{M}(\spadesuit,\heartsuit)}$, with $\spadesuit \in \{\emptyset, S\}$ and $\heartsuit \in \{\emptyset, F\}$) allows us to properly monitor the different influences. There is however a drawback: consider a SL[BG] formula $\phi = (Q_i x_i)_{i\leq l}\xi(\beta_j\varphi_j)_{j\leq n}$ and two parameters $\spadesuit, \heartsuit$ respectively in $\{\emptyset, S\}$ and $\{\emptyset, F\}$; from the definition, nothing prevents ϕ and $\neg\phi$ to both hold on a game. Indeed, there may well be two $\mathcal{M}(\spadesuit, \heartsuit)$ maps θ relative to $(Q_i x_i)_{i\leq l}$ and $\overline{\theta}$ relative to $(\overline{Q_i}x_i)_{i\leq l}$ such that

$$\forall w\colon (\mathcal{V}^{\forall} \to \mathsf{Strat}) \qquad \mathcal{G}, q \models_{\theta(w)} \xi\big(\beta_j\varphi_j\big)_{j\leq n}$$
$$\forall \overline{w}\colon (\mathcal{V}^{\exists} \to \mathsf{Strat}) \qquad \mathcal{G}, q \models_{\overline{\theta}(\overline{w})} \neg\xi\big(\beta_j\varphi_j\big)_{j\leq n}$$

The theorem below shows that this cannot happen.

Theorem 5.5. *For any formula* ϕ *in SL[BG], any game* $\mathcal{G}$, *any initial state* q_{ini} *and any two parameters* $\spadesuit, \heartsuit$ *respectively in* $\{\emptyset, S\}$ *and* $\{\emptyset, F\}$, *it holds that*

$$\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit)} \phi \quad \Rightarrow \quad \mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\spadesuit,\heartsuit)} \neg\phi$$

Proof. Consider the original satisfaction relation $\models$ defined in Chapter 1. By definition of the $\neg$ operator, if $\mathcal{G}, q_{ini} \models \phi$ holds, then $\mathcal{G}, q_{ini} \not\models \neg\phi$. As said before, $\models$ corresponds to $\models^{\mathcal{M}(S,F)}$ and therefore

$$\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F)} \phi \quad \Rightarrow \quad \mathcal{G}, q_{ini} \not\models^{\mathcal{M}(S,F)} \neg\phi \tag{5.1}$$

Toward a contradiction, assume that Theorem 5.5 does not hold. There must exist two parameters $\spadesuit \in \{\emptyset, S\}$ and $\heartsuit \in \{\emptyset, F\}$ such that

$$\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit)} \phi \quad \text{and} \quad \mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit)} \neg\phi \tag{5.2}$$

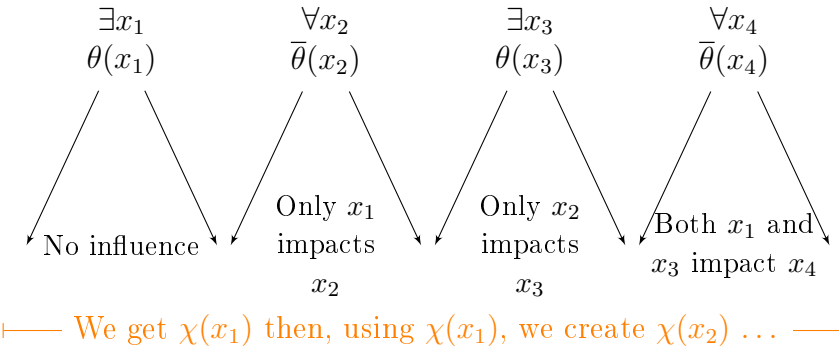


Figure 5.5: The idea behind Theorem 5.5 for $\phi := \exists x_1.\forall x_2.\exists x_3.\forall x_4.\ \xi\big(\beta_j\varphi_j\big)_{j\leq n}$.

Now, a $\mathcal{M}(\spadesuit,\heartsuit)$ map can be seen as a $\mathcal{M}(S,F)$ map, hence

$$\begin{array}{ccc} \mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit)} \phi & \text{and} & \mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit)} \neg\phi \\ \Downarrow & & \Downarrow \\ \mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F)} \phi & \text{and} & \mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F)} \neg\phi \end{array}$$

The last line is in contradiction with Equation (5.1), therefore Equation (5.2) must be false and Theorem 5.5 must hold. □

There is another way, more intuitive, to understand Theorem 5.5. Consider a formula $\phi := (Q_i x_i)_{i\leq l}\ \xi\big(\beta_j\varphi_j\big)_{j\leq n}$ and suppose that both $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit)} \phi$ and $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit)} \neg\phi$. Then there are two $\mathcal{M}(\spadesuit,\heartsuit)$ maps θ and $\overline{\theta}$ relative respectively to $(Q_i x_i)_{i\leq l}$ and $(\overline{Q_i} x_i)_{i\leq l}$ witnessing respectively that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit)} \phi$ and $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit)} \neg\phi$, i.e.

$$\begin{array}{rcl} & \forall w\colon (\mathcal{V}^{\forall} \to \mathsf{Strat}) & \mathcal{G}, q \models_{\theta(w)} \xi\big(\beta_j\varphi_j\big)_{j\leq n} \\ \text{and} & \forall \overline{w}\colon (\mathcal{V}^{\exists} \to \mathsf{Strat}) & \mathcal{G}, q \models_{\overline{\theta}(\overline{w})} \neg\xi\big(\beta_j\varphi_j\big)_{j\leq n} \end{array}$$

We can then confront θ and $\overline{\theta}$ to get a valuation χ of domain $\mathsf{dom}(\chi) = \{x_i \mid i \leq l\}$ such that $\theta(\chi_{|\mathcal{V}^\forall}) = \chi$ and $\overline{\theta}(\chi_{|\mathcal{V}^\exists}) = \chi$. This means that χ must satisfy both $\xi\big(\beta_j\varphi_j\big)_{j\leq n}$ and $\neg\xi\big(\beta_j\varphi_j\big)_{j\leq n}$, this is impossible (Remark 5.4). An intuition of this can be found in Figure 5.5. This idea of confronting maps for a formula and its negation will be particularly interesting in later chapters.

5.2.4 Can a formula and its negation both fail to hold on a game ?

One can also wonder if it is possible for neither a formula nor its negation to hold on a game. The correspondence between $\models$ and $\models^{\mathcal{M}(S,F)}$ gives us the following lemma.

Lemma 5.6. *For any formula ϕ in SL[BG], any game $\mathcal{G}$ and any initial state q_{ini}, it holds that*

$$\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(S,F)} \phi \quad \Rightarrow \quad \mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F)} \neg\phi$$

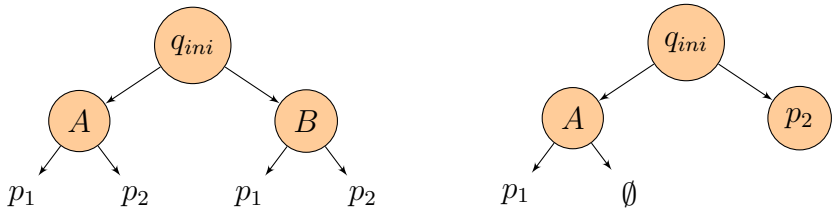


Figure 5.6: The games of Lemma 5.7 on the left and of Lemma 5.8 on the right.

The situation is however more complex for the other three relations, as illustrated by the two lemmas below.

Lemma 5.7. *There exists a formula ϕ in SL[BG], a game $\mathcal{G}$ and an initial state q_{ini} such that for any $\heartsuit \in \{\emptyset, F\}$, it holds that*

$$\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\heartsuit)} \phi \quad \textit{and} \quad \mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\heartsuit)} \neg\phi$$

Proof. Fix a parameter $\heartsuit \in \{\emptyset, F\}$. Consider the formula below and the one-player game on the left of Figure 5.6.

$$\phi := \forall x.\exists y \begin{cases} \mathsf{assign}(\bigcirc, x).\ \mathbf{F}\, p_1 \Leftrightarrow \mathsf{assign}(\bigcirc, y).\ \mathbf{F}\, p_1 \\ \wedge \\ \mathsf{assign}(\bigcirc, y).\ \mathbf{F}\, B \end{cases}$$

We start by proving that $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\heartsuit)} \phi$. Trivially, the strategy $y(q_{ini})$ must play to B for ϕ to be satisfied. Now, if $x(q_{ini})$ plays to A, the strategy y needs side dependencies of $x(q_{ini}.A)$ to play adequately on $q_{ini}.B$, and this dependency is not allowed in $\mathcal{M}(\emptyset, \heartsuit)$ maps.

On the other hand, $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\heartsuit)} \neg\phi$. Indeed, whatever the choice of x we can find a strategy y such that ϕ holds. So whatever the $\mathcal{M}(\emptyset, \heartsuit)$ map $\overline{\theta}$ for $\neg\phi$, there will always exist a function $\overline{w}$ such that $\overline{\theta}(\overline{w})$ fails to satisfy the conjunction of equivalences of goals. □

Lemma 5.8. *There exists a formula ϕ in SL[BG], a game $\mathcal{G}$ and an initial state q_{ini} such that for any $\spadesuit \in \{\emptyset, S\}$, it holds*

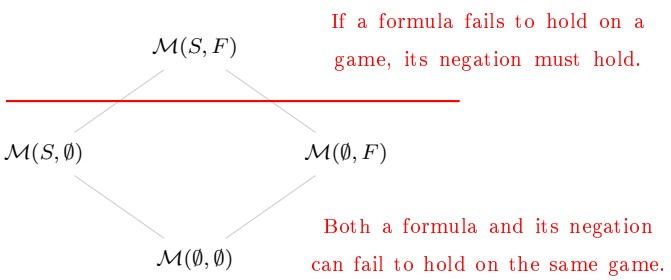
$$\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\spadesuit,\emptyset)} \phi \quad \textit{and} \quad \mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\spadesuit,\emptyset)} \neg\phi$$

Sketch of proof. The proof works in a fashion similar to the one of Lemma 5.7, this time using the game on the right of Figure 5.6 and the formula below.

$$\phi := \forall x.\exists y.\ (\mathsf{assign}(\bigcirc, x).\ \mathbf{F}\, p_1 \Leftrightarrow \mathsf{assign}(\bigcirc, y).\ \mathbf{F}\, p_2)$$

□

This situation (where neither a formula nor its negation hold) occurs whenever an existential strategy can make a lucky guess in order to successfully satisfy the formula but lacks some dependencies in order to ensure success.



5.2.5 What do these relations represent ?

The previous section raises an obvious question: are the relations poorly defined ? We recall that

$$\mathcal{G}, q \models^{\mathcal{M}(\spadesuit,\heartsuit)} (Q_i x_i)_{i\leq l}\, \xi\big(\beta_j\varphi_j\big)_{j\leq n} \quad \Leftrightarrow \quad \begin{cases} \exists \theta \in \mathcal{M}(\spadesuit,\heartsuit) \\ \forall w\colon (\mathcal{V}^{\forall} \to \mathsf{Strat}) \\ \mathcal{G}, q \models_{\theta(w)} \xi\big(\beta_j\varphi_j\big)_{j\leq n} \end{cases}$$

This definition entails two important things. The first one is obvious, the existentially quantified strategies are allowed the dependencies given by the parameters $\spadesuit$ and $\heartsuit$. Second, the universally quantified strategies are allowed a complete knowledge of the map through the universal quantification over w. Indeed, consider a quantification block $\forall x_1\ \exists x_2$. In the satisfaction relation, the universal quantification over the functions of the form $(\mathsf{Hist} \to \mathsf{Act})^{\{x_1\}}$ creates an omniscience of x_1 about the way x_2 will answer x_1 choices. In other words, x_1 has knowledge that "If x_1 plays such that..., then x_2 will play such that...". Note however that x_2 still depends on x_1.

In this chapter and the followings, it will be important not to overinterpret what a formula holding onto a game means for a given relation. For example, "$\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,F)} (Q_i x_i)_{i\leq l}\xi(\beta_j\varphi_j)_{j\leq n}$" states that there exists a behaviour for the existentially quantified strategies of $(Q_i x_i)_{i\leq l}$ that uses only (local and) future dependencies and that satisfies $\xi(\beta_j\varphi_j)_{j\leq n}$ no matter what the universally quantified strategies play. On the other hand, "$\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,F)} (\overline{Q_i} x_i)_{i\leq l}\neg\xi(\beta_j\varphi_j)_{j\leq n}$" does not means that the universally quantified strategies of $(Q_i x_i)_{i\leq l}$ have an (omniscient) answer for all $\mathcal{M}(\emptyset,F)$ behaviour of the existentially strategies. It states the existence of a $\mathcal{M}(\emptyset,F)$ behaviour for the universally quantified strategies of $(Q_i x_i)_{i\leq l}$ to avoid $\xi(\beta_j\varphi_j)_{j\leq n}$ against all (omniscient) behaviour of the existentially quantified strategy of $(Q_i x_i)_{i\leq l}$. The syntactic negation of a formula reverses the roles of the quantifiers but also the knowledge associated with them. This implies that the syntactic and semantic negations differ and gives us results such as the ones of Section 5.2.4.

To cope with the possibility for neither a formula nor its syntactic negation to hold, it is important to understand that inputs are two dimensional. The first dimension is the quantification block while the second is the set of dependency parameters $(\emptyset, S, F)$. What

a negation means is therefore ambiguous: should we consider negation of the quantifiers, negation of the dependencies or negation of both? We opt for the first option. More precisely, we choose to give the universally quantified strategies the knowledge of the map to echo the implicit universal quantification in ATL*. This choice is reminiscent of the (usually implicit) omniscience of the environment in worst-case analysis of open-systems. The framework is adapted to our goal, a worst-case verification of multi-agents systems, but also incomplete. Other approaches are possible; for example, we could have add other parameters to handle the dependencies of the universal strategies. In such a framework, the negation would be on both the quantifiers and the parameters.

5.3 Narrowing SL[BG]

The subject of dependencies was studied in [40, 41]; the authors showed that side and future dependencies can be removed from SL[BG] formulas based on conjunctions and disjunctions of goals. They however forgot to take into account the lack of knowledge that strategies have on prefixes of the current history[4]. As we will see later on, this impacts significantly the results. The ideas and ways of restricting $\mathsf{SL[BG]}^\flat$ however look promising. They introduced four fragments of SL[BG] by restricting the boolean combination of goals after the quantifiers block.

- SL[1G], introduced in [38], restrict SL[BG] to a unique goal. (The flat fragment) $\mathsf{SL[1G]}^\flat$ is defined from the grammar of $\mathsf{SL[BG]}^\flat$ by skipping the ξ line. More precisely, ξ's type sub-formulas must avoid any boolean operator, i.e. must be of form $\xi ::= \beta$.

$$\begin{aligned} \mathsf{SL[1G]}^\flat \ni \phi &::= \exists x.\phi \mid \forall x.\phi \mid \xi \\ \xi &::= \beta \\ \beta &::= \mathsf{assign}(A, x).\ \beta \mid \varphi \\ \varphi &::= \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi \mid p \end{aligned}$$

- SL[CG], introduced in [40], is the fragment where only *conjunctions* of goals are allowed. Formally, (the flat fragment) $\mathsf{SL[CG]}^\flat$ is defined from the grammar of $\mathsf{SL[BG]}^\flat$ with the restriction below on the ξ's type sub-formulas $\xi ::= \xi \wedge \xi \mid \beta$.

- Similarly, SL[DG] only allows *disjunctions* of goals, i.e. $\xi ::= \xi \vee \xi \mid \beta$.

- SL[CG] and SL[DG] have been further extended into SL[AG] [41], where limited alternation of goals are allowed. Formally, (the flat fragment of) this logic is defined

[4] More precisely, a strategy x on a history ρ does not have knowledge of what strategies quantified after x played on prefixes of ρ. This implies that $x(\rho)$ does not have knowledge of which goals are still active on ρ and which goals have deviated on other histories.

as follows:

$$\begin{aligned}
\mathsf{SL[AG]}^\flat \ni \phi &::= \exists x.\phi \mid \forall x.\phi \mid \xi \\
\xi &::= \xi \wedge \beta \mid \xi \vee \beta \mid \beta \\
\beta &::= \mathsf{assign}(A, x).\ \beta \mid \varphi \\
\varphi &::= \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi \mid p
\end{aligned}$$

Notice that the formula (that we recall below) used to characterise qualitative Nash equilibrium in Section 1.5.4 is not (syntactically) in SL[AG]. Indeed, its boolean combination of goals is a conjunctions of implication (i.e. a conjunctions of disjunctions) and therefore cannot be expressed linearly, as required in SL[AG].

$$\exists x_1, \dots x_n.\ \forall y_1, \dots, y_n.\ \bigwedge_{1\leq i\leq n} \mathsf{assign}(A_j, x_j)_{j\neq i}\big(\mathsf{assign}(A_i, y_i)\varphi_i \Rightarrow \mathsf{assign}(A_i, x_i)\varphi_i\big),$$

5.4 Needed dependencies

As exposed in the introduction of this chapter, there are unnatural and unexpected dependencies between strategies in the same quantifier block of a SL[BG] formula. In this section, we study the cases where certain dependencies are needed in order to satisfy SL[BG] formulas. For this, we consider the different logics exposed in Section 5.3 and prove that the four satisfaction relations are distinct from one another with respect to these logics. We will also see later cases where we can suppress some of the dependencies.

5.4.1 Future dependencies

We start by showing that some formulas require future dependencies to be satisfied. The condition for them to appear is rather thin as even the most simple formulas have them.

Lemma 5.9. *There exist a formula $\phi \in$ SL[DG], a turn-based game $\mathcal{G}$ and one of its states q_{ini} such that*

- $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset)} \phi$ *and* $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,F)} \phi$
- $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(S,\emptyset)} \phi$ *and* $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F)} \phi$

Proof. Take the game of Figure 5.7a on two agents 🟠 and 🟥, and the following formula

$$\phi := \forall y.\exists x_A.\forall x_B \begin{cases} \mathsf{assign}(🟠, y; 🟥, x_A).\ \mathbf{F}\,p_1 \\ \vee \\ \mathsf{assign}(🟠, y; 🟥, x_B).\ \mathbf{F}\,p_2 \end{cases}$$

First, note that by construction of ϕ and $\mathcal{G}$, no side dependency is possible. Indeed, only the variable x_A is existentially quantified and may have side dependencies. x_A is

however only relevant on the state q_{ini}, and due to the acyclic nature of $\mathcal{G}$, q_{ini} can only appear at the root of a history. At the root there is no possible counter-factual play and therefore no side dependency.

Take $\spadesuit \in \{\emptyset, S\}$, we start by proving $\mathcal{G}, q \models^{\mathcal{M}(\spadesuit,F)} \phi$. By definition of $\mathcal{M}(\spadesuit, F)$, x_A has knowledge of what y would play on suffixes after q_{ini}. So, if y goes from a to p_1 then x_A goes from q_{ini} to a and the first goal is satisfied, making ϕ hold. If y goes from a to p_2 and from b to p_1, x_A goes from q_{ini} to b and the first goal is satisfied. There remains the last case where y goes from both a and b to p_2, then no matter the universal choice of x_B, the second goal will be satisfied and ϕ will hold true (we can take any choice for x_A).

Next in order is to prove that $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\spadesuit,\emptyset)} \phi$. As no future dependency is permitted, x_A does not have any knowledge of y choices. If x_A decides to play from q_{ini} to a, then for y going from a to p_2 and from b to p_1 and x_B going to b, neither objective will be satisfied; if x_A decides to play from q_{ini} to b then for y going from a to p_1 and from b to p_2 and x_B going to a, again neither objective will be satisfied. So the choice of x_A on q_{ini} needs knowledge of what y plays on suffixes after q_{ini} and $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\spadesuit,\emptyset)} \phi$. □

(a) A turn-based game with two agents.

(b) Another turn-based game but with three agents.

Figure 5.7: Different games used in Lemmas 5.9, 5.10 and 5.12

Lemma 5.10. *There exist a formula $\phi \in$ SL[CG], a turn based game $\mathcal{G}$ and one of its states q_{ini} such that*

- $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(S,\emptyset)} \phi$ *and* $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F)} \phi$
- $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,F)} \phi$ *and* $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F)} \phi$

Proof. Consider the game on Figure 5.7b with three agents ● , ■ and ◆ , and the SL[CG] formula

$$\phi := \forall y.\exists z.\exists x_A.\exists x_B \begin{cases} \mathsf{assign}(●, y; ■, x_A; ◆, z).\ \mathbf{F}\, p_1 \\ \wedge \\ \mathsf{assign}(●, y; ■, x_B; ◆, z).\ \mathbf{F}\, p_2 \end{cases}$$

We start by proving that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F)} \phi$. Note that as both future and side dependencies are allowed, both x_A and x_B but also z may depend on the choices of $y(q_{ini}.a)$. If $y(q_{ini}.a) = p_1$ then we set $x_A(q_{ini}) := a$, $x_B(q_{ini}) := b$ and $z(q_{ini}.b) := p_2$, in the end both goals are satisfied. On the other hand if $y(q_{ini}.a) = p_2$, we set $x_A(q_{ini}) := b$, $x_B(q_{ini}) := a$

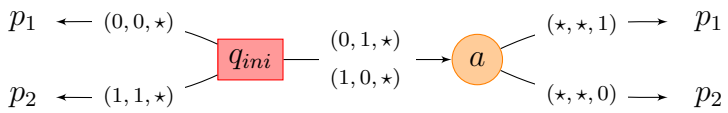


Figure 5.8: A concurrent game $\mathcal{G}$ where $\mathsf{Agt} := \{A, B, \bigcirc\}$ and $\mathsf{Act} = \{0, 1\}$.

and $z(q_{ini}.b) := p_1$. Again both goals are satisfied. From this we can build a $\mathcal{M}(S, F)$ witness of $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F)} \phi$.

We then prove that $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(S,\emptyset)} \phi$. This time, x_A and x_B cannot depend on the value of $y(q_{ini}.a)$; only z can. First if $x_A(q_{ini}) = x_B(q_{ini})$, one of the goals will not be satisfied (no matter y and z). Now, if $x_A(q_{ini}) = a$ and $x_B(q_{ini}) = b$ then having $y(q_{ini}.a) = p_2$ means that the first goal does not hold. Finally, if $x_A(q_{ini}) = b$ and $x_B(q_{ini}) = a$, for $y(q_{ini}.a) = p_1$, the second goal will not hold. This ensures that x_A and x_B need knowledge for q_{ini} on what y plays on $q_{ini}.a$ and therefore $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(S,\emptyset)} \phi$ and the first point of Lemma 5.10 holds.

We show now that $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,F)} \phi$. Here, x_A and x_B may depend on y but z may not. Assuming $z(q_{ini}.b) = p_1$ then if $y(q_{ini}.a) = p_1$, the second goal cannot hold no matter x_A and x_B. On the contrary, if $z(q_{ini}.b) = p_2$ then with $y(q_{ini}.a) = p_2$, we get that the first goal cannot hold. Hence, side dependencies are mandatory and $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,F)} \phi$. Combining this with $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F)} \phi$ (proved for the first point) gives us the second point of Lemma 5.10. □

Lemma 5.11. *There exist a formula $\phi \in$ SL[CG], a* concurrent *game $\mathcal{G}$ and one of its states q_{ini} such that $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset)} \phi$ and $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,F)} \phi$.*

Proof. Consider the game on Figure 5.8 with three agents A, B and $\bigcirc$, and two actions 1 and 0. Agents A and B play concurrently in state q_{ini} while agent $\bigcirc$ controls the circle state. Consider also the SL[CG] formula

$$\phi := \forall y.\exists x^A.\exists x_1^B.\exists x_2^B \begin{cases} \mathsf{assign}(A, x^A; B, x_1^B; \bigcirc, y).\ \mathbf{F}\, p_1 \\ \wedge \\ \mathsf{assign}(A, x^A; B, x_2^B; \bigcirc, y).\ \mathbf{F}\, p_2 \end{cases}$$

We start by proving that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,F)} \phi$. As future dependencies are allowed, knowledge of $y(q_{ini}.a)$ can be assumed when building x^A, x_1^B, x_2^B. If $y(q_{ini}.a) = 1$, then we set $x^A(q_{ini}) = 1$, $x_1^B(q_{ini}) = 0$ and $x_2^B(q_{ini}) = 1$. The first goal goes to a and p_1 while the second goes to p_2 thus making the conjunction holds. If $y(q_{ini}.a) = 0$, we set $x^A(q_{ini}) := 0$, $x_1^B(q_{ini}) := 0$ and $x_2^B(q_{ini}) := 1$. The first goal goes to p_1 and the second to a and p_2. Whatever the choice made by y, we have a solution hence we have a $\mathcal{M}(\emptyset, F)$ witness that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,F)} \phi$.

It remains to show that $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset)} \phi$. We proceed by contradiction. Assume there is a $\mathcal{M}(\emptyset, \emptyset)$ map Δ witnessing ϕ on $\mathcal{G}$. By nature of Δ, $\Delta(x^A)$ is independent of y choices. If $x^A(q_{ini}) = 1$ then with $y(q_{ini}.a) = 0$ there is no way to ensure the first goal. Similarly, if $x^A(q_{ini}) = 0$ then with $y(q_{ini}.a) = 1$ we cannot ensure the second goal. So the hypothesis must be false and $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset)} \phi$. □

5.4.2 Side dependencies

We then tackle the cases where side dependencies are needed.

Lemma 5.12. *There exist a formula $\phi \in$ SL[DG], a turn-based game $\mathcal{G}$ and one of its states q_{ini} such that*

- $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset)} \phi$ *and* $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset)} \phi$
- $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,F)} \phi$ *and* $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F)} \phi$

Proof. Consider the game of Figure 5.7b and the formula

$$\phi := \forall y.\exists z.\forall x_A.\forall x_B \begin{cases} \mathsf{assign}(\blacksquare, x_A; \bigcirc, y; \diamond, z)\mathbf{F}\, p_1 \\ \vee \\ \mathsf{assign}(\blacksquare, x_B; \bigcirc, y; \diamond, z)\mathbf{F}\, p_2 \end{cases}$$

First, note that in Figure 5.7b and ϕ there can be no future dependencies; hence proving the first point will immediately imply the second. We begin by showing that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset)} \phi$. Side dependencies are allowed, hence z may depend on the choice of y. If $y(q_{ini}.a) = p_1$, then we set $z(q_{ini}.b) := p_1$; whatever the choice of x_A, the first goal holds. On the other hand, if $y(q_{ini}.a) = p_2$, we set $z(q_{ini}.b) := p_2$ and for any choice of x_B, the second goal holds. We can then combine these two cases to build a $\mathcal{M}(S,\emptyset)$ witness that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset)} \phi$.

It remains to prove that $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset)} \phi$. In the present case, z cannot depend on the choices of y. Assume $z(q_{ini}.b) = p_1$ then for $y(q_{ini}.a) = p_2$, $x_A(q_{ini}) = a$ and $x_B(q_{ini}) = b$, neither of the objectives holds. Similarly, if $z(q_{ini}.b) = p_2$ then with $y(q_{ini}.a) = p_1$, $x_A(q_{ini}) = b$ and $x_B(q_{ini}) = a$, neither of the objectives holds. This means that z needs knowledge of y choices and that $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset)} \phi$. □

5.5 Removable dependencies

5.5.1 $\mathcal{M}(S,\emptyset)$ and $\mathcal{M}(\emptyset,\emptyset)$ coincide on SL[CG]

Lemma 5.12 shows that there are some SL[DG] formulas on which adequate behaviours must rely on side dependencies. The case of SL[CG] is however different.

Theorem 5.13. *For any concurrent game $\mathcal{G}$, any state q_{ini} of $\mathcal{G}$ and any formula $\phi \in$ SL[CG],*

$$\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset)} \phi \quad \Leftrightarrow \quad \mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset)} \phi$$

Proof. Fix any game $\mathcal{G}$, any state q_{ini} of $\mathcal{G}$ and any SL[CG] formula ϕ of the form

$$\phi := (Q_i x_i)_{i \leq l} \bigwedge_{j \leq n} \beta_j \varphi_j$$

with $Q_i x_i$ a quantification, β_j an assignment and φ_j an LTL formula. Assuming that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset)} \phi$, there exists Δ a $\mathcal{M}(S,\emptyset)$ witness of ϕ holding on $\mathcal{G}$ from q_{ini}, i.e.

$$\forall w : (\mathsf{Hist}_{\mathcal{G}} \to \mathsf{Act})^{\mathcal{V}^\forall} \qquad \mathcal{G}, q_{ini} \models_{\Delta(w)} \bigwedge_{j \leq n} \beta_j \varphi_j \tag{5.3}$$

We create a $\mathcal{M}(\emptyset,\emptyset)$ map Δ' and prove it is a $\mathcal{M}(\emptyset,\emptyset)$ witness of ϕ on $\mathcal{G}$. For this, choose any fixed function $w_0 : (\mathsf{Hist}_{\mathcal{G}} \to \mathsf{Act})^{\mathcal{V}^\forall}$. Given another function $w' : (\mathsf{Hist}_{\mathcal{G}} \to \mathsf{Act})^{\mathcal{V}^\forall}$ and a history ρ, we write w'_0 for the function equal to w' on ρ and its prefixes and equal to w_0 elsewhere (i.e on extensions and counter-factual histories of ρ). We then set, for any variable $x_i \in \mathcal{V}$,

$$\Delta'(w')(x_j)(\rho) := \Delta(w'_0)(x_j)(\rho)$$

Δ' is indeed a $\mathcal{M}(\emptyset,\emptyset)$ map: on a history ρ, we fixed universal choices on counter-factual histories of ρ to be "as in" w_0 before using Δ (on input made of w', x_j, ρ, the values of w' on counter-factual play do not influence the definition of w'_0) hence we do not import the side dependencies from Δ.

It remains to prove that Δ' is a witness for $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset)} \phi$. Toward a contradiction, assume this is not the case. Then there exists some $w_1 : (\mathsf{Hist}_{\mathcal{G}} \to \mathsf{Act})^{\mathcal{V}^\forall}$ such that $\mathcal{G}, q_{ini} \not\models_{\Delta'(w_1)} \bigwedge_{j \leq n} \beta_j \varphi_j$. In particular there is some $j_0 \in [1, \dots n]$ such that $\mathcal{G}, q_{ini} \not\models_{\Delta'(w_1)} \beta_{j_0} \varphi_{j_0}$. Let $\pi_{j_0} := \mathsf{out}(\beta_{j_0}(\Delta'(w_1)), q_{ini})$ be the outcome obtained when applying the assignments of β_{j_0} to the valuation $\Delta'(w_1)$. We build another function $w_2 : (\mathsf{Hist}_{\mathcal{G}} \to \mathsf{Act})^{\mathcal{V}^\forall}$ equal to w_1 (for any variable) on any prefix of π_{j_0} and equal to w_0 on any other history. By construction, for any prefix $\pi^p_{j_0}$ of π_{j_0} we have that $\Delta'(w_1)(x_i)(\pi^p_{j_0}) = \Delta(w_2)(x_i)(\pi^p_{j_0})$ hence π_{j_0} is also equal to $\mathsf{out}(\beta_{j_0}(\Delta(w_2)), q_{ini})$, the outcome obtained when applying β_{j_0} on $\Delta(w_2)$. This immediately implies that $\mathcal{G}, q_{ini} \not\models_{\Delta(w_2)} \beta_{j_0} \varphi_{j_0}$ thus $\mathcal{G}, q_{ini} \not\models_{\Delta(w_2)} \bigwedge_{j \leq n} \beta_j \varphi_j$ which is a contradiction with Formula (5.3). So, Δ' is a $\mathcal{M}(\emptyset,\emptyset)$ witness of ϕ on $\mathcal{G}$ and therefore $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset)} \phi$ and Theorem 5.13 holds. □

5.5.2 The case of SL[1G]

SL[1G] formulas are in many ways similar to ATL*, the main differences are the ability to share strategies among multiple agents and having multiple quantifier alternations (as opposed to only one in ATL*). The ability to share strategies does not amount to much: we can always merge agents sharing a common strategy in a new game, slightly modify the formula to suppress the strategy-sharing aspects, then solve the new game with the modified formula in order to deduce whether the original formula holds on the original game[5]. The expressive powers of both logics are closely related though not equal, and allowing more than one quantifier alternation does not seem to impact the complexity results (though there may be differences in the refined complexity). For example, SL[1G]

[5]The fact that we only have one goal is essential as it means that within a formula, an agent can only have one strategy (or more precisely only one strategy that is not overridden by some other strategy). With multiple goals, two agents may share strategies within one goal but not within the other, then the correctness of the reduction collapses.

satisfiability was proven *2*-EXPTIME-complete in [38], the same as ATL* satisfiability (relatively to the $\models$ satisfaction relation, or equivalently –see Section 5.2– the $\models^{\mathcal{M}(S,F)}$ satisfaction relation).

We prove that, as for[6] ATL*, to build a correct behaviour for a SL[1G] formula we do not need any side or future information. Due to the similar expressive power of SL[1G] and ATL*, this is not surprising. For the theorem below, we could have adapted to the model checking problem the techniques developed in [38] and used to solve SL[1G] satisfiability. We however chose to develop a new proof: the method presented below will become useful later on.

Theorem 5.14. *For any SL[1G] formula ϕ, any concurrent game $\mathcal{G}$ and any state q_{ini} of $\mathcal{G}$,*

$$\forall \spadesuit \in \{\emptyset, S\}\ \forall \heartsuit \in \{\emptyset, F\} \qquad \mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset)} \phi \Leftrightarrow \mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit)} \phi$$

Proof. Consider a game $\mathcal{G}$, one of its state q_{ini} and let $\phi := (Q_i x_i)_{i \leq l}\ \beta\varphi$ be a SL[1G] formula where Q_i is a quantification and $\beta\varphi$ forms a goal with β a total assignment and φ an LTL objective. We prove the following equivalence:

$$\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F)} \phi \quad \Leftrightarrow \quad \mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset)} \phi \tag{5.4}$$

The other cases will be inferred from it: equivalences involving other kind of maps can all be deduced from Equivalence (5.4) simply by seeing a $\mathcal{M}(\spadesuit, \heartsuit)$ map as a $\mathcal{M}(S, F)$ map, then applying the equivalences above. For the same reason, we have the right-to-left implication for free, hence to prove Theorem 5.14 it remains to show the left-to-right implication of Equivalence (5.4).

Specifying a turn-based parity game $\mathcal{H}$

We start by defining some specific turn-based arenas that in some way will "flatten" the game and formula. Figure 5.9 illustrates the construction. First, relatively to $\mathcal{G}$ and ϕ we define the following turn-based finite tree-like arena, which we call *cluster*:

- there are two players $P_\exists$ and $P_\forall$.
- the set of states is $S_{cluster} := \{\mathfrak{m} \in \mathsf{Act}^* \mid 0 \leq |\mathfrak{m}| \leq l\}$, thereby defining a tree of depth $l+1$ with directions Act. A state $\mathfrak{m}$ in $S_{cluster}$ with $|\mathfrak{m}| < l$ belongs to $P_\exists$ if and only if $Q_{|\mathfrak{m}|+1} = \exists$. To whom the states with $|\mathfrak{m}| = l$ belong does not matter.
- there is a transition from each $\mathfrak{m}$ of size strictly less than l to all $\mathfrak{m} \cdot a$ for all $a \in \mathsf{Act}$. In particular, the empty word $\varepsilon \in S_{cluster}$ is the starting node of the cluster, and it has no incoming transitions, while all words of length l have no outgoing transitions;

A leaf in such a cluster represents a move vector of domain $\mathcal{V} = \{x_i \mid 1 \leq i \leq l\}$: the leaf $\mathfrak{m}$ represents the move vector m where $m(x_i) = \mathfrak{m}(i)$.

[6] There exists no formal proof of it but it can be derived from the standard algorithm for ATL* model checking.

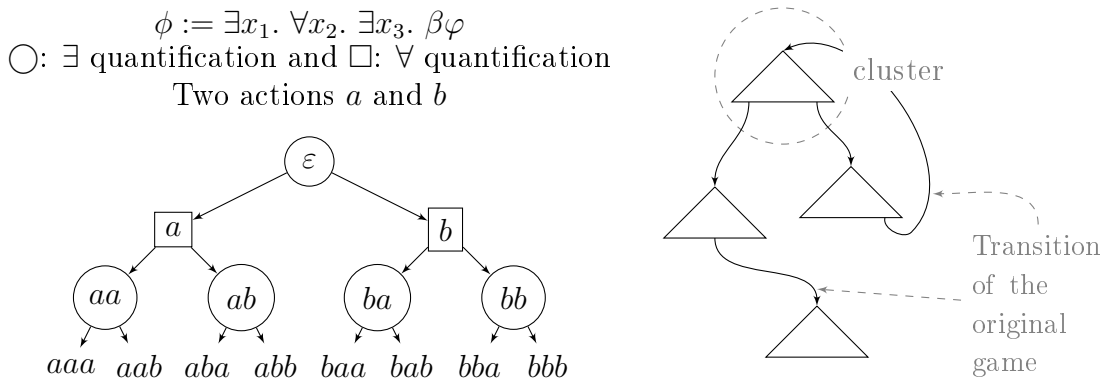


Figure 5.9: On the left: representation of a cluster. On the right: the overall shape of $\mathcal{H}$.

We denote by D a deterministic parity automaton over 2^{AP} associated with φ. We also write d_{ini} for the initial state of D. We let $\mathsf{succ}(d, q)$ be the function associating with each $d \in D$ its successor upon reading the labelling $\mathsf{labels}(q)$ where q is a state of $\mathcal{G}$. Using the notion of clusters, we define a turn-based parity game $\mathcal{H}$ as follows:

- the players are the same as before: $P_\exists, P_\forall$.
- for each state q of $\mathcal{G}$ and each state d of D, $\mathcal{H}$ contains a copy of a cluster which we call the (q, d) *cluster*. For any $\mathfrak{m} \in \mathsf{Act}^*$ with $|\mathfrak{m}| \leq l$, we refer to the state $\mathfrak{m}$ of the (q, d) cluster as the $(q, d, \mathfrak{m})$ state.
- the transitions in $\mathcal{H}$ are of two types:
  - internal transitions in the clusters are preserved;
  - consider a state $(q, d, \mathfrak{m})$ where $\mathfrak{m}$ is a leaf. If there exists a state q' such that $q' = \Delta(q, m_\beta)$ where $m_\beta : \mathsf{Agt} \to \mathsf{Act}$ is the move vector over Agt defined by $m_\beta(A) = \mathfrak{m}(i-1)$ where $x_i = \beta(A)$ (i.e. applying the choices of $\mathfrak{m}$ according to β in $\mathcal{G}$ leads from q to q'), then we add a transition from $(q, d, \mathfrak{m})$ to (q', d', ε) where $d' = \mathsf{succ}(d, q')$.
- the set of priorities are the same as in D and each $(q, d, \mathfrak{m})$ state has the same priority as d.

Correspondence between paths in $\mathcal{G}$ and in $\mathcal{H}$

There is not a clear one-to-one correspondence between the histories in $\mathcal{H}$ and the ones in $\mathcal{G}$, however there exists nevertheless some degree of connection. We introduce the notion of *lanes* before going into the details.

Definition 5.15. *A* lane *in $\mathcal{G}$ is a tuple (ρ, u, i, t) made of a history $\rho := (q_j)_{j \leq a}$ (for some integer a); a function $u : \mathcal{V} \times \mathsf{Pref}_{<\rho} \to \mathsf{Act}$; an integer $i \in [0; l]$; a function*

$t : [x_1; x_i] \to \mathsf{Act}$ *(with the convention that* $t = \emptyset$ *if* $i = 0$*); and such that*

$$\forall j < a \quad T(q_j, \beta(m_j)) = q_{j+1} \qquad \begin{cases} \text{where } m_j : \mathcal{V} \to \mathsf{Act} \text{ is the move vector over } \mathcal{V} \\ \text{with } m_j(x) := u(x)(\rho_{\leq j}) \end{cases}$$

We can build a one-to-one application $HtoG_{pth}$ between histories in $\mathcal{H}$ and lanes in $\mathcal{G}$. On a history π in $\mathcal{H}$, of shape

$$\pi \ := \ \Pi_{j<a}(\Pi_{0\leq i\leq l}(q_j, d_j, \mathfrak{m}_{j,i})).\Pi_{0\leq i\leq b}(q_a, d_a, \mathfrak{m}_{a,i})$$

of length $a.(l+1)+b$ with $0 \leq b < l+1$, we define $HtoG_{pth}(\pi)$ by

$$HtoG_{pth}(\pi) := ((q_j)_{j\leq a}, u, b, t)$$

with $\quad u : \mathcal{V} \times \mathsf{Pref}_{<\rho} \to \mathsf{Act} \qquad t : \mathcal{V} \cap [x_1; x_b] \to \mathsf{Act}$

$$\forall a' < a \quad (x_i, (q_j)_{j\leq a'}) \mapsto \mathfrak{m}_{j,i} \qquad \forall i \leq b \quad x_i \mapsto \mathfrak{m}_{a,i}$$

The application $HtoG_{pth}$ is clearly injective (two different histories will correspond to two different lanes), but also surjective. To prove it, we build the reciprocal function $GtoH_{pth}$: from a lane $((q_j)_{j\leq a}, u, i, t)$, we set $GtoH_{pth}((q_j)_{j\leq a}, u, i, t) := \pi$ where π is a history in $\mathcal{H}$ of length $a.(l+1)+|\mathsf{dom}(t)|+1$ of shape

$$\pi := \Pi_{j<a}\big(\Pi_{0\leq i\leq l}\big(q_j, d_j, u(x_i, (q_{j'})_{j'\leq j})\big)\big).\Pi_{0\leq i\leq b}\big(q_a, d_a, t(x_i, (q_j)_{j\leq a})\big)$$

where d_j is the vector of states reachable through $(q_{j'})_{j'\leq j}$

Because of the coherence condition imposed on lanes (see their definition), we get that the transition between clusters of $GtoH_{pth}((q_j)_{j\leq a}, u, i, t)$ are valid relatively to the transition table of $\mathcal{H}$. $GtoH_{pth}((q_j)_{j\leq a}, u, i, t)$ is therefore a valid history in $\mathcal{H}$ and $GtoH_{pth}$ is well defined. From the definitions, one can easily check that

$$\forall \pi \in \mathsf{Hist}_{\mathcal{H}} \qquad GtoH_{pth}(HtoG_{pth}(\pi)) = \pi$$

and deduce that $GtoH_{pth}$ is the inverse function of $HtoG_{pth}$; therefore

Proposition 5.16. *The application* $HtoG_{pth}$ *is a bijection between lanes of* $\mathcal{G}$ *and histories in* $\mathcal{H}$*, and* $GtoH_{pth}$ *is its inverse function.*

Extending the correspondence to strategies

We can use the $HtoG_{pth}$ correspondence to describe another correspondence $HtoG$ between positional strategies for $P_\exists$ in $\mathcal{H}$ and $\mathcal{M}(\emptyset, \emptyset)$ maps in $\mathcal{G}$. We recall that a map θ is a function $(\mathsf{Hist}_{\mathcal{G}} \to \mathsf{Act})^{\mathcal{V}^\forall} \to (\mathsf{Hist}_{\mathcal{G}} \to \mathsf{Act})^{\mathcal{V}}$ taking three inputs: a function $w : (\mathsf{Hist}_{\mathcal{G}} \to \mathsf{Act})^{\mathcal{V}^\forall}$, a variable x_i and a history π. We also recall that if $Q_j = \forall$, then $\theta(w)(x_i)(\rho) = w(x_i)(\rho)$, hence we will only define $HtoG$ for the existentially quantified variables. Moreover, by the nature of $\mathcal{M}(\emptyset, \emptyset)$ maps (see Section 5.2), if we consider a variable x_i for the second argument of θ, the first entry of θ can be simplified to a function of type $w : \mathsf{Pref}_{\leq\rho} \times \mathcal{V}^\forall \cap [x_1; x_i] \to \mathsf{Act}$. This simplifies the definition of $HtoG$.

Formally, the application $HtoG$ takes as input a positional strategy δ for player $P_\exists$ in $\mathcal{H}$ and returns a $\mathcal{M}(\emptyset, \emptyset)$ map. The value of $HtoG(\delta)(w)(x_i)(\pi)$ is defined by induction:

- **initial step (empty history):**
  - **initial step (x_1 assuming $x_1 \in \mathcal{V}^\exists$):** We set $HtoG(\delta)(w)(x_1)(\varepsilon) = \delta(\varepsilon)$.
  - **induction step (x_i assuming $x_i \in \mathcal{V}^\exists$):** As the history and the variable are fixed, the only variable part of the input for the map we are building is a function $w : \{\varepsilon\} \times (\mathcal{V}^\forall \cap [x_1; x_{i-1}]) \to \mathsf{Act}$. From x_i and w we create a function $t_{i,w} : [x_1; x_{i-1}] \to \mathsf{Act}$ that associates to $x \in \mathcal{V}^\forall$ the action $w(x)(\varepsilon)$ and to $x \in \mathcal{V}^\exists$ the action $\theta(w)(x)(\varepsilon)$. We can then create the lane $lane_{i,w} = (\varepsilon, \emptyset, i-1, t_{i,w})$ and define

$$HtoG(\delta)(w)(x_i)(\varepsilon) := \delta(GtoH_{pth}(lane_{i,w}))$$

- **induction step (non empty history):** we work on a history π assuming we have define $HtoG(\delta)$ on prefixes of π. Like before, the history and the variable are fixed and the only changing part of the input is a function w of type $\mathsf{Pref}_{\leq\rho} \times (\mathcal{V}^\forall \cap [x_1; x_{i-1}]) \to \mathsf{Act}$.
  - **initial step (x_1 assuming $x_1 \in \mathcal{V}^\exists$):** We set $HtoG(\delta)(w)(x_1)(\pi) = \delta(\mathfrak{q})$ where $\mathfrak{q} = (\mathsf{lst}(\pi), d_\pi, \varepsilon)$ is the state of $\mathcal{H}$ with d_π the state reached by π in D (the deterministic parity automaton associated with the goal of ϕ).
  - **induction step (x_i assuming $x_i \in \mathcal{V}^\exists$):** From x_i and w, we create a function $t_{i,w} : \mathcal{V} \cap [x_1; x_{i-1}] \to \mathsf{Act}$ where $t_{i,w}$ associate to $x \in \mathcal{V}^\forall$ the action $w(x)$ and to $x \in \mathcal{V}^\exists$ the action $\theta(w)(x)(\pi)$. We can then create the lane $lane_{i,w} = (\pi, \emptyset, i-1, t_{i,w})$ and define

$$HtoG(\delta)(w)(x_i)(\pi) := \delta(GtoH_{pth}(lane_{i,w}))$$

    Because δ is positional, feeding the empty function in $lane_{i,w}$ for the second component of the input is without consequence. Indeed, the second component of $lane_{i,w}$ in $GtoH_{pth}(lane_{i,w})$ only describes the actions played on π until the last state and these actions have no influences (because δ is positional).

At the end of the induction, we get a map that by construction has no side nor future dependency. Figure 5.10 illustrates the construction.

Concluding the proof

The winner of the parity game $\mathcal{H}$ gives us informations about $\mathcal{G}$.

Proposition 5.17. *Assume that $P_\exists$ is winning in $\mathcal{H}$ and let δ be a positional winning strategy, then the $\mathcal{M}(\emptyset,\emptyset)$ map $HtoG(\delta)$ is a witness that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset)} \phi$.*

Proof. Assume that $P_\exists$ is winning in $\mathcal{H}$. Toward a contradiction, assume further that $HtoG(\delta)$ is not a witness of $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset)} \phi$; then

$$\text{There is } w_0 : (\mathsf{Hist}_\mathcal{G} \to \mathsf{Act})^{\mathcal{V}^\forall} \text{ such that } \mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset)}_{HtoG(\delta)(w_0)} \beta\varphi \tag{5.5}$$

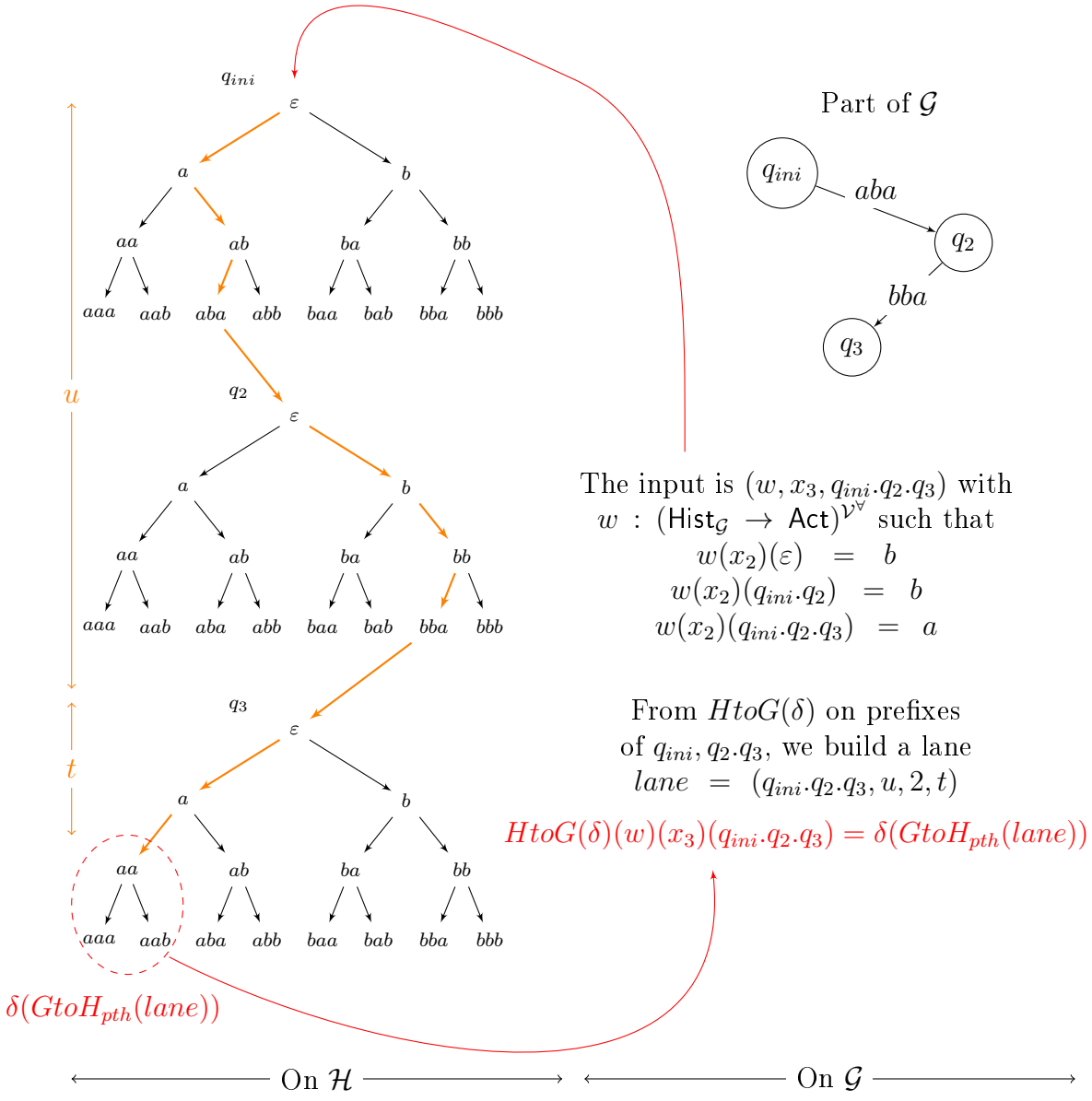


Figure 5.10: From $\mathcal{H}$ to $\mathcal{G}$ on the formula $\exists x_1.\ \forall x_2.\ \exists x_3.\ \mathsf{assign}(A_1, x_1; A_2, x_2; A_3, x_3)\ \varphi_0$.

We use w_0 to build a strategy $\overline{\delta}$ for $P_\forall$ in $\mathcal{H}$. Given a history ρ in $\mathcal{H}$ of the form

$$\rho := \Pi_{j<a}(\Pi_{0\leq i\leq l}(q_j, d_j, \mathfrak{m}_{j,i})).\Pi_{0\leq i\leq b}(q_a, d_a, \mathfrak{m}_{a,i})$$

we define $\pi = \Pi_{j\leq a}(q_j)$ and set $\overline{\delta}(\rho) := w_0(x_{b+1})(\eta)$.

Write $\eta = (q_j)_{j\in\mathbb{N}}$ for the outcome of $\beta(HtoG(\delta)(w_0))$ in $\mathcal{G}$. Then, by construction of $\overline{\delta}$, the outcome of δ and $\overline{\delta}$ in $\mathcal{H}$ will pass through the clusters $(q_j, d_j)_{j\in\mathbb{N}}$, with d_j the state reached by $(q_{j'})_{j'\leq j}$ in the automaton D associated with the LTL formula φ. Now,

because of Formula (5.5), we get that η does not satisfy φ and therefore the outcome of δ and $\overline{\delta}$ does not satisfy the parity condition. This is in contradiction with δ being the winning strategy of $P_\exists$. In the end, $HtoG(\delta)$ must be a witness that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset)} \phi$. □

We are now ready to prove the left-to-right implication of Equation (5.4), that we recall below

$$\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F)} \phi \quad \Leftrightarrow \quad \mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset)} \phi \tag{5.4}$$

We assume that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F)} \phi$. Turn-based parity games are positionally determined, meaning that one of the players is winning and that this player has a positional winning strategy. First of two possibilities, $P_\exists$ is winning. We then let δ be his positional winning strategy. By Proposition 5.17, $HtoG(\delta)$ is a witness that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset)} \phi$ and therefore the left-to-right implication of Equation (5.4) must hold. Second possibility, $P_\forall$ is winning. Then we do a similar reasoning with $P_\forall$ and $\neg\phi$ as we did with $P_\exists$ and ϕ. We get that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset)} \neg\phi$, therefore $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F)} \neg\phi$. Now combining this with Theorem 5.5, we get a contradiction with the hypothesis $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F)} \phi$. Therefore $P_\forall$ cannot be winning and the left-to-right implication of Equation (5.4) must hold.

As said at the beginning of the proof, the right-to-left implication is given by definition and from Equation (5.4) we deduce Theorem 5.14. □

Corollary 5.18. *SL[1G] model checking is 2-*EXPTIME*-complete for any of the four satisfaction relations ($\models^{\mathcal{M}(\spadesuit,\heartsuit)}$ with $\spadesuit \in \{\emptyset, S\}$ and $\heartsuit \in \{\emptyset, F\}$).*

Proof. Given a formula ϕ, we can build the parity game $\mathcal{H}$ in time $2^{2^{|P(\phi)|}}$ for some polynomial P. The game $\mathcal{H}$ has $2^{2^{|P(\phi)|}} \times |\mathcal{G}|$ states and $2^{Q(|\phi|)}$ indexes for some other polynomial Q. It can then be solved in time polynomial in the number of states and exponential in the number of indexes [56] which gives us a *2-*EXPTIME algorithm. □

5.6 Conclusion

In this chapter, we highlighted problems about dependencies in SL[BG], proposed a framework to study them and presented a few results. The work developed in this chapter draws similarities in spirit with the independence-friendly logic [35] and the dependence logic [58] (two extensions of FO to handle dependency problems). The temporal aspects of SL[BG] however pushed us toward a different framework.

As shown in Section 5.4.1, future dependencies are needed even for simple SL[CG] or SL[DG] formulas. The case of side dependencies is however different and we can remove side dependencies when we forbid future ones with SL[CG] formulas. A compilation of our results obtained so far can be found in Figure 5.11.

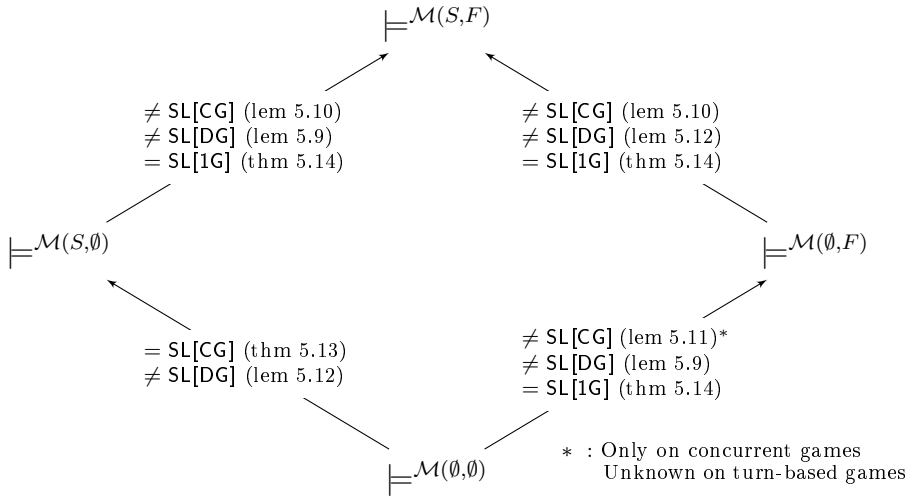


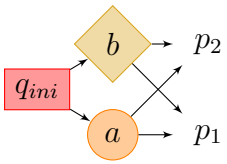
Figure 5.11: The inclusion graph of the four satisfaction relations.

Chapter 6

Unordered prefix dependencies

As we have seen in the previous chapter, when we consider formulas outside SL[1G], we can remove side dependencies in a single case and future dependencies are needed. The proofs of Lemmas 5.9, 5.10 and 5.11 (pages 122 to 124) leave us little hope to improve our results: the quantifier block is simple (2 alternations at most), the goals are trivial (reachability), the boolean combination is a simple disjunction of two elements and the games are turn-based (with the exception of Lemma 5.11).

Nonetheless, Theorem 5.13 (stating that $\models^{\mathcal{M}(S,\emptyset)}$ and $\models^{\mathcal{M}(\emptyset,\emptyset)}$ are equivalent on SL[CG]) suggests that side dependencies are easier to remove than future ones. Moreover, in Lemma 5.12 (showing that $\models^{\mathcal{M}(S,\emptyset)}$ and $\models^{\mathcal{M}(\emptyset,\emptyset)}$ are not equivalent on SL[DG]), there exists a potential improvement. We recall the game and formula on Figure 6.1.



$$\phi := \forall y.\exists z.\forall x_A.\forall x_B. \begin{cases} \mathsf{assign}(\blacksquare, x_A; \bigcirc, y; \diamond, z)\mathbf{F}\, p_1 \\ \vee \\ \mathsf{assign}(\blacksquare, x_B; \bigcirc, y; \diamond, z)\mathbf{F}\, p_2 \end{cases}$$

Figure 6.1: The game and formula of Lemma 5.12's proof.

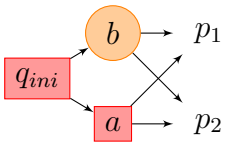
Consider a valuation χ where the strategies δ_y, δ_z, δ_{x_A} and δ_{x_B} are assigned respectively to the variables y, z, x_A and x_B. In the proof, δ_z must rely on the choices of $\delta_y(q_{ini}.a)$ because it does not know which goal will reach $q_{ini}.b$. This incapacity for a strategy to know which goal is active when making a choice is not without consequences. If we add to δ_z the information (through some sort of black box/oracle) about the choices of x_A and x_B, the formula becomes true. In this case, this corresponds to a change in the quantification order and we can rewrite the formula as

$$\forall y.\forall x_A.\forall x_B.\exists z \begin{cases} \mathsf{assign}(\blacksquare, x_A; \bigcirc, y; \diamond, z)\mathbf{F}\, p_1 \\ \vee \\ \mathsf{assign}(\blacksquare, x_B; \bigcirc, y; \diamond, z)\mathbf{F}\, p_2 \end{cases}$$

There are however cases where we could want to add this information but cannot

rewrite the formula. For example take the game and formula of Figure 6.2. One can see that

- the formula ϕ does not hold (for all four satisfaction relations $\models^{\mathcal{M}(S,\emptyset)}$, $\models^{\mathcal{M}(S,F)}$, $\models^{\mathcal{M}(\emptyset,F)}$ and $\models^{\mathcal{M}(\emptyset,\emptyset)}$).
- the formula is still false for $\models^{\mathcal{M}(S,\emptyset)}$ when we add an (intuitive and informal) notion of oracle giving to y the choices of $x_A(q_{ini})$ and $x_B(q_{ini})$.
- the modified formula with quantifiers $\forall x_A.\ \forall x_B.\ \exists y$, where $\exists y$ is moved at the end of the quantifier block, is however true for $\models^{\mathcal{M}(S,\emptyset)}$. This is due to y having knowledge of $x_A(q_{ini}.a)$ and $x_B(q_{ini}.a)$.



$$\phi := \exists y.\forall x_A.\forall x_B \begin{cases} \begin{cases} \mathsf{assign}(\square, x_A; \bigcirc, y)\mathbf{X}\, a \\ \Leftrightarrow \\ \mathsf{assign}(\square, x_B; \bigcirc, y)\mathbf{X}\, a \end{cases} \\ \vee \\ \begin{cases} \mathsf{assign}(\square, x_A; \bigcirc, y)\mathbf{F}\, p_2 \\ \Leftrightarrow \\ \mathsf{assign}(\square, x_B; \bigcirc, y)\mathbf{F}\, p_2 \end{cases} \end{cases}$$

Figure 6.2: A game and formula where we cannot reorder the quantifications

Rewriting the formula adds more information to y than what we want. The quantification order does not take into account the temporality that a strategy has relatively to the game: x_A and x_B are not only interesting on q_{ini} but also on $q_{ini}.a$, and modifying the order of the quantifiers means modifying the order on both q_{ini} and $q_{ini}.a$. Therefore, reordering the quantifications is not a sufficient solution.

For a strategy δ trying to make a choice on a history ρ, the information about which goals are still active on ρ can be added by making available the actions played by all the active strategies on prefixes of ρ. The present chapter explores this idea; for this we extend the previous framework and find when we can remove the dependencies in a similar fashion to the previous chapter.

6.1 Extending the current framework

We proceed by adding a new kind of influence: *unordered prefix*[1]. It appears when a strategy y on history π wishes to have information on the choices made on any strict prefix of π by strategies quantified before y but also from strategies quantified after y.

[1]"Unordered prefix" as we add the knowledge of all strategies on all prefixes of the current history , no matter the order of the quantifications

Remark 6.1. *The knowledge of the actions played by all strategies on prefixes of the current history π is equivalent to adding informations about which objectives are on π and which ones have deviated from the expected behaviour. Indeed, consider a full valuation χ and a SL[BG] formula ϕ. With the knowledge of $\chi_{|Pref_{<\rho}}$, for each assignment β we can check by a step by step procedure that π is a prefix of $\mathsf{out}(\beta(\chi), \pi(1))$. Note however that what happens to the objectives that deviated is not known (see Figure 6.3).*

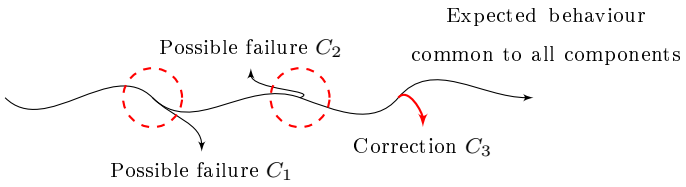


Figure 6.3: The component C_3 has knowledge of C_1 and C_2 failures and acts accordingly.

In a way, unordered prefix dependencies make the timeline of the game more important than the timeline of the quantifications: as we move through the game, knowledge of previous steps must be available in their entirety (whatever the order of the quantifications) to the system so it can make its choice. This adds up to side and future dependencies.

We update the notion of maps presented in Section 5.2. We reuse the notations $\wp$ for the block of quantifiers under consideration, $\mathcal{V}$ for the variables of $\wp$, $\mathcal{V}^\forall$ for the variables of $\mathcal{V}$ universally quantified in $\wp$ and $\mathcal{V}^\exists$ for the ones existentially quantified. We recall that a *$\wp$-map* (or simply map when $\wp$ is clear of context) is a function

$$\theta : (\mathsf{Hist} \to \mathsf{Act})^{\mathcal{V}^\forall} \to (\mathsf{Hist} \to \mathsf{Act})^{\mathcal{V}} \quad \text{or equivalently} \quad \theta : (\mathcal{V}^\forall \to \mathsf{Strat}) \to (\mathcal{V} \to \mathsf{Strat})$$

with $\theta(w)(x_i)(\rho) = w(x_i)(\rho)$ for any $w : (\mathsf{Hist} \to \mathsf{Act})^{\mathcal{V}^\forall}$, any universally quantified variable $x_i \in \mathcal{V}^\forall$ and any history ρ.

As we did in Section 5.2, we can forcibly remove some given dependencies by applying restrictions on the maps. We reuse the four conditions $\mathcal{C}(Local)$, $\mathcal{C}(\emptyset)$, $\mathcal{C}(S)$ and $\mathcal{C}(F)$ defined in Section 5.2 and define an additional one $\mathcal{C}(P)$ for unordered prefix dependencies.

- $\mathcal{C}(\emptyset)$: empty condition (always satisfied)
- $\mathcal{C}(Local)$: w_1 and w_2 coincide on $\mathcal{V}^\forall \cap [x_1; x_{i-1}]$ and on ρ, i.e.
  $$\forall y \in \mathcal{V}^\forall \cap [x_1; x_{i-1}].\ \forall \mu \leq \rho \quad w_1(y)(\mu) = w_2(y)(\mu)$$
- $\mathcal{C}(S)$: w_1 and w_2 coincide on $\mathcal{V}^\forall \cap [x_1; x_{i-1}]$ and on side histories of ρ (side), i.e.
  $$\forall y \in \mathcal{V}^\forall \cap [x_1; x_{i-1}].\ \forall \mu \text{ counter-factual of } \rho \quad w_1(y)(\mu) = w_2(y)(\mu)$$
- $\mathcal{C}(F)$: w_1 and w_2 coincide on $\mathcal{V}^\forall \cap [x_1; x_{i-1}]$ and on extensions of ρ (future), i.e.
  $$\forall y \in \mathcal{V}^\forall \cap [x_1; x_{i-1}].\ \forall \mu \text{ extension of } \rho \quad w_1(y)(\mu) = w_2(y)(\mu)$$

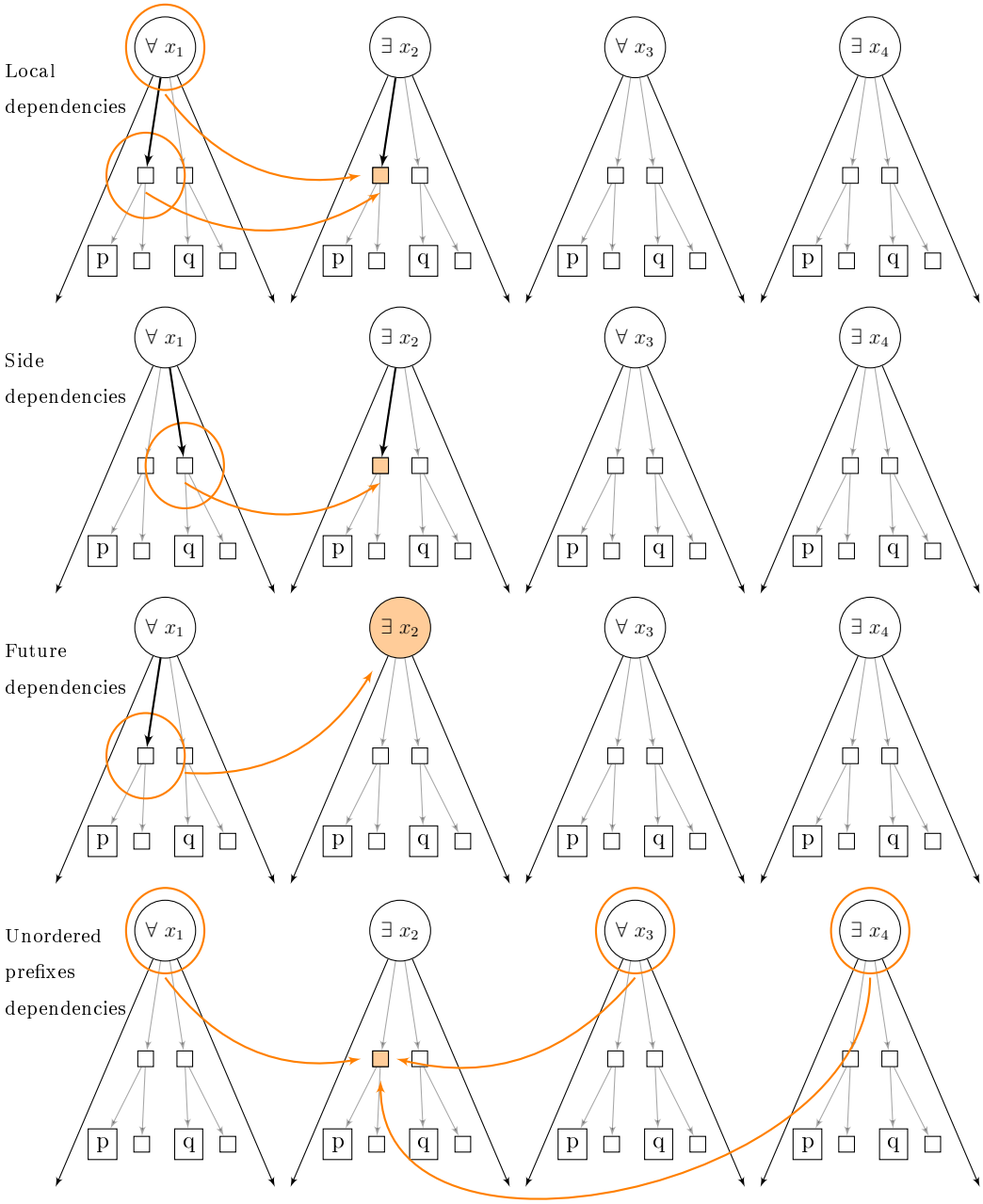


Figure 6.4: The four kinds of dependencies

- $\mathcal{C}(P)$: $\theta(w_1)$ and $\theta(w_2)$ coincide on strict prefixes of ρ (unordered prefix), i.e.
$$\forall \mu < \rho.\ \forall y \in \mathcal{V} \quad \theta(w_1)(y)(\mu) = \theta(w_2)(y)(\mu)$$

We can redefine the type of maps of Section 5.2, to which we add the parameter P to signify the addition of unordered prefix dependencies. Formally a $\mathcal{M}(\spadesuit, \heartsuit, P)$ map is a map such that

$$\left.\begin{array}{l}\forall \rho \in \mathsf{Hist},\ \forall x_i \in \mathcal{V} \\ \forall w_1, w_2 : (\mathcal{V}^{\forall} \to \mathsf{Strat})\end{array}\right\} \quad \left(\begin{array}{c}\mathcal{C}(Local) \wedge \mathcal{C}(\spadesuit) \\ \wedge\, \mathcal{C}(\heartsuit) \wedge \mathcal{C}(P)\end{array}\right) \Rightarrow \theta(w_1)(x_i)(\rho) = \theta(w_2)(x_i)(\rho)$$

To avoid confusion, we also rename the four maps $(\mathcal{M}(\spadesuit, \heartsuit))_{\spadesuit \in \{\emptyset, S\}, \heartsuit \in \{\emptyset, F\}}$ of Section 5.2 by $(\mathcal{M}(\spadesuit, \heartsuit, \emptyset))_{\spadesuit \in \{\emptyset, S\}, \heartsuit \in \{\emptyset, F\}}$ to make explicit the lack of unordered prefix dependencies.

The new condition to handle unordered prefix dependencies

$$\forall \mu < \rho.\ \forall y \in \mathcal{V} \quad \theta(w_1)(y)(\mu) = \theta(w_2)(y)(\mu)$$

is a formal way to indicate the knowledge of what all strategies played on prefixes of ρ: upon two entries w_1, w_2, the choices of an existentially quantified strategy x_j can differ between $\theta(w_1)(x_j)(\rho)$ and $\theta(w_2)(x_j)(\rho)$ only if at some prefix μ of ρ, $\theta(w_1)$ and $\theta(w_2)$ give two different outputs.

As we did in Section 5.2, we define a new satisfaction relation per kind of map. Given a closed formula $(Q_i x_i)_{i \leq l}\ \xi\big(\beta_j \varphi_j\big)_{j \leq n}$ in $\mathsf{SL[BG]}$, we set

$$\mathcal{G}, q \models^{\mathcal{M}(\spadesuit, \heartsuit, \clubsuit)} (Q_i x_i)_{i \leq l}\ \xi\big(\beta_j \varphi_j\big)_{j \leq n} \quad \Leftrightarrow \quad \begin{cases} \exists \theta \in \mathcal{M}(\spadesuit, \heartsuit, \clubsuit) \\ \forall w : (\mathcal{V}^{\forall} \to \mathsf{Strat}_{\mathcal{V}}) \\ \mathcal{G}, q \models_{\theta(w)} \xi\big(\beta_j \varphi_j\big)_{j \leq n} \end{cases}$$

A note on other frameworks

In Section 5.2.5, we explained why we chose to use maps. Another reason for maps is that they naturally adapt to the addition of actions on the paths. This is not the case of all frameworks; consider one where the universal and existential strategies are treated one after the other (as in the original semantic presented in Chapter 1). This framework could not handle more than one quantifier alternation without running into a paradox. Figure 6.5 presents the problem: to define $x_2(q_{ini}.a)$ we need $x_1(q_{ini}.a.p)$, to define $x_1(q_{ini}.a.p)$ we need $x_4(q_{ini}.a)$ thus also $x_3(q_{ini}.a)$ and $x_2(q_{ini}.a)$; we loop indefinitely. This is due to the two timelines: game and formula. The framework therefore must follow one of three options:

- the universal strategies are not treated equally to the existential ones.
- privilege the timeline of quantifications: in case of loop, suppress unordered prefix dependencies
- privilege the timeline of the game: in case of loop, suppress side and future dependencies.

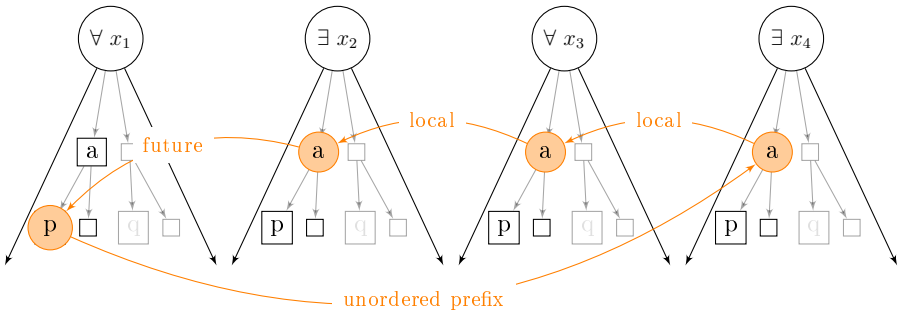


Figure 6.5: Why can't both universal and existential strategies be treated equally when making available all the actions.

Our framework (maps) follow the first choice: the universal strategies are all quantified at the same time, as a bloc. In Figure 6.5, we ask for a map θ that receive both x_1 and x_3 at the same time; the local dependency of x_3 on x_2 disappears thus breaking the loop.

6.2 The negation problem

6.2.1 Can a formula and its syntactic negation both hold on a game ?

Side and future dependencies both follow the order of quantifications and override the timeline of the game (omniscience about the future); on the other hand unordered prefix dependencies follow the timeline of the game but override by definition the order of the quantifications (once again see Figure 6.4). This difference (about which timeline to override) between side/future and unordered prefix influences is not without consequences. We find ourselves in a situation similar to the one described along Sections 5.2.4 and 5.2.5 (pages 118 and 120). We recall that the odd behaviour of the (syntactic) negation results from the two dimensions of the inputs (in the quantifier block and in the dependency parameters).

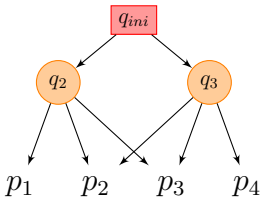


Figure 6.6: A game $\mathcal{G}$

Lemma 6.2. *There exists a game $\mathcal{G}$, a state q of $\mathcal{G}$ and a SL[BG] formula ϕ such that*

$$\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F,P)} \phi \qquad \textit{and} \qquad \mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F,P)} \neg\phi$$

Proof. Consider the turn-based game on Figure 6.6 with two agents ■ and ● , and ϕ the SL[BG] formula below.

$$\phi \;:=\; \forall x_1.\exists y_1.\exists y_2.\exists x_2 \begin{cases} \mathsf{assign}(■, y_1; ●, x_1)\mathbf{F}\, p_2 \;\Rightarrow \mathsf{assign}(■, y_2; ●, x_2)\mathbf{F}\, p_1 \\ \wedge \\ \mathsf{assign}(■, y_1; ●, x_1)\mathbf{F}\, p_3 \;\Rightarrow \mathsf{assign}(■, y_2; ●, x_2)\mathbf{F}\, p_4 \end{cases}$$

In the following, we regroup the four quantifications in the usual notation $\wp$. We also write ψ_1 for the first implication and ψ_2 for the second one. We first prove that

$$\exists \theta \in \mathcal{M}(S,F,P)\ \forall w : (\mathsf{Hist}_{\mathcal{G}} \to \mathsf{Act})^{\mathcal{V}^{\forall}} \qquad \mathcal{G}, q_{ini} \models_{\theta(w)} \psi_1 \wedge \psi_2 \tag{6.1}$$

Consider a function $w : (\mathsf{Hist}_{\mathcal{G}} \to \mathsf{Act})^{\{x_1\}}$ as input. To build θ, we decompose by cases based upon the choice of x_1 on the history $q_{ini}.q_2$.

- If $w(x_1)(q_{ini}.q_2) = p_2$ then both $y_1(q_{ini})$ and $y_2(q_{ini})$ play to q_2 while $x_2(q_{ini}.q_2)$ plays to p_1. This ensures that the outcomes of $\{● \to x_1; ■ \to y_1\}$ and $\{● \to x_2; ■ \to y_2\}$ will respectively see p_2 and p_1, ensuring $\mathcal{G}, q_{ini} \models_{\theta(w)} \psi_1 \wedge \psi_2$. Note that y_1 and y_2 use future dependencies to gather knowledge of $w(x_1)(q_{ini}.q_2)$.

- If $w(x_1)(q_{ini}.q_2) = p_3$ then $y_1(q_{ini})$ plays to q_2, $y_2(q_{ini})$ play to q_3 and $x_2(q_{ini}.q_3)$ plays to p_4. The outcomes of $\{● \to x_1; ■ \to y_1\}$ and $\{● \to x_2; ■ \to y_2\}$ will respectively see p_3 and p_4, ensuring $\mathcal{G}, q_{ini} \models_{\theta(w)} \psi_1 \wedge \psi_2$. Beside future dependencies from y_1 and y_2 on $w(x_1)(q_{ini}.q_2)$, we also have a side dependency from x_2 on $w(x_1)(q_{ini}.q_2)$.

- If $w(x_1)(q_{ini}.q_2) = p_1$ then no matter what x_2, y_1, y_2 play, the outcome of $\{● \to x_1; ■ \to y_1\}$ will see neither p_2 nor p_3, ensuring $\mathcal{G}, q_{ini} \models_{\theta(w)} \psi_1 \wedge \psi_2$.

We can then merge all three cases into a single $\mathcal{M}(S,F,\emptyset)$ map to ensure that (6.1) holds.

It remains to show that, writing $\overline{\wp} := \exists x_1.\ \forall y_1.\ \forall y_2.\ \forall x_2$,

$$\exists \overline{\theta} \in \mathcal{M}(S,F,P)\ \forall \overline{w} : (\mathsf{Hist}_{\mathcal{G}} \to \mathsf{Act})^{\{y_1,y_2,x_2\}} \qquad \mathcal{G}, q_{ini} \models_{\overline{\theta}(\overline{w})} \neg\psi_1 \vee \neg\psi_2 \tag{6.2}$$

As we are looking for a $\mathcal{M}(S,F,P)$ map, to decide the action of x_1 on $q_{ini}.q_2$ or $q_{ini}.q_3$, we may assume knowledge of what y_1, y_2 played on q_{ini}. Again we proceed by cases.

- If y_2 plays to q_2 then $x_1(q_{ini}.q_2) = x_1(q_{ini}.q_3) := p_3$, this makes the outcome of $\{● \to x_1; ■ \to y_1\}$ see p_3 while the choices of y_2 ensure that $\{● \to x_2; ■ \to y_2\}$ will not see p_4 thus $\mathcal{G}, q_{ini} \models_{\overline{\theta}(\overline{w})} \neg\psi_1 \vee \neg\psi_2$.

- Similarly, if y_2 plays to q_3 then choosing $x_1(q_{ini}.q_2) = x_1(q_{ini}.q_3) = p_2$ ensures $\mathcal{G}, q_{ini} \models_{\overline{\theta}(\overline{w})} \neg\psi_1 \vee \neg\psi_2$.

Again, merging the cases into a single $\mathcal{M}(\emptyset,\emptyset,P)$ map we get that (6.2) holds. □

In a sense, the first among $\{x_1\}$ and $\{x_2, y_1, y_2\}$ to declare its choices will lose. In the spirit of Section 5.2.4, Lemma 6.2 shows that what we could expect to be the *syntactic* negation does not correspond to the *semantic* negation. This discrepancy between a formula and its negation raises an obvious question: for which formulas $\phi \in \mathsf{SL[BG]}$ can we ensure that, whatever the game $\mathcal{G}$,

$$\mathcal{G}, q_{ini} \models^{\mathcal{M}(♠,♡,♣)} (Q_i x_i)_{i\leq l}\ \xi(\beta_j\varphi_j)_{j\leq n} \quad \Rightarrow \quad \mathcal{G}, q_{ini} \not\models^{\mathcal{M}(♠,♡,♣)} (\overline{Q_i} x_i)_{i\leq l}\ \neg\xi(\beta_j\varphi_j)_{j\leq n}$$

where $\spadesuit \in \{\emptyset, S\}$, $\heartsuit \in \{\emptyset, F\}$ and $\clubsuit \in \{\emptyset, P\}$?

As we have seen in Section 5.2, this is already the case for $\mathcal{M}(\emptyset,\emptyset,\emptyset)$, $\mathcal{M}(\emptyset,F,\emptyset)$, $\mathcal{M}(S,\emptyset,\emptyset)$ and $\mathcal{M}(S,F,\emptyset)$. The two lemmas and theorem below complete the answer. We provide only a sketch of proof for Lemma 6.3 and move the proof of Lemma 6.4 in annex 6.A (page 153).

Lemma 6.3. *There exists a game $\mathcal{G}$, one of its states q_{ini} and a SL[BG] formula ϕ such that*

$$\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,F,P)} \phi \qquad \textit{and} \qquad \mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,F,P)} \neg\phi$$

Sketch of proof. The proof consists in taking the formula ϕ defined below and use it over the game of Figure 5.7a (page 123).

$$\phi \;:=\; \forall y.\, \exists x_A.\, \forall x_B.\, \big(\mathsf{assign}(\bigcirc, y; \square, x_A).\, \mathbf{F}\, p_1 \vee \mathsf{assign}(\bigcirc, y; \square, x_B).\, \mathbf{F}\, p_2\big)$$

Then one can easily build two witnesses to ensure that both $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,F,P)} \phi$ and $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,F,P)} \neg\phi$. □

Lemma 6.4. *There exists a game $\mathcal{G}$, one of its states q_{ini} and a SL[BG] formula ϕ such that*

$$\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,P)} \phi \qquad \textit{and} \qquad \mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,P)} \neg\phi$$

Note that while both Lemmas 6.3 and 6.4 imply Lemma 6.2, Lemma 6.2 has the merit of using all three parameters of the $\models^{\mathcal{M}(S,F,P)}$ satisfaction relation, hence the result on $\models^{\mathcal{M}(S,F,P)}$ is not a gimmick heritage from the other relations.

Theorem 6.5. *For any game $\mathcal{G}$, any initial state q_{ini}, any formula ϕ in SL[BG], it holds*

$$\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi \quad \Rightarrow \quad \mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} \neg\phi$$

Proof. The idea, similarly to what we explained after Theorem 5.5 (page 117), is to assume that both a formula and its negation hold on the same game from the same state, to deduce two witnesses and to confront them. The only change lies in the way we confront the maps as their nature has changed. First, we write $\mathcal{V} := \{x_i \mid i \leq n\}$, $\mathcal{V}^\exists := \{x_i \mid Q_i = \exists\}$ and $\mathcal{V}^\forall := \{x_i \mid Q_i = \forall\}$. Now, assume that both formulas below hold:

$$\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{i\leq l}\, \xi(\beta_j\varphi_j)_{j\leq n} \quad \textit{and} \quad \mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} (\overline{Q_i} x_i)_{i\leq l}\, \neg\xi(\beta_j\varphi_j)_{j\leq n}$$

We then have two $\mathcal{M}(\emptyset,\emptyset,P)$ maps θ and $\overline{\theta}$ such that

$$\forall w : (\mathsf{Hist}_\mathcal{G} \to \mathsf{Act})^{\mathcal{V}^\forall} \qquad \mathcal{G}, q_{ini} \models_{\theta(w)} \xi(\beta_j\varphi_j)_{j\leq n} \tag{6.3}$$

$$\forall \overline{w} : (\mathsf{Hist}_\mathcal{G} \to \mathsf{Act})^{\mathcal{V}^\exists} \qquad \mathcal{G}, q_{ini} \models_{\overline{\theta}(\overline{w})} \neg\xi(\beta_j\varphi_j)_{j\leq n} \tag{6.4}$$

We confront θ with $\overline{\theta}$ to get a contradiction: we build a valuation χ of domain $\mathcal{V}$ inductively (Figure 6.7 gives an intuition):

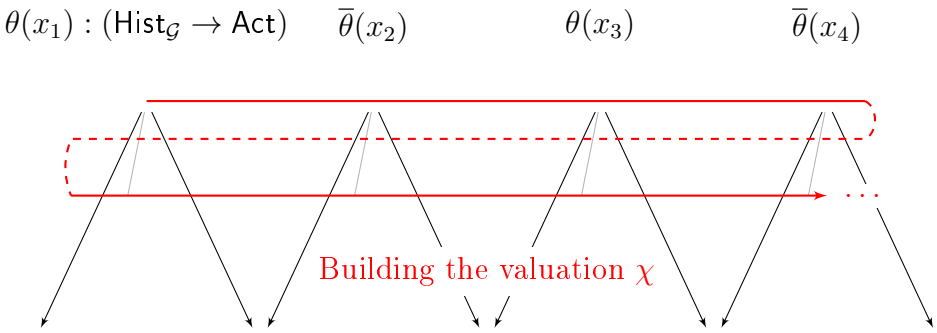


Figure 6.7: How to confront θ and $\overline{\theta}$, two $\mathcal{M}(\emptyset, \emptyset, P)$ maps

- consider the empty history ε. For any $x_i \in \mathcal{V}$ and having build χ on ε for any $x_{i'}$ where $i' < i$, then
  - if $x_i \in \mathcal{V}^{\exists}$, we set $\chi(x_i)(\varepsilon) := \theta(\bigcup_{i'<i} \chi_{|\varepsilon \text{ and } x_{i'}})(x_i)(\varepsilon)$
  - if $x_i \in \mathcal{V}^{\forall}$, we set $\chi(x_i)(\varepsilon) := \overline{\theta}(\bigcup_{i'<i} \chi_{|\varepsilon \text{ and } x_{i'}})(x_i)(\varepsilon)$

  Because θ and $\overline{\theta}$ are $\mathcal{M}(\emptyset, \emptyset, P)$ maps, feeding them with $\bigcup_{i'<i} \chi_{|\varepsilon \text{ and } x_{i'}}$ is sufficient.
- consider a history ρ where χ has been defined on any prefix of ρ, for any $x \in \mathcal{V}$ and any prefix ρ' of ρ. For a variable $x_i \in \mathcal{V}$, having build χ on ρ for any $x_{i'}$ where $i' < i$, then we define $\chi_{\rho,i}$ to be the valuation

$$\chi_{\leq(\rho,i)} \;:=\; \chi_{|\mathsf{Pref}_{<\rho}} \;\cup\; \bigcup_{i'<i} \chi(x_{i'})(\rho)$$

  - if $x_i \in \mathcal{V}^{\exists}$, we then set $\chi(x_i)(\rho) := \theta(\chi_{\leq(\rho,i)})(x_i)(\rho)$
  - if $x_i \in \mathcal{V}^{\forall}$, we then set $\chi(x_i)(\rho) := \overline{\theta}(\chi_{\leq(\rho,i)})(x_i)(\rho)$

  Again, because θ and $\overline{\theta}$ are $\mathcal{M}(\emptyset, \emptyset, P)$ maps, feeding them a partial first entry is sufficient.

Now, by construction, $\theta(\chi_{|\mathcal{V}^{\forall}}) = \overline{\theta}(\chi_{|\mathcal{V}^{\exists}}) = \chi$. Then, by Formulas (6.3) and (6.4), we have both $\mathcal{G}, q_{ini} \models_{\chi} \xi(\beta_j \varphi_j)_{j \leq n}$ and $\mathcal{G}, q_{ini} \models_{\chi} \neg\xi(\beta_j \varphi_j)_{j \leq n}$ which is a contradiction. □

6.2.2 Can a formula and its negation both fail to hold on a game ?

The three lemmas below outline the possibilities for both a formula and its negation to fail on a game relatively to the satisfaction relations with unordered prefix dependencies. The situation is akin to Chapter 5.

Lemma 6.6. *There exists a formula ϕ in SL[BG], a game $\mathcal{G}$ and an initial state q_{ini} such that for any $\heartsuit \in \{\emptyset, F\}$, it holds that*

$$\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\heartsuit,P)} \phi \quad \text{and} \quad \mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\heartsuit,P)} \neg\phi$$

Proof. Similar to the one of Lemma 5.7 (page 119), same game and same formula. □

Lemma 6.7. *There exists a formula ϕ in SL[BG], a game $\mathcal{G}$ and an initial state q_{ini} such that for any $\spadesuit \in \{\emptyset, S\}$, it holds that*

$$\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\spadesuit,\emptyset,P)} \phi \quad \text{and} \quad \mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\spadesuit,\emptyset,P)} \neg\phi$$

Proof. Similar to the one of Lemma 5.8 (page 119), same game and same formula. □

Lemma 6.8. *For any formula ϕ in SL[BG], any game $\mathcal{G}$ and any initial state q_{ini},*

$$\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(S,F,P)} \phi \quad \Rightarrow \quad \mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F,P)} \neg\phi$$

Proof. Due to the correspondence of $\models^{\mathcal{M}(S,F,\emptyset)}$ and $\models$ (the original relation defined in Chapter 1), at least one of ϕ and $\neg\phi$ must hold on $\mathcal{G}$ relatively to $\models^{\mathcal{M}(S,F,\emptyset)}$. The one that holds for $\models^{\mathcal{M}(S,F,\emptyset)}$ also holds for $\models^{\mathcal{M}(S,F,P)}$. □

6.3 Needed and avoidable dependencies

We start by an adaptation of Lemma 5.9 (page 122) in Section 5.4 to the case with unordered prefix dependencies. The proof of the result below is similar to the one of Lemma 5.9, we simply add that, by construction, unordered prefix dependencies cannot play a role.

Lemma 6.9. *There exist a formula $\phi \in$ SL[DG], a turn-based game $\mathcal{G}$ and one of its states q_{ini} such that*

- $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$ *and* $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,F,P)} \phi$
- $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(S,\emptyset,P)} \phi$ *and* $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F,P)} \phi$

Similarly, we can adapt the first part of Lemma 5.10 (page 123) to satisfaction relations with unordered prefix dependencies. However the second part of Lemma 5.10 cannot be adapted to work with $\models^{\mathcal{M}(\emptyset,F,P)}$ and $\models^{\mathcal{M}(S,F,P)}$: unordered prefix dependencies can replace the role of side dependencies thus making side dependencies non essential[2]. Reusing the game and formula of Lemma 5.10's proof, we get $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,F,P)} \phi$.

Lemma 6.10. *There exist a formula $\phi \in$ SL[CG], a turn based game $\mathcal{G}$ and one of its states q_{ini} such that*

$$\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(S,\emptyset,P)} \phi \text{ and } \mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F,P)} \phi$$

[2]Page 123, the strategies x_A and x_B dispatch one goal on $q_{ini}.a$ and the other on $q_{ini}.b$ based on the choices of y such that the goal passing through $q_{ini}.a$ is satisfied. In Chapter 5, the strategy z used side dependency to play the opposite of y and satisfy the goal passing through $q_{ini}.b$. With knowledge of the actions played on q_{ini}, the strategy z knows the goal active on $q_{ini}.b$ and thus can deduce the choice of y without side dependencies.

The proofs of Lemma 5.11 and Theorem 5.13 (page 124 and 125) also extend to unordered prefix dependencies, hence:

Lemma 6.11. *There exist a formula $\phi \in$ SL[CG], a* concurrent *game $\mathcal{G}$ and one of its states q_{ini} such that $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$ and $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,F,P)} \phi$.*

Theorem 6.12. *For any formula $\phi \in$ SL[CG], any concurrent game $\mathcal{G}$ and any state q_{ini} of $\mathcal{G}$,*

$$\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,P)} \phi \quad \Rightarrow \quad \mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$$

With methods and models similar to the ones of Section 5.4, we study the remaining cases.

(a) A turn-based game with two agents. (b) Another turn-based game.

Figure 6.8: The games used in Lemmas 6.13 and 6.14.

Lemma 6.13. *There exist a formula $\phi \in$ SL[CG], a turn-based game $\mathcal{G}$ and one of its states q_{ini} such that for all $\spadesuit \in \{\emptyset, S\}$ and all $\heartsuit \in \{\emptyset, F\}$ it holds $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\spadesuit,\heartsuit,\emptyset)} \phi$ and $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit,P)} \phi$.*

Proof. Take the game $\mathcal{G}$ of Figure 6.8a and the following formula

$$\phi := \exists y. \forall x_A. \exists x_B \begin{cases} \mathsf{assign}(\bigcirc, y; \square, x_A).\ \mathbf{F}\, p_1 \\ \wedge \\ \mathsf{assign}(\bigcirc, y; \square, x_B).\ \mathbf{F}\, p_2 \end{cases}$$

By construction of ϕ and $\mathcal{G}$, no side and future dependencies are needed hence we treat all items at once. Take $\spadesuit \in \{\emptyset, S\}$ and $\heartsuit \in \{\emptyset, F\}$. We prove $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit,P)} \phi$. As unordered prefix dependencies are allowed, $y(q_{ini}.a)$ and $y(q_{ini}.b)$ may assume knowledge of what x_A played on q_{ini}. If $x_A(q_{ini}) = a$ then we set $x_B(q_{ini}) = b$ (x_B having a local influence from x_A), $y(q_{ini}.a) = p_1$ and $y(q_{ini}.b) = p_2$. One can then see that both objectives hold. On the other hand, if $x_A(q_{ini}) = b$, we set $x_B(q_{ini}) = a$, $y(q_{ini}.a) = p_2$ and $y(q_{ini}.b) = p_1$; again both goals will hold. We can then build a $\mathcal{M}(\spadesuit, \heartsuit, P)$ map witnessing that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit,P)} \phi$.

Now, if ϕ was to admit a $\mathcal{M}(\spadesuit, \heartsuit, \emptyset)$ map as witness, then which strategy y prescribes in a (resp. in b) would not depend on the values of x_A. If $y(q_{ini}.a) = y(q_{ini}.b)$ then trivially one of the two goals will not be satisfied. If $y(q_{ini}.a) = p_1$ then for $x_A(q_{ini}) = b$, the first goal is not satisfied. Finally, if $y(q_{ini}.a) = p_2$ and $y(q_{ini}.b) = p_1$ then for $x_A(q_{ini}) = a$, the first goal is not satisfied. So knowledge on the prefixes for strategies quantified before is required to build an adequate y. □

Lemma 6.14. *There exist a formula $\phi \in$ SL[DG], a turn-based game $\mathcal{G}$ and one of its states q_{ini} such that for any $\spadesuit \in \{\emptyset, S\}$ and any $\heartsuit \in \{\emptyset, F\}$, it holds $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\spadesuit,\heartsuit,\emptyset)} \phi$ and $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit,P)} \phi$.*

Proof. Take the game of Figure 6.8b and the following formula

$$\phi := \exists y.\forall x_A.\forall x_B.\forall z \begin{cases} \mathsf{assign}(\bigcirc, y; \square, x_A; \diamond, z).\ \mathbf{F}\, p_1 \\ \vee \\ \mathsf{assign}(\bigcirc, y; \square, x_B; \diamond, z).\ \mathbf{F}\, p_2 \end{cases}$$

Again, by construction of ϕ and $\mathcal{G}$, no side and future dependencies are possible hence we do all items at once: fix $\spadesuit \in \{\emptyset, S\}$ and $\heartsuit \in \{\emptyset, F\}$. We start by showing that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit,P)}$. To build a strategy on $q_{ini}.a$, y has knowledge of what x_A and x_B played on the prefix q_{ini}, if $x_A(q_{ini}) = x_B(q_{ini}) = a$ then whatever the choice of y, one of the goals will be satisfied. Similarly, if $x_A(q_{ini}) = x_B(q_{ini}) = b$ then whatever the choice of z, one of the goals will be satisfied. Two cases remain, the first is when $x_A(q_{ini}) = a$ and $x_B(q_{ini}) = b$, then by taking $y(q_{ini}.a)$ playing to p_1 the first goal will be achieved; the second is when $x_A(q_{ini}) = b$ and $x_B(q_{ini}) = a$ then taking $y(q_{ini}.a) = p_2$ ensures the second objective. Hence $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit,P)}$.

Next in order is to show that $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\spadesuit,\heartsuit,\emptyset)}$. To build y on $q_{ini}.a$, we do not have knowledge of the choices of x_A and x_B. Assume we set $y(q_{ini}.a) = p_1$ then x_A can play to b, x_B to a and z to p_2 and neither of the goals will be satisfied. On the other hand, if $y(q_{ini}.a) = p_2$ then x_A can play to a, x_B to b and z to p_1 and, again, neither of the goals will be satisfied. Hence $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\spadesuit,\heartsuit,\emptyset)}$ and knowledge on the prefixes is needed to ensure ϕ. □

6.4 Moving informations through the two timelines

The addition of unordered prefix dependencies to the side and future ones gives rise to original situations. As we will see with the theorem below, we can use them to pass informations that one formerly thought to be inaccessible.

Theorem 6.15. *There exist a concurrent game $\mathcal{G}$, one of its states q_{ini} and a SL[DG] formula ϕ such that*

- $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$ *and* $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,P)} \phi$.
- $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,F,P)} \phi$ *and* $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F,P)} \phi$.

Proof. Consider the game $\mathcal{G}$ of Figure 6.9 with 6 agents □, ⬠, ⬠, ◇, ①, and ②. Each agent can only influence the state represented in its name, for example ① and ② are the only agents having an influence on the state named c. To ease the reading we

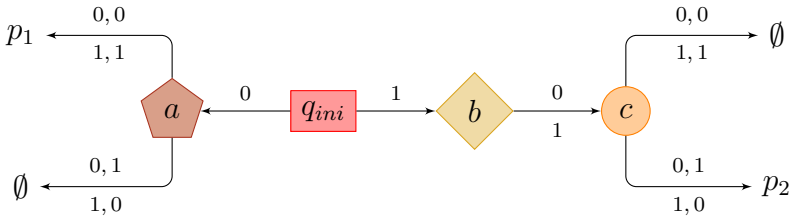


Figure 6.9: A concurrent game $\mathcal{G}$.

only represented the actions of the active agents on the transitions. We define a SL[DG] formula

$$\phi := \exists x_{p1}^{\blacksquare}.\ \exists x_{p2}^{\blacksquare}.\ \exists x_1^{\bullet}.\ \forall x_2^{\bullet}.\ \exists x_1^{\pentagon}.\ \forall x_2^{\pentagon}.\ \exists x^{\blacklozenge}$$
$$\begin{cases} \mathsf{assign}(\blacksquare, x_{p1}^{\blacksquare}; 1, x_1^{\pentagon}; 2, x_2^{\pentagon}; \blacklozenge, x^{\blacklozenge}; 1, x_1^{\bullet}; 2, x_2^{\bullet}).\mathbf{F}\, p_1 \\ \vee \\ \mathsf{assign}(\blacksquare, x_{p2}^{\blacksquare}; 1, x_1^{\pentagon}; 2, x_2^{\pentagon}; \blacklozenge, x^{\blacklozenge}; 1, x_1^{\bullet}; 2, x_2^{\bullet}).\mathbf{F}\, p_2 \end{cases}$$

For the first point, Table 6.1a induces a $\mathcal{M}(S, \emptyset, P)$ witness of $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,P)} \phi$ and Table 6.1b proves that $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$. The first table gives a description of each existentially quantified strategy (on the state it is active on) based upon the choice of the two universally quantified strategies ($x_2^{\bullet}$ and $x_2^{\pentagon}$). We also indicate the knowledge of each strategy on the right. Table 6.1b does a case-by-case analysis to show that, whatever the $\mathcal{M}(\emptyset, \emptyset, P)$ map chosen, it cannot be a witness of ϕ holding on $\mathcal{G}$ from q_{ini} for $\models^{\mathcal{M}(\emptyset,\emptyset,P)}$.

We can get the second point simply by noticing that future dependencies may only intervene for $x^{\blacklozenge}$ but do not modify the results, hence $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F,P)} \phi$ and $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,F,P)} \phi$. □

Intuitively, the proof of Theorem 6.15 shows that we can pass the information of a universal variable $x_2^{\pentagon}$ to an existential variable $x_1^{\pentagon}$ on the same history, despite $x_1^{\pentagon}$ being quantified before $x_2^{\pentagon}$. This comes from having a state c privileging the timeline of the game and the states a, b privileging the timeline of the quantifiers. Figure 6.10 illustrates the reasoning: the choice of $x_2^{\pentagon}$ is passed to $x^{\blacklozenge}$ using side dependencies (by the timeline of the quantifications), then to $x_1^{\bullet}$ using unordered prefix dependencies (by the timeline of the game), next to $x_2^{\bullet}$ using local dependencies (if $x_2^{\bullet}$ refuses to carry the choice of $x_2^{\pentagon}$, the formula is trivially satisfied), and back to $x_1^{\pentagon}$ using side dependencies (by the timeline of the quantifications).

We use the actions in the transition from b to c to hide the information about $x_2^{\pentagon}$. We could have also passed the information using the objectives instead of the actions, hence Theorem 6.15 still holds for turn-based games. Indeed, we can add a new goal ψ_{add} that carries the information from b to c: if ψ_{add} goes from b to c, this means $x_2^{\pentagon}(q_{ini}.a) = 1$ and if it goes from b to some new state branching from b, it means $x_2^{\pentagon}(q_{ini}.a) = 0$. In a

w ($\forall x$)	x_2	0	0	1	1	
	x_2	0	1	0	1	
$\theta(w)$ ($\exists x$)	x_{p1}	0	0	0	0	Depend on nothing
	x_{p2}	1	1	1	1	Depend on nothing
	x_1	0	1	0	1	Depend on x_{p1}, x_{p2} and x ($x_1 = x$)
	x_1	0	0	1	1	Depend on x_{p1}, x_{p2}, x and x_2 ($x_1 = x_2$)
	x	0	1	0	1	Depend on x_{p1}, x_{p2} and x_2 ($x = x_2$)
result	Goal 1	ok	no	no	ok	
	Goal 2	no	ok	ok	no	

(a) Building the map θ to prove $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,P)}$ in Theorem 6.15.

		if $x_{p2} = 0$		if $x_{p1} = 1$		if $x_{p1} = 0$ and $x_{p2} = 1$
		if ($x_1 = 0$)	if ($x_1 = 1$)	if ($x_1 = 0$)	if ($x_1 = 1$)	Case with $x_1 = 0$ and $x_1 = 0$; other cases are similar
w	x_2	-	-	0	1	0
	x_2	1	0	-	-	1
$\theta(w)$	x_{p1}	-	-	1	1	0
	x_{p2}	0	0	-	-	1
	x	-	-	-	-	-
	x_1	0	1	-	-	0
	x_1	-	-	0	1	0
result	Goal 1	$q_{ini}.a.\emptyset$ no	$q_{ini}.a.\emptyset$ no	$q_{ini}.b.c.\emptyset$ no	$q_{ini}.b.c.\emptyset$ no	$q_{ini}.a.\emptyset$ no
	Goal 2	$q_{ini}.a.\emptyset$ no	$q_{ini}.a.\emptyset$ no	$q_{ini}.b.c.\emptyset$ no	$q_{ini}.b.c.\emptyset$ no	$q_{ini}.b.c.\emptyset$ no

(b) Proving $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)}$ in Theorem 6.15. Variables x_{p1} and x_{p2} depend on nothing. Variable x depends on x_{p1} and x_{p2}. Variable x_1 depends on x_{p1} and x_{p2}. Variable x_1 depends on x_{p1}, x_{p2} and x. In particular, x_1 and x_1 are effectively independent of w.

Table 6.1: Tables used for the proof of Theorem 6.15.

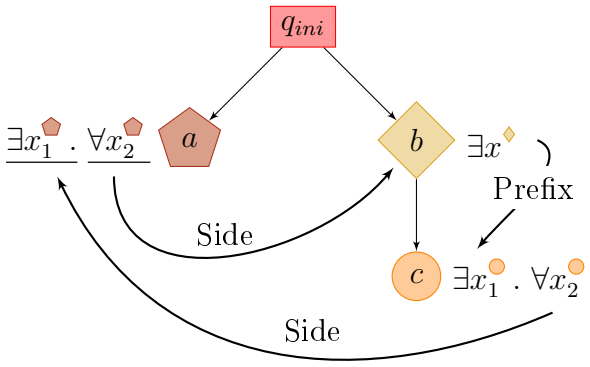


Figure 6.10: Idea behind Theorem 6.15

broader sense, the choice of the framework (adding the full sequence of actions) does not impact the result of Theorem 6.15, as long as a strategy can know which goals are active on the current history.

This passing of informations using both timelines (of the game and the quantifiers) can also be done when there is a single goal. The idea of Theorems 6.16 and 6.17 (below) are similar to the one of Theorem 6.15 and we move the proofs to annexes 6.B and 6.C (pages 156 and 158).

Theorem 6.16. *There exist a concurrent game $\mathcal{G}$, one of its states q_{ini} and a SL[1G] formula ϕ such that $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,F,P)} \phi$ and $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F,P)} \phi$.*

Theorem 6.17. *There exist a formula $\phi \in$ SL[1G] and a concurrent game structure $\mathcal{G}$ such that*

- $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,F,\emptyset)} \phi$ *and* $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,F,P)} \phi$
- $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$ *and* $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,F,P)} \phi$
- $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(S,F,\emptyset)} \phi$ *and* $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F,P)} \phi$
- $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(S,\emptyset,P)} \phi$ *and* $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F,P)} \phi$

The hope we had to suppress some of the dependencies (by adding unordered prefix dependencies) dies with theorems 6.15, 6.16 and 6.17. Theorems 6.16 and 6.17 are particularly interesting: they show that even for a single objective, adding information about the actions played on a history gives rise to some unexpected behaviour. This comforts us in the idea that future and side dependencies are a hassle: making the actions of all variables available has the side effect of bypassing quantification order. Note that Theorems 6.16 and 6.17 cannot be adapted to LTL as we need more than one quantifier alternation. They can however be adapted for other logics: BSIL, CATL, ATL_{sc} and CHP-SL (the first version of SL, see [15]). The proofs use the same games as Theorems 6.16 and 6.17 but adapt the formulas to the syntax of these logics. Concerning

the dependency problem, the combination of Theorems 6.15, 6.16 and 6.17 means that adding information about the history is not enough to suppress the impact of side and future dependencies.

6.5 Additional results on SL[1G]

Theorem 6.18. *For any SL[1G] formula ϕ, any concurrent game $\mathcal{G}$ and any state q_{ini} of $\mathcal{G}$,*

$$\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi \quad \Leftrightarrow \quad \mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,\emptyset)} \phi$$

Proof. The right-to-left part of the equivalence is trivially satisfied. The proof of the left-to-right implication is for the most part similar to the one of Theorem 5.14 (page 127). We build the same game $\mathcal{H}$ and draw the same correspondences. We only change the paragraph "Concluding the proof". We recall Proposition 5.17, which is the result we get at the end of the construction.

Proposition 5.17. *Assume that $P_\exists$ is winning in $\mathcal{H}$ and let δ be a positional winning strategy, then the $\mathcal{M}(\emptyset,\emptyset)$ map $HtoG(\delta)$ is a witness that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset)} \phi$.*

We assume that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$. Turn-based parity game are positionally determined, meaning that one of the player has a positional winning strategy. First of two possibilities, $P_\exists$ is winning. We then let δ be a positional winning strategy for $P_\exists$. By Proposition 5.17, $HtoG(\delta)$ is a witness of $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,\emptyset)} \phi$ and the left-to-right implication holds. Second possibility, $P_\forall$ is winning. Then, we do a similar reasoning with $P_\forall$ and $\neg\phi$ as we did with $P_\exists$ and ϕ, we get $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,\emptyset)} \neg\phi$ and therefore $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} \neg\phi$. Now combining this with Theorem 6.5 (page 142), we get that $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$ which is in contradiction with the hypothesis made at the start of the proof. Therefore $P_\forall$ cannot be winning in $\mathcal{H}$ and the left-to-right implication of the equivalence must hold. □

In a similar vein to Corollary 5.18 (page 132), we get

Corollary 6.19. *The model checking problem of SL[1G] formulas relatively to the $\models^{\mathcal{M}(\emptyset,\emptyset,P)}$ satisfaction relation is 2-EXPTIME-complete.*

Theorem 6.20. *For any SL[1G] formula ϕ, any concurrent game $\mathcal{G}$ and any state q_{ini} of $\mathcal{G}$,*

$$\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,\emptyset)} \phi \quad \Leftrightarrow \quad \mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,P)} \phi$$

Proof. $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,\emptyset)} \phi$ if and only if $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,\emptyset)} \phi$ (Theorem 5.13) if and only if $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$ (Theorem 6.18) if and only if $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,P)} \phi$ (Theorem 6.12). □

6.6 Conclusion

Pushed by the disappointing results of Chapter 5, we extended the framework: a strategy δ choosing an action on a history ρ has access to all the actions played by all the strategies on strict prefixes of ρ (we call this additional information unordered prefix). In this

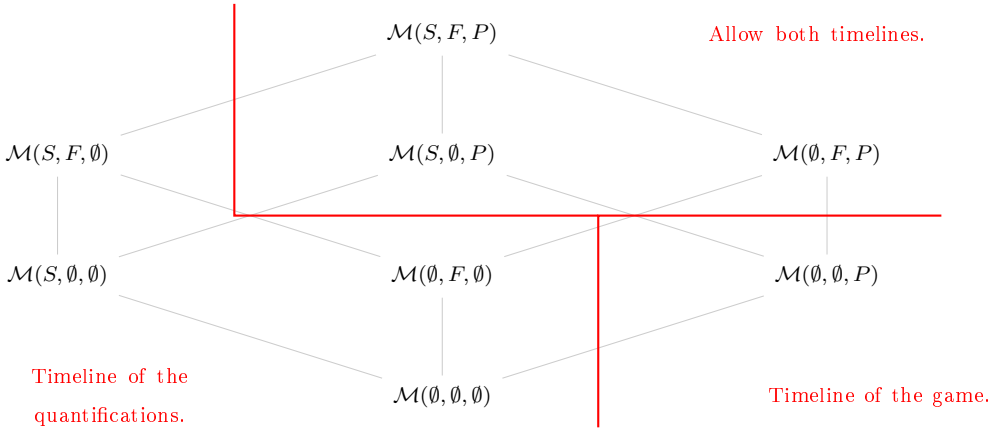


Figure 6.11: Which maps follow which timeline.

framework, the combination of side and future dependencies with unordered prefix dependencies gives rise to some unexpected results: certain dependencies cannot be suppressed in SL[1G], a formula and its negation can both hold onto the game (see Figures 6.11 and 6.12). While the framework makes it possible (and understandable), the result is still surprising because, in essence, we just made the actions played earlier available in their entirety to the strategies. Ultimately, adding all the actions played on the prefix of the current history does not help us remove some of the dependencies. Our results are summed up on Figure 6.13.

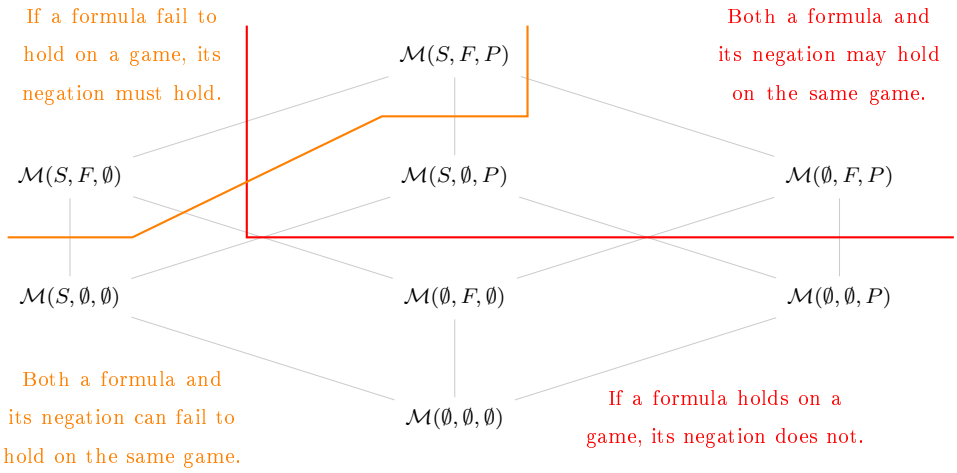


Figure 6.12: When can a formula and its (syntactic) negation hold both hold on a game? When can a formula and its (syntactic) negation hold both fail to hold on a game?

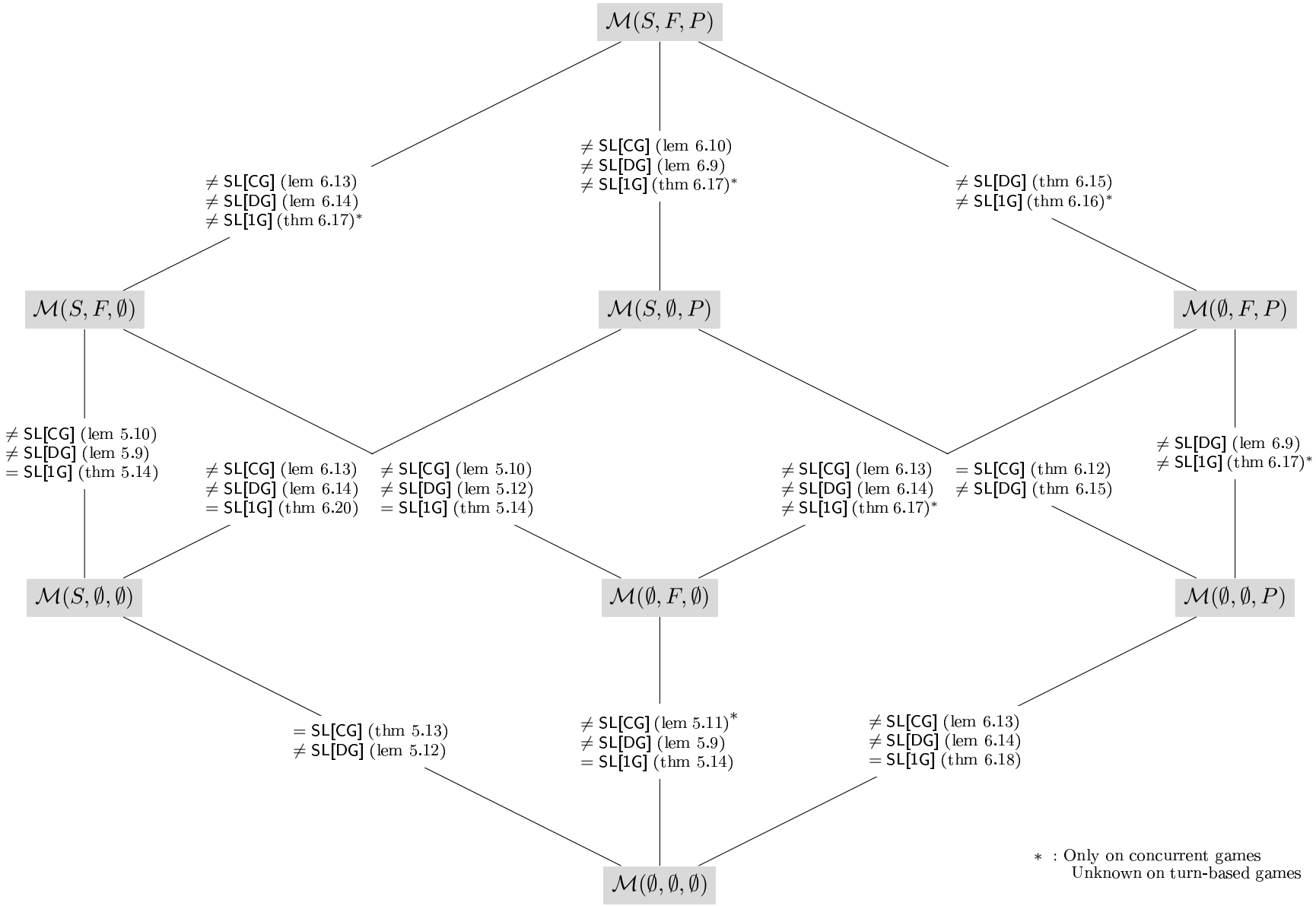


Figure 6.13: Inclusion graph of the different satisfaction relations

6.A Annex A

Lemma 6.4. *There exists a game $\mathcal{G}$, one of its states q_{ini} and a SL[BG] formula ϕ such that*

$$\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,P)} \phi \qquad \textit{and} \qquad \mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,P)} \neg\phi$$

Proof. First, we consider the turn-based game of Figure 6.14 and the state q_{ini} as starter. The game has four agents , , , and , and six atomic propositions $p_1, p_2, p_3, p_4, p_5, p_6$. Second, we define ϕ

$$\phi := \forall x_2.\ \forall x_4.\ \forall x_5.\ \exists x_1.\ \exists x_3.\ \exists x_6.\ \exists z_1.\ \exists y_1.\ \forall v_5.\ \exists v_6.\ \forall w_2.\ \exists w_3.\ \forall z_4.\ \phi'$$

$$\text{with } \phi' := \begin{cases} \psi_1 \Leftrightarrow \psi_5 \\ \vee \\ \psi_5 \Leftrightarrow \psi_6 \wedge \psi_2 \Leftrightarrow \psi_3 \wedge \begin{cases} \psi_1 \Leftrightarrow \psi_4 \\ \vee \\ \psi_3 \Leftrightarrow \psi_4 \end{cases} \end{cases}$$

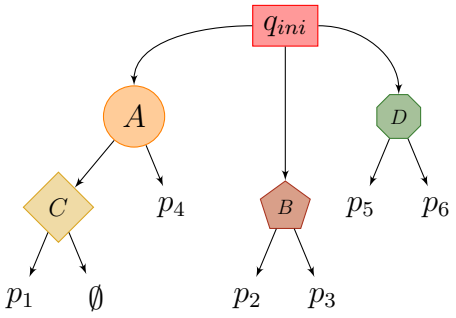


Figure 6.14: Game of Lemma 6.4.

and

$$\psi_1 := \mathsf{assign}(, x_1; , z_1; , y_1; , w_2; , v_5) F p_1$$
$$\psi_2 := \mathsf{assign}(, x_2; , z_1; , y_1; , w_2; , v_5) F p_2$$
$$\psi_3 := \mathsf{assign}(, x_3; , z_1; , y_1; , w_3; , v_5) F p_3$$
$$\psi_4 := \mathsf{assign}(, x_4; , z_4; , y_1; , w_2; , v_5) F p_4$$
$$\psi_5 := \mathsf{assign}(, x_5; , z_1; , y_1; , w_2; , v_5) F p_5$$
$$\psi_6 := \mathsf{assign}(, x_6; , z_1; , y_1; , w_2; , v_6) F p_6$$

A witness that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,P)} (Q_i x_i)_{i\leq l}\, \xi(\beta_j \varphi_j)_{j\leq n}$ can be derivated from Figure 6.15 and a witness that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,P)} (\overline{Q_i} x_i)_{i\leq l}\, \neg\xi(\beta_j \varphi_j)_{j\leq n}$ can be derivated from Figure 6.16. We present the variables following the order of the quantifications. An existentially quantified variable then depends on the variables universally quantified on the left (with an exception of $y_1(q_{ini}.A.C)$ in the first table which also depends on $z_4(q_{ini}.A)$). We have also made the different dependencies appear in red in both figures.

□

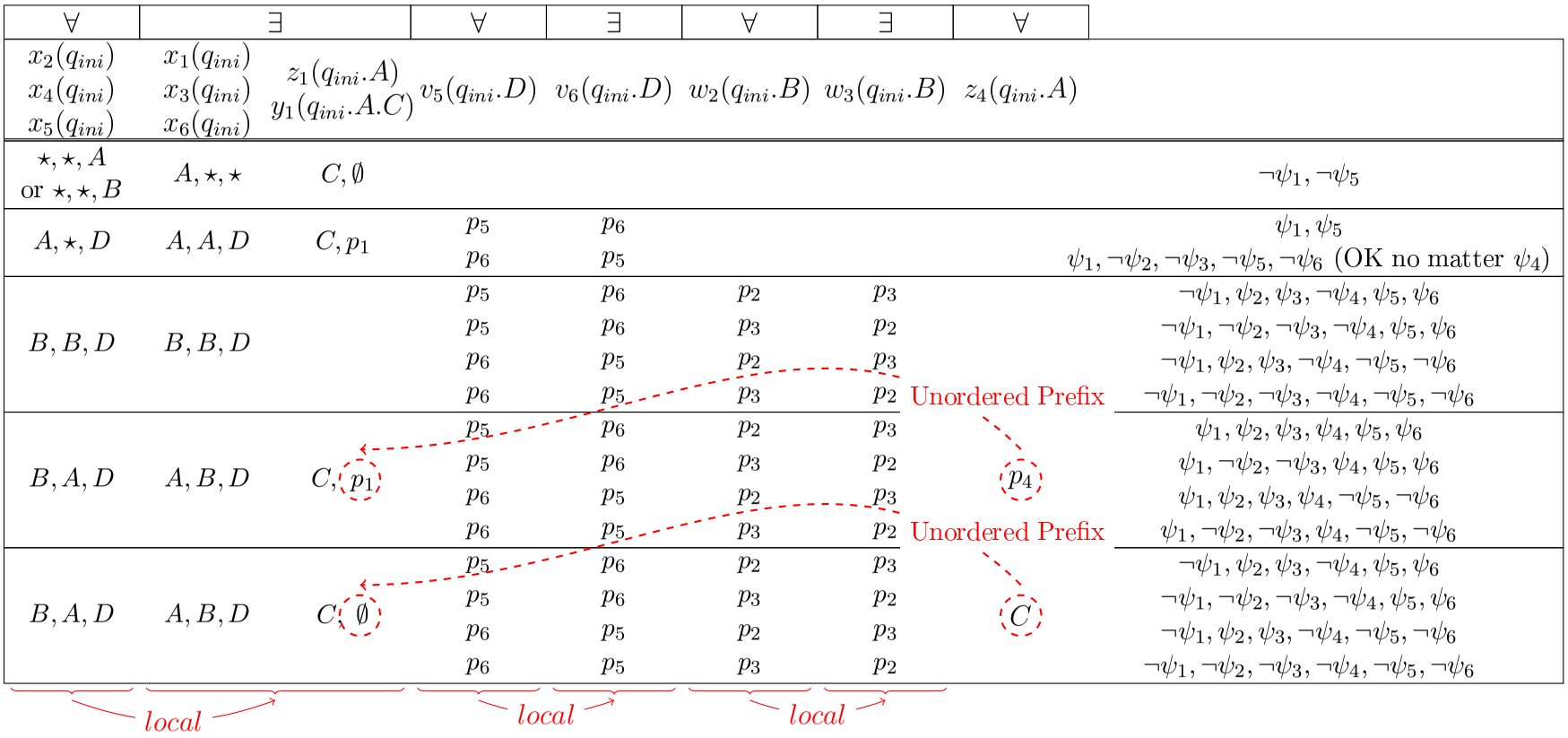


Figure 6.15: Interactions between strategies in Lemma 6.4 in order to prove $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,P)} (Q_i x_i)_{i \leq l}\ \xi(\beta_j \varphi_j)_{j \leq n}$.

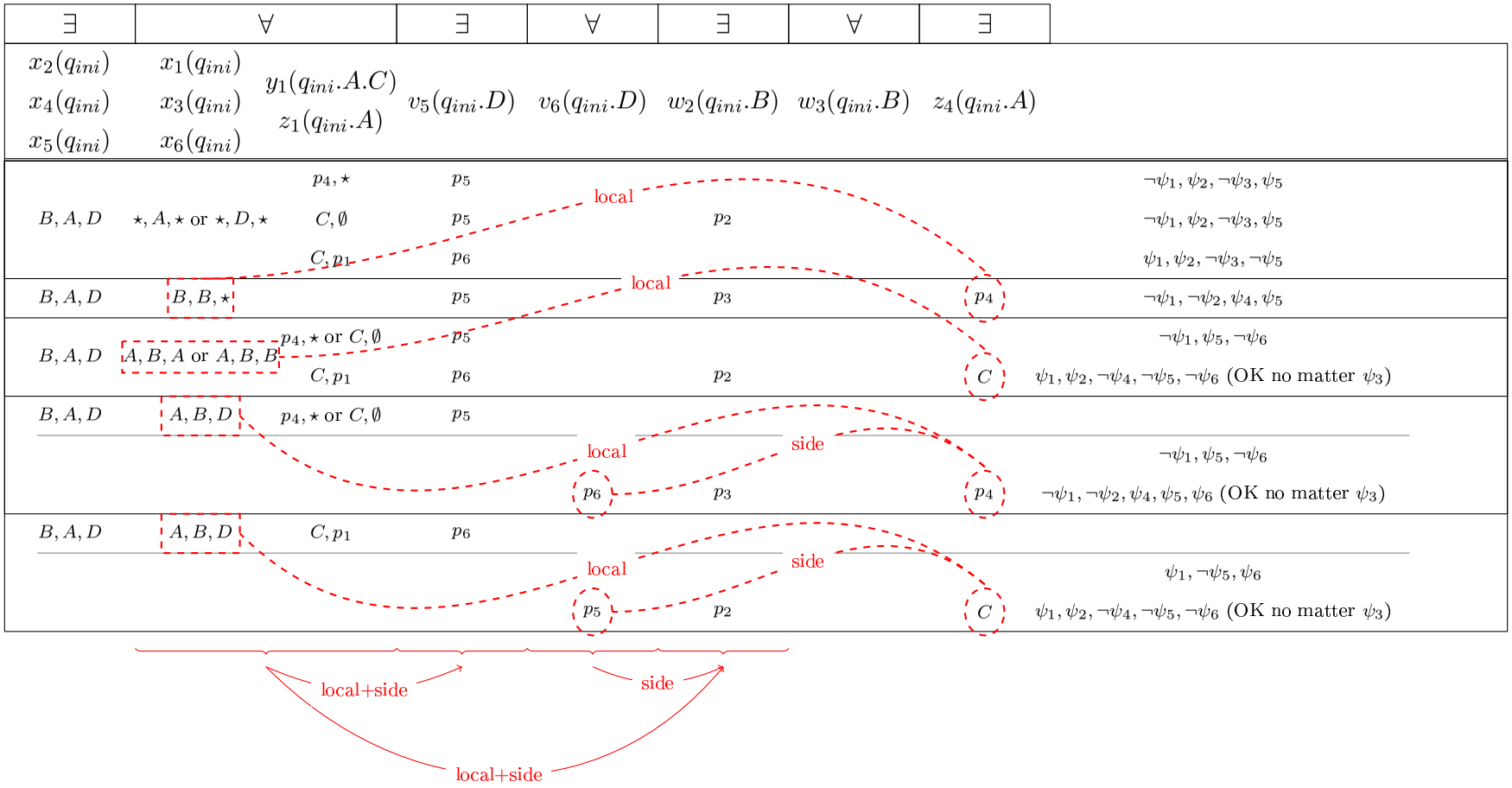


Figure 6.16: Interactions between strategies in Lemma 6.4 in order to prove $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,P)} (\overline{Q_i}x_i)_{i\leq l}\ \neg\xi(\beta_j\varphi_j)_{j\leq n}$.

6.B Annex B

Theorem 6.16. *There exist a concurrent game $\mathcal{G}$, one of its states q_{ini} and a SL[1G] formula ϕ such that $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,F,P)} \phi$ and $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F,P)} \phi$.*

Proof. The idea is very similar to the one used in the proof of Theorem 6.15. Consider once again the game $\mathcal{G}$ of Figure 6.9 that we recall on Figure 6.17 with 6 agents □, ①, ②, ◇, ① and ②. We recall that each agent can only influence the state represented in its name, for example ① and ② are the only agents having an influence on the state c. We define a SL[1G] formula

$$\phi := \exists x_1^{\bigcirc} . \forall x_2^{\bigcirc} . \exists x_1^{⬠} . \forall x_2^{⬠} . \exists x^{◇} . \exists x^{□} .$$
$$\mathsf{assign}(□, x^{□}; ①, x_1^{⬠}; ②, x_2^{⬠}; ◇, x^{◇}; ①, x_1^{\bigcirc}; ②, x_2^{\bigcirc}). \mathbf{F}\,(p_1 \vee p_2)$$

As we did in Theorem 6.15, we present the proof through two Tables: 6.2a and 6.2b. On the first one, we detail how to build a witness θ by specifying $\theta(w)$ in function of w; we give the dependencies on the right and the result at the bottom. On the second table, we do a case analysis to prove that $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,F,P)} \phi$; again the dependencies allowed appear on the right and the result, case by case, at the bottom.

□

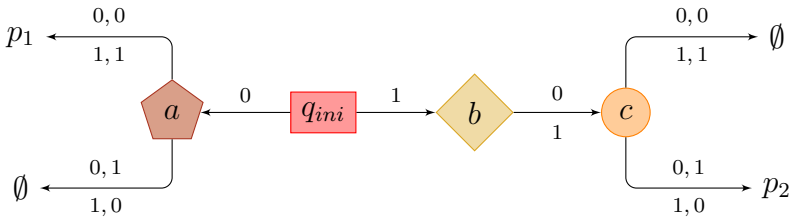


Figure 6.17: The game of Figure 6.9 used in Theorems 6.15 and 6.16.

Note that $x^{□}$ uses a future dependency, hence Theorem 6.16 cannot be extended to the case of $\models^{\mathcal{M}(S,\emptyset,P)}$ and $\models^{\mathcal{M}(\emptyset,\emptyset,P)}$. Theorem 6.12 even showed that $\models^{\mathcal{M}(S,\emptyset,P)}$ and $\models^{\mathcal{M}(\emptyset,\emptyset,P)}$ are equivalent.

w	x_2	0	0	1	1	
	x_2	0	1	0	1	
$\theta(w)$	x	0	1	0	1	Depends x_2 , x_2 and x $(x = x_2)$
	x	0	1	1	0	Depends x_2 and x_2 $(x = x_2 + x_2 \ mod[2])$
	x_1	0	1	0	1	Depends on x and x $(x_1 = x)$
	x_1	0	0	1	1	Depends on x and x_2 $(x_1 = x_2)$
result		$q_{ini}.a.p_1$ ok	$q_{ini}.b.c.p_2$ ok	$q_{ini}.b.c.p_2$ ok	$q_{ini}.a.p_1$ ok	

(a) Building the map θ to prove $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F,P)} \phi$.

		if $x_1 = x$				if $x_1 \neq x$				
		if $x_1 \neq x$		if $x_1 = x$		if $x_1 \neq x$		if $x_1 = x$		
w	x_2	1	1	1	1	0	0	0	0	
	x_2	0	0	1	1	0	0	1	1	
$\theta(w)$	x	1	0	1	0	1	0	1	0	Depends x_2 and x_2
	x	-	-	-	-	-	-	-	-	Depends x_2 and x
	x_1	1	0	1	0	0	1	0	1	Depends on x and x
	x_1	-	1	-	0	-	1	-	0	Depends on x
result		$q_{ini}.b.c.\emptyset$ no	$q_{ini}.a.\emptyset$ no	$q_{ini}.b.c.\emptyset$ no	$q_{ini}.a.\emptyset$ no	$q_{ini}.b.c.\emptyset$ no	$q_{ini}.a.\emptyset$ no	$q_{ini}.b.c.\emptyset$ no	$q_{ini}.a.\emptyset$ no	

(b) Proving $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,F,P)} \phi$. No matter the values fixed, we cannot find a working map.

Table 6.2: The tables for the proof of Theorem 6.16.

6.C Annex C

Theorem 6.17. *There exist a formula $\phi \in$ SL[1G] and a concurrent game structure $\mathcal{G}$ such that*

- $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,F,\emptyset)} \phi$ *and* $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,F,P)} \phi$
- $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$ *and* $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,F,P)} \phi$
- $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(S,F,\emptyset)} \phi$ *and* $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F,P)} \phi$
- $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(S,\emptyset,P)} \phi$ *and* $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,F,P)} \phi$

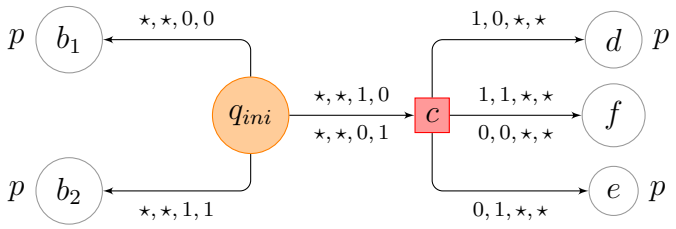


Figure 6.18: The game $\mathcal{G}$ used in Theorem 6.17's proof.

Proof. Consider the game of Figure 6.18 with four agents 1 , 2 , 1 and 2 , and two actions 0, 1. We define a SL[1G] formula ϕ

$$\phi := \exists x_1.\ \forall x_2.\ \exists x_3.\ \forall x_4.\ \mathsf{assign}(1, x_3; 2, x_4; 1, x_1; 2, x_2)\mathbf{F}\, p$$

Proving $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,F,P)} \phi$: To do so we specify a $\mathcal{M}(\emptyset, F, P)$ witness:

- To specify $x_3(q_{ini})$ we may assume knowledge of x_2 on $q_{ini}.c$ through the future dependencies. We then can set $x_3(q_{ini}) = 1 - x_2(q_{ini}.c)$.
- To specify $x_1(q_{ini}.c)$ we may assume knowledge of x_3/x_4 on q_{ini} through the unordered prefix dependencies. We simply set $x_1(q_{ini}.c) = 1 - x_4(q_{ini})$.

From this we can get a coherent $\mathcal{M}(\emptyset, F, P)$ map θ. It remains to check if it is an appropriate witness, for this we refer to Table 6.3.

$x_1(q_{ini}.c)$	$x_2(q_{ini}.c)$	$x_3(q_{ini})$	$x_4(q_{ini})$	Result
1	0	1	0	ok (d)
0	0	1	1	ok (b_2)
1	1	0	0	ok (b_1)
0	1	0	1	ok (e)

Table 6.3: Checking if θ is a witness of ϕ

Proving $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$: If we forbid future dependencies, $x_3(q_{ini})$ may not rely on any knowledge to make its choice. Assume that you have θ a $\mathcal{M}(\emptyset,\emptyset,P)$ map with $\theta(w)(x_3)(q_{ini}) = 1$ no matter w. θ has two possibilities:

- either $\theta(w)(x_1)(q_{ini}.c) = 1$ for w's with $w(x_4)(q_{ini}) = 0$. Then for w_0 with $w_0(x_4)(q_{ini}) = 0$ but also $w_0(x_2)(q_{ini}.c) = 1$, $\theta(w_0)$ will lead the goal to f and fail.
- or $\theta(w)(x_1)(q_{ini}.c) = 0$ for w's with $w(x_4)(q_{ini}) = 0$. Then taking w_0 with $w_0(x_4)(q_{ini}) = 0$ but also $w_0(x_2)(q_{ini}.c) = 0$ also leads $\theta(w_0)$ to f and fail.

In either cases, θ is not a witness. The case with $\theta(w)(x_3)(q_{ini}) = 0$ is similar. In the end, $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$

Proving $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,F,\emptyset)} \phi$: If we forbid unordered prefixes dependencies, $x_1(q_{ini}.c)$ may not relies on what has been played on q_{ini}. Consider a $\mathcal{M}(\emptyset,F,\emptyset)$ map θ. Either $\theta(w)(x_1)(q_{ini}.c) = 1$ or $\theta(w)(x_1)(q_{ini}.c) = 0$, no matter w. Take the first case, again we decompose by cases:

- either $\theta(w)(x_3)(q_{ini}) = 1$ for w's with $w(x_2)(q_{ini}.c) = 1$. Then for w_0 with $w_0(x_2)(q_{ini}.c) = 1$ and $w_0(x_4)(q_{ini}) = 0$, $\theta(w_0)$ leads to f
- or $\theta(w)(x_3)(q_{ini}) = 0$ for w's with $w(x_2)(q_{ini}.c) = 1$. Then for w_0 with $w_0(x_2)(q_{ini}.c) = 1$ and $w_0(x_4)(q_{ini}) = 1$, $\theta(w_0)$ also leads to f.

In both cases we need the unordered prefix dependencies to satisfy ϕ.

Notice that by construction of the formula and the game there cannot be side dependencies. So, from the three previous paragraphs we can easily deduce all the points of the theorem. □

Chapter 7

The SL[EG] fragment

In the previous chapter, we have seen that adding all the actions on the history (via unordered prefix dependencies) leads to unexpected consequences. In particular, one can override the order of quantification locally by using the side or future dependencies in conjunction with the actions made available; this can even be done with SL[1G] formulas. The previous chapter fell short: we wanted to suppress side dependencies by adding the actions played onto the histories [1], but in the end this cannot be done.

In this chapter we show a strong correlation between the ability to satisfy a formula ϕ without side or future dependencies and the possibility for both ϕ and $\neg\phi$ to hold on a game. More precisely we define a fragment SL[EG] of SL[BG] for which the following theorem holds. We also show that SL[EG] is maximal for Theorem 7.1.

Theorem 7.1. *Consider a formula $\phi \in$ SL[EG], a game $\mathcal{G}$, a state q_{ini} of $\mathcal{G}$ and two parameters $\spadesuit \in \{\emptyset, S\}$ and $\heartsuit \in \{\emptyset, F\}$. If $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit,P)} \phi$ and $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\spadesuit,\heartsuit,P)} \neg\phi$ then $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$.*

A more fashionable way to state Theorem 7.1 is to say that when a formula $\phi \in$ SL[EG] holds and its syntactic negation does not, we can remove both side and future dependencies. A naive approach would be to assume that if $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$, then $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} \neg\phi$ and therefore $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit,P)} \neg\phi$. However from $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$, we cannot deduce $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} \neg\phi$; indeed, we may be in one of those cases where neither ϕ nor $\neg\phi$ hold on $\mathcal{G}$ for $\models^{\mathcal{M}(\emptyset,\emptyset,P)}$. Hence Theorem 7.1 is not trivial.

Theorem 7.1 cannot be extended as an equivalence between the possibility for both a formula and its negation to hold and the capacity to remove side dependencies. There are cases where both ϕ and $\neg\phi$ hold on a game for $\models^{\mathcal{M}(S,\emptyset,P)}$ and where we can remove side dependencies from one of the two. Indeed, consider the game (that we refer to as $\mathcal{G}$ in the following) and formula (referred to as ϕ) in the proof of Theorem 6.15 (page 146). We have $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,P)} \phi$ (as proven in the proof) and $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} \neg\phi$ (this can easily be checked).

[1] Formally we use unordered prefix dependencies but as explained during Chapter 6, the framework is equivalent to adding all the actions played before on the histories.

7.1 Semi-stable sets and SL[EG]

Before defining our logic, we first fix some notations: for $n \in \mathbb{N}$, we let $\{0,1\}^n$ be the set of mappings from $[1,n]$ to $\{0,1\}$. We write $\mathbf{0}^n$ (or $\mathbf{0}$ if the size n is clear) for the function that maps all integers in $[1,n]$ to 0, and $\mathbf{1}^n$ (or $\mathbf{1}$) for the function that maps $[1,n]$ to 1. For $f \in \{0,1\}^n$ and $k \leq n$, $f_{[1,k]}$ is the restriction of f to $[1,k]$. The size of $f \in \{0,1\}^n$ is defined as $|f| = \sum_{1 \leq i \leq n} f(i)$. For two elements f and g of $\{0,1\}^n$, we write $f \leq g$ whenever $f(i) = 1$ implies $g(i) = 1$ for all $i \in [1,n]$. Given $B^n \subseteq \{0,1\}^n$, we write $\uparrow B^n = \{g \in \{0,1\}^n \mid \exists f \in B^n.\ f \leq g\}$. A set $F^n \subseteq \{0,1\}^n$ is *upward closed* if $F^n = \uparrow F^n$. Finally, we also define the following operators:

$$\overline{f}\colon i \mapsto 1 - f(i) \qquad f \curlywedge g\colon i \mapsto \min\{f(i), g(i)\} \qquad f \curlyvee g\colon i \mapsto \max\{f(i), g(i)\}.$$

As a way to to get the reader accustomed to these three operators, we start with a simple lemma.

Lemma 7.2. *For any $f, g, s \in \{0,1\}^n$, $\overline{(f \curlywedge s) \vee (g \curlywedge \overline{s})} = (\overline{f} \curlywedge s) \vee (\overline{g} \curlywedge \overline{s})$.*

Proof. Consider some $1 \leq i \leq n$, then

$$\begin{aligned}
\overline{(f \curlywedge s) \vee (g \curlywedge \overline{s})}(i) = 1 &\Leftrightarrow (f \curlywedge s) \vee (g \curlywedge \overline{s})(i) = 1 \\
&\Leftrightarrow s(i) = 1 \Rightarrow f(i) = 0 \text{ and } s(i) = 0 \Rightarrow g(i) = 0 \\
&\Leftrightarrow s(i) = 1 \Rightarrow \overline{f}(i) = 1 \text{ and } s(i) = 0 \Rightarrow \overline{g}(i) = 1 \\
&\Leftrightarrow (\overline{f} \curlywedge s) \vee (\overline{g} \curlywedge \overline{s})(i) = 1
\end{aligned}$$

□

In order to define SL[EG], we introduce the notion of semi-stable subset of $\{0,1\}^n$:

Definition 7.3. *A set $F^n \subseteq \{0,1\}^n$ is* semi-stable *if for any f and g in F^n, it holds that*

$$\forall s \in \{0,1\}^n \qquad (f \curlywedge s) \curlyvee (g \curlywedge \overline{s}) \in F^n \text{ or } (g \curlywedge s) \curlyvee (f \curlywedge \overline{s}) \in F^n.$$

Example 7.4. *Obviously, $\{0,1\}^n$ is semi-stable, as well as the empty set. For $n = 2$, the set $\{\langle 0,1\rangle, \langle 1,0\rangle\}$ is easily seen not to be semi-stable: taking $f = \langle 0,1\rangle$ and $g = \langle 1,0\rangle$ with $s = \langle 1,0\rangle$, we get $(f \curlywedge s) \curlyvee (g \curlywedge \overline{s}) = \langle 0,0\rangle$ and $(g \curlywedge s) \curlyvee (f \curlywedge \overline{s}) = \langle 1,1\rangle$. Similarly, $\{\langle 0,0\rangle, \langle 1,1\rangle\}$ is not semi-stable. It can be checked that any other subset of $\{0,1\}^2$ is semi-stable.*

We now define SL[EG]. As for SL[BG] and its fragments, we focus on its *flat* fragment, which is defined as follows:

$$\begin{aligned}
\mathsf{SL[EG]} \ni \phi &::= \forall x.\phi \mid \exists x.\phi \mid \xi \\
\xi &::= F^n((\beta_i)_{1 \leq i \leq n}) \\
\beta &::= \mathsf{assign}(A, x).\ \beta \mid \varphi \\
\varphi &::= \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi \mid p
\end{aligned}$$

where p ranges over AP, n ranges over $\mathbb{N}$ and for each n, F^n ranges over a set $\mathcal{F}^n$ of semi-stable subsets of $\{0,1\}^n$. The semantics of the F^n operator is defined as

$$\mathcal{G}, q \models_\chi F^n((\beta_i)_{i\leq n}) \quad \Leftrightarrow \quad \exists f \in F^n \text{ where } f(i) = 1 \text{ iff } \mathcal{G}, q \models_\chi \beta_i$$

Notice that if F^n would range over all subsets of $\{0,1\}^n$, then this definition would exactly correspond to SL[BG]. Similarly, the case where $F^n = \{\mathbf{1}^n\}$ corresponds to SL[CG], while $F^n = \{0,1\}^n \setminus \{\mathbf{0}^n\}$ gives rise to SL[DG]. Both $\{\mathbf{1}^n\}$ and $\{0,1\}^n \setminus \{\mathbf{0}^n\}$ are semi-stable, hence SL[CG] $\subseteq$ SL[EG] and SL[DG] $\subseteq$ SL[EG].

Example 7.5. *Consider the formula of Example 1.5.4 expressing the existence of a Nash equilibrium. For two agents, it can be written as*

$$\exists x_1.\exists x_2.\forall y_1.\forall y_2 \begin{cases} (\mathsf{assign}(A_1, y_1; A_2, x_2).\varphi_1) \Rightarrow (\mathsf{assign}(A_1, x_1; A_2, x_2).\varphi_1) \\ \wedge \\ (\mathsf{assign}(A_1, x_1; A_2, y_2).\varphi_2) \Rightarrow (\mathsf{assign}(A_1, x_1; A_2, x_2).\varphi_2) \end{cases} \tag{7.1}$$

This formula has four goals, and it corresponds to the set

$$F^4 = \{\langle 1,1,1,1\rangle, \langle 0,1,1,1\rangle, \langle 1,1,0,1\rangle, \langle 0,1,0,1\rangle, \langle 0,0,1,1\rangle, \langle 1,1,0,0\rangle, \\ \langle 0,0,0,1\rangle, \langle 0,1,0,0\rangle, \langle 0,0,0,0\rangle\}$$

Taking $f = \langle 1,1,0,0\rangle$ *and* $g = \langle 0,0,1,1\rangle$, *with* $s = \langle 1,0,1,0\rangle$ *we have* $(f \curlywedge s) \curlyvee (g \curlywedge \overline{s}) = \langle 1,0,0,1\rangle$ *and* $(g \curlywedge s) \curlyvee (f \curlywedge \overline{s}) = \langle 0,1,1,0\rangle$, *which are not in* F^4. *Hence Formula* (7.1) *is not syntactically in SL[EG]. We conjecture that the existence of Nash equilibria cannot be expressed in SL[EG].*

7.2 Expressiveness of SL[EG]

We now investigate the relative expressiveness of SL[EG] w.r.t the (flat) fragment SL[AG] of SL[BG], defined in Section 5.3. To this aim, pick $\phi \in$ SL[BG] with n goals, and write $\phi = \wp.\ \xi(\psi_i)_{1\leq i\leq n}$, where $\wp$ is the quantification part, and ξ is a boolean combination of the goals $(\psi_i)_{1\leq i\leq n}$. We define

$$F^n_\xi = \{f \in \{0,1\}^n \mid \xi(f) \text{ evaluates to true}\}.$$

Proposition 7.6. *For any formula* $\phi \in$ *SL[AG] with* n *goals,* F^n_ξ *is semi-stable.*

Proof. Take $\phi \in$ SL[AG] with n goals. By definition of SL[AG], the boolean formula ξ of ϕ may be written in one of the following two ways:

$$\xi(x_i)_{1\leq i\leq n} = \xi'((x_i)_{1\leq i\leq n-1}) \wedge x_n$$
$$\xi(x_i)_{1\leq i\leq n} = \xi'((x_i)_{1\leq i\leq n-1}) \vee x_n$$

As a consequence, we are in one of the following two cases:

$$F^n_\xi = \{f \mid f(n) = 1 \text{ and } f_{[1,n-1]} \in F^{n-1}_{\xi'}\} \tag{7.2}$$
$$F^n_\xi = \{f \mid f(n) = 1\} \cup \{g \mid g(n) = 0 \text{ and } g_{[1,n-1]} \in F^{n-1}_{\xi'}\} \tag{7.3}$$

By induction on n, we prove that such sets are semi-stable. The base case, for $n = 1$, is trivial. Now, assume that the result holds up to step $n-1$, and pick a formula $\phi \in \mathsf{SL[AG]}$ with n goals (for $n \geq 2$). We first consider the case where $\xi = \xi'((x_i)_{1\leq i\leq n-1}) \wedge x_n$. Then $F^n_\xi = \{f \mid f(n) = 1 \text{ and } f_{[1,n]} = F^{n-1}_{\xi'}\}$, and by induction hypothesis $F^{n-1}_{\xi'}$ is semi-stable. Pick any two elements f and g in F^n_ξ, and any $s \in \{0,1\}^n$. Let $s' = s_{[1,n-1]} \in \{0,1\}^{n-1}$, $f' = f_{[1,n-1]}$ and $g' = g_{[1,n-1]}$. Since $f(n) = g(n) = 1$, we have $[(f \curlywedge s) \curlyvee (g \curlywedge \overline{s})](1) = [(g \curlywedge s) \curlyvee (f \curlywedge \overline{s})](1) = 1$. Moreover, $[(f \curlywedge s) \curlyvee (g \curlywedge \overline{s})]_{[1,n-1]} = (f' \curlywedge s') \curlyvee (g' \curlywedge \overline{s'})$, and $[(g \curlywedge s) \curlyvee (f \curlywedge \overline{s})]_{[1,n-1]} = (g' \curlywedge s') \curlyvee (f' \curlywedge \overline{s'})$. Since $F^{n-1}_{\xi'}$ is semi-stable, it contains one of these two elements, so that one of $(f \curlywedge s) \curlyvee (g \curlywedge \overline{s})$ and $(g \curlywedge s) \curlyvee (f \curlywedge \overline{s})$ is in F^n_ξ.

The case where $\xi = \xi'((x_i)_{1\leq i\leq n-1}) \vee x_n$ relies on similar arguments: assuming that $F^{n-1}_{\xi'}$ is semi-stable, we pick two elements f and g in F^n_ξ, and $s \in \{0,1\}^n$. In case $f(n) = 1$ or $g(n) = 1$, then one of $(f \curlywedge s) \curlyvee (g \curlywedge \overline{s})$ and $(g \curlywedge s) \curlyvee (f \curlywedge \overline{s})$ takes value 1 in n, and thus belongs to F^n_ξ. Otherwise, the argument is similar to the case of conjunctive formulas. □

It follows that any formula in SL[AG] can be written as a formula in SL[EG]. One can wonder if the converse translation is possible, which would mean that SL[EG] and SL[AG] would have the same expressive power. The answer is negative:

Proposition 7.7. *Fix $n = 3$ and let $H^n = \{\langle 1,1,1\rangle, \langle 1,1,0\rangle, \langle 1,0,1\rangle, \langle 0,1,1\rangle\}$. Then H^n is semi-stable, and for any formula $\phi = \wp.\ \xi(\psi_i)_{1\leq i\leq n}$ in SL[AG], we have $F^n_\xi \neq H^n$.*

Proof. That H^n is semi-stable is easily obtained by brute force. Now, for any formula $\phi = \wp\xi$ in SL[AG], $\xi(x_1, x_2, x_3)$ must be in one of the following four forms:

$$\xi'(x_1, x_2) \wedge x_3 \qquad \xi'(x_1, x_2) \wedge \neg x_3$$
$$\xi'(x_1, x_2) \vee x_3 \qquad \xi'(x_1, x_2) \vee \neg x_3$$

Again, it is easily checked that none of these cases can give rise to exactly H^n: for instance, in the first case, $\langle 1,1,0\rangle$ would not belong to F^n_ξ, while the last case would allow $\langle 0,0,0\rangle$ to be in H^n. □

7.3 Properties of SL[EG]

We exhibit some properties of semi-stable sets of valuations. In particular, we show that semi-stable sets can be rearranged by flipping bits into an upward-closed set. We fix an index n for the rest of this section.

7.3.1 Closure under bit flipping

Fix a vector $b \in \{0,1\}^n$. We define the operation $\mathsf{flip}_b \colon \{0,1\}^n \to \{0,1\}^n$ that maps any vector f to $(f \curlywedge b) \curlyvee (\overline{f} \curlywedge \overline{b})$. In other terms, flip_b flips the i-th bit of its argument if $b_i = 0$, and keeps this bit unchanged if $b_i = 1$. Notice that flip_b is a permutation of $\{0,1\}^n$. Notice also that $\mathsf{flip}_{\mathbf{0}}(f) = \overline{f}$ and $\mathsf{flip}_f(f) = \mathbf{1}$ for all $f \in \{0,1\}^n$.

The following lemma shows that flipping bits preserves semi-stability. This is a natural property for our logic, since flipping bits corresponds to negating goals. More precisely, for $b \in \{0,1\}^n$, open formulas $F^n((\beta_i.\ \varphi_i)_{1 \le i \le n})$ and $\mathsf{flip}_b(F^n)((\beta_i.\ \varphi'_i)_{1 \le i \le n})$, where $\varphi'_i = \varphi_i$ if $b(i) = 1$ and $\varphi'_i = \neg\varphi_i$ if $b(i) = 0$, are equivalent.

Lemma 7.8. *If $F^n \subseteq \{0,1\}^n$ is semi-stable, then so is $\mathsf{flip}_b(F^n)$.*

Proof. We assume that F^n is semi-stable. Take $f' = \mathsf{flip}_b(f)$ and $g' = \mathsf{flip}_b(g)$ in $\mathsf{flip}_b(F^n)$, and $s \in \{0,1\}^n$. Then

$$\begin{aligned}(f' \curlywedge s) \curlyvee (g' \curlywedge \overline{s}) &= \big(((f \curlywedge b) \curlyvee (\overline{f} \curlywedge \overline{b})) \curlywedge s\big) \curlyvee \big(((g \curlywedge b) \curlyvee (\overline{g} \curlywedge \overline{b})) \curlywedge \overline{s}\big) \\ &= \big(((f \curlywedge s) \curlyvee (g \curlywedge \overline{s})) \curlywedge b\big) \curlyvee \big(((\overline{f} \curlywedge s) \curlyvee (\overline{g} \curlywedge \overline{s})) \curlywedge \overline{b}\big)\end{aligned}$$

Write $\alpha = (f \curlywedge s) \curlyvee (g \curlywedge \overline{s})$ and $\beta = (\overline{f} \curlywedge s) \curlyvee (\overline{g} \curlywedge \overline{s})$. By Lemma 7.2, $\beta = \overline{\alpha}$. We then have

$$\begin{aligned}(f' \curlywedge s) \curlyvee (g' \curlywedge \overline{s}) &= (\alpha \curlywedge b) \curlyvee \big(\overline{\alpha} \curlywedge \overline{b}\big) \\ &= \mathsf{flip}_b(\alpha). \end{aligned} \tag{7.4}$$

This computation being valid for any f and g, we also have

$$\begin{aligned}(g' \curlywedge s) \curlyvee (f' \curlywedge \overline{s}) &= (\gamma \curlywedge b) \curlyvee \big(\overline{\gamma} \curlywedge \overline{b}\big) \\ &= \mathsf{flip}_b(\gamma) \end{aligned} \tag{7.5}$$

with $\gamma = (g \curlywedge s) \curlyvee (f \curlywedge \overline{s})$. By hypothesis, at least one of α and γ belongs to F^n, so that also at least one of $(f' \curlywedge s) \curlyvee (g' \curlywedge \overline{s})$ and $(g' \curlywedge s) \curlyvee (f' \curlywedge \overline{s})$ belongs to $\mathsf{flip}_b(F^n)$. □

Corollary 7.9. *F^n is semi-stable if, and only if, its complement is.*

Proof. Assume F^n is not semi-stable, and pick f and g in F^n and $s \in \{0,1\}^n$ such that none of $\alpha = (f \curlywedge s) \curlyvee (g \curlywedge \overline{s})$ and $\gamma = (g \curlywedge s) \curlyvee (f \curlywedge \overline{s})$ are in F^n. It cannot be the case that $g = f$, as this would imply $\alpha = f \in F^n$. Hence $\alpha \neq \gamma$. We claim that α and γ are our witnesses for showing that the complement of F^n is not semi-stable: both of them belong to the complement of F^n, and $(\alpha \curlywedge s) \curlyvee (\gamma \curlywedge \overline{s})$ can be seen to equal f, hence it is not in the complement of F^n. Similarly for $(\gamma \curlywedge s) \curlyvee (\alpha \curlywedge \overline{s}) = g$. □

7.3.2 Transformation into upward-closed set

Lemma 7.10. *If $F^n \subseteq \{0,1\}^n$ is semi-stable, then for any $s \in \{0,1\}^n$ and any non-empty subset H^n of F^n, it holds that*

$$\exists f \in H^n.\ \forall g \in H^n.\ (f \curlywedge s) \curlyvee (g \curlywedge \overline{s}) \in F^n.$$

Proof. For a contradiction, assume that there exist $s \in \{0,1\}^n$ and $H^n \subseteq F^n$ such that, for any $f \in H^n$, there is an element $g \in H^n$ for which $(f \curlywedge s) \curlyvee (g \curlywedge \overline{s}) \notin F^n$. Then there must exist a minimal integer $2 \leq \lambda \leq |H^n|$ and λ elements $\{f_i \mid 1 \leq i \leq \lambda\}$ of H^n such that

$$\forall 1 \leq i \leq \lambda - 1 \ (f_i \curlywedge s) \curlyvee (f_{i+1} \curlywedge \overline{s}) \notin F^n \text{ and } (f_\lambda \curlywedge s) \curlyvee (f_1 \curlywedge \overline{s}) \notin F^n.$$

By Corollary 7.9, the complement of F^n is semi-stable. Hence, considering $(f_{\lambda-1} \curlywedge s) \curlyvee (f_\lambda \curlywedge \overline{s})$ and $(f_\lambda \curlywedge s) \curlyvee (f_1 \curlywedge \overline{s})$, one of the following two vectors is not in F^n:

$$\big([(f_{\lambda-1} \curlywedge s) \curlyvee (f_\lambda \curlywedge \overline{s})] \curlywedge s\big) \curlyvee \big([(f_\lambda \curlywedge s) \curlyvee (f_1 \curlywedge \overline{s})] \curlywedge \overline{s}\big)$$
$$\big([(f_\lambda \curlywedge s) \curlyvee (f_1 \curlywedge \overline{s})] \curlywedge s\big) \curlyvee \big([(f_{\lambda-1} \curlywedge s) \curlyvee (f_\lambda \curlywedge \overline{s})] \curlywedge \overline{s}\big)$$

The second expression equals f_λ, which is in F^n. Hence we get that $(f_{\lambda-1} \curlywedge s) \curlyvee (f_1 \curlywedge \overline{s})$ is not in F^n, contradicting minimality of λ. □

Lemma 7.11. *For any semi-stable set F^n, there exists $B \in \{0,1\}^n$ such that $\mathsf{flip}_B(F^n)$ is upward closed.*

Proof. The lemma trivially holds for $F^n = \emptyset$ thus, in the following, we assume F^n to be non-empty. For $1 \leq i \leq n$, let $s_i \in \{0,1\}^n$ be the vector such that $s_i(j) = 1$ if, and only if, $j = i$. Applying Lemma 7.10, we get that for any i, there exists some $f_i \in F^n$ such that for any $f \in F^n$, it holds

$$(f_i \curlywedge s_i) \curlyvee (f \curlywedge \overline{s}_i) \in F^n. \tag{7.6}$$

We fix such a family $(f_i)_{i \leq n}$ then define $g \in \{0,1\}^n$ as $g = \bigvee_{1 \leq i \leq n} (f_i \curlywedge s_i)$, i.e. $g(i) = f_i(i)$ for all $1 \leq i \leq n$. Starting from any element of F^n and applying Equation (7.6) iteratively for each i, we get that $g \in F^n$. Since $g \curlywedge s_i = f_i \curlywedge s_i$, we also have

$$\forall f \in F^n \qquad (g \curlywedge s_i) \curlyvee (f \curlywedge \overline{s}_i) \in F^n$$

By Equation (7.5), since $\mathsf{flip}_g(g) = \mathbf{1}$, we get

$$\forall f \in F^n \qquad (\mathbf{1} \curlywedge s_i) \curlyvee (\mathsf{flip}_g(f) \curlywedge \overline{s}_i) \in \mathsf{flip}_g(F^n). \tag{7.7}$$

Now, assume that $\mathsf{flip}_g(F^n)$ is not upward closed: then there exist elements $f \in F^n$ and $h \notin F^n$ such that $\mathsf{flip}_g(f)(i) = 1 \Rightarrow \mathsf{flip}_g(h)(i) = 1$ for all i. Starting from f and iteratively applying Equation (7.7) for those i for which $\mathsf{flip}_g(h)(i) = 1$ and $\mathsf{flip}_g(f)(i) = 0$, we get that $\mathsf{flip}_g(h) \in \mathsf{flip}_g(F^n)$ and $h \in F^n$. Hence $\mathsf{flip}_g(F^n)$ must be upward closed. □

Remark 7.12. *Notice that being upward closed is not a sufficient condition for being semi-stable. Consider for instance the set $F^n = \uparrow\{\langle 0,0,1,1\rangle; \langle 1,1,0,0\rangle\}$. Then F^n is not semi-stable: taking $f = \langle 0,0,1,1\rangle$ and $g = \langle 1,1,0,0\rangle$, and $s = \{1,0,0,1\}$, we get $(f \curlywedge s) \curlyvee (g \curlywedge \overline{s}) = \langle 0,1,0,1\rangle \notin F^n$ and $(g \curlywedge s) \curlyvee (f \curlywedge \overline{s}) = \langle 1,0,1,0\rangle \notin F^n$.*

7.3.3 Ordering $\{0,1\}^n$

Consider $F^n \subseteq \{0,1\}^n$ and $s \in \{0,1\}^n$, for any $h \in \{0,1\}^n$, we define

$$\mathbb{F}^n(h,s) := \{h' \in \{0,1\}^n \mid (h \curlywedge s) \curlyvee (h' \curlywedge \overline{s}) \in F^n\}$$

and, writing $\overline{F^n}$ for the complement of F^n,

$$\overline{\mathbb{F}^n}(h,s) := \{h' \in \{0,1\}^n \mid (h \curlywedge s) \curlyvee (h' \curlywedge \overline{s}) \in \overline{F^n}\}$$

Trivially $\mathbb{F}^n(h,s) \bigcap \overline{\mathbb{F}^n}(h,s) = \emptyset$ and $\mathbb{F}^n(h,s) \bigcup \overline{\mathbb{F}^n}(h,s) = \{0,1\}^n$. If F^n is a semi-stable set, then the family $(\mathbb{F}^n(h,s))_{h\in\{0,1\}^n}$ obeys certain properties:

Lemma 7.13. *Fix a semi-stable set F^n and $s \in \{0,1\}^n$. For any $h_1, h_2 \in \{0,1\}^n$, either $\mathbb{F}^n(h_1,s) \subseteq \mathbb{F}^n(h_2,s)$ or $\mathbb{F}^n(h_2,s) \subseteq \mathbb{F}^n(h_1,s)$.*

Proof. Assume that the two relations do not hold: there are $h'_1 \in \mathbb{F}^n(h_1,s) \backslash \mathbb{F}^n(h_2,s)$ and $h'_2 \in \mathbb{F}^n(h_2,s) \backslash \mathbb{F}^n(h_1,s)$. We then have:

$$\begin{array}{ll} (h_1 \curlywedge s) \curlyvee (h'_1 \curlywedge \overline{s}) \in F^n & (h_2 \curlywedge s) \curlyvee (h'_1 \curlywedge \overline{s}) \notin F^n \\ (h_2 \curlywedge s) \curlyvee (h'_2 \curlywedge \overline{s}) \in F^n & (h_1 \curlywedge s) \curlyvee (h_2 \curlywedge \overline{s}) \notin F^n \end{array}$$

Now consider $(h_1 \curlywedge s) \curlyvee (h'_1 \curlywedge \overline{s})$, $(h_2 \curlywedge s) \curlyvee (h'_2 \curlywedge \overline{s})$ and s. As F^n is semi-stable, one of the two following vectors is in F^n :

$$\begin{array}{l} \big((h_1 \curlywedge s) \curlyvee (h'_1 \curlywedge \overline{s}) \curlywedge s\big) \curlyvee \big((h_2 \curlywedge s) \curlyvee (h'_2 \curlywedge \overline{s}) \curlywedge \overline{s}\big) \\ \big((h_2 \curlywedge s) \curlyvee (h'_2 \curlywedge \overline{s}) \curlywedge s\big) \curlyvee \big((h_1 \curlywedge s) \curlyvee (h'_1 \curlywedge \overline{s}) \curlywedge \overline{s}\big) \end{array}$$

The first vector is equal to $(h_1 \curlywedge s) \curlyvee (h'_2 \curlywedge \overline{s})$ and the second to $(h_2 \curlywedge s) \curlyvee (h'_1 \curlywedge \overline{s})$ and both are supposed to be in $\overline{F^n}$, we get a contradiction. □

Given a semi-stable set F^n and $s \in \{0,1\}^n$, we can use the inclusion relation of Lemma 7.13 to defines a quasi-order relation $\preceq_s^{F^n}$ over the elements of $\{0,1\}^n$.

Definition 7.14. *Fix F^n semi-stable and $s \in \{0,1\}^n$. We define $\preceq_s^{F^n} \subseteq \{0,1\}^n \times \{0,1\}^n$ so that $h_1 \preceq_s^{F^n} h_2$ iff $\mathbb{F}^n(h_1,s) \subseteq \mathbb{F}^n(h_2,s)$. In particular, $h_1 \preceq_{\mathbf{1}}^{F^n} h_2$ whenever either $h_1 \in \overline{F^n}$ or $h_2 \in F^n$.*

To ease the reading, whenever F^n is clear from context, we write $\preceq_s$ instead of $\preceq_s^{F^n}$. Reflexiveness and transitivity of $\preceq_s$ follows from the reflexiveness and transitivity of the inclusion relation $\subseteq$. We use the usual notations $\prec_s, \succeq_s, \succ_s$ and $=_s$ respectively for the strict, the inverse, the strict of the inverse quasi orders and the equality up to quasi-order, all derived from $\preceq_s$. Intuitively, $\preceq_s$ orders the elements of $\{0,1\}^n$ based on how "easy" it is to complete their restriction to s so that the completion belongs to F^n. Figure 7.1 shows an application while Figure 7.2 gives an illustration of two orders.

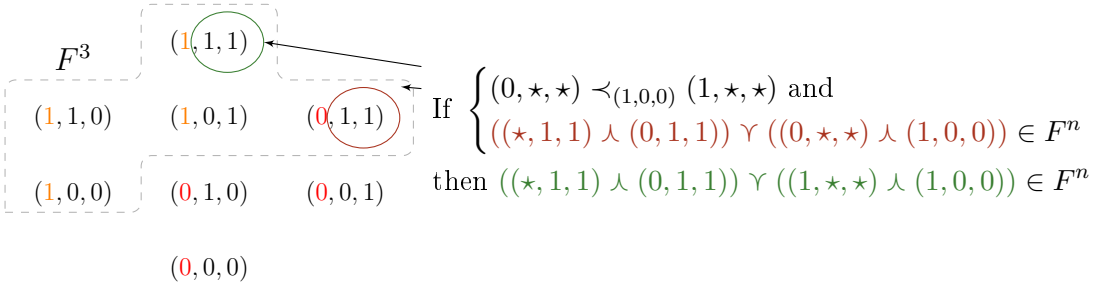


Figure 7.1: Link between the $F^n(h,s)$'s sets and the orders.

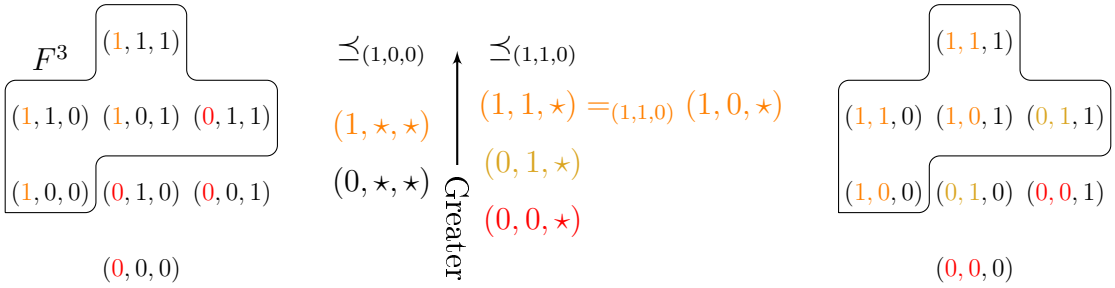


Figure 7.2: An illustration with $F^3 := \{\langle 1,1,1\rangle; \langle 1,1,0\rangle; \langle 1,0,1\rangle; \langle 0,1,1\rangle; \langle 1,0,0\rangle\}$ of the orders $\preceq_{(1,0,0)}$ and $\preceq_{(1,1,0)}$.

Remark 7.15. *To avoid new notations, we defined $\preceq_s$ over $\{0,1\}^n \times \{0,1\}^n$ but only the indexes on which s takes value 1 are of interest: given $h_1, h_2 \in \{0,1\}^n$ such that $(f_1 \curlywedge s) = (f_2 \curlywedge s)$, we have $\mathbb{F}(f_1,s) = \mathbb{F}(f_2,s)$ and $f_1 =_s f_2$. The fact that all orders belong to the same product set $\{0,1\}^n \times \{0,1\}^n$ allows us to easily compare elements of $\{0,1\}^n$ on multiple orders. The other option would have been to work with order over $\{0,1\}^{|s|} \times \{0,1\}^{|s|}$; we would then have need some new operations to navigate between the sets $\{0,1\}^{|s|}$ and $\{0,1\}^n$.*

Lemma 7.16. *Given a semi-stable set F^n, $s_1, s_2 \in \{0,1\}^n$ such that $s_1 \curlywedge s_2 = \mathbf{0}$ and $f, g \in \{0,1\}^n$ such that $f \preceq_{s_1} g$ and $f \preceq_{s_2} g$. Then $f \preceq_{s_1 \curlyvee s_2} g$.*

Proof. Because $f \preceq_{s_1} g$ and $f \preceq_{s_2} g$, we have

$$\forall i \in \{1,2\}\ \forall h \in \{0,1\}^n \qquad (f \curlywedge s_i) \curlyvee (h \curlywedge \overline{s_i}) \in F^n \Rightarrow (g \curlywedge s_i) \curlyvee (h \curlywedge \overline{s_i}) \in F^n \quad (7.8)$$

Consider $h' \in \{0,1\}^n$ such that $\alpha := (f \curlywedge (s_1 \curlyvee s_2)) \curlyvee (h' \curlywedge \overline{(s_1 \curlyvee s_2)})$ is in F^n. Define the element $h := \alpha \curlywedge \overline{s_2}$, then $(f \curlywedge s_2) \curlyvee (h \curlywedge \overline{s_2}) = (f \curlywedge (s_1 \curlyvee s_2)) \curlyvee (h' \curlywedge \overline{(s_1 \curlyvee s_2)}) \in F^n$. Using (7.8) with s_2 and h, we get $\beta := (g \curlywedge s_2) \curlyvee (h \curlywedge \overline{s_2})$. As $s_1 \curlywedge s_2 = \mathbf{0}$, we can write $\beta = (f \curlywedge s_1) \curlyvee (g \curlywedge s_2) \curlyvee (h' \curlywedge \overline{(s_1 \curlyvee s_2)}) \in F^n$.

Now consider $h = \beta \curlywedge \overline{s_1}$, we have $(f \curlywedge s_1) \curlyvee (h \curlywedge \overline{s_1}) = \beta \in F^n$. Using (7.8) with s_1 and h, we get $(g \curlywedge (s_1 \curlyvee s_2)) \curlyvee (h' \curlywedge \overline{(s_1 \curlyvee s_2)}) \in F^n$. Therefore $\mathbb{F}^n(f, s_1 \curlyvee s_2) \subseteq \mathbb{F}^n(g, s_1 \curlyvee s_2)$ and $f \preceq_{s_1 \curlyvee s_2} g$. □

7.4 Side dependencies in SL[EG]

This section is devoted to the proof of Theorem 7.1, which we recall below.

Theorem 7.1. *Consider a formula $\phi \in$ SL[EG], a game $\mathcal{G}$, a state q_{ini} of $\mathcal{G}$ and two parameters $\spadesuit \in \{\emptyset, S\}$ and $\heartsuit \in \{\emptyset, F\}$. If $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit,P)} \phi$ and $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\spadesuit,\heartsuit,P)} \neg\phi$ then $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$.*

We fix a concurrent game $\mathcal{G} = \langle \mathsf{AP}, \mathsf{Agt}, \mathsf{Q}, \mathsf{Act}, \Delta, \mathsf{labels} \rangle$ and consider a closed SL[EG] formula

$$\phi := (Q_i x_i)_{1 \leq i \leq l}.\ F^n((\beta_j \varphi_j)_{1 \leq j \leq n})$$

where for any $1 \leq i \leq l$, $Q_i \in \{\exists, \forall\}$, and for any $1 \leq j \leq n$, $\beta_j \colon \mathsf{Agt} \to \{x_j \mid 1 \leq j \leq l\}$ is a *full* assignment and φ_j is a LTL formula. We also write $\mathcal{V} := \{x_i \mid 1 \leq i \leq l\}$, $\mathcal{V}^\forall := \{x_i \mid Q_i = \forall\}$ and $\mathcal{V}^\exists := \{x_i \mid Q_i = \exists\}$. Finally, following Lemmas 7.8 and 7.11, we assume that F^n is upward closed (even if it means negating some of the LTL objectives).

Outline of the proof

The proof is quite long and technical, therefore we first sketch it to give its intuition to the reader. The idea at the heart of our proof is that, when making a decision for an existentially quantified variable x_i on a history ρ, we have knowledge of which goals are still active on ρ (as opposed to which ones have deviated) through the unordered prefix dependencies. We then represent the set of goals enabled on ρ (according to a given context) by an element s of $\{0,1\}^n$ and use the quasi-order $\preceq_s$ to get a clear cut hierarchy of the potential results. This way, when making a decision for x_j on ρ, we have a set of ordered potential results and just act to achieve the highest possible.

Sketch of proof

1. We define a set $\{D_{s,h} \mid s, h \in \{0,1\}^n\}$ of parity automata. Given a path ρ, we can associate an element $k \in \{0,1\}^n$ such that $k(i) = 1$ iff ρ satisfy φ_i. An automaton $D_{s,h}$ accepts a path ρ iff $h \preceq_s k$. The intuitive idea is that $D_{s,h}$ accepts a path ρ if and only if ρ produces a result at least as good as h relatively to $\preceq_s$.

2. Using these automata, we define two new sets of operators: $\Gamma^{stick}_{d,s,h}$ and $\Gamma^{sep}_{d,s,\Upsilon}$ for $s, h \in \{0,1\}^n$ and some parameters d, Υ that will be defined later on. The Γ^{stick} operators are used to encode the $\{D_{s,h}\}_{s,h \in \{0,1\}^n}$ automata while the $\Gamma^{sep}_{d,s,\Upsilon}$ operators handle the junction between the different Γ^{stick} operators. The operators are essential to the step below; as we will see in the technical proof, they are however rather technical and therefore we cannot provide much intuition about them.

3. We highlight some specific elements $b_{q,d,s}$ of $\{0,1\}^n$ where q is a state of $\mathcal{G}$, d represents some knowledge about the history and s is a set of active goals. A $b_{q,d,s}$ element represents the best we can hope to achieve relatively to $\preceq_s$ when we consider

a history ρ finishing in q, carrying the information on d and where s represents the set of active goals on ρ.

The construction of the b elements is done inductively based on the size of s: we start with $|s| = 1$ toward $|s| = n$ and use the elements defined at previous steps to decide if we should keep the goals active in s together or if we should separate them from one another.

4. We then produce two $\mathcal{M}(\emptyset, \emptyset, P)$ maps θ and $\overline{\theta}$. Using the b elements built at the previous step, the map θ deduces an optimal strategy in order to satisfy ϕ. θ is in some way (made clear within the technical proof) an optimal map for ϕ. Similarly, $\overline{\theta}$ is the best map to avoid the satisfaction of ϕ.

5. Finally, we show the following lemma:

Lemma 7.17. *There exists a valuation χ of domain $\mathcal{V}$ such that $\theta(\chi_{|\mathcal{V}^\forall}) = \chi$ and $\overline{\theta}(\chi_{|\mathcal{V}^\exists}) = \chi$. Moreover χ satisfies*

$$\begin{aligned}
&\mathcal{G}, q_{ini} \models_\chi \ \Gamma_{F^n} \quad &\Rightarrow \quad &\forall w : (\mathsf{Hist}_\mathcal{G} \to \mathsf{Act})^{\mathcal{V}^\forall} \ \mathcal{G}, q_{ini} \models_{\theta(w)} F^n(\beta_j \varphi_j)_{1 \leq j \leq n} \\
&\mathcal{G}, q_{ini} \models_\chi \ \neg\Gamma_{F^n} \quad &\Rightarrow \quad &\forall \overline{w} : (\mathsf{Hist}_\mathcal{G} \to \mathsf{Act})^{\mathcal{V}^\exists} \ \mathcal{G}, q_{ini} \models_{\overline{\theta}(\overline{w})} \overline{F^n}(\beta_j \varphi_j)_{1 \leq j \leq n}
\end{aligned}$$

We can then apply a simple reasoning to get Theorem 7.1. Assume that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit,P)} \phi$ and $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\spadesuit,\heartsuit,P)} \neg\phi$. If $\mathcal{G}, q_{ini} \models_\chi \ \neg\Gamma_{F^n}$, then by the second point of Lemma 7.17, $\overline{\theta}$ would be a witness that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} \neg\phi$ and it would hold that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit,P)} \neg\phi$. So $\mathcal{G}, q_{ini} \models_\chi \ \Gamma_{F^n}$ and using Lemma 7.17 once again, we get that θ is a witness that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$.

7.4.1 Automata

We build a large set of *deterministic parity word automata* over 2^{AP}. For $s \in \{0,1\}^n$ and $h \in \{0,1\}^n$, we let $D_{s,h}$ be a deterministic parity automaton accepting exactly the words over 2^{AP} along which the following formula[2] holds:

$$\bigvee_{\substack{k \in \{0,1\}^n \\ h \preceq_s k}} \ \bigwedge_{\substack{j \text{ s.t.} \\ (k \curlywedge s)(j)=1}} \varphi_j. \tag{7.9}$$

where a conjunction over an empty set (i.e., if $(k \curlywedge s)(j) = 0$ for all j) is true. As an example, take $s \in \{0,1\}^n$ with $|s| = 1$, writing j for the index where $s(j) = 1$, for any $h \in \{0,1\}^n$ we get that $D_{s,h}$ is universal iff there is $k \succeq_s h$ with $k(j) = 0$; otherwise $D_{s,h}$ accepts the set of words over 2^{AP} along which φ_j holds.

[2]Likewise to Section 7.3.3, to retain a rigorous definition and avoid too many notations we use $k \in \{0,1\}^n$ despite k being in essence an element of $\{0,1\}^{|s|}$: indeed k only matters in Formula (7.9) on the indexes j where $s(j) = 1$, therefore only on $|s|$ indexes.

Write $\mathcal{D} = \{D_{s,h} \mid s \in \{0,1\}^n,\ h \in \{0,1\}^n\}$ for the set of automata just defined. A *vector of states* of $\mathcal{D}$ is a function associating with each automaton $D \in \mathcal{D}$ one of its states. We write VS for the set of all vectors of states of $\mathcal{D}$. Let d be a vector of states of $\mathcal{D}$ and let q be a state of $\mathcal{G}$. We set $\mathsf{succ}(d,q)$ to be the function associating with each $D \in \mathcal{D}$ the successor of $d(D)$ upon reading the labelling $\mathsf{labels}(q)$ of q; we also extend succ to take an input (d,ρ) and to return the state reachable by ρ from d. As usual, a path $(q_i)_{i\in\mathbb{N}}$ in $\mathcal{G}$ is accepted by an automaton D of $\mathcal{D}$ whenever its labels sequence $(\mathsf{labels}(q_i))_{i\in\mathbb{N}}$ is accepted by D. We use the customary notation $\mathcal{L}(D)$ for the set of words accepted by an automaton D. Finally we denote by $\mathcal{L}(D^d_{s,h})$ the set of words that are accepted by $D_{s,h}$ starting from the state $d(D_{s,h})$.

Proposition 7.18. *The following holds for any $s \in \{0,1\}^n$:*

1. *for any $h_1, h_2 \in \{0,1\}^n$ where $h_1 \preceq_s h_2$, we have $\mathcal{L}(D_{s,h_1}) \supseteq \mathcal{L}(D_{s,h_2})$.*
2. *$D_{s,\mathbf{0}}$ is universal.*
3. *for any $h \in F^n$, $D_{\mathbf{1},h}$ accepts the words satisfying $\bigvee_{f\in F^n} \bigwedge_{j \text{ s.t. } f(j)=1} \varphi_j$.*

Proof. The first and third points are immediate. In Formula (7.9) applied to $h = \mathbf{0}$, take $k = \mathbf{0}$ in the disjunction; then the conjunction is empty thus trivially true and therefore $D_{s,\mathbf{0}}$ accepts any word over 2^{AP}. □

7.4.2 Supervising goals going on different paths

Using the automata in $\mathcal{D}$, we define two new families of temporal operators for the proof of Theorem 7.1. Their semantics differ from the *until* and *next* operators: they are relative to the values of a valuation on the variables and are not asking to assign a strategy to each agent. The first family of operators simply transfers the conditions of the automata of $\mathcal{D}$ onto an operator for a later usage. For any $d \in \mathsf{VS}$ and any two s, h in $\{0,1\}^n$, the *parity* operator $\Gamma^{stick}_{d,s,h}$ obeys the following semantics[3]: given a context χ with $\mathcal{V} \subseteq \mathsf{dom}(\chi)$ and a state q of $\mathcal{G}$,

$$\mathcal{G}, q \models_\chi \Gamma^{stick}_{d,s,h} \quad \Leftrightarrow \quad \begin{array}{c}\exists\ \rho \text{ infinite in} \\ \mathcal{G} \text{ from } q \text{ with}\end{array} \begin{cases} \forall j \leq n,\ s(j) = 1 \Rightarrow \mathsf{out}(\beta_j(\chi), q) = \rho \\ \\ \rho \in \mathcal{L}(D^d_{s,h}) \end{cases}$$

Intuitively, the outcome of the assignments enabled by s must follow a common path that is accepted by $D^d_{s,h}$.

The main difficulty of $\mathsf{SL[EG]}$ (or $\mathsf{SL[BG]}$ more generally) lies in the separation of the different goals along different histories. The second batch of operators must tackle this difficulty but before defining them, we need some formalism (we recall that Q is the set of states of $\mathcal{G}$ and VS is the set of all vectors of states of $\mathcal{D}$):

[3] We recall that the different maps introduced in Chapters 5 and 6 are for finding suitable behaviour for the quantifications and that there is a common semantics to all temporal and boolean operators, therefore there is only one satisfaction relation ($\models$) for the Γ^{stick} operators.

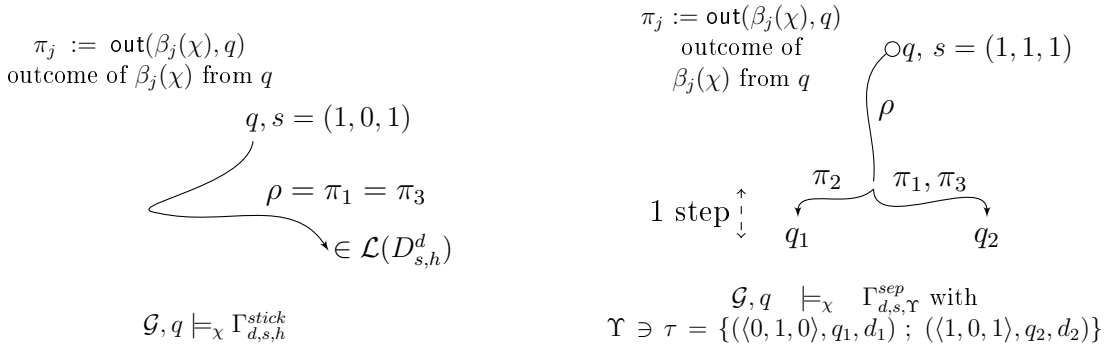


Figure 7.3: The Γ^{stick} and Γ^{sep} operators.

Definition 7.19. *A* partition *of an element $s \in \{0,1\}^n$ is a set $\{s_\kappa \mid 1 \leq \kappa < \lambda\}$ of two or more elements of $\{0,1\}^n$ with $s_1 \curlyvee \ldots \curlyvee s_\lambda = s$ and where for any two $\kappa \neq \kappa'$ and any $j \leq n$ we have $s_\kappa(j) = 1 \Rightarrow s_{\kappa'}(j) = 0$.*

An extended *partition of s is a set $\tau := \{(s_\kappa, q_\kappa, d_\kappa) \in \{0,1\}^n \times Q \times VS \mid 1 \leq \kappa \leq \lambda\ ,\ \lambda \geq 2\}$ with $(s_\kappa)_{\kappa \leq \lambda}$ a partition of s.*

Note that we only consider nontrivial partitions. We write $\mathsf{Part}(s)$ for the set of all extended partitions of s. If $|s| =\leq 1$, then $\mathsf{Part}(s) = \emptyset$. For any $d \in \mathsf{VS}$, any s in $\{0,1\}^n$ and any set of partitions Υ of s, the condition $\Gamma^{sep}_{d,s,\Upsilon}$ looks for the assignments enabled by s to all follow a common history ρ for some time then partition themselves according to some partition in Υ. Its semantics are defined upon a context χ with $\mathcal{V} \subseteq \mathsf{dom}(\chi)$ and a state q of $\mathcal{G}$ by the formula below. Figure 7.3 gives an intuition on both operators.

$$\mathcal{G}, q \models_\chi \Gamma^{sep}_{d,s,\Upsilon} \quad \Leftrightarrow \quad \begin{array}{c}\exists \tau \in \Upsilon.\ \exists \rho \\ \text{finite history} \\ \text{in } \mathcal{G} \text{ from } q \\ \text{such that}\end{array} \left\{ \begin{array}{l} \forall j \leq n, \\ s(j) = 1 \Rightarrow \rho \in \mathsf{Pref}_{<\mathsf{out}(\beta_j(\chi), q)} \\ \\ \forall \kappa \leq |\tau|,\ \forall j \leq n, \\ s_\kappa(j) = 1 \Rightarrow q_\kappa = \Delta(\mathsf{lst}(\rho), m_j) \text{ with} \\ \forall A \in \mathsf{Agt},\ m_j(A) := \chi(\beta_j(A), \rho) \\ \\ \forall \kappa \leq |\tau|, \text{ applying } \mathsf{succ} \text{ inductively} \\ \text{from } d \text{ on the path } \rho.q_\kappa \text{ leads to } d_\kappa \end{array} \right.$$

7.4.3 Finding optimal elements

By an induction on $|s|$ ranging from 1 to n,

1. for every s with $|s| = \alpha$, every $h \in \{0,1\}^n$ and every $d \in \mathsf{VS}$, we define a new temporal operator $\Gamma_{d,s,h}$ based on the Γ^{stick} and Γ^{sep} operators.

   The Γ^{stick}'s operators handle the case where all goals stay on the same path; the Γ^{sep}'s operators handle the case where the goals split in different directions. The

operator $\Gamma_{d,s,h}$ will regroup both possibilities and ask that starting with information d, the goals of s do at least as good as h for $\preceq_s$.

2. for every s with $|s| = \alpha$, every $d \in \mathsf{VS}$ and every state q of $\mathcal{G}$, we define an element $b_{q,d,s}$ of $\{0,1\}^n$.

   The $b_{q,d,s}$ element carries the information about the highest $h \in \{0,1\}^n$ possible for $\preceq_s$ so that the operator $\Gamma_{d,s,h}$ is satisfied.

3. if $\alpha \neq n$, for all $s \in \{0,1\}^n$ with $|s| = \alpha + 1$ and all $\tau \in \mathsf{Part}(s)$, we define yet another element $c_{s,\tau}$ of $\{0,1\}^n$.

   The c's elements carry information about previous step of the induction in the form of an element of $\{0,1\}^n$. Past the initial step, $c_{s,\tau}$ is used to determine the b's elements of the form $b_{\star,\star,s}$.

This induction allows us to condense information about the best course possible in the form of elements of $\{0,1\}^n$: the b's and c's elements. Theses elements will then be used to build an optimal behaviour in later sections.

Initial step ($\alpha = 1$)

1. For any $d \in \mathsf{VS}$ and any two s, h of $\{0,1\}^n$ with $|s| = 1$ we set $\Gamma_{d,s,h} := \Gamma^{stick}_{d,s,h}$.

2. For any state q of $\mathcal{G}$, any $d \in \mathsf{VS}$ and any $s \in \{0,1\}^n$ with $|s| = 1$, there is a maximal element $b_{q,d,s} \in \{0,1\}^n$ for the order $\preceq_s$ such that

$$\mathcal{G}, q \models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1 \leq i \leq l}\ \Gamma_{d,s,b_{q,d,s}} \tag{7.10}$$

   By Proposition 7.18, $D^d_{s,\mathbf{0}}$ is universal; therefore, for any complete valuation χ, $\mathcal{G}, q \models_\chi \Gamma_{d,s,\mathbf{0}}$. This trivially implies that any $\mathcal{M}(\emptyset,\emptyset,P)$ map Δ is a witness that Formula (7.10) holds for $b_{q,d,s} = \mathbf{0}$. So there is at least one element of $\{0,1\}^n$ to fill the role of $b_{q,d,s}$ for Formula (7.10) and, because $\preceq_s$ is a total quasi order, there must exist a maximal element. On the other hand, unicity is not guaranteed : if $h_1 =_s h_2$ then $\mathcal{L}(D_{s,h_1}) = \mathcal{L}(D_{s,h_2})$ thus $\Gamma_{d,s,h_1} = \Gamma_{d,s,h_2}$ and

$$\mathcal{G}, q \models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1 \leq i \leq l}\ \Gamma_{d,s,h_1} \quad \Leftrightarrow \quad \mathcal{G}, q \models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1 \leq i \leq l}\ \Gamma_{d,s,h_2}$$

   Characterisation $b_{q,d,s} \in \{0,1\}^n$ is an element such that

   (a) $\mathcal{G}, q \models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1 \leq i \leq l}\ \Gamma_{d,s,b_{q,d,s}}$

   (b) for any $h \in \{0,1\}^n$ with $b_{q,d,s} \prec_s h$, we have $\mathcal{G}, q \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1 \leq i \leq l}\ \Gamma_{d,s,h}$

3. Fix some $s \in \{0,1\}^n$ with $|s| = 2$ and an extended partition $\tau := \{(s_\kappa, q_\kappa, d_\kappa) \mid 1 \leq \kappa \leq 2\}$ of s. By definition of τ, for any $\kappa \leq 2$ we have $|s_\kappa| < |s| = 2$ i.e. $|s_\kappa| = 1$ thus $b_{q_\kappa,d_\kappa,s_\kappa}$ have been defined just before. We define $c_{s,\tau}$ by

$$c_{s,\tau} = (s_1 \curlywedge b_{q_1,d_1,s_1}) \curlyvee (s_2 \curlywedge b_{q_2,d_2,s_2})$$

The partition τ models a possible way for the goals to split; $c_{s,\tau}$ then regroups the b elements adequate to τ in a single element of $\{0,1\}^n$. Ergo $c_{s,\tau}$ carries information about the best that can be achieved just after the goals split along τ. The $c_{s,\tau}$ belonging to $\{0,1\}^n$, we can compare it to other elements of $\{0,1\}^n$ carrying other information using the quasi-orders of Section 7.3.3. Using these comparisons, we will then deduce an optimal approach.

Induction step $(1 < \alpha \leq n)$

The induction step is slightly more involved.

1. For any $d \in \mathsf{VS}$ and any two s, h of $\{0,1\}^n$ with $|s| = \alpha$, we define an operator $\Gamma_{d,s,h}$ by
$$\Gamma_{d,s,h} \;=\; \Gamma^{stick}_{d,s,h} \;\vee\; \Gamma^{sep}_{d,s,\Upsilon} \qquad \text{where } \Upsilon := \{\tau \in \mathsf{Part}(s) \mid h \preceq_s c_{s,\tau}\}$$
We recall that $c_{s,\tau}$ was defined at the previous step of the induction. Figure 7.4 gives an intuition.

2. As before, for any q, any $d \in \mathsf{VS}$ and any $s \in \{0,1\}^n$ with $|s| = \alpha$, there is a maximal element $b_{q,d,s} \in \{0,1\}^n$ for the order $\preceq_s$ such that
$$\mathcal{G}, q \models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1\leq i\leq l}\ \Gamma_{d,s,b_{q,d,s}} \tag{7.11}$$
Similarly to the initial step, we show that such an element $b_{q,d,s}$ exists by proving that Formula (7.11) holds for $b_{q,d,s} = \mathbf{0}$. F^n is upward closed so $\mathbf{0}$ is a minimal element of $\preceq_s$ (no matter s) and for any $\tau \in \mathsf{Part}(s)$, $\mathbf{0} \preceq_s c_{s,\tau}$. Now, consider any given complete valuation χ. First of two possibilities: after some finite history ρ, χ splits the outcomes of the goals enabled by s into different paths following a partition τ_0, then we get
$$\mathcal{G}, q \models_\chi \Gamma^{sep}_{d,s,\Upsilon} \quad \text{for } \Upsilon := \{\tau \in \mathsf{Part}(s) \mid \mathbf{0} \preceq_s c_{s,\tau}\} = \mathsf{Part}(s)$$
Second possibility: all the outcomes (enabled by s) follow the same infinite path. $D_{s,\mathbf{0}}$ is universal (Proposition 7.18) so we get $\mathcal{G}, q \models_\chi \Gamma^{stick}_{d,s,\mathbf{0}}$. This means that, whatever the value of χ, it holds that $\mathcal{G}, q \models_\chi \Gamma_{d,s,\mathbf{0}}$. Hence, as for the initial case, any $\mathcal{M}(\emptyset,\emptyset,P)$ map is a witness that Formula (7.11) holds for $\Gamma_{d,s,\mathbf{0}}$. As for the initial step, unicity is not guaranteed : if $h_1 =_s h_2$ then $\mathcal{L}(D_{s,h_1}) = \mathcal{L}(D_{s,h_2})$ thus $\Gamma^{stick}_{d,s,h_1} = \Gamma^{stick}_{d,s,h_2}$ and $h_1 \preceq_s c_{s,\tau}$ iff $h_2 \preceq_s c_{s,\tau}$, so
$$\mathcal{G}, q \models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1\leq i\leq l}\ \Gamma_{d,s,h_1} \quad\Leftrightarrow\quad \mathcal{G}, q \models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1\leq i\leq l}\ \Gamma_{d,s,h_2}$$

Characterisation $b_{q,d,s} \in \{0,1\}^n$ is an element such that

(a) $\mathcal{G}, q \models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1\leq i\leq l}\ \Gamma_{d,s,b_{q,d,s}}$

(b) for any $h \in \{0,1\}^n$ with $b_{q,d,s} \prec_s h$, we have $\mathcal{G}, q \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1\leq i\leq l}\ \Gamma_{d,s,h}$

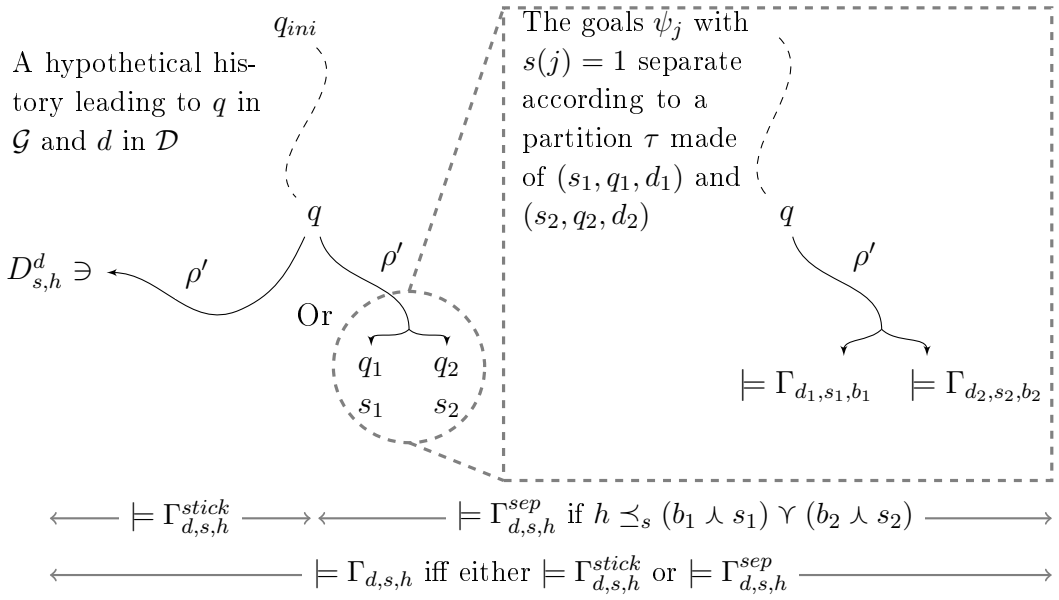


Figure 7.4: The $\Gamma_{d,s,h}$ operator

3. In the case of $\alpha < n$, fix some $s \in \{0,1\}^n$ with $|s| = \alpha+1$ and an extended partition $\tau := \{(s_\kappa, q_\kappa, d_\kappa) \mid 1 \leq \kappa \leq \lambda\ ,\ \lambda \geq 2\}$ of s. By definition of τ, for any $\kappa \leq \lambda$ we have $|s_\kappa| < |s| = \alpha+1$, and the element $b_{q_\kappa,d_\kappa,s_\kappa}$ has been defined on previous steps of the induction. We define $c_{s,\tau}$ by

$$c_{s,\tau} = (s_1 \curlywedge b_{q_1,d_1,s_1}) \curlyvee \ldots \curlyvee (s_\lambda \curlywedge b_{q_\lambda,d_\lambda,s_\lambda})$$

Intermediary results

We now focus on results derived from the elements defined previously.

Lemma 7.20. *For any state q, any $d \in \mathsf{VS}$ and any $s \in \{0,1\}^n$, there is a $\mathcal{M}(\emptyset,\emptyset,P)$ map $\varrho_{q,d,s}$ for $(Q_i x_i)_{1\leq i\leq l}$ witnessing that*

$$\mathcal{G}, q \models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1\leq i\leq l}\ \Gamma_{d,s,b_{q,d,s}}$$

There is another $\mathcal{M}(\emptyset,\emptyset,P)$ map $\overline{\varrho_{q,d,s}}$ for $(\overline{Q_i} x_i)_{1\leq i\leq l}$ witnessing that

$$\mathcal{G}, q \models^{\mathcal{M}(\emptyset,\emptyset,P)} (\overline{Q_i} x_i)_{1\leq i\leq l} \bigwedge_{b_{q,d,s} \prec_s h} \neg\Gamma_{d,s,h}$$

The proof of the first part is an immediate consequences of the definitions of the b's elements. Combining the optimality of the b's elements with Theorem 6.5 gives us the second part.

We also highlight a peculiar Γ operator whose parameters are set by ϕ. In the big induction, we inductively defined both the Γ's operators, the b's elements and the c's

elements. In a way, Γ_{F^n} is what we find at the very top of the induction. Notice that for any two $f, f' \in F^n$, we have $f =_{\mathbf{1}} f'$ and thus $\mathcal{L}(D_{\mathbf{1},f}) = \mathcal{L}(D_{\mathbf{1},f'})$. By definition of the Γ^{stick} operators, for $d \in \mathsf{VS}$, $\Gamma^{stick}_{d,\mathbf{1},f} = \Gamma^{stick}_{d,\mathbf{1},f'}$. We then set

$$\Gamma^{stick}_{F^n} \;=\; \Gamma^{stick}_{d_{ini},\mathbf{1},f} \qquad \text{and} \qquad \Gamma_{F^n} \;=\; \Gamma^{stick}_{F^n} \;\vee\; \Gamma^{sep}_{d_{ini},\mathbf{1},\Upsilon_{F^n}}$$

where f is any element in F^n, d_{ini} is the initial vector of states and with $\Upsilon_{F^n} := \{\tau \in \mathsf{Part}(\mathbf{1}) \mid c_{\mathbf{1},\tau} \in F^n\}$. Thus the operator Γ_{F^n} is the element of the Γ family of operators where d is the initial vector of state, $s = \mathbf{1}$ and h is an element of F^n.

The same way we highlighted Γ_{F^n} in the family of all Γ's operators, we highlight two $\mathcal{M}(\emptyset,\emptyset,P)$ maps $\varrho_{\mathbf{1}}$ and $\overline{\varrho_{\mathbf{1}}}$. The map $\varrho_{\mathbf{1}}$ corresponds to $\varrho_{q_{ini},d_{ini},\mathbf{1}}$ while the map $\overline{\varrho_{\mathbf{1}}}$ corresponds to $\overline{\varrho_{q_{ini},d_{ini},\mathbf{1}}}$.

Lemma 7.21. *If $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1\leq i\leq l}\ \Gamma_{F^n}$, then $\varrho_{\mathbf{1}}$ witness that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1\leq i\leq l}\Gamma_{F^n}$. If $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1\leq i\leq l}\ \Gamma_{F^n}$, then $\overline{\varrho_{\mathbf{1}}}$ witness that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} (\overline{Q_i} x_i)_{1\leq i\leq l}\neg\Gamma_{F^n}$.*

Proof. The first case is a simple application of Lemma 7.20. For the second case, assume that $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1\leq i\leq l}\ \Gamma_{F^n}$; this implies that $b_{q_{ini},d_{ini},\mathbf{1}} \notin F^n$. We can then apply Lemma 7.20 to an element $f \in F^n$ to get that $\overline{\varrho_{\mathbf{1}}}$ witnesses $\mathcal{G}, q \models^{\mathcal{M}(\emptyset,\emptyset,P)} (\overline{Q_i} x_i)_{1\leq i\leq l}\neg\Gamma_{d,s,f}$. Now by definition of Γ_{F^n} this means that $\overline{\varrho_{\mathbf{1}}}$ also witnesses $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} (\overline{Q_i} x_i)_{1\leq i\leq l}\neg\Gamma_{F^n}$. □

7.4.4 Assembling optimal $\mathcal{M}(\emptyset,\emptyset,P)$ maps

Having done this preliminary work, we may now build two $\mathcal{M}(\emptyset,\emptyset,P)$ maps θ and $\overline{\theta}$ to define a behaviour respectively for $(Q_i x_i)_{1\leq i\leq l}$ and $(\overline{Q_i} x_i)_{1\leq i\leq l}$. They are optimal (in a sense that will become clear later) respectively for $F^n((\beta_j\varphi_j)_{1\leq j\leq n})$ and $\overline{F^n}((\beta_j\varphi_j)_{1\leq j\leq n})$. To define the two maps, we start on the root and progress inductively along the histories. Given a history ρ and a function $w : (\mathsf{Hist} \to \mathsf{Act})^{\mathcal{V}^\forall}$, we can know which goal is still following ρ. Indeed, assume θ has been defined on strict prefixes of ρ, we say that a goal $\psi_j := \beta_j\varphi_j$ is *active* on ρ w.r.t $\theta(w)$ whenever

$$\forall i < |\rho|,\ \Delta(\rho(i), m_i) = \rho(i+1) \begin{cases} \text{with } m_i : \mathsf{Agt} \to \mathsf{Act} \text{ is defined by} \\ \forall A \in \mathsf{Agt},\ m_i(A) := \theta(w)(\beta_j(A))(\rho_{\leq i}) \end{cases}$$

Under these circumstances, we denote by $s_{\rho,\theta(w)} \in \{0,1\}^n$ the unique element such that $s_{\rho,\theta(w)}(j) = 1$ iff β_j is active on ρ w.r.t $\theta(w)$.

The idea behind θ is to combine together the maps defined in Lemmas 7.20 and 7.21. We start by defining θ that aims to satisfy $(Q_i x_i)_{i\leq l}\ F^n(\beta_j\varphi_j)_{j\leq n}$.

- If $x_i \in \mathcal{V}^\forall$, we must set $\theta(w)(x_i)(\rho) := w(x_i)(\rho)$ whatever the inputs $w : (\mathsf{Hist} \to \mathsf{Act})^{\mathcal{V}^\forall}$ and $\rho \in \mathsf{Hist}$ by definition of maps (of any kind).

- If $x_i \in \mathcal{V}^\exists$, we use the maps defined in Lemmas 7.20 and 7.21. Consider a history ρ, a function $w : (\mathsf{Hist} \to \mathsf{Act})^{\mathcal{V}^\forall}$ and a variable $x_i \in \mathcal{V}$ such that θ has been defined on strict prefixes of ρ. Then, θ being already defined on strict prefixes of ρ and having unordered prefix dependencies, we can know the active goals on ρ for $\theta(w)$, and we represent them by an element $s_{\rho,\theta(w)}$ of $\{0,1\}^n$. We decompose ρ in two parts ρ_1 and ρ_2 ($\rho = \rho_1.\rho_2$) such that ρ_2 represents the part of ρ that is followed by exactly the goals of $s_{\rho,\theta(w)}$, i.e. ρ_1 is the maximal prefix such that $s_{\rho_1,\theta(w)} \neq s_{\rho,\theta(w)}$.

  - First of two possibilities: $s_{\rho,\theta(w)} = \mathbf{1}$ and $\rho_1 = \varepsilon$. Then θ follows the map $\varrho_{\mathbf{1}}$ of Lemma 7.21 and we set $\theta(w)(x_i)(\varepsilon) := \varrho_{\mathbf{1}}(w)(x_i)(\varepsilon)$.
  - Second possibility: $s_{\rho,\theta(w)} \neq \mathbf{1}$ and ρ_1 is not empty. The map θ then regroups the important information of ρ_1 in the vector of state $d_{\rho_1} := \mathsf{succ}(d_{ini}, \rho_1)$ of $\mathcal{D}$. The behaviour of θ on ρ then follows the maps of Lemma 7.20, meaning that we set $\theta(w)(x_i)(\rho) := \varrho_{\mathsf{lst}(\rho_1),d_{\rho_1},s_{\rho,u}}(w_{\overrightarrow{\rho_1}})(x_i)(\rho_2)$.

Having defined θ, we proceed similarly to define $\overline{\theta}$, a $\mathcal{M}(\emptyset,\emptyset,P)$ map trying to ensure $\overline{F^n}(\beta_j\varphi_j)_{j\leq n}$.

- If $x_i \in \mathcal{V}^\exists$, we must set $\overline{\theta}(\overline{w})(x_i)(\rho) := \overline{w}(x_i)(\rho)$ whatever the inputs $\overline{w} : (\mathsf{Hist} \to \mathsf{Act})^{\mathcal{V}^\exists}$ and $\rho \in \mathsf{Hist}$ by definition of maps.

- If $x_i \in \mathcal{V}^\forall$, consider a history ρ, a function $\overline{w} : (\mathsf{Hist} \to \mathsf{Act})^{\mathcal{V}^\exists}$ and a variable $x_i \in \mathcal{V}$ such that $\overline{\theta}$ has been defined on strict prefixes of ρ. We proceed again by decomposing ρ in two parts ρ_1 and ρ_2 such that ρ_1 is the maximal prefix with $s_{\rho_1,\overline{\theta}(\overline{w})} \neq s_{\rho,\overline{\theta}(\overline{w})}$.

  - if $s_{\rho,\theta(w)} = \mathbf{1}$ and ρ_1 is empty, we set $\overline{\theta}(\overline{w})(x_i)(\varepsilon) := \overline{\varrho_{\mathbf{1}}}(\overline{w})(x_i)(\varepsilon)$.
  - if $s_{\rho,\theta(w)} \neq \mathbf{1}$ and ρ_1 is non-empty, we set $\overline{\theta}(\overline{w})(x_i)(\rho) := \overline{\varrho_{\mathsf{lst}(\rho_1),d_{\rho_1},s_{\rho,u}}}(\overline{w}_{\overrightarrow{\rho_1}})(x_i)(\rho_2)$ with $d_{\rho_1} = \mathsf{succ}(d_{ini}, \rho_1)$.

7.4.5 Optimality of θ and $\overline{\theta}$

Under the assumption that only one of ϕ and $\neg\phi$ can hold on $\mathcal{G}$ under $\mathcal{M}(\spadesuit, \heartsuit, P)$ maps, θ is in a sense an optimal map for $F^n(\beta_j\varphi_j)_{1\leq j\leq n}$ while $\overline{\theta}$ is optimal for $\overline{F^n}(\beta_j\varphi_j)_{1\leq j\leq n}$. The lemma below formalises this optimality property and characterises when θ can ensure $F^n(\beta_j\varphi_j)_{1\leq j\leq n}$ and when $\overline{\theta}$ can ensure $\overline{F^n}(\beta_j\varphi_j)_{1\leq j\leq n}$. The proof can be found in the annex 7.A (page 182).

Lemma 7.17. *There exists a valuation χ of domain $\mathcal{V}$ such that $\theta(\chi_{|\mathcal{V}^\forall}) = \chi$ and $\overline{\theta}(\chi_{|\mathcal{V}^\exists}) = \chi$. Moreover χ satisfies*

$$\begin{array}{lll} \mathcal{G}, q_{ini} \models_\chi \Gamma_{F^n} & \Rightarrow & \forall w : (\mathsf{Hist}_\mathcal{G} \to \mathsf{Act})^{\mathcal{V}^\forall}\ \mathcal{G}, q_{ini} \models_{\theta(w)} F^n(\beta_j\varphi_j)_{1\leq j\leq n} \\ \mathcal{G}, q_{ini} \models_\chi \neg\Gamma_{F^n} & \Rightarrow & \forall \overline{w} : (\mathsf{Hist}_\mathcal{G} \to \mathsf{Act})^{\mathcal{V}^\exists}\ \mathcal{G}, q_{ini} \models_{\overline{\theta}(\overline{w})} \overline{F^n}(\beta_j\varphi_j)_{1\leq j\leq n} \end{array}$$

We can now prove Theorem 7.1. Assume that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit,P)} \phi$ and $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\spadesuit,\heartsuit,P)} \neg\phi$. If $\mathcal{G}, q_{ini} \models_{\chi} \neg\Gamma_{F^n}$, then by the second point of Lemma 7.17, $\overline{\theta}$ would be a witness that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} \neg\phi$ and it would hold that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit,P)} \neg\phi$. So $\mathcal{G}, q_{ini} \models_{\chi} \Gamma_{F^n}$ and using Lemma 7.17 once again, we get that θ is a witness that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$. This concludes the proof of Theorem 7.1.

7.4.6 Closing remarks on SL[EG] complexity

From our work, we can also deduce a *2*-EXPTIME algorithm for the model checking of SL[EG] relatively to $\models^{\mathcal{M}(\emptyset,\emptyset,P)}$.

Theorem 7.22. *The model checking problem of SL[EG] formulas relatively to the* $\models^{\mathcal{M}(\emptyset,\emptyset,P)}$ *satisfaction relation is 2-*EXPTIME*-complete.*

Proof. We reuse the notations in the proof of Theorem 7.1 (Sections 7.4.1 to 7.4.5). The proof consists in building a procedure to check if $\mathcal{G}, q_{ini} \models_{\chi} \Gamma_{F^n}$ holds; we can then deduce if $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$ holds by using Lemma 7.17. The procedure is given in the annex 7.B (page 185).

Lemma 7.23. *Consider* s *in* $\{0,1\}^n$ *and assume that we know the values of all* $b_{q',d',s'}$ *for all* $s' \in \{0,1\}^n$ *with* $|s'| < |s|$, $q' \in Q$ *and* $d' \in VS$. *We then have a 2-*EXPTIME *algorithm that computes*

- *the truth value of* $\mathcal{G}, q \models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1\leq i\leq l}\Gamma_{d,s,h}$ *for any* $q \in Q$, $d \in VS$ *and* $h \in \{0,1\}^n$.
- *the value of* $b_{q,d,s}$ *for any* $d \in VS$ *and* $q \in Q$.

Sketch of proof. The proof consists in finding the potential candidates in $\{0,1\}^n$ for $b_{q,d,s}$ and choosing the optimal one for $\preceq_s$. Each candidate is checked by building and solving a parity game based on the ideas developed for Theorems 5.14 and 6.5 (pages 127 and 142). □

The operator Γ_{F^n} is defined as a special case (see Section 7.4.3) of the family of all Γ's operators based on the b's elements. We can then use the procedure of Lemma 7.23 to build the b's elements, define Γ^{F^n}, check the truth value of $\mathcal{G}, q_{ini} \models_{\chi} \Gamma_{F^n}$ and deduce if $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$.

In the end, we need to build and solve at most $2^n.|Q|.2^{n.2^{|\phi|}}$ parity games. The state space of each game is of size two exponential in the size of the formula and polynomial in the size of the game. The number of indexes is exponential in the size of the formula. Using standard techniques to solve parity game we retrieve a *2*-EXPTIME algorithm for Lemma 7.23 and for SL[EG] model checking relatively to $\models^{\mathcal{M}(\emptyset,\emptyset,P)}$. We can derive a matching lower bound from the *2*-EXPTIME lower bound of ATL* model checking.

□

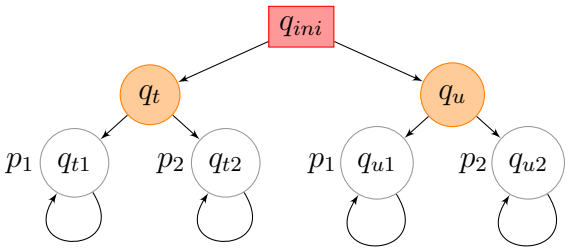


Figure 7.5: The two-agents turn-based game $\mathcal{G}$

7.5 Maximality of SL[EG]

In this section, we prove that SL[EG] is, in a sense, the maximal fragment of SL[BG] possible for the results of Theorem 7.1: we prove that if F^n is not semi-stable, we can build a formula that holds true in a given game for $\models^{\mathcal{M}(S,\emptyset,P)}$, where its negation does not hold for $\models^{\mathcal{M}(S,\emptyset,P)}$ but where we cannot remove side dependencies.

Theorem 7.24. *For any $n \in \mathbb{N}^*$ and any non-semi-stable set F^n there exists a SL[BG] formula ϕ built on F^n, a game $\mathcal{G}$ and a state q_{ini} of $\mathcal{G}$ such that*

$$\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,P)} \phi \quad \textit{and} \quad \mathcal{G}, q_{ini} \not\models^{\mathcal{M}(S,\emptyset,P)} \neg\phi \quad \textit{and} \quad \mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$$

Proof. We consider the two-agents game $\mathcal{G}$ depicted on Figure 7.5 with two agents $\square$ and $\bigcirc$. Let F^n be a non-semi-stable set over $\{0,1\}^n$. Then there must exist $f_1, f_2 \in F^n$, and $s \in \{0,1\}^n$, such that $(f_1 \curlywedge s) \curlyvee (f_2 \curlywedge \overline{s}) \notin F^n$ and $(f_2 \curlywedge s) \curlyvee (f_1 \curlywedge \overline{s}) \notin F^n$. We then let

$$\phi \;:=\; \forall y_t^{\square}\, \forall y_u^{\square}\, \forall x_t^{\bigcirc}\, \exists x_u^{\bigcirc}\, \cdot F^n(\beta_1\varphi_1, \ldots, \beta_n\varphi_n)$$

where

$$\beta_i = \begin{cases} \mathsf{assign}(\square, y_t^{\square}; \bigcirc, x_t^{\bigcirc}) & \text{if } s(i) = 1 \\ \mathsf{assign}(\square, y_u^{\square}; \bigcirc, x_u^{\bigcirc}) & \text{if } s(i) = 0 \end{cases}$$

and

$$\varphi_i = \begin{cases} \mathbf{F}\, p_1 \vee \mathbf{F}\, p_2 & \text{if } f_1(i) = f_2(i) = 1 \\ \mathbf{F}\, p_1 & \text{if } f_1(i) = 1 \text{ and } f_2(i) = 0 \\ \mathbf{F}\, p_2 & \text{if } f_1(i) = 0 \text{ and } f_2(i) = 1 \\ \mathtt{false} & \text{if } f_1(i) = f_2(i) = 0 \end{cases}$$

It is not hard to check that the following holds:

Lemma 7.25. *Let ρ be a maximal run of $\mathcal{G}$ from q_{ini}. Let $k \in \{1,2\}$ be such that ρ visits a state labelled with p_k. Then for any $1 \leq i \leq n$, we have $\rho \models \varphi_i$ if, and only if, $f_k(i) = 1$.*

We obtain

Proposition 7.26. $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,P)} \phi$

Proof. Let σ_t be a strategy for y_t, and let σ_u for y_u. Let τ_t be a strategy for x_t. Note that only the value $\tau_t(q_{ini} \cdot \sigma_t(q_{ini}))$ is important for ϕ since x_t is only applied jointly with y_t. Let $k \in \{1, 2\}$ be such that the state reached by applying $\tau_t(q_{ini} \cdot \sigma_t(q_{ini}))$ is labelled with p_k.

We then define the strategy τ_u for x_u by $\tau_u(q_{ini}.q_u) = q_{uk}$ and $\tau_u(q_{ini}.q_t) = q_{tk}$. Note that there is a potential side dependency of strategy τ_u w.r.t. τ_t. We now prove that, if χ is such that

$$\chi(y_t) = \sigma_t \qquad \chi(y_u) = \sigma_u \qquad \chi(x_t) = \tau_t \qquad \chi(x_u) = \tau_u$$

then:

$$\mathcal{G}, q_{ini} \models_\chi F^n(\beta_1\varphi_1, \ldots, \beta_n\varphi_n)$$

Fix an integer $1 \leq i \leq n$. Let $\star$ be t if $s(i) = 1$ and u otherwise. The outcome ρ^i of $\{\square \mapsto \sigma_\star, \bigcirc \mapsto \tau_\star\}$ (which corresponds to binding $\wp_i$) is $q_{ini}q_\star q_{\star k}$. Applying Lemma 7.25, we get that $\rho^i \models \phi_i$ if, and only if, $f_k(i) = 1$. This shows that $F^n(\beta_1\varphi_1, \ldots, \beta_n\varphi_n)$ is true, which concludes our proof[4]. □

As mentioned in the proof above, satisfaction of ϕ in $\mathcal{G}$ from q_{ini} uses strategies with side dependencies. We will now show that this is indeed required.

Proposition 7.27. $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$

Proof. Towards a contradiction, assume that ϕ is satisfied in $\mathcal{G}$ from q_{ini} using only strategies without side dependencies.

We let σ_t (resp. σ_u) be the strategy that maps history q_{ini} to q_t (resp. q_u). We fix strategy τ_t such that $\tau_t(q_{ini} \cdot q_t) = q_{t1}$. There is a strategy τ_u (without side dependencies) such that

$$\mathcal{G}, q_{ini} \models_\chi F^n(\beta_1\varphi_1, \ldots, \beta_n\varphi_n)$$

where χ maps y_t to σ_t, y to σ_u, x_t to τ_t and x_u to τ_u.

Since x_u is only jointly applied with y_u, the only important information about τ_u is its value on history $q_{ini}\sigma_u(q_{ini}) = q_{ini}q_u$. Because there is no side dependency, this value is independent on the value of $\tau_t(q_{ini}q_t) = \tau_t(q_{ini}\sigma_t(q_{ini}))$. In particular, writing χ' for the context obtained from χ by replacing $\chi(y_t) = \tau_t$ with τ'_t, where $\tau'_t(q_{ini}q_t) = q_{t2}$, we also have

$$\mathcal{G}, q_{ini} \models_{\chi'} F^n(\beta_1\varphi_1, \ldots, \beta_n\varphi_n)$$

Let v and v' be the elements in $\{0, 1\}^n$ representing the values of the goals $(\beta_1\varphi_1, \ldots, \beta_n\varphi_n)$ under χ and χ'. Then v and v' are in F^n. However:

- If $\tau_u(q_{ini}q_u) = q_{u1}$, then $v' = (f_1 \curlywedge \overline{s}) \curlyvee (f_2 \curlywedge s)$.
- If $\tau_t(q_{ini}q_t) = q_{t2}$, then $v = (f_1 \curlywedge s) \curlyvee (f_2 \curlywedge \overline{s})$.

In both cases, by hypothesis, this does not belong to F^n, which is a contradiction. □

[4]Note that there is no unordered prefix dependency and we could have shown $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,\emptyset)} \phi$.

Only one point remains,

Proposition 7.28. $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(S,\emptyset,P)} \neg\phi$

Proof. Consider a $\mathcal{M}(S,\emptyset,P)$ map $\overline{\theta}$ over $\exists y_t \exists y_u \exists x_t \forall x_u$. Fix a function $\overline{w} : (\mathsf{Hist} \to \mathsf{Act})^{x_u}$ and write ρ for the outcome of $\{ \to \overline{\theta}(\overline{w})(y_t); \to \overline{\theta}(\overline{w})(x_t)\}$. The choices of $\overline{\theta}$ on y_t, y_u and x_t do not depend on x_u, therefore ρ is independent of $\overline{w}$. Let $k \in \{1,2\}$ be such that ρ visits a state labelled p_k and write $\overline{w_k}$ for the function such that $\overline{w_k}(x_u)(q_{ini}.q_t) = q_{tk}$ and $\overline{w_k}(x_u)(q_{ini}.q_u) = q_{tk}$. Then the element v in $\{0,1\}^n$ representing the values of $(\beta_1\varphi_1, \dots \beta_n\varphi_n)$ under $\overline{\theta}(\overline{w_k})$ can either be

$$v = (f_1 \curlywedge \overline{s}) \curlyvee (f_1 \curlywedge s) = f_1$$
$$\text{or} \quad v = (f_2 \curlywedge s) \curlyvee (f_2 \curlywedge \overline{s}) = f_2$$

and both are in F^n. Therefore, $\overline{\theta}$ is not a witness of $\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,P)} \neg\phi$ and $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(S,\emptyset,P)} \neg\phi$ □

With Propositions 7.26, 7.27 and 7.28 we conclude the proof of Theorem 7.24. □

7.6 Conclusion

Theorem 7.1 shows that for any formula $\phi \in \mathsf{SL[EG]}$, when allowing unordered prefix dependencies and under the hypothesis that ϕ holds but $\neg\phi$ does not, we can remove side and future dependencies. We have also seen that this result is the best we can do. It would therefore be interesting to find a characterisation for the case when both a formula and its negation hold on a game. Using this characterisation and Theorem 7.1, we could then obtain a large class of inputs (game and formula) on which we can remove the side and future dependencies. This class would also admit a *2*-EXPTIME-complete model checking: all the steps in Theorem 7.1 are constructive and their combination gives an algorithm *2*-EXPTIME in the formula and PTIME in the game for the $\models^{\mathcal{M}(\emptyset,\emptyset,P)}$ satisfaction relation (see Section 7.4.6).

7.A Annex A

Lemma 7.17. *There exists a valuation χ of domain $\mathcal{V}$ such that $\theta(\chi_{|\mathcal{V}^\forall}) = \chi$ and $\overline{\theta}(\chi_{|\mathcal{V}^\exists}) = \chi$. Moreover χ satisfies*

$$\begin{aligned}
\mathcal{G}, q_{ini} \models_\chi \ \Gamma_{F^n} \quad &\Rightarrow \quad \forall w : (\mathit{Hist}_\mathcal{G} \to \mathit{Act})^{\mathcal{V}^\forall} \ \mathcal{G}, q_{ini} \models_{\theta(w)} F^n(\beta_j \varphi_j)_{1 \leq j \leq n} \\
\mathcal{G}, q_{ini} \models_\chi \ \neg\Gamma_{F^n} \quad &\Rightarrow \quad \forall \overline{w} : (\mathit{Hist}_\mathcal{G} \to \mathit{Act})^{\mathcal{V}^\exists} \ \mathcal{G}, q_{ini} \models_{\overline{\theta}(\overline{w})} \overline{F^n}(\beta_j \varphi_j)_{1 \leq j \leq n}
\end{aligned}$$

Proof. Both θ and $\overline{\theta}$ are $\mathcal{M}(\emptyset, \emptyset, P)$ maps, we can therefore apply the technique used in the proof of Theorem 6.5 to get a valuation χ such that $\theta(\chi_{|\mathcal{V}^\forall}) = \chi$ and $\overline{\theta}(\chi_{|\mathcal{V}^\exists}) = \chi$.

It remains to prove the two implications, we start by proving the first one. In the following, assume that

$$\mathcal{G}, q_{ini} \models_\chi \ \Gamma_{F^n} \tag{7.12}$$

We first fix some notations specific to this proof then prove some intermediary results.

Notations. *For a fixed parameter $w : (\mathit{Hist}_\mathcal{G} \to \mathit{Act})^{\mathcal{V}^\forall}$,*

- *we call π_j^w the outcome $\mathsf{out}(\beta_j(\theta(w)), q_{ini})$.*
- *we set f^w to be the $\{0,1\}^n$ element such that $f^w(j) = 1$ iff π_j^w satisfy φ_j.*
- *likewise to when we defined θ in Section 7.4.4, for any history ρ we call $s_{\rho,w}$ the $\{0,1\}^n$ element such that $s_{\rho,w}(j) = 1$ iff $\mathsf{out}(\beta_j(\theta(w)), q_{ini})$ follows ρ.*
- *finally, we define $\mathbb{R}^w \subseteq \{0,1\}^n \times \mathit{Hist}_\mathcal{G}$, the relation such that $(s, \rho) \in \mathbb{R}^w$ if and only if $s = s_{\rho,w}$ and ρ is minimal (meaning for any prefix ρ' of ρ, $(s, \rho') \notin \mathbb{R}^w$).*

Proposition 7.29. *For any $w : (\mathit{Hist}_\mathcal{G} \to \mathit{Act})^{\mathcal{V}^\forall}$, using the notations presented above, it holds*

$$\forall s \in \{0,1\}^n. \ \forall \rho \in \mathit{Hist}_\mathcal{G}. \quad (s, \rho) \in \mathbb{R}^w \ \Rightarrow \ b_{\mathsf{lst}(\rho), d_\rho, s} \preceq_s f^w$$

where $d_\rho := \mathsf{succ}(d_{ini}, \rho)$ (the vector of states accessible by ρ from the initial vector of states).

Proof. Fix some $w : (\mathsf{Hist}_\mathcal{G} \to \mathsf{Act})^{\mathcal{V}^\forall}$, we proceed by induction on the size of s from 1 to n.

The initial case ($|s| = 1$) Consider any history ρ such that $(s, \rho) \in \mathbb{R}^w$. As $|s| = 1$ and $(s, \rho) \in \mathbb{R}^w$, there is a unique goal, say $\beta_{j_0}\varphi_{j_0}$, following ρ. By definition of θ, $\pi_{j_0} = \rho.\eta$ where η is the outcome[5] obtained through $\beta_{j_0}\big(\varrho_{\mathsf{lst}(\rho), d_\rho, s}(w_{\overrightarrow{\rho}})\big)$ starting in $\mathsf{lst}(\rho)$.

Note that because $|s| = 1$, $\Gamma_{d_\rho, s, b_{\mathsf{lst}(\rho), d_\rho, s}} = \Gamma^{stick}_{d_\rho, s, b_{\mathsf{lst}(\rho), d_\rho, s}}$. The map $\varrho_{\mathsf{lst}(\rho), d_\rho, s}$ is a $\mathcal{M}(\emptyset, \emptyset, P)$ witness that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1 \leq i \leq l} \Gamma_{d_\rho, s, b_{\mathsf{lst}(\rho), d_\rho, s}}$, therefore it also witnesses that $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1 \leq i \leq l} \Gamma^{stick}_{d_\rho, s, b_{\mathsf{lst}(\rho), d_\rho, s}}$. By definition of the Γ^{stick} operators,

[5] which can be written in the following barbaric way: $\mathsf{out}(\beta_{j_0}(\varrho_{\mathsf{lst}(\rho), d_\rho, s}(w_{\overrightarrow{\rho}})), \mathsf{lst}(\rho))$

this implies that all its outcomes are accepted by the automaton $D^{d_\rho}_{s,b_{|\mathsf{st}(\rho),d_\rho,s}}$; in particular, η is accepted by $D^{d_\rho}_{s,b_{|\mathsf{st}(\rho),d_\rho,s}}$.

The automaton $D^{d_\rho}_{s,b_{|\mathsf{st}(\rho),d_\rho,s}}$ accepts paths which give better results for the objectives $(\beta_j\varphi_j)_{j|s(j)=1}$ than $b_{|\mathsf{st}(\rho),d_\rho,s}$. In our case this means that f^w does better than $b_{|\mathsf{st}(\rho),d_\rho,s}$ for s, i.e. $b_{|\mathsf{st}(\rho),d_\rho,s} \preceq_s f^w$.

The induction step ($|s| = \alpha$) We assume that the Proposition 7.29 holds for elements s of size $|s| < \alpha$. Consider for the induction step a history ρ such that $(s, \rho) \in \mathbb{R}^w$.

- Either there exists a common (infinite) path η such that for any j with $s(j) = 1$, $\pi_j = \rho.\eta$, i.e. all goals enabled by s always follow the same path $\rho.\eta$. We then apply the same reasoning as done in the initial case and deduce $b_{|\mathsf{st}(\rho),d_\rho,s} \preceq_s f^w$.

- Or, somewhere after ρ, the goals enabled by s split themselves along some extended partition $\tau = (s_\kappa, q_\kappa, d_\kappa)_{\kappa\leq\lambda}$. We call η the history from the last state of ρ to the point where the goals split from each other; formally η is obtained by applying $\beta_j\big(\varrho_{|\mathsf{st}(\rho),d_\rho,s}(w_{\overrightarrow{\rho}})\big)$ where j is such that $s(j) = 1$.

  We recall the notation for $c_{s,\tau}$ from Section 7.4.3

  $$c_{s,\tau} = (s_1 \curlywedge b_{q_1,d_1,s_1}) \curlyvee \ldots \curlyvee (s_\lambda \curlywedge b_{q_\lambda,d_\lambda,s_\lambda})$$

  The map $\varrho_{|\mathsf{st}(\rho),d_\rho,s}$ witnesses that $\mathcal{G}, |\mathsf{st}(\rho) \models^{\mathcal{M}(\emptyset,\emptyset,P)} \Gamma_{d,s,b_{|\mathsf{st}(\rho),d_\rho,s}}$, therefore η may reach only a partition τ such that

  $$b_{|\mathsf{st}(\rho),d_\rho,s} \preceq_s c_{s,\tau} \tag{7.13}$$

  For any $\kappa \leq \lambda$ we have $(s_\kappa, \rho.\eta.q_\kappa) \in \mathbb{R}^w$, and using the induction hypothesis we get

  $$s_\kappa \curlywedge b_{q_\kappa,d_\kappa,s_\kappa} \preceq_{s_\kappa} f^w \tag{7.14}$$

  so, using Lemma 7.16 (page 168) repeatedly on the $(s_\kappa)_{\kappa\leq\lambda}$ and Inequality 7.14, we obtain

  $$\begin{aligned}
  & s_1 \curlywedge b_{q_1,d_1,s_1} \preceq_{s_1} f^w \\
  \Rightarrow\ & (s_1 \curlywedge b_{q_1,d_1,s_1}) \curlyvee (s_2 \curlywedge b_{q_2,d_2,s_2}) \preceq_{s_1 \curlyvee s_2} f^w \\
  & \ldots \\
  \Rightarrow\ & (s_1 \curlywedge b_{q_1,d_1,s_1}) \curlyvee \ldots \curlyvee (s_\lambda \curlywedge b_{q_\lambda,d_\lambda,s_\lambda}) \preceq_{s_1 \curlyvee \ldots \curlyvee s_\lambda} f^w \\
  \Rightarrow\ & c_{s,\tau} \preceq_s f^w
  \end{aligned}$$

  Combined with Inequality 7.13, we get $b_{|\mathsf{st}(\rho),d_\rho,s} \preceq_s c_{s,\tau} \preceq_s f^w$.

This concludes the induction and the proof of Proposition 7.29. □

Proposition 7.30. $b_{q_{ini},d_{ini},\mathbf{1}} \in F^n$

Proof. Towards a contradiction, assume that $b_{q_{ini},d_{ini},\mathbf{1}} \in \overline{F^n}$. Then, by definition of the $b_{q_{ini},d_{ini},\mathbf{1}}$ element[6], $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1\leq i\leq l}\ \Gamma_{F^n}$. Applying this to Lemma 7.21, we have that the map $\overline{\varrho_{\mathbf{1}}}$ (and therefore $\overline{\theta}$ which act as $\overline{\varrho_{\mathbf{1}}}$ before goal goes on different paths) witness $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1\leq i\leq l}\ \Gamma_{F^n}$. This immediately implies that $\mathcal{G}, q_{ini} \not\models_{\chi} \Gamma_{F^n}$ which is in contradiction with the Hypothesis 7.12. □

With these preliminary results, we are now ready to prove the first implication of the lemma. Consider a function $w : (\mathsf{Hist}_{\mathcal{G}} \to \mathsf{Act})^{\mathcal{V}^{\forall}}$. By Proposition 7.29 applied to w, $\mathbf{1}$, ε we get that $b_{q_{ini},d_{ini},\mathbf{1}} \preceq_{\mathbf{1}} f^w$. Now by Proposition 7.30, $b_{q_{ini},d_{ini},\mathbf{1}} \in F^n$, therefore the element f^w which is greater than $b_{q_{ini},d_{ini},\mathbf{1}}$ for $\preceq_{\mathbf{1}}$ must also be in F^n, which is equivalent to $\mathcal{G}, q_{ini} \models_{\theta(w)} F^n(\beta_j \varphi_j)_{1\leq j\leq n}$.

The second implication of the lemma works similarly: produce an equivalent to Lemma 7.29 for $\overline{\theta}$, use the left side of the implication and Lemma 7.21 to get a counterpart to Proposition 7.30, and deduce the right hand side of the implication.

□

[6]See the definitions and explanations between Lemma 7.20 and Lemma 7.21 page 175

7.B Annex B

Lemma 7.23. *Consider s in $\{0,1\}^n$ and assume that we know the values of all $b_{q',d',s'}$ for all $s' \in \{0,1\}^n$ with $|s'| < |s|$, $q' \in Q$ and $d' \in VS$. We then have a 2-EXPTIME algorithm that computes*

- *the truth value of $\mathcal{G}, q \models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1\leq i\leq l} \Gamma_{d,s,h}$ for any $q \in Q$, $d \in VS$ and $h \in \{0,1\}^n$.*
- *the value of $b_{q,d,s}$ for any $d \in VS$ and $q \in Q$.*

Proof. The proof has many elements similar to the ones of Theorems 5.14 (page 127) and 6.5 (page 142), where we showed that $\models^{\mathcal{M}(S,F,\emptyset)}$, $\models^{\mathcal{M}(\emptyset,\emptyset,\emptyset)}$ and $\models^{\mathcal{M}(\emptyset,\emptyset,P)}$ are equivalent over SL[1G].

Checking the first point

We build a parity game $\mathcal{H}$ as we did in Theorem 5.14, only adapting the transitions and parities to $\Gamma_{d,s,h}$. We shorten the automaton $D^d_{s,h}$ to D in the following. Using the notion of clusters defined in the proof of Theorem 5.14 (page 127), we define $\mathcal{H}$ by

- there are two players: $P_\exists, P_\forall$.
- for each state q of $\mathcal{G}$ and each state d of D, $\mathcal{H}$ contains a copy of a cluster which we call the (q,d) *cluster*. For any $\mathfrak{m} \in \mathsf{Act}^*$ with $|\mathfrak{m}| \leq l$, we refer to the state $\mathfrak{m}$ of the (q,d) cluster as the $(q,d,\mathfrak{m})$ state. We also add two new states q_{even} and q_{odd}.
- the transitions in $\mathcal{H}$ are of three types:
  - internal transitions in the cluster are preserved;
  - consider a state $(q,d,\mathfrak{m})$ where $\mathfrak{m}$ is a leaf.
    - If there exists a state q' such that

$$\forall j \leq n \qquad s(j) = 1 \Rightarrow q' = \Delta(q, m_{\beta_j}) \quad \text{where} \begin{cases} m_{\beta_j} & : \mathsf{Agt} \to \mathsf{Act} \\ m_{\beta_j}(A) & = \mathfrak{m}(i-1) \\ \text{with } x_i & = \beta_j(A) \end{cases}$$

      I.e. applying the choices of $\mathfrak{m}$ according to β_j in $\mathcal{G}$ leads from q to q'. Then we add a transition from $(q,d,\mathfrak{m})$ to (q',d',ε) where $d' = \mathsf{succ}(d,q')$.
    - If there exists a partition τ of s such that for any $\kappa \leq |\tau|$

$$\Delta(q, m_{\beta_j}) = q_\kappa \quad \text{where} \begin{cases} m_{\beta_j} & : \mathsf{Agt} \to \mathsf{Act} \\ m_{\beta_j}(A) & = \mathfrak{m}(i-1) \\ \text{with } x_i & = \beta_j(A) \end{cases}$$

$$\mathsf{succ}(d, q_\kappa) = d_\kappa$$

$$b_{q,d,s} \preceq_s c_{s,\tau}$$

I.e. the goals enabled by s partition themselves from q based on $\mathfrak{m}$ along a partition whose c element is above $b_{q,d,s}$. Then we add a transition from $(q, d, \mathfrak{m})$ to q_{even}.

$*$ If there exists a partition τ such that

$$\Delta(q, m_{\beta_j}) = q_\kappa \quad \text{where} \begin{cases} m_{\beta_j} & : \mathsf{Agt} \to \mathsf{Act} \\ m_{\beta_j}(A) & = \mathfrak{m}(i-1) \\ \text{with } x_i & = \beta_j(A) \end{cases}$$

$$\mathsf{succ}(d, q_\kappa) = d_\kappa$$

$$c_{s,\tau} \prec_s b_{q,d,s}$$

I.e. the goals enabled by s partition themselves from q based on $\mathfrak{m}$ along a partition whose c element is below $b_{q,d,s}$. Then we add a transition from $(q, d, \mathfrak{m})$ to q_{odd}.

- the set of priorities are the same as in D and each $(q, d, \mathfrak{m})$ state has the same priority as d. We also label the states q_{even} and q_{odd} respectively by an even and an odd parity.

In the end, using the same arguments as the ones in the proof of Theorem 5.14, we get that if $P_\exists$ wins the parity game $\mathcal{H}$, we can build a $\mathcal{M}(\emptyset, \emptyset, \emptyset)$ map that witnesses $\mathcal{G}, q \models^{\mathcal{M}(\emptyset,\emptyset,\emptyset)} (Q_i x_i)_{1\leq i\leq l} \Gamma_{d,s,h}$ and therefore also witnesses $\mathcal{G}, q \models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1\leq i\leq l} \Gamma_{d,s,h}$. On the other hand if $P_\forall$ wins $\mathcal{H}$, there is a $\mathcal{M}(\emptyset, \emptyset, \emptyset)$ witness that $\mathcal{G}, q \models^{\mathcal{M}(\emptyset,\emptyset,\emptyset)} (Q_i x_i)_{1\leq i\leq l} \neg\Gamma_{d,s,h}$. Combining this with Theorem 6.5, we get[7] that $\mathcal{G}, q\neg \models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1\leq i\leq l} \Gamma_{d,s,h}$.

Checking the second point

We recall that an element $b_{q,d,s} \in \{0,1\}^n$ is such that

1. $\mathcal{G}, q \models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1\leq i\leq l}\ \Gamma_{d,s,b_{q,d,s}}$
2. for any $h \in \{0,1\}^n$ with $b_{q,d,s} \prec_s h$, we have $\mathcal{G}, q \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1\leq i\leq l}\ \Gamma_{d,s,h}$

We can check $\mathcal{G}, q \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} (Q_i x_i)_{1\leq i\leq l}\ \Gamma_{d,s,h}$ for all $h \in \{0,1\}^n$. The element $b_{q,d,s}$ is the maximal h relatively to $\preceq_s$ for which it holds. The algorithm for the second point derives from the algorithm of the first point.

□

[7] Rigorously, Theorem 6.5 is not proved for formulas with the new operator $\Gamma_{d,s,h}$. The proof however also works without modifications.

Conclusion

In this thesis, we investigated the complexity and the semantics of Strategy Logic (SL). Unlike others temporal logics for multi-agent systems, SL dissociates strategies from players. This dissociation is one of the keys of SL expressiveness; for example, a strategy δ can be created at a time t and used at a later time t'. Another important aspect of SL is the possibility for complex interplay between the different executions of the system. In ATL*, there are no strategy-sharing aspects between two executions of the system; the executions are totally dissociated from one another. SL on the other hand allows an agent to share its strategy in two executions. SL is therefore a very expressive logic and a good framework to reason about temporal properties of multi-agents systems.

Complexity

As always, the large expressive power of SL leads to poor complexity results. Mogavero, Murano, Perelli and Vardi showed that SL admits an undecidable satisfiability problem and a decidable but NONELEMENTARY model checking problem; we present their algorithm in Chapter 2. We also complete the picture with our own results: SL is at least PH-hard with respect to the size of the game and the SL[BG] fragment cannot admit an ELEMENTARY algorithm (thus the complexity of SL[BG] is no better than the one of SL). Some gaps remain in SL complexity when considering formulas with a fixed number of alternations and matching the lower and upper bound would greatly improve the understanding of SL. In particular, results on the complexity relative to the size of the game would precise the usability of SL for practical cases; until then SL remains mainly a theoretical framework.

The time between creation and usage

Sometimes in temporal verification, a strategy δ must be created at a time t and used at a later time t'. At a time $t'' > t'$, SL semantics then considers that δ must have knowledge of the history between t' and t'' but also between t to t'. In Chapter 3, we argued that the situation where δ is deprived the knowledge of what happened between t and t' is more adequate to model server/client interactions. For this, we created an alternative version FSL of SL and study in depth its complexity. Theorems 3.2 and 3.13 show that a large part of the logic unfortunately becomes undecidable under the new semantics.

In the end, the choice of the logic depends on the practical problem. A property where a strategy δ is created and used at different times should be modelled in SL if the

knowledge between creation and usage is available to δ, and in FSL if this knowledge is not available to δ. If all strategies are created and used at the same time, the fragment SL[BG] (where the two semantics intersect) is probably the best candidate.

Strategy dependencies

Most extensions of ATL* are subject to knowledge-based problems: about the goals of other agents, about the histories on which strategies are evaluated, about the other agents strategies... Along Chapters 5 to 7, we investigated the degree of knowledge needed about the universally quantified strategies in order to build the ones which are existentially quantified. There exist many subtleties and one must be careful when dealing with the dependencies between strategies choices. For example, on our journey to cartography dependencies, we found that adding all the actions played on the current history complicates the situation greatly. In some case, it allows to transfer informations between strategies. In particular, Theorems 6.15, 6.16 and 6.17) show that one can locally override the order of the quantifications. The result not only holds for SL[BG], but also for all extensions of ATL* that allow three or more quantifier alternations.

In simple requests, such as ω-regular conditions or ATL* formulas, the strategies only need local knowledge. In systems where choices are made locally (within a state), we must forcefully forbid any dependency, otherwise a specification could be erroneously interpreted as true. On the other hand, when a strategy obeys a known and public protocol, other strategies can depend on it on its entirety. Finally, global objectives must allow a total dependence between the strategies. Consider the scenario of an electronic attack on a system. The attacker may repeat the attack numerous times, gathering information about the system. He may then guess the protocol in place. To model such knowledge, the strategies of the attack must have a total knowledge of the strategies of the system to the point of overriding the order of the quantifications.

Long term perspectives

Among the challenges of this thesis, the dependency problem between strategies stands out. Many logics are introduced in the literature, but semantical aspects are often omitted, and only the decidability of the satisfiability and model checking problems are considered. A deeper look at SL revealed many subtleties and all extensions of ATL* have taken the same approach to dependencies: everything is allowed as long as it does not mess the order of the quantifiers. I hope that the second part of this thesis gives insight into these subtleties, and is convincing the reader that semantical aspects are really important.

The framework proposed in Chapters 5 and 6 has its merits: it treats the quantifications as a compact block and allows to bypass the order of the quantifications. However, it also suffers from severe drawbacks: the syntactic negation is completely dissociated from the semantic negations. Indeed, the semantic negation swaps the role of existential and universal strategies but it does not change the knowledge they have about each other; the syntactic negation reverses both. As shown in Chapter 7, the syntactic negation plays

an important role in the dependency problem. It would be worth developing a framework better at handling the two notions of negations. Potential answers could be found in the works on the independence-friendly logic and the dependence logic.

Another imperfection of the framework is that universally quantified strategies are allowed an important degree of knowledge by the universal quantification on all functions of the form $(\mathsf{Hist} \to \mathsf{Act})^{\nu^\forall}$. In the manuscript, we do not investigate what happens when both universal and existential strategies are severely limited in their knowledge of other strategies. A deeper look into this issue would complete our work in an appropriate manner.

Bibliography

[1] Thomas Ågotnes, Valentin Goranko, and Wojciech Jamroga. Alternating-time temporal logics with irrevocable strategies. In *Proceedings of the 11th conference on Theoretical aspects of rationality and knowledge*, pages 15–24. ACM, 2007.

[2] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, September 2002.

[3] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[4] Aaron Bohy, Véronique Bruyère, Emmanuel Filiot, and Jean-François Raskin. Synthesis from ltl specifications with mean-payoff objectives. In *Proceedings of the 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'13, pages 169–184, Berlin, Heidelberg, 2013. Springer-Verlag.

[5] Ahmed Bouajjani, Marius Bozga, Peter Habermehl, Radu Iosif, Pierre Moro, and Tomáš Vojnar. Programs with lists are counter automata. In *International Conference on Computer Aided Verification*, pages 517–531. Springer, 2006.

[6] Patricia Bouyer, Uli Fahrenberg, Kim G. Larsen, Nicolas Markey, and Jiří Srba. *Infinite Runs in Weighted Timed Automata with Energy Constraints*, pages 33–47. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[7] Patricia Bouyer, Patrick Gardy, and Nicolas Markey. Weighted strategy logic with boolean goals over one-counter games. In Prahladh Harsha and G. Ramalingam, editors, *Proceedings of the 35th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'15)*, volume 45, pages 69–83, Bangalore, India, December 2015.

[8] Patricia Bouyer, Patrick Gardy, and Nicolas Markey. On the semantics of strategy logic. *Information Processing Letters*, 116(2):75–79, 2016.

[9] Tomáš Brázdil, Petr Jančar, and Antonín Kučera. Reachability games on extended vector addition systems with states. In *International Colloquium on Automata, Languages, and Programming*, pages 478–489. Springer, 2010.

[10] Thomas Brihaye, Arnaud Da Costa, François Laroussinie, and Nicolas Markey. *ATL with Strategy Contexts and Bounded Memory*, pages 92–106. Springer Berlin Heidelberg, 2009.

[11] Thierry Cachat. Tree automata make ordinal theory easy. In *FSTTCS06*, volume 4337 of *LNCS*, pages 285–296. Springer, 2006.

[12] Olivier Carton and Chloé Rispal. Complementation of rational sets on scattered linear orderings of finite rank. *Theoretical Computer Science*, 382(2):109–119, 2007.

[13] Krishnendu Chatterjee and Laurent Doyen. Energy parity games. *Theoretical Computer Science*, 458:49–60, 2012.

[14] Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Generalized Mean-payoff and Energy Games. In *FSTTCS 2010*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 505–516, 2010.

[15] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Strategy logic. *Information and Computation*, 208(6):677 – 693, 2010.

[16] Edmund M. Clarke and E. Allen Emerson. *Design and synthesis of synchronization skeletons using branching time temporal logic*, pages 52–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 1982.

[17] René Cori and Daniel Lascar. *Mathematical Logic: Recursion theory, Gödel's theorems, set theory, model theory*. OUP Oxford, 2001.

[18] Arnaud Da Costa, François Laroussinie, and Nicolas Markey. *Quantified CTL: Expressiveness and Model Checking*, pages 177–192. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[19] René David and Hassane Alla. Petri nets for modeling of dynamic systems: A survey. *Automatica*, 30(2):175–202, 1994.

[20] Manfred Droste and Paul Gastin. Weighted automata and weighted logics. *Theoretical Computer Science*, 380(1):69 – 86, 2007.

[21] E Allen Emerson and Joseph Y Halpern. 'sometimes' and 'not never' revisited: on branching versus linear time temporal logic. *Journal of the ACM (JACM)*, 33(1):151–178, 1986.

[22] U.S-Canada Power System Outrage Task Force. Final report on the august 14, 2003 blackout in the united states and canada. causes and recommendations. Technical report, DoE and Ministry of Natural Resources Canada, 2004.

[23] Paul Gastin and Denis Oddoux. Fast ltl to büchi automata translation. In *Computer Aided Verification: 13th International Conference, CAV 2001 Paris, France, July 18–22, 2001 Proceedings*, pages 53–65, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[24] Rob Gerth, Doron Peled, Moshe Y. Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Proceedings of the Fifteenth IFIP WG6.1 International Symposium on Protocol Specification, Testing and Verification XV*, pages 3–18, London, UK, UK, 1996. Chapman & Hall, Ltd.

[25] Stefan Goller and Markus Lohrey. Branching-time model checking of one-counter processes and timed automata. *SIAM Journal on Computing*, 42(3):884–923, 2013.

[26] Christoph Haase, Stephan Kreutzer, Joël Ouaknine, and James Worrell. Reachability in succinct and parametric one-counter automata. In *International Conference on Concurrency Theory*, pages 369–383. Springer, 2009.

[27] Thomas A. Henzinger, Benjamin Horowitz, and Rupak Majumdar. *Rectangular Hybrid Games*, pages 320–335. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[28] Kenneth Kunen. *Set Theory*. Elsevier, 1980.

[29] Orna Kupferman and Moshe Y. Vardi. Weak alternating automata and tree automata emptiness. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 224–233. ACM, 1998.

[30] Francois Laroussinie and Nicolas Markey. Quantified CTL: Expressiveness and Complexity. *Logical Methods in Computer Science*, Volume 10, Issue 4, December 2014.

[31] Francois Laroussinie and Nicolas Markey. Augmenting atl with strategy contexts. *Information and Computation*, 245:98 – 123, 2015.

[32] Nancy G. Leveson and Clark S. Turner. An investigation of the therac-25 accidents. *Computer*, 26(7):18–41, July 1993.

[33] Leonid Libkin. *Elements of finite model theory*. Springer Science & Business Media, 2013.

[34] Inquiry Board (Chairman: Prof. Lacques-Louis LIONS). Ariane 5 flight 501 failure. Technical report, CNES (centre national d'etude spatiales) and ESA (european space agency), 1996.

[35] Allen L Mann, Gabriel Sandu, and Merlijn Sevenster. *Independence-friendly logic: A game-theoretic approach*, volume 386. Cambridge University Press, 2011.

[36] Robin Milner. *Operational and algebraic semantics of concurrent processes*, chapter Operational and Algebraic Semantics of Concurrent Processes, pages 1201–1242. MIT Press, Cambridge, MA, USA, 1990.

[37] Marvin L Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Inc., 1967.

[38] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi. What makes atl* decidable? a decidable fragment of strategy logic. In *Proceedings of the 23rd International Conference on Concurrency Theory*, CONCUR'12, pages 193–208, Berlin, Heidelberg, 2012. Springer-Verlag.

[39] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi. Reasoning about strategies: On the model-checking problem. *ACM Trans. Comput. Logic*, 15(4):34:1–34:47, November 2014.

[40] Fabio Mogavero, Aniello Murano, and Luigi Sauro. On the boundary of behavioral strategies. In *Proceedings of the 28th Annual Symposium on Logic in Computer Science (LICS'13)*, pages 263–272. IEEE Comp. Soc. Press, June 2013.

[41] Fabio Mogavero, Aniello Murano, and Luigi Sauro. A behavioral hierarchy of strategy logic. In *International Workshop on Computational Logic and Multi-Agent Systems*, pages 148–165. Springer, 2014.

[42] Martin J Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 1994.

[43] Nir Piterman. From nondeterministic buchi and streett automata to deterministic parity automata. In *21st Annual IEEE Symposium on Logic in Computer Science (LICS'06)*, pages 255–264. IEEE, 2006.

[44] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, pages 46–57, Washington, DC, USA, 1977. IEEE Computer Society.

[45] Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 179–190. ACM, 1989.

[46] Charles Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6(2):223 – 231, 1978.

[47] Julien Reichert. On the complexity of counter reachability games. *Fundamenta Informaticae*, 143(3-4):415–436, 2016.

[48] Stéphane Riedweg and Sophie Pinchinat. Quantified mu-calculus for control synthesis. In *International Symposium on Mathematical Foundations of Computer Science*, pages 642–651. Springer, 2003.

[49] Gareth Scott Rohde. *Alternating automata and the temporal logic of ordinals*. PhD thesis, University of Illinois at Urbana-Champaign, 1997.

[50] Sylvain Schmitz. Complexity hierarchies beyond elementary. *ACM Trans. Comput. Theory*, 8(1):3:1–3:36, February 2016.

[51] Olivier Serre. Parity games played on transition graphs of one-counter processes. In *International Conference on Foundations of Software Science and Computation Structures*, pages 337–351. Springer, 2006.

[52] Michael Sipser. *Introduction to the Theory of Computation*, volume 2. Thomson Course Technology Boston, 2006.

[53] Aravinda P. Sistla. *Theoretical Issues in the Design and Verification of Distributed Systems*. PhD thesis, Harvard University, Cambridge, MA, USA, 1983. AAI8403047.

[54] Aravinda P. Sistla, Moshe Y. Vardi, and Pierre Wolper. The complementation problem for buchi automata with applications to temporal logic. *Theoretical Computer Science*, 49(2):217 – 237, 1987.

[55] Colin Stirling. *Modal and temporal logics for processes*, pages 149–237. Springer Berlin Heidelberg, 1996.

[56] Wolfgang Thomas. Languages, automata, and logic. In *Handbook of formal languages*, pages 389–455. Springer, 1997.

[57] Wolfgang Thomas, Thomas Wilke, et al. *Automata, logics, and infinite games: a guide to current research*, volume 2500. Springer Science & Business Media, 2002.

[58] Jouko Väänänen. *Dependence logic: A new approach to independence friendly logic*, volume 70. Cambridge University Press, 2007.

[59] Wiebe van der Hoek, Wojciech Jamroga, and Michael Wooldridge. A logic for strategic reasoning. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 157–164. ACM, 2005.

[60] Moshe Y. Vardi. *An automata-theoretic approach to linear temporal logic*, pages 238–266. Springer Berlin Heidelberg, 1996.

[61] Steen Vester. *On the Complexity of Model-Checking Branching and Alternating-Time Temporal Logics in One-Counter Systems*, pages 361–377. Springer International Publishing, Cham, 2015.

[62] Igor Walukiewicz. Pushdown processes: Games and model-checking. *Information and computation*, 164(2):234–263, 2001.

[63] Farn Wang, Chung-Hao Huang, and Fang Yu. A temporal logic for the interaction of strategies. In *International Conference on Concurrency Theory*, pages 466–481. Springer, 2011.

[64] Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1):343–359, 1996.

Index

Glossary

$f \curlywedge g$ Function of $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ such that $f \curlywedge g(i) = 1$ if and only if $f(i) = 1$ and $g(i) = 1$.

$f \curlyvee g$ Function of $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ such that $f \curlyvee g(i) = 1$ if and only if $f(i) = 1$ or $g(i) = 1$.

$\pi \cdot \rho$ Singular concatenation of two histories π and ρ.

$\Pi_{i \in I}(\pi_i)$ Concatenation of the histories $(\pi_i)_{i \in I}$ following the order on I.

$\cdot^\flat$ Flat fragment of....

$\delta_{\vec{\pi}}$ Strategy obtained by $\delta_{\vec{\pi}}(\rho) = \delta(\pi.\rho)$.

$\chi_{\vec{\pi}}$ Valuation obtained by $\chi_{\vec{\pi}}(x) = \chi(x)_{\vec{\pi}}$ for any agent or variable x.

$\chi_{\underrightarrow{\pi}}$ Valuation obtained by $\chi_{\vec{\pi}}(x) = \chi(x)_{\vec{\pi}}$ for any agent x.

$\uparrow B$ Upward closure of a set B (of $\mathbb{N}$ or $\{0,1\}^n$).

Act Set of all actions.

Agt Set of all agents.

AP Set of all atomic propositions.

$\mathsf{dom}(\chi)$ Domain of definition of χ.

flip_b Function of $\{0,1\}^n \to \{0,1\}^n$ such that $\mathsf{flip}_b(f) = (f \curlywedge b) \curlyvee (\overline{f} \curlywedge \overline{b})$.

$\mathsf{free}(\phi)$ Free variables and free agents of ϕ.

$\mathsf{Hist}_{\mathcal{G}}$ Set of all histories in $\mathcal{G}$.

$\mathsf{Inf}(\pi)$ Set of states appearing infinitely often in π.

$\mathcal{L}(\mathcal{N})$ Language of $\mathcal{N}$ (set of all words accepted by $\mathcal{N}$).

labels Labelling function.

$\mathsf{lst}(\pi)$ Last state of π.

$\mathsf{out}(\chi, q)$ Outcome produced by χ starting from q.

$\mathsf{Part(s)}$ Set of all partitions of s (as defined in the proof of Theorem 7.1).

$\mathsf{Path}_{\mathcal{G}}$ Set of all paths in $\mathcal{G}$.

$\mathsf{Pref}_{\leq\rho}$ Set of all prefix of ρ including ρ..

$\mathsf{Pref}_{<\rho}$ Set of all prefix of ρ excluding ρ..

$\mathsf{Shift}_{\nu}(\pi)$ History obtained by shifting the counter values in π by a factor ν.

$\mathsf{Strat}_{\mathcal{G}}$ Set of all strategies possible in $\mathcal{G}$.

$\mathsf{Tower}(a, b)$ Function returning a tower of exponential of height b and input a.

$\mathcal{V}^{\exists}$ Set of existentially quantified variables (of a formula or a quantifier block).

$\mathcal{V}^{\forall}$ Set of universally quantified variables (of a formula or a quantifier block).

VS Set of vector of states in the proof of Theorem 7.1.

$\mathsf{Weights}$ Weight function of a WCGS.

Résumé Substentiel en français

Introduction

Les systèmes électroniques et numériques ont envahi nos vies: administration, armée, éducation, énergie, médias, santé, transports. L'augmentation du nombre de ces systèmes créé de nombreux défis, des plus connus comme la cryptographie ou les problèmes de confidentialité aux plus obscurs comme les problèmes de concurrence ou de compteur...

Il existe trois principales méthodes pour assurer du bon comportement de ces systèmes: tester, trouver un certificat de bon fonctionnement et explorer l'arbre des possibilités. Cette thèse se concentre principalement sur cette dernière approche. Il convient alors de se poser deux questions: "Qu'est ce qu'un bon comportement de mon système ?" (spécification) et "Mon système peut-il sortir de ce comportement ?" (vérification). Le travail du chercheur est alors de développer un cadre de travail capable de s'appliquer à un grand nombre de cas pratiques, il serait coûteux de redévelopper un algorithme pour chaque système. Une fois le cadre fixé, la procédure (connue sous le nom de "model-checking") pour vérifier le comportement d'un système est la suivante:

1. Spécifier le système dans un modèle mathématique (représentation formelle).
2. Transcrire la propriété en une formule d'un formalisme connu (une condition qui approuve ou invalide l'exécution du système).
3. Vérifier que la formule est vraie dans le modèle en explorant les diverses possibilités. On retrouve ici les questions autour de l'efficacité des algorithmes.
4. Transposer la réponse de l'étape précédente au cas pratique.

Systèmes multi-agents

La vérification de systèmes monolithiques ou de systèmes à deux agents (un agent représentant le système électronique et l'autre représentant l'environnement) est relativement bien comprise. Le cas des systèmes multi-agents est en revanche moins bien compris. La modélisation des systèmes multi-agents se fait généralement par des jeux sur des graphes, chaque agent est alors représenté par un joueur. Dans cette thèse, nous considérons des systèmes avec un nombre fini mais arbitrairement grand d'agents. La difficulté est alors d'incorporer les objectifs de chaque agent dans un système global, cohérent et sans dysfonctionnement. La manière dont les différents objectifs des différents agents limitent leurs comportements et impactent le système seront au centre de notre étude.

Plan du manuscrit

Cette thèse se concentre sur l'étude du formalisme connu sous le nom de "Strategy Logic" (SL). Cette logique est un langage de premier ordre pour raisonner sur les liens unissant plusieurs exécutions d'un même système possédant un nombre arbitraire (mais fini) d'agents. Notre travail tourne autour de deux problèmes:

- trouver des algorithmes pour le model-checking (étant donnés un système et une formule, la formule est-t-elle vérifiée sur le jeu ?).
- étudier les choix sémantiques faits dans la définition de SL, ainsi que leur conséquences.

La thèse est organisé en deux partie regroupant respectivement quatre et trois chapitres. La première partie présente et explore SL dans ses grandes lignes. La seconde se concentre sur le problème des dépendances entre stratégies dans un fragment (SL[BG]) de SL.

Le premier chapitre présente les différents formalismes présents dans la littérature et utilisés pour la vérification de systèmes multi-agents. On y retrouve une définition formelle de SL et de ses principaux fragments. Le second chapitre explore la complexité des problèmes de satisfiabilité et de model-checking de SL. Dans le troisième chapitre, nous nous intéressons aux problème d'historique qui apparaissent quand une stratégie est créée à un instant t mais utilisée seulement à un instant t' (avec $t' > t$). Enfin, dans le quatrième chapitre, nous explorons les versions quantitatives de SL, avec compteurs et énergies.

Les chapitres de la seconde partie (cinq à sept) sont exclusivement consacrée aux problèmes de dépendances entre stratégies. Partant d'une quantification "Pour tout x_1, il existe x_2", il s'agit de cartographier les choix de x_1 que x_2 a besoin de connaître pour répondre de manière appropriée.

Part 1: "Strategy Logic"

ATL* et Strategy Logic (chapitre 1)

Il existe de nombreux formalismes pour décrire les systèmes multi-agents: ATL, ATL*, BSIL, IATL, ATL$_{sc}$, GL ... Cependant, dans cette thèse nous nous intéressons à un seul d'entre eux: la Strategy Logic (SL).

Modèles pour systèmes multi-agents

Définition (Jeux concurrents)**.**
Un jeu concurrent ("concurrent game structure", CGS) est un n-uplet $\mathcal{G} := \langle AP, Agt, Q, Act, \Delta, labels\rangle$ *où* AP *est un ensemble non-vide et fini de propositions atomiques,* Agt *est un ensemble non-vide et fini d'agents,* Q *est un ensemble non-vide et fini d'états,* Act *est un ensemble non-vide d'actions,* $\Delta : Q \times Act^{Agt} \to Q$ *est une fonction de transition et* $labels : Q \to 2^{AP}$ *une fonction d'étiquetage.*

L'exécution du système est alors représentée en posant une pierre sur un état initial et en la faisant bouger dans le jeu. Au temps t, chaque agent joue une action et l'ensemble des actions jouées définit un mouvement qui déplace la pierre d'état en état. La suite des états visités représente alors l'évolution du système.

Définition (Historiques et chemins)**.**
Un historique dans un jeu $\mathcal{G} := \langle AP, Agt, Q, Act, \Delta, labels\rangle$ *est une suite finie d'états* $(q_i)_{i\leq L}$ *(*$L \in \mathbb{N}$*) telle que pour tout* $i < L$*, on peut trouver* $d \in Act^{Agt}$ *avec* $\Delta(q_i, d) = q_{i+1}$*.*
Un chemin dans $\mathcal{G}$ *est une suite infinie d'états* $(q_i)_{i\in\mathbb{N}}$ *tel que pour tout* $i \in \mathbb{N}$*, il existe* $d \in Act^{Agt}$ *avec* $\Delta(q_i, d) = q_{i+1}$*.*

Pour représenter les comportements des différents agents, nous utilisons la notion de stratégie. Pour cette thèse, les stratégies seront déterministes et auront une mémoire infinie.

Définition (Strategy)**.**
Une strategie dans un jeu $\mathcal{G} := \langle AP, Agt, Q, Act, \Delta, labels\rangle$ *est une fonction (potentiellement partielle)* $\delta : Hist_{\mathcal{G}} \to Act$ *qui associe une action à chaque historique.*

Une valuation (sur un ensemble Agt d'agents et un ensemble $\mathcal{V}$ de variables) attribue une stratégie à chaque variable ou agent.

Définition (Un résultat).
Soit $\mathcal{G} := \langle AP, Agt, Q, Act, \Delta, labels\rangle$ *un CGS,* q *un de ses états et* χ *une valuation. Le résultat* $out(\chi, q) := (q_i)_{i\in\mathbb{N}^*}$ *de* χ *depuis* q *est l'unique chemin avec* $q_1 = q$ *et* $\forall i \in \mathbb{N}^*$, $q_{i+1} = \Delta(q_i, d_i)$, *où* $d_i \in Act^{Agt}$ *est une fonction qui assigne une action à chaque agent en suivant* χ *sur l'historique* $(q_{i'})_{i'<i}$.

La Strategy Logic

SL est construit sur un ensemble d'agents Agt, un ensemble de propositions atomiques AP et un ensemble de variables $\mathcal{V}$. Sa grammaire est la suivante:

$$\mathsf{SL} \ni \phi \ ::= \ \exists x.\phi \mid \mathsf{assign}(A, x).\phi \mid \phi \vee \phi \mid \neg\phi \mid \phi \mathbf{U}\, \phi \mid \mathbf{X}\, \phi \mid p$$

avec $x \in \mathcal{V}$ une variable, $A \in \mathsf{Agt}$ un agent et $p \in \mathsf{AP}$ une proposition atomique.

Les opérateurs $\mathsf{assign}(A, x)$ et $\exists x$ sont respectivement appelés *assignation* de x à A et quantification existentielle de x.

Avant de définir la sémantique de SL, nous devons parler des variables et agents libres d'une formule. L'ensemble des variables et agents libres dans une formule ϕ est noté $\mathsf{free}(\phi)$ et représente les éléments qui n'ont pas encore de stratégies. $\mathsf{free}(\phi)$ est définie inductivement par

$$\begin{aligned}
\mathsf{free}(p) &= \varnothing \quad \text{pour tout } p \in \mathsf{AP} & \mathsf{free}(\mathbf{X}\, \phi) &= \mathsf{Agt} \cup \mathsf{free}(\phi) \\
\mathsf{free}(\neg\phi) &= \mathsf{free}(\phi) & \mathsf{free}(\phi\, \mathbf{U}\, \psi) &= \mathsf{Agt} \cup \mathsf{free}(\phi) \cup \mathsf{free}(\psi) \\
\mathsf{free}(\phi \vee \psi) &= \mathsf{free}(\phi) \cup \mathsf{free}(\psi) & \mathsf{free}(\exists x.\ \phi) &= \mathsf{free}(\phi) \setminus \{x\}
\end{aligned}$$

$$\mathsf{free}(\mathsf{assign}(A, x).\ \phi) = \begin{cases} \mathsf{free}(\phi) & \text{si } A \notin \mathsf{free}(\phi) \\ (\mathsf{free}(\phi) \cup \{x\}) \setminus \{A\} & \text{sinon} \end{cases}$$

Quand $\mathsf{free}(\phi) = \varnothing$, on parle alors de formule fermée.

Les formules de SL sont évaluées sur un CGS $\mathcal{G} := \langle \mathsf{AP}, \mathsf{Agt}, \mathsf{Q}, \mathsf{Act}, \Delta, \mathsf{labels}\rangle$ à un état q relativement à une valuation χ (et tel que les ensembles d'agents et de propositions atomiques de la formule coïncident avec ceux du CGS).

$$\begin{aligned}
\mathcal{G}, q \models_\chi p &\quad\Leftrightarrow\quad p \in \mathsf{labels}(q) \\
\mathcal{G}, q \models_\chi \phi \vee \phi' &\quad\Leftrightarrow\quad \mathcal{G}, q \models_\chi \phi \text{ ou } \mathcal{G}, q \models_\chi \phi' \\
\mathcal{G}, q \models_\chi \neg\phi &\quad\Leftrightarrow\quad \mathcal{G}, q \not\models_\chi \phi
\end{aligned}$$

Si $\mathsf{free}(\phi) \setminus \{x\} \subseteq \mathsf{dom}(\chi)$, alors

$$\mathcal{G}, q \models_\chi \exists x.\phi \quad\Leftrightarrow\quad \exists \delta \in \mathsf{Strat}_\mathcal{G} \text{ tel que } \mathcal{G}, q \models_{\chi[x\mapsto\delta]} \phi$$

De plus, étant donné $A \in \mathsf{Agt}$, si $(\mathsf{free}(\phi) \setminus \{A\}) \cup \{x\} \subseteq \mathsf{dom}(\chi)$ alors

$$\mathcal{G}, q \models_\chi \mathsf{assign}(A, x).\phi \quad\Leftrightarrow\quad \mathcal{G}, q \models_{\chi[A\mapsto\chi(x)]} \phi$$

Si $\mathsf{Agt} \cup \mathsf{free}(\phi) \cup \mathsf{free}(\psi) \subseteq \mathsf{dom}(\chi)$ alors χ produit un chemin résultat π depuis q, i.e. $\pi := \mathsf{out}(\chi, q)$. Pour tout entier j, on note $\chi_{\overrightarrow{j}}$ la valuation $\chi_{\overrightarrow{\pi_{\leq j}}}$. Ensuite,

$$\begin{array}{lcl} \mathcal{G}, q \models_\chi \mathbf{X}\,\phi & \Leftrightarrow & \mathcal{G}, \mathsf{out}(\chi, q)(1) \models_{\chi_{\overrightarrow{1}}} \phi \\ \mathcal{G}, q \models_\chi \phi \mathbf{U}\,\phi' & \Leftrightarrow & \exists k \in \mathbb{N}.\ \mathcal{G}, \mathsf{out}(\chi, q)(k) \models_{\chi_{\overrightarrow{k}}} \phi' \text{ et} \\ & & \forall j \in \mathbb{N}.\ 0 \leq j < k \Rightarrow \mathcal{G}, \mathsf{out}(\chi, q)(j) \models_{\chi_{\overrightarrow{j}}} \phi \end{array}$$

Les fragments (plats) SL[NG] et SL[BG]

À travers la thèse, nous étudierons aussi des fragments de SL. Le premier fragment d'importance est le fragment SL[NG] avec des objectifs imbriqués (*Nested Goals*). Une formule de SL[NG] regroupera les quantifications par blocs de sorte qu'un nouveau bloc marque le début d'une formule fermée.

Définition (SL avec objectifs imbriqués, SL[NG])**.**

$$\begin{array}{rcl} \mathit{SL[NG]}^\flat \ni \phi & ::= & \exists x.\phi \mid \forall x.\phi \mid \xi \\ \xi & ::= & \xi \vee \xi \mid \xi \wedge \xi \mid \beta \\ \beta & ::= & \mathit{assign}(A, x).\beta \mid \varphi \\ \varphi & ::= & \varphi \vee \varphi \mid \neg\varphi \mid \varphi \mathbf{U}\,\varphi \mid \mathbf{X}\,\varphi \mid p \mid \beta \end{array}$$

avec $x \in \mathcal{V}$ une variable, $A \in \mathsf{Agt}$ un agent et $p \in \mathsf{AP}$ une proposition atomique.

Les fameux objectifs de SL[NG] sont les sous-formules de type β. Une formule de ce fragment est alors décomposable en un bloc de quantifications suivies d'une combinaison booléenne d'objectifs (potentiellement imbriqués).

Si l'on enlève la possibilité d'imbriquer les objectifs, on obtient le fragment à objectif booléen, SL[BG] (*Boolean Goals*).

Définition (SL avec objectifs booléens, SL[BG])**.**

$$\begin{array}{rcl} \mathit{SL[BG]}^\flat \ni \phi & ::= & \exists x.\phi \mid \forall x.\phi \mid \xi \\ \xi & ::= & \xi \vee \xi \mid \xi \wedge \xi \mid \beta \\ \beta & ::= & \mathit{assign}(A, x).\beta \mid \varphi \\ \varphi & ::= & \varphi \vee \varphi \mid \neg\varphi \mid \varphi \mathbf{U}\,\varphi \mid \mathbf{X}\,\varphi \mid p \end{array}$$

On peut aussi restreindre les combinaisons booléennes et obtenir de nouveaux fragments: SL[1G] avec un unique objectif, SL[CG] qui n'autorise que des conjonctions d'objectifs, ou encore SL[DG] qui n'autorise que des disjonctions d'objectifs.

Complexité (chapitre 2)

Dans ce chapitre, nous présentons des résultats de complexité sur SL (certains développés dans cette thèse, d'autres venant de différents auteurs de la communauté scientifique). Il existe deux questions principales:

Définition (Satisfiabilité)**.**
Le problème de satisfiabilité d'une logique $\mathcal{L}$ demande l'existence d'un algorithme qui prend en entrée une formule $\phi \in \mathcal{L}$ et qui répond est-ce qu'il existe un CGS sur lequel ϕ est vraie.

Définition (Model-checking)**.**
Le problème de model-checking pour une logique $\mathcal{L}$ demande un algorithme qui prend en entrée une formule $\phi \in \mathcal{L}$ et un jeu $\mathcal{G}$, et qui répond si ϕ est vraie sur $\mathcal{G}$.

L'ensemble des résultats connus sont présentés dans le tableau ci-dessous. Les résultats de borne inférieure du model-checking relativement au jeu (PH-hard) pour SL, SL[NG] et SL[BG], et de borne inférieure pour SL[BG] relativement à la formule sont développés dans la thèse. Les autres résultats sont issus de la littérature scientifique sur SL.

<table>
<tr><th colspan="2" rowspan="2"></th><th colspan="2">Model Checking</th><th rowspan="2">Satisfiability</th></tr>
<tr><th>Formula</th><th>Data</th></tr>
<tr><td rowspan="2">SL</td><td>Upper Bound</td><td colspan="2">NONELEMENTARY [39]</td><td rowspan="4">UNDECIDABLE</td></tr>
<tr><td>Lower Bound</td><td>TOWER [39]</td><td>PH-hard</td></tr>
<tr><td rowspan="2">SL[NG] and SL[BG]</td><td>Upper Bound</td><td colspan="2">In $(k+1)$-EXPTIME for k quantifier alternations [39]</td></tr>
<tr><td>Lower Bound</td><td>TOWER</td><td>PH-hard</td></tr>
<tr><td rowspan="2">SL[1G]</td><td>Upper Bound</td><td colspan="3" rowspan="2">2-EXPTIME-complete [38]
(PTIME-complete for the model checking data complexity)</td></tr>
<tr><td>Lower Bound</td></tr>
</table>

Résultats de complexité. En *orange*, les aboutissements de la thèse.

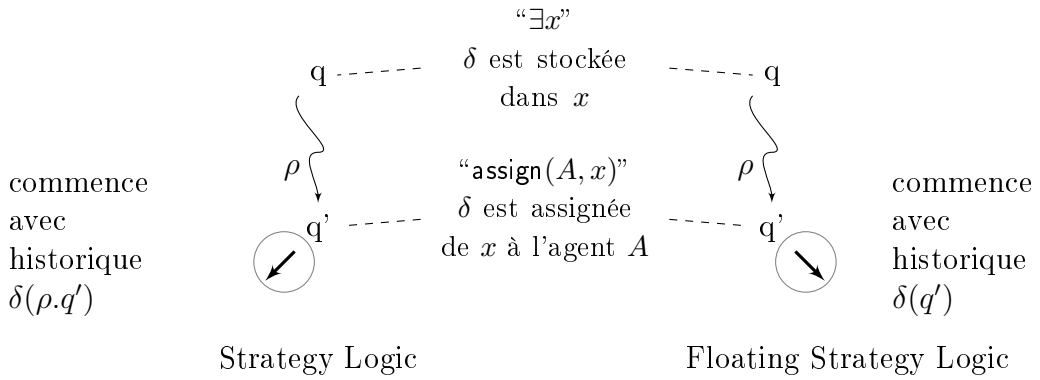
La sémantique flottante de SL (chapitre 3)

Prenons l'exemple d'un serveur qui propose à plusieurs clients l'accès à une ressource. Le serveur doit s'assurer de deux choses. Un, qu'à tout moment, l'accès à la ressource n'est possible qu'à au plus un client. Deux, qu'il existe un protocole pour chaque client qui lui assure l'accès (éventuellement) à la ressource (pourvu que le client collabore). Écrit dans SL, cela donne la formule suivante:

$$\exists \delta_{\text{serveur}}.\ \exists \delta_{\text{client}}.\ \text{ si serveur utilise } \delta_{\text{serveur}} \text{ alors } \big[(\text{ Toujours exclusion mutuelle})$$
$$\wedge\ (\text{ toujours } (\text{ si client utilise } \delta_{\text{client}} \text{ alors éventuellement accès }).)\big]$$

Intuitivement, au moment où la stratégie δ_{client} est assignée, elle devrait avoir un historique vide. En particulier, δ_{client} ne devrait pas avoir connaissance des demandes faites par les autres clients. Cependant, la sémantique de SL donne cette information. En effet, les stratégies ont des historiques qui commencent au moment de leur quantification et non de leur assignation. Ce choix limite l'expressivité de SL.

L'importance de ce détail sémantique a été longtemps sous-évalué par la communauté scientifique. Dans ce chapitre, nous proposons une nouvelle sémantique (nommée flottante) dans laquelle les stratégies tirent leurs historiques depuis leurs assignations (par opposition à leurs quantifications). Nous étudions ensuite la complexité et l'algorithmique qu'entraîne ce changement. La situation peut être illustrée par le dessin suivant.



Connaissance des historiques dans les deux sémantiques.

Définitions de la version flottante FSL de SL

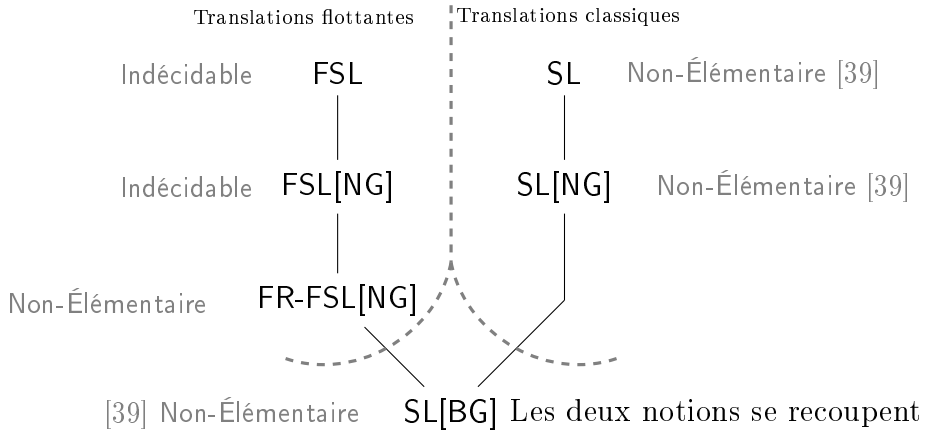
Définition (Translations (de valuations) flottantes)**.**
Pour une valuation χ sur un jeu $\mathcal{G}$, pour un ensemble $\mathcal{V}$ de variables et pour un historique ρ dans $\mathcal{G}$, on définit la translation flottante $\chi_{\underset{\rho}{\rightarrow}}$ par

$$\forall x \in \mathit{Agt} \cap \mathit{dom}(\chi),\ \chi_{\underset{\rho}{\rightarrow}}(x) := \chi(x)_{\overrightarrow{\rho}}$$
$$\forall x \in \mathcal{V} \cap \mathit{dom}(\chi),\ \chi_{\underset{\rho}{\rightarrow}}(x) := \chi(x)$$

La version flottante FSL de SL obéit à la même grammaire que SL, seule la sémantique des opérateurs temporels change.

$$\mathsf{FSL} \ni \phi \ ::= \ \exists x.\phi \mid \mathsf{assign}(A,x).\phi \mid \phi \vee \phi \mid \neg\phi \mid \phi \mathbf{U}\ \phi \mid \mathbf{X}\ \phi \mid p$$

Fixons une valuation χ avec $\mathsf{Agt} \cup \mathsf{free}(\phi) \cup \mathsf{free}(\phi') \subseteq \mathsf{dom}(\chi)$. En utilisant la translation flottante sur χ, comme dans la sémantique originale, on obtient un unique chemin



Graphe d'inclusion et complexité.

résultant ρ. On note alors $\chi_{\underset{\rightarrow}{j}}$ pour $\chi_{\rho_{\underset{\rightarrow}{\leq j}}}$ et on définit la sémantique de $\mathbf{X}$ et $\mathbf{U}$ par

$$\mathcal{G}, q \models_{\chi} \mathbf{X}\,\phi \quad \Leftrightarrow \quad \mathcal{G}, q \models_{\chi_{\underset{\rightarrow}{1}}} \phi$$

$$\mathcal{G}, q \models_{\chi} \phi' \mathbf{U}\,\phi \quad \Leftrightarrow \quad \exists k \in \mathbb{N}.\ \begin{cases} \mathcal{G}, q \models_{\chi_{\underset{\rightarrow}{k}}} \\ \forall j \in \mathbb{N}.\ 0 \leq j < k \Rightarrow \mathcal{G}, q \models_{\chi_{\underset{\rightarrow}{j}}} \phi' \end{cases}$$

On note FSL[NG] le fragment de FSL qui reprend la grammaire de SL[NG]. On note aussi FR-FSL[NG] le fragment de FSL[NG] qui force une réassignation de chaque agent lors de l'utilisation de l'opérateur assign (par opposition à seulement une réassignation d'une partie des agents).

Les deux sémantiques (originale et flottante) coïncident sur SL[BG]. On rappelle que les formules de SL[BG] sont formées d'un bloc de quantifications suivi par une combinaison booléenne d'objectifs (assignations suivies d'opérateurs temporels). Il ne peut donc pas y avoir d'assignation après un opérateur temporel. La quantification et l'assignation d'une variable ont lieu au même moment. De fait, la sémantique choisie n'a donc plus d'importance dans SL[BG].

Résultats de complexité

Les résultats de complexité sont représentés dans la Figure en haut de cette page. On peut voir que FSL admet un model-checking indécidable et que seul le fragment FR-FSL[NG] admet un algorithme. Ce petit changement sémantique a donc d'importantes conséquences algorithmiques.

Ces subtilités sémantiques dans SL et SL[NG] (ainsi que les résultats de complexité) nous laissent à penser que SL[BG] est le fragment à privilégier pour la suite, celui-ci étant relativement plus simple.

Aspects quantitatifs dans SL (chapitre 4)

De nombreux systèmes possèdent des contraintes quantitatives: gestion des ressources, grilles d'énergies, programmes avec des boucles... Les formalismes qualitatifs et monolithiques ne peuvent gérer ces aspects. La communauté scientifique a donc développé de nouveaux outils. Ici, nous nous intéressons aux adaptations possibles de SL pour traiter les aspects quantitatifs.

La vérification des systèmes ouverts quantitatifs est bien avancée et possède de multiple branches: jeux à énergies, jeux à coût moyen, VASS alternants, jeux à piles, jeux temporisés, jeux hybrides... Chacun de ces formalismes permet de modéliser un type de système différent. Dans ce chapitre, nous nous intéressons à deux d'entre eux: les jeux à compteurs et les jeux à énergies. Dans un premier temps, il convient d'adapter les jeux sur lesquels nous travaillons.

Ajouter des poids aux jeux

Définition (Jeux avec des poids: WCGS (weighted concurrent game structures))**.**
Un jeu à n-poids est un uplet $\mathcal{G} := \langle AP, Agt, Q, Act, \Delta, Weights, labels\rangle$ *où* AP, Agt, Q, Act, Δ, *et* $labels$ *représentent les même notions que dans le cas qualitatif, et où* $Weights : Q \times Act^{Agt} \to \{-1, 0, 1\}^n$ *est une fonction de poids qui associe n entiers dans* $\{-1, 0, 1\}$ *(les fameux poids) à chaque transition de* Δ.

On notera par WCGS *un jeu à n-poids, pour tout entier* $n > 0$.

Hormis les jeux, d'autres notions doivent être adaptées. Fixons un WCGS $\mathcal{G}$. Une *configuration* est un couple $(q, c) \in \mathsf{Q} \times \mathbb{N}^n$ composé d'un état et d'un n -uplet d'entiers positifs ou nuls. Un *historique* (resp. *chemin*) est une suite $(q_i, c_i)_{i<L}$ de configurations où $L \in \mathbb{N}$ (resp. $L = \infty$), pour tout $i \leq L$ (resp. $i \in \mathbb{N}$) $c_i \in \mathbb{N}^n$ et tel que pour tout $i < L-1$ (resp. $i \in \mathbb{N}$) $q_{i+1} = \Delta(q_i, m_i)$, et $c_{i+1} = c_i + \mathsf{Weights}(q_i, m_i)$ pour un $m_i \in \mathsf{Act}^{\mathsf{Agt}}$. Les autres concepts sont inchangés mais se basent sur ces nouvelles définitions d'historique et de chemin.

Ajouter des contraintes de compteurs à SL

Les compteurs sont omniprésents en l'informatique, notamment de par les boucles dans les programmes. Il existe donc une volumineuse littérature sur la vérification de système à compteurs. Pour notre travail, nous placerons les contraintes de compteurs dans la logique (SL). Notons qu'il aurait été possible de les ajouter sur les transitions du jeu (cela revient grosso-modo au même).

Il existe quelques résultats d'importance sur les systèmes à compteurs. Premièrement, l'accessibilité dans les jeux à deux compteurs et plus est indécidable. Dans les jeux à un unique compteur, cette accessibilité devient PSPACE. D'autre part, le model checking de ATL* est *2*-EXPTIME complet.

Définition (Contraintes de compteurs)**.**
Une contrainte de compteur sur n poids est un sous-ensemble S *de* $\mathbb{N}^n$ *fait d'une union*

(finie) d'ensembles periodiques d'entiers (ensembles de forme $a + b \cdot \mathbb{N}^n$, pour un vecteur a fixé et un vecteur de période b, tous deux dans $\mathbb{N}^n$).

Définition (cSL, Strategy Logic avec des contraintes de compteurs)**.**
La logique cSL est construite sur un nombre n de poids, un ensemble Agt d'agents, un ensemble APde propositions atomiques et un ensemble $\mathcal{V}$ de variables. Sa grammaire est la suivante:

$$cSL \in \phi \ ::= \ \exists x.\phi \mid assign(A, x).\phi \mid \phi \vee \phi \mid \neg\phi \mid \phi \mathbf{U}\, \phi \mid \mathbf{X}\, \phi \mid p \mid cnt \in S$$

où $x \in \mathcal{V}$ est une variable, $A \in Agt$ un agent, $p \in AP$ une proposition atomique et S une contrainte atomique sur n poids.

La notion d'agent et de variable libre est similaire à SL, avec l'addition ci-dessous pour les contraintes:

$$\mathsf{free}(\mathsf{cnt} \in \mathsf{S}) = \varnothing \quad \text{pour toute contrainte S}$$

De même, la sémantique des opérateurs déjà dans SL est conservée. On y ajoute celle des contraintes:

$$\mathcal{G}, (q, c) \models_\chi \mathsf{cnt} \in \mathsf{S} \quad \Leftrightarrow \quad c \in \mathsf{S}$$

Comme précisé plus haut, le model-checking de cSL est, dans le cas général, indécidable. Une solution potentielle est de réduire le nombre de poids à un; pour la question de l'accessibilité, cela ramène la décidabilité. Nous nous intéressons donc particulièrement à cSL avec un unique poids; on notera alors la logique 1cSL. De plus, nous nous contenterons d'étudier le fragment SL[BG]; une étude de la logique entière serait trop long (particulièrement s'il fallait traiter les deux sémantiques, flottante et originale). Dans cette thèse, nous avons développé deux résultats:

Théorème. *Soit $\mathcal{G}$ un 1-WCGS, et ϕ une formule de 1cSL[BG]. Il existe alors un seuil $\Delta \geq 0$ et une période $\Lambda \geq 0$ pour la valeur de vérité de ϕ sur $\mathcal{G}$. C·à·d pour toute configuration (q, c) de $\mathcal{G}$ avec $c \geq \Delta$, pour tout $k \in \mathbb{N}$, $\mathcal{G}, (q, c) \models \phi$ si et seulement si $\mathcal{G}, (q, c + k \cdot \Lambda) \models \phi$.*

De plus la somme $\Delta + \Lambda$ est majorée par

$$Tower\Big(\max_{\theta \in SubForm(\phi)} n_\theta, \max_{\theta \in SubForm(\phi)} k_\theta + 1\Big)^{|Q| \cdot 2^{2^{|\phi|}}}$$

où Q est l'ensemble d'états de $\mathcal{G}$, $SubForm(\phi)$ est l'ensemble des sous-formules de ϕ, k_θ est le nombre d'alternance des quantifications dans θ, et n_θ est le nombre d'assignations dans θ.

Théorème. *Faisons l'hypothèse de l'existence d'un algorithme pour le model-checking de 1cSL[BG] qui travaille en temps $Tower(|\phi|, k + c)$ sur une formule ϕ à k alternance de quantifications, pour une constante c fixée. Alors, la logique MSO sur le modèle $(\omega^\omega, <)$ peut être décidée en temps $Tower(|\Phi|, k + c)$ pour une formule Φ avec k alternance et où c est la constante de l'algorithme pour 1cSL[BG].*

Ajouter des contraintes d'énergies à SL

À l'inverse des contraintes de compteurs, le couple accessibilité plus contraintes d'énergies ne mène pas à l'indécidabilité dans les jeux avec deux poids ou plus. À travers cette partie, nous allons essayer de savoir si cela s'applique aussi à SL enrichie des contraintes d'énergies.

Un ensemble S de $\mathbb{N}^n$ est "clos par le haut" si, dès que $(s_1, \dots, s_n) \in \mathsf{S}$, il est vrai que $(s_1 + i_1, \dots, s_n + i_n) \in \mathsf{S}$ pour tout $(i_1, \dots, i_n) \in \mathbb{N}^n$.

Définition (Contraintes d'énergies)**.**
Une contrainte d'énergies sur n poids est un ensemble S of $\mathbb{N}^n$ *clos par le haut.*

Définition (eSL, Strategy logic avec contraintes d'énergies)**.**
Les formules d'eSL sont construites similairement à cSL, sur la grammaire suivante:

$$\mathit{eSL} \in \phi \ ::= \ \exists x.\phi \mid \mathit{assign}(A,x).\phi \mid \phi \vee \phi \mid \neg\phi \mid \phi \mathbf{U}\, \phi \mid \mathbf{X}\, \phi \mid p \mid \mathit{cnt} \in \mathit{S}$$

mais où S est une contrainte d'énergie.

La sémantique est la même que pour cSL, seul le type de contrainte change.

En utilisant les restrictions vues précédemment, on peut ainsi définir le fragment eSL[BG] de eSL. Nous nous intéresserons aussi au fragment conjonctif, dont la grammaire (du fragment plat) est rappelée ci-dessous.

$$\begin{aligned} \mathsf{eSL[CG]}^\flat \ni \phi \ &::= \exists x.\phi \mid \forall x.\phi \mid \xi \\ \xi \ &::= \xi \wedge \xi \mid \beta \\ \beta \ &::= \mathsf{assign}(A,x).\beta \mid \varphi \\ \varphi \ &::= \varphi \vee \varphi \mid \neg\varphi \mid \varphi \mathbf{U}\, \varphi \mid \mathbf{X}\, \varphi \mid p \mid \mathsf{cnt} \in \mathsf{S} \end{aligned}$$

Comme dit précédemment, l'accessibilité dans les jeux multi-dimensionnels à énergies est décidable. Nous avions donc bon espoir que le model-checking de eSL[BG] soit décidable lui aussi. Il n'en est rien, comme le prouve le résultat de la thèse ci-dessous.

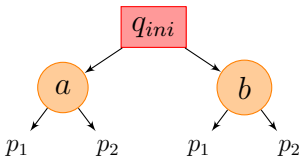
Théorème. *Le model-checking de eSL[CG] sur les 2-*WCGS *est indécidable*

Part 2: “Problèmes de dépendances dans SL[BG]”

Dépendances dans SL (chapitres 5 et 6)

Dans la sémantique des quantifications, une stratégie δ dépend de toutes les stratégies quantifiées avant elle dans leurs totalités. Cela rend la synthèse de δ difficile et force le système à utiliser beaucoup de mémoire. Cela a aussi un côté contre-intuitif. L'exemple ci-dessous illustre le problème.

Fixons deux agents $\square$ et $\bigcirc$, deux propositions atomiques p_1 et p_2 ainsi que quatre variables $x^{\square}_A$, $x^{\square}_B$, $y^{\bigcirc}_A$ et $y^{\bigcirc}_B$. Considérons le jeu $\mathcal{G}$ de la figure ainsi que la formule ϕ_0 de SL[BG] définies ci-dessous:



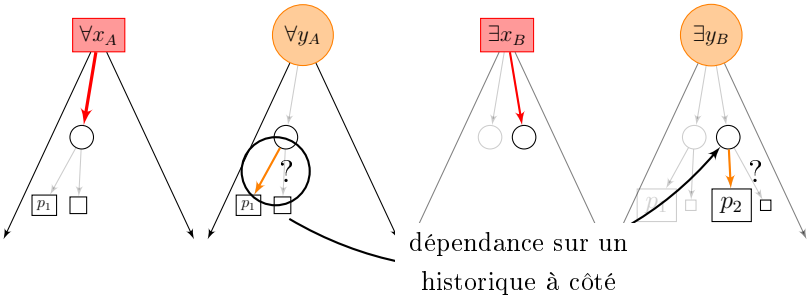
$$\phi_0 := \forall x^{\square}_A . \forall y^{\bigcirc}_A . \exists x^{\square}_B . \exists y^{\bigcirc}_B . \underbrace{\begin{cases} (\mathsf{assign}(\square, x^{\square}_A; \bigcirc, y^{\bigcirc}_A)\mathbf{F}\, p_1 \\ \Leftrightarrow\ \mathsf{assign}(\square, x^{\square}_B; \bigcirc, y^{\bigcirc}_B)\mathbf{F}\, p_2) \\ \wedge \\ \mathsf{assign}(\square, x^{\square}_B; \bigcirc, y^{\bigcirc}_B)\mathbf{F}\, b \end{cases}}_{\phi_1}$$

On peut voir que, suivant la sémantique donnée précédemment, ϕ_0 est vraie sur $\mathcal{G}$ depuis l'état initial q_{ini}[8].

En y regardant de plus près, la formule ϕ_0 oblige la stratégie de $y^{\bigcirc}_B$ sur l'historique $q_{ini}.b$ à dépendre du choix de $y^{\bigcirc}_A$ sur l'historique $q_{ini}.a$. On voit apparaître une dépendance d'une stratégie sur une autre dans sa globalité; la figure de la page suivante illustre la situation. Comprendre quand et pourquoi ces dépendances apparaissent nous permettrait de mieux aborder les problèmes mémoire dans SL[BG] et aiderait grandement le procédé de modélisation en vue d'applications pratiques.

Avant toute chose, certaines définitions sont nécessaires. En premier lieu, nous rappelons les notions de préfixes et d'extensions. Pour un historique $\rho = (q_i)_{i \leq L}$ fixé où $L \in \mathbb{N}$, un préfixe de ρ est un historique de forme $\rho' := (q_i)_{i \leq L'}$ où $L' < L$. Une extension

[8]Le détail est donné dans la thèse, mais nous laissons la vérification en exercice pour ce résumé substantiel en français



Dépendance de y_B sur y_A.

de ρ est un historique de forme $\rho'' := (q_i)_{i<L''}$ où $L'' > L$. Dans la suite nous écrirons $\mathsf{Pref}_{\leq\rho}$ pour l'ensemble des préfixes de ρ, ρ inclus, et $\mathsf{Pref}_{<\rho}$ pour ce même ensemble excluant ρ. Nous regroupons l'ensemble des historiques n'étant ni préfixe ni extension dans la notion d'historique "à côté".

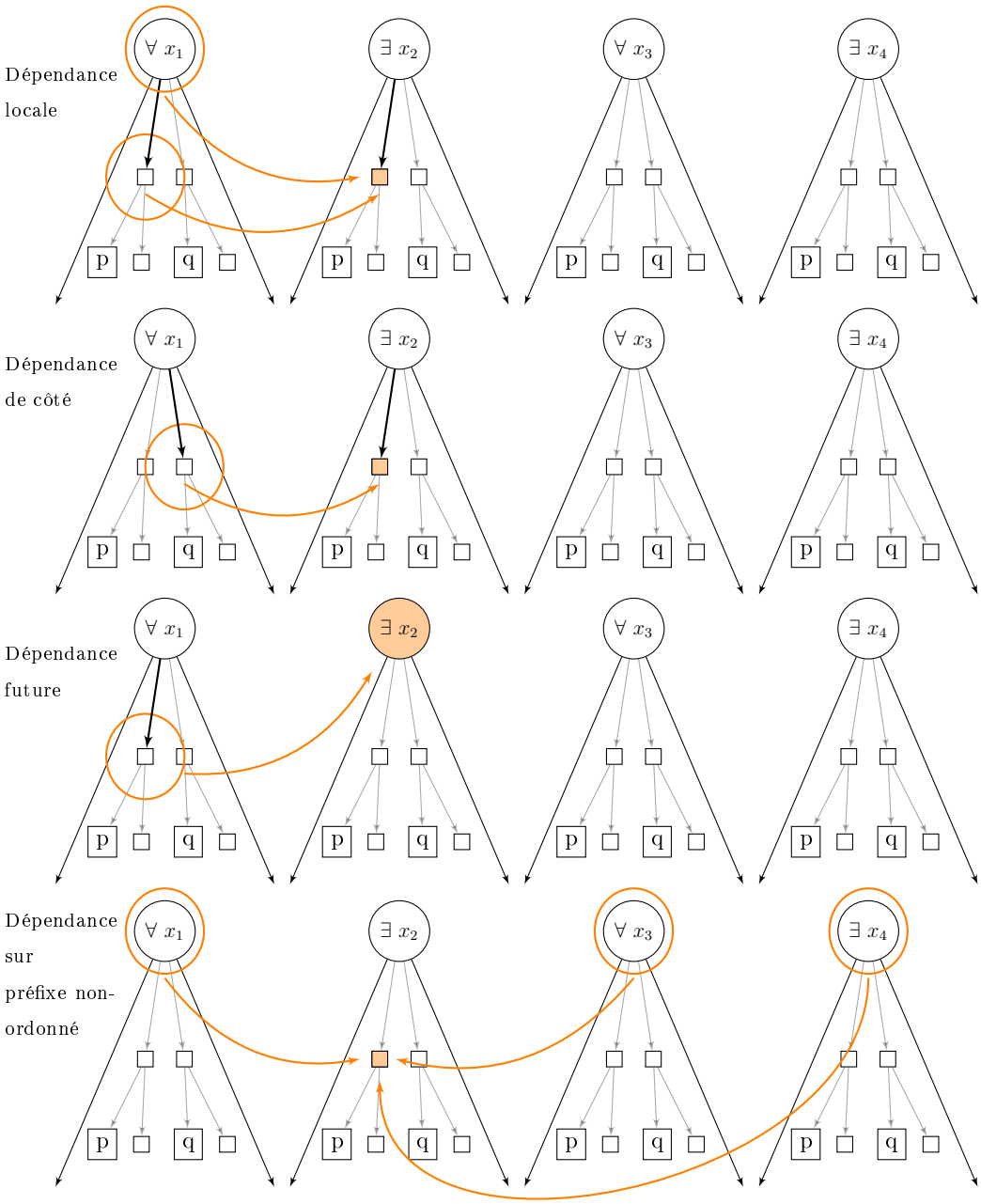
Définition (Historique à côté)**.**
Pour deux historiques ρ et π, on dit que π est un historique à côté de ρ quand ce n'est ni un préfixe ni une extension de ρ.

Nous identifions quatre types de dépendances et explorons leurs impacts respectifs sur SL[BG]. Plutôt qu'une longue explication, nous les présentons à travers une illustration: la figure de la page suivante. Des détails sur chacune d'entre elles sont donnés dans la thèse.

Pour étudier ces dépendances, nous réutilisons et améliorons un cadre présent dans la littérature: les "maps". L'idée reprend les concepts de la Skolémisation de la logique du premier ordre. L'astuce est de représenter les choix existentiels comme la réponse aux choix universels à travers une fonction (une *map* en anglais). On peut alors encoder les différentes dépendances en appliquant des restrictions à la dite fonction. La notion de "il existe une stratégie tel que pour toute stratégie... telle que le contexte résultant satisfait ..." du bloc de quantifications devient alors "il existe une map qui pour toute entrée retourne un contexte satisfaisant ...".
Pour des raisons de place, nous ne définirons pas les maps dans ce résumé et renvoyons à la thèse.

En utilisant ces maps, nous pouvons construire des relations de satisfaction adaptées aux différentes dépendances: $(\models^{\mathcal{M}(\spadesuit,\heartsuit)})_{\spadesuit\in\{\emptyset,S\},\ \heartsuit\in\{\emptyset,F\}}$. Pour une formule $(Q_i x_i)_{i\leq l}\, \xi\big(\beta_j\varphi_j\big)_{j\leq n}$ fermée de SL[BG], on pose

$$\mathcal{G}, q \models^{\mathcal{M}(\spadesuit,\heartsuit)} (Q_i x_i)_{i\leq l}\, \xi\big(\beta_j\varphi_j\big)_{j\leq n} \quad \Leftrightarrow \quad \begin{cases} \exists \text{ une map } \theta \text{ avec restrictions } \mathcal{M}(\spadesuit,\heartsuit) \text{ telle que} \\ \text{Pour tout comportement des stratégies universelles} \\ \text{le contexte } \chi \text{ résultant satisfait} \\ \mathcal{G}, q \models_\chi \xi\big(\beta_j\varphi_j\big)_{j\leq n} \end{cases}$$



Les quatre types de dépendance

La map θ est en quelque sorte un *témoin* d'un comportement possible des stratégies existentielles en fonction des stratégies universelles. La relation de satisfiabilité $\models^{\mathcal{M}(S,F)}$ redéfinit alors celle présentée au début du résumé ($\models$).

Remarque. *Notons que seule la sémantique des quantifications change (à travers l'utilisation des maps), mais pas celle des autres opérateurs.*

Complications regardant la notion de négation

L'étude des dépendances nous oblige à décomposer les relations entre les différentes stratégies. Cela a quelques effets indésirables. Le principal est que la notion de négation n'est plus aussi évidente que dans les cas simples (ATL^* par exemple). Quand on considère la négation d'une formule, que fait-on avec les dépendances? Faut-il les inverser aussi ? Faut-il les laisse tel-quel? La négation est donc à deux dimensions. On a ainsi plusieurs notions de négation et il faut faire attention à ce qu'on fait. On peut se retrouver avec des résultats comme celui dessous (après la figure).

$$\mathcal{G}, q \models^{\mathcal{M}(S,\emptyset)} (Q_i x_i)_{i\leq l}\ \xi\big(\beta_j\varphi_j\big)_{j\leq n}$$

les dépendances ne sont pas inversées

$$\mathcal{G}, q \models^{\mathcal{M}(S,\emptyset)} (\overline{Q_i} x_i)_{i\leq l}\ \xi\big(\beta_j\varphi_j\big)_{j\leq n}$$

La notion de négation devient plus complexe.

Lemme. *Il existe une formule ϕ de SL[BG], un jeu $\mathcal{G}$ ainsi qu'un état initial q_{ini} tel que, pour tout $\heartsuit \in \{\emptyset, F\}$,*

$$\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\heartsuit)} \phi \quad \textit{et} \quad \mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\heartsuit)} \neg\phi$$

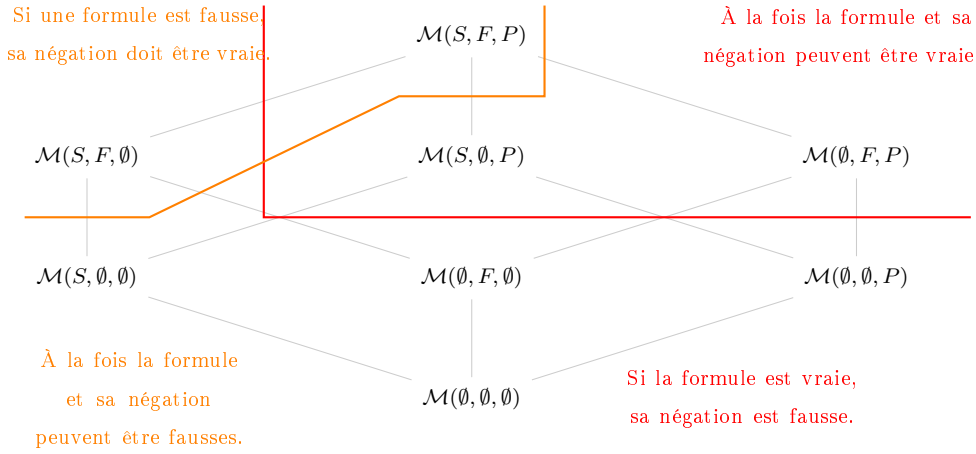
Combinaisons booléennes

Suivant une idée présente dans la littérature, nous décomposons SL[BG] selon plusieurs fragments basés sur le type de combinaisons booléennes d'objectifs. Il y a trois fragments d'intérêt:

- SL[1G] est le fragment le plus simple. Il restreint SL[BG] à n'avoir qu'un seul objectif. Dans sa grammaire, la ligne ξ devient $\xi ::= \beta$.

$$\begin{aligned} \mathsf{SL[1G]}^\flat \ni \phi &::= \exists x.\phi \mid \forall x.\phi \mid \xi \\ \xi &::= \beta \\ \beta &::= \mathsf{assign}(A, x).\ \beta \mid \varphi \\ \varphi &::= \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi \mid p \end{aligned}$$

- SL[CG] ne permet que des conjonctions d'objectifs; la ligne ξ devient alors $\xi ::= \xi \wedge \xi \mid \beta$.
- Similairement, SL[DG] n'autorise que des disjonctions d'objectifs.



Situation regardant la négation

Résultats

Plutôt qu'un long texte, pour ce résumé nous opterons pour une option visuellement plus simple, plus propre. L'ensemble des résultats est donc représentés à travers les deux figures ci-dessus et de la page suivante.

Le fragment SL[EG] (chapitre 7)

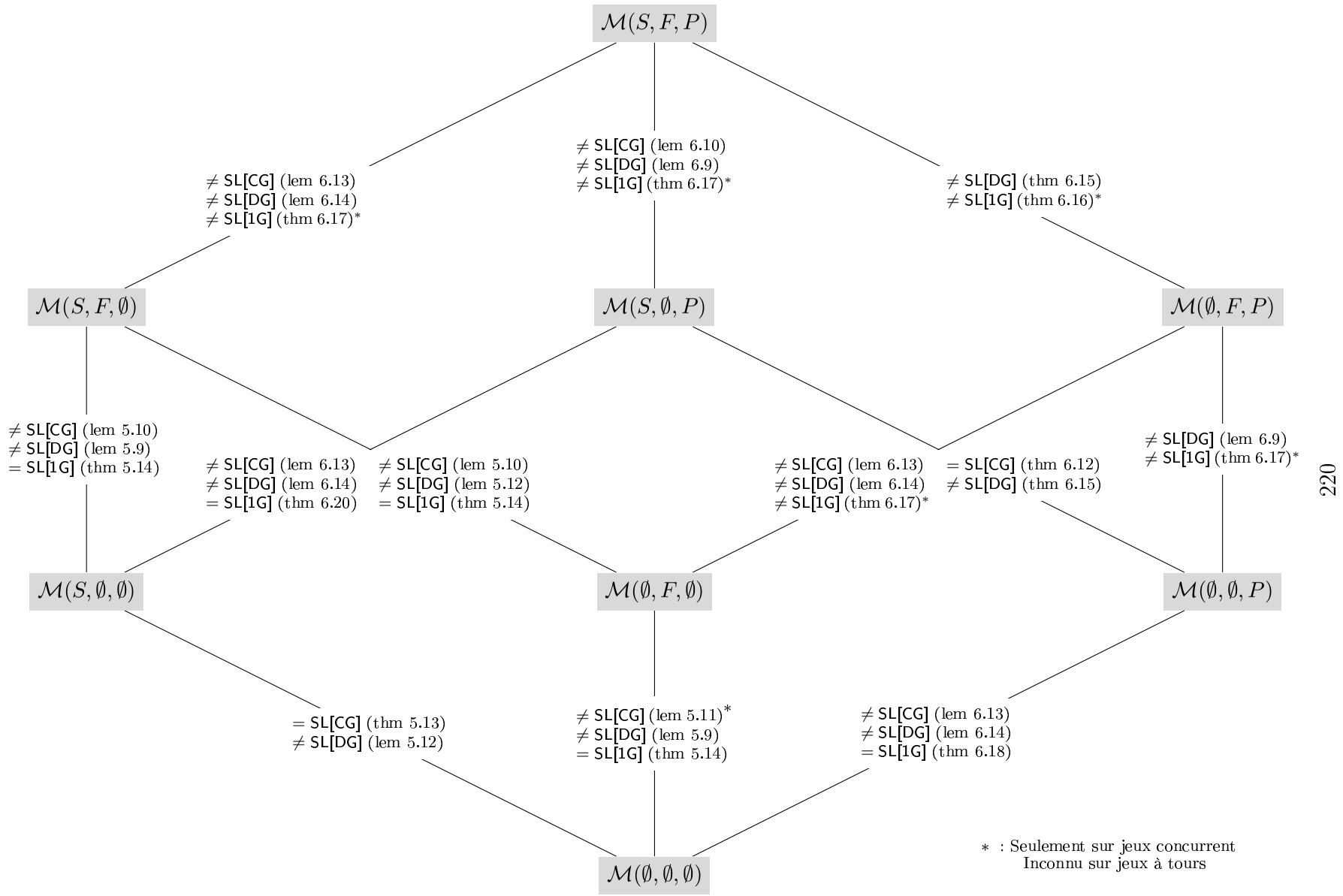
Nous poursuivons notre étude des dépendances dans SL. Nous nous intéresserons en particulier aux trois résultats ci-dessous. Ceux-ci reposent sur une restriction algébrique de SL[BG]: les ensembles semi-stables, et un nouveau fragment: SL[EG].

Théorème. *Fixons une formule $\phi \in$ SL[EG], un jeu $\mathcal{G}$, un état q_{ini} de $\mathcal{G}$ et deux paramètres $\spadesuit \in \{\emptyset, S\}$ et $\heartsuit \in \{\emptyset, F\}$. Si $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\spadesuit,\heartsuit,P)} \phi$ et $\mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\spadesuit,\heartsuit,P)} \neg\phi$ alors $\mathcal{G}, q_{ini} \models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$.*

Théorème. *Pour tout $n \in \mathbb{N}^*$ et tout ensemble non-semi-stable F^n, il existe un formule ϕ de SL[BG] construite sur F^n, un jeu $\mathcal{G}$ et un état q_{ini} de $\mathcal{G}$ tel que*

$$\mathcal{G}, q_{ini} \models^{\mathcal{M}(S,\emptyset,P)} \phi \quad et \quad \mathcal{G}, q_{ini} \not\models^{\mathcal{M}(S,\emptyset,P)} \neg\phi \quad et \quad \mathcal{G}, q_{ini} \not\models^{\mathcal{M}(\emptyset,\emptyset,P)} \phi$$

Théorème. *Le model checking de SL[EG] pour la relation $\models^{\mathcal{M}(\emptyset,\emptyset,P)}$ de satisfaction est 2-EXPTIME-complete.*



Graphe d'inclusion des différentes sémantiques. Les références correspondent à leur numérotation dans le manuscrit de thèse.

Ces résultats ont plusieurs conséquences. La principale est de fournir un fragment (SL[EG]) avec un model-checking en 2-EXPTIME pour la relation $\models^{\mathcal{M}(\emptyset,\emptyset,P)}$. Une seconde est de montrer que des restrictions algébriques permettent de limiter les dépendances.

Nous débutons par des notations: pour $n \in \mathbb{N}$, on note $\{0,1\}^n$ l'ensemble des fonctions de $[1,n]$ dans $\{0,1\}$. On note aussi $\mathbf{0}^n$ (ou $\mathbf{0}$ si n est clair) pour la fonction qui associe tout entier de $[1,n]$ à 0, et $\mathbf{1}^n$ (ou $\mathbf{1}$) pour celle qui leur associe toujours 1. Pour $f \in \{0,1\}^n$ et $k \leq n$, $f_{[1,k]}$ sera la restriction de f à $[1,k]$. La taille de $f \in \{0,1\}^n$ est définie par $|f| = \sum_{1 \leq i \leq n} f(i)$. Pour deux f et g dans $\{0,1\}^n$, on dit que $f \leq g$ quand $f(i) = 1$ implique $g(i) = 1$ pour tout $i \in [1,n]$. Pour $B^n \subseteq \{0,1\}^n$, on écrit $\uparrow B^n = \{g \in \{0,1\}^n \mid \exists f \in B^n.\ f \leq g\}$. Un ensemble $F^n \subseteq \{0,1\}^n$ sera dit *clos par le haut* quand $F^n = \uparrow F^n$. Enfin, on définit les opérateurs suivants:

$$\overline{f}\colon i \mapsto 1 - f(i) \qquad f \curlywedge g\colon i \mapsto \min\{f(i), g(i)\} \qquad f \curlyvee g\colon i \mapsto \max\{f(i), g(i)\}.$$

Pour définir SL[EG], le fragment au coeur des trois résultats cités plus haut, on introduit la notion de sous-ensemble semi-stable:

Définition. *Un ensemble $F^n \subseteq \{0,1\}^n$ est* semi-stable *si pour tout f et g dans F^n, il est vrai que*

$$\forall s \in \{0,1\}^n \qquad (f \curlywedge s) \curlyvee (g \curlywedge \overline{s}) \in F^n \text{ or } (g \curlywedge s) \curlyvee (f \curlywedge \overline{s}) \in F^n.$$

Exemple. *Clairement, $\{0,1\}^n$ est semi-stable, comme l'ensemble vide. Pour $n = 2$, l'ensemble $\{\langle 0,1\rangle, \langle 1,0\rangle\}$ n'est pas semi-stable: il suffit de prendre $f = \langle 0,1\rangle$ et $g = \langle 1,0\rangle$ avec $s = \langle 1,0\rangle$, on a alors $(f \curlywedge s) \curlyvee (g \curlywedge \overline{s}) = \langle 0,0\rangle$ et $(g \curlywedge s) \curlyvee (f \curlywedge \overline{s}) = \langle 1,1\rangle$. De même, $\{\langle 0,0\rangle, \langle 1,1\rangle\}$ n'est pas semi-stable. Tout autre sous-ensemble de $\{0,1\}^2$ est semi-stable.*

Nous pouvons maintenant définir le fragment SL[EG]:

$$\begin{aligned} \mathsf{SL[EG]} \ni \phi &::= \forall x.\phi \mid \exists x.\phi \mid \xi \\ \xi &::= F^n((\beta_i)_{1 \leq i \leq n}) \\ \beta &::= \mathsf{assign}(A,x).\ \beta \mid \varphi \\ \varphi &::= \neg\varphi \mid \varphi \vee \varphi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi \mid p \end{aligned}$$

avec p dans AP, n dans $\mathbb{N}$ et pour tout n, F^n est un ensemble semi-stable de $\{0,1\}^n$. La sémantique de F^n est alors

$$\mathcal{G}, q \models_\chi F^n((\beta_i)_{i \leq n}) \quad \Leftrightarrow \quad \exists f \in F^n \text{ avec } f(i) = 1 \text{ ssi } \mathcal{G}, q \models_\chi \beta_i$$

Si F^n n'avait pas la restriction semi-stable, on retrouverait alors la logique SL[BG]. Le cas où $F^n = \{\mathbf{1}^n\}$ correspond à SL[CG], tandis que $F^n = \{0,1\}^n \setminus \{\mathbf{0}^n\}$ donne SL[DG]. Puisque $\{\mathbf{1}^n\}$ et $\{0,1\}^n \setminus \{\mathbf{0}^n\}$ sont semi-stables, on a SL[CG] $\subseteq$ SL[EG] and SL[DG] $\subseteq$ SL[EG].

Exemple. *Considérons la formule de l'existence d'un équilibre de Nash. Pour deux agents, cela donne*

$$\exists x_1.\exists x_2.\forall y_1.\forall y_2 \begin{cases} (\mathsf{assign}(A_1, y_1; A_2, x_2).\varphi_1) \Rightarrow (\mathsf{assign}(A_1, x_1; A_2, x_2).\varphi_1) \\ \wedge \\ (\mathsf{assign}(A_1, x_1; A_2, y_2).\varphi_2) \Rightarrow (\mathsf{assign}(A_1, x_1; A_2, x_2).\varphi_2) \end{cases} \tag{7.15}$$

La formule a quatre objectifs et est en correspondance avec l'ensemble

$$F^4 = \{\langle 1,1,1,1\rangle, \langle 0,1,1,1\rangle, \langle 1,1,0,1\rangle, \langle 0,1,0,1\rangle, \langle 0,0,1,1\rangle, \langle 1,1,0,0\rangle, \\ \langle 0,0,0,1\rangle, \langle 0,1,0,0\rangle, \langle 0,0,0,0\rangle\}$$

Cet ensemble n'est pas semi-stable et donc la formule n'est pas syntaxiquement dans SL[EG].

Grâce à sa définition algébrique, SL[EG] admet des propriétés intéressantes, par exemple un ensemble F^n est semi-stable si et seulement si son complément l'est aussi. Une autre propriété d'importance est l'existence de pré-ordres qui classent les objectifs par ordre d'importance.

Pour voir cela, nous avons besoin de nouvelles notations. Pour un ensemble $F^n \subseteq \{0,1\}^n$, un $s \in \{0,1\}^n$ et un $h \in \{0,1\}^n$, on définit

$$\mathbb{F}^n(h,s) := \{h' \in \{0,1\}^n \mid (h \curlywedge s) \curlyvee (h' \curlywedge \overline{s}) \in F^n\}$$

et (en écrivant $\overline{F^n}$ pour le complément de F^n),

$$\overline{\mathbb{F}^n}(h,s) := \{h' \in \{0,1\}^n \mid (h \curlywedge s) \curlyvee (h' \curlywedge \overline{s}) \in \overline{F^n}\}$$

Trivialement, $\mathbb{F}^n(h,s) \bigcap \overline{\mathbb{F}^n}(h,s) = \emptyset$ et $\mathbb{F}^n(h,s) \bigcup \overline{\mathbb{F}^n}(h,s) = \{0,1\}^n$.

Lemme. *Si F^n est semi-stable, alors la famille $(\mathbb{F}^n(h,s))_{h\in\{0,1\}^n}$ est tel que pour tout $h_1, h_2 \in \{0,1\}^n$, soit $\mathbb{F}^n(h_1,s) \subseteq \mathbb{F}^n(h_2,s)$ soit $\mathbb{F}^n(h_2,s) \subseteq \mathbb{F}^n(h_1,s)$.*

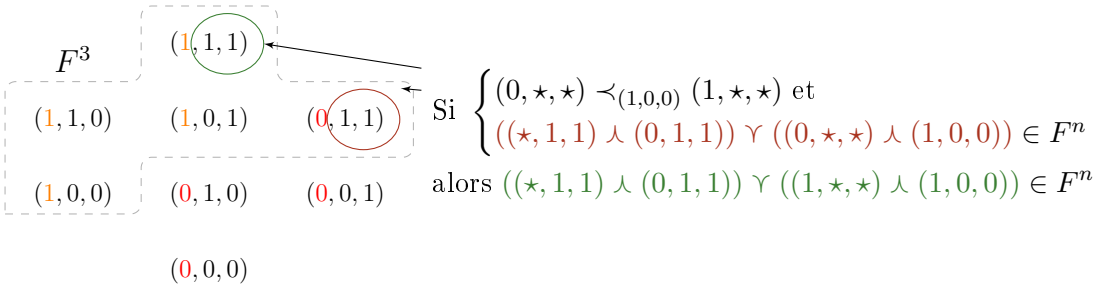
On peut alors utiliser l'inclusion du lemme précédant des pré-ordres $\preceq_s^{F^n}$ sur les éléments de $\{0,1\}^n$.

Définition. *On définit $\preceq_s^{F^n} \subseteq \{0,1\}^n \times \{0,1\}^n$ de sorte que $h_1 \preceq_s^{F^n} h_2$ ssi $\mathbb{F}^n(h_1,s) \subseteq \mathbb{F}^n(h_2,s)$. En particulier, $h_1 \preceq_{\mathbf{1}}^{F^n} h_2$ dès lors que $h_1 \in \overline{F^n}$ ou $h_2 \in F^n$.*

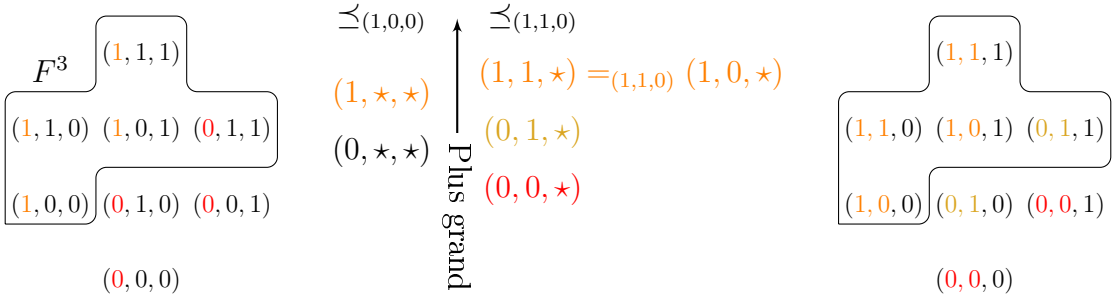
Dans la suite, F^n est fixé et nous utiliserons la notation simplifiée $\preceq_s$ pour faciliter la lecture. L'intuition derrière $\preceq_s$ est d'ordonner les éléments de $\{0,1\}^n$ en fonction de la facilité à les compléter en un élément de F^n. Les figures de la page suivante illustrent l'idée.

Lemme. *Fixons un ensemble semi-stable F^n, $s_1, s_2 \in \{0,1\}^n$ tel que $s_1 \curlywedge s_2 = \mathbf{0}$ et $f, g \in \{0,1\}^n$ tel que $f \preceq_{s_1} g$ et $f \preceq_{s_2} g$. Alors, $f \preceq_{s_1 \curlyvee s_2} g$.*

Ces pré-ordres peuvent alors être utilisés pour prouver les trois résultats présentés au début de la section. L'idée étant qu'une stratégie existentielle peut utilisé le pré-ordre approprié pour choisir l'action la mieux adaptée. Nous reportons les preuves à (la version complête) la thèse.



Lien entre $F^n(h,s)$ et les pré-ordres.



Un exemple avec $F^3 := \{\langle 1,1,1\rangle; \langle 1,1,0\rangle; \langle 1,0,1\rangle; \langle 0,1,1\rangle; \langle 1,0,0\rangle\}$ et les pré-ordres $\preceq_{(1,0,0)}$ et $\preceq_{(1,1,0)}$.

Conclusion

Cette thèse porte sur les problèmes de sémantique et de complexité dans la Strategy Logic (SL). SL tire sa particularité de la dissociation entre stratégies et agents; cette dissociation joue un rôle central dans l'expressivité de la logique. Il est par exemple possible de créer une stratégie à un temps t et de l'appliquer à un autre temps t'. On peut aussi exprimer des liens complexes entre les exécutions potentielles du système: on peut forcer un agent à agir suivant le même comportement dans deux exécutions tout en laissant les autres agents changer de stratégies entre les deux exécutions. De manière générale, SL est une logique possédant un grand pouvoir d'expression. Cela en fait un cadre idéal pour travailler sur les aspects temporels des systèmes multi-agents.

Complexité: Comme toujours, une grande expressivité va de paire avec une mauvaise complexité. En particulier, Mogavero, Murano, Perelli et Vardi ont montré que la satisfiabilité de SL est indécidable et que son model-checking est (décidable mais) non-élémentaire. À travers cette thèse, nous avons complété le tableau. Premièrement nous avons trouvé une borne inférieure TOWER au fragment SL[BG]; ce faisant, la complexité du fragment SL[BG] rejoint celle de SL. Deuxièmement nous avons montré que le model-checking relatif à la taille du système (pour une formule fixée) est dure pour tout les niveaux de la hiérarchie polynomiale. Il reste malheureusement un écart important entre les bornes supérieures et inférieures (TOWER et PH) du model-checking de SL et SL[BG] relativement à la taille du système. Résoudre ce décalage permettrait d'améliorer les applications pratiques de SL.

Phase entre la création et l'utilisation d'une stratégie: Il se peut qu'une stratégie δ soit créée à un temps t mais appliquée seulement après un temps de latence au moment t' (avec $t' > t$). Dans ce cas, SL considère que δ au moment t'' ($t'' > t'$) a non seulement connaissance de l'évolution du système entre t' et t'' mais aussi entre t et t'. Dans cette thèse, nous argumentons pour un autre choix sémantique: δ ne pourrait alors connaître que l'évolution du système entre t' et t'', le passage entre t et t' lui serait inaccessible. Une telle sémantique (que nous nommons FSL) serait particulièrement adaptée aux interactions client/serveur et aux problèmes d'ouverture et de fermeture de sessions. Cette différence sémantique n'avait jusqu'alors pas été identifiée. Fâcheusement, le model-checking de FSL est indécidable. Nos résultats identifient la frontière entre décidabilité et indécidabilité dans FSL. En particulier, nous identifions un fragment dont le model-checking est décidable.

Dépendances entre stratégies: La plupart des extensions d'ATL* sont sujettes à des problèmes sémantiques d'accès à l'information (sur l'historique du système, sur les stratégies des autres agents, sur la différentiation des états du système...). Dans la seconde partie de la thèse, nous étudions le degré de connaissance sur les stratégies universelles nécessaire pour synthétiser les stratégies existentielles. Pour de simple requêtes, les informations nécessaire ne sont que locales. A l'inverse, si on considère des propriétés plus complexes, le degré d'information sur les choix des autres stratégies créé différentes sémantiques.

Les résultats présentés dans cette thèse permettent de connaître précisément les différentes sémantiques possibles. Nous identifions aussi un fragment, SL[EG], de SL qui admet un model-checking *2*-EXPTIME. SL[EG] est basé sur différentes restrictions algébriques, cela laisse à penser l'existence d'élégant résultats basés sur des méthodes combinatoires et algébriques.

Réflexions et Perspectives: Le problème de dépendance entre stratégies universelles et existentielles a été peu étudié. Il est pourtant central à la compréhension de SL (et des systèmes multi-agents de manière générale). La communauté scientifique délaisse majoritairement ce problème pour se concentrer sur les résultats de complexité. Ainsi, l'immense majorité des travaux utilise la même sémantique: une stratégie avec une connaissance totale des stratégies quantifiées avant elle. Ce choix est particulièrement malheureux pour deux raisons. Premièrement car, comme montré dans cette thèse, les questions de dépendance entre stratégies dans les systèmes multi-agents induisent de nombreuses subtilités. Deuxièmement car la majorité des applications industrielles ne se modélisent pas avec cette sémantique. Ainsi une majorité des travaux académiques sur SL qui n'abordent pas les dépendances ne resteront que théorique.

En vue d'une diffusion de la vérification des systèmes multi-agents dans l'industrie, il convient d'approfondir notre connaissance des problèmes de dépendance. En particulier, il est impératif de simplifier le cadre mathématique, d'étudier les différentes sémantiques possibles (relativement aux différentes dépendances) et, pour chacune, de développer des algorithmes adaptés.

Abstract

With the proliferation of computerised devices, software verification is more prevalent than ever. Since the 80's, multiple costly software failures have forced both private and public actors to invest in software verification. Among the main procedures we find the model-checking, developed by Clarke and Emerson in the 80's. It consists in abstracting both the system into a formal model and the property of expected behaviour in some logical formalism, then checking if the property's abstraction holds on the system's abstraction. The difficulty lies in finding appropriate models and efficient algorithms.

In this thesis, we focus on one particular logical formalism: the *Strategy Logic* SL, used to express multi-objectives properties of multi-agents systems. Strategy Logic is a powerful and expressive formalism that treats strategies (i.e. potential behaviours of the agents) like first-order objects. It can be seen as an analogue to first-order logic for multi-agents systems. Many semantic choices were made in its definition without much discussion. Our main contributions are relative to the possibilities left behind by the original definition.

We first introduce SL and present some complexity results (including some of our owns). We then outline some other semantic choices within SL's definition and study their influence. Third, we study the logic's behaviour under quantitative multi-agents systems (games with energy and counter constraints). Finally, we adress the problem of dependencies within SL[BG], a fragment of SL.

Keywords: game theory, strategy logic, model-checking, formal methods...

Résumé

De nombreux bugs informatiques ont mis en lumière le besoin de certifier les programmes informatiques et la vérification de programmes a connu un développement important au cours des quarante dernières années. Parmi les méthodes possibles, on trouve le model checking, développé par Clarke et Emerson dans les années 80. Le model checking consiste à trouver un modèle abstrait pour le système et un formalisme logique pour le comportement puis a vérifier si le modèle vérifie la propriété exprimée dans la logique. La difficulté consiste alors à développer des algorithmes efficaces pour les différents formalismes.

Nous nous intéresserons en particulier au formalisme logique de *Strategy logic* SL, utilisée sur les systèmes multi-agents. SL est particulièrement expressif de par son traitement des stratégies (comportements possibles pour les agents du système) comme des objets du premier ordre. Dans sa définition, divers choix sémantiques sont faits et, bien que ces choix se justifient, d'autres possibilités n'en sont pas plus absurdes: tel ou tel choix donne telle ou telle logique et chacune permet d'exprimer des propriétés différentes. Dans cette thèse, nous étudions les différentes implications des différents choix sémantiques.

Nous commencerons par introduire SL et préciserons l'étendue des connaissances actuelles. Nous nous intéresserons ensuite aux possibilités non explorées par la sémantique originale. Nous étudierons aussi la logique sur des systèmes quantitatifs (ajout de contraintes d'énergie et de contraintes de compteurs). Finalement, nous examinerons la question des dépendances dans SL[BG] (un fragment de SL).

Mots-clefs: théorie des jeux, strategy logic, model-checking, méthodes formelles...