Représentation symbolique pour la logique épistémique dynamique probabiliste

Sebastien Gamblin

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité INFORMATIQUE

Préparée au sein de l'Université de Caen Normandie

Représentation symbolique pour la logique épistémique dynamique probabiliste

Présentée et soutenue par
SEBASTIEN GAMBLIN

Thèse soutenue le 12/12/2022 devant le jury composé de		
M. TIAGO DE LIMA	Maître de conférences HDR, Université d'Artois	Rapporteur du jury
M. FRANCESCO BELARDINELLI	Maître de conférences, Imperial College London, Blackett Labo.	Membre du jury
M. ALEXANDRE NIVEAU	Maître de conférences, Université de Caen Normandie	Membre du jury
MME SOPHIE PINCHINAT	Professeur des universités, UNIVERSITE RENNES 1	Membre du jury
M. ANDREAS HERZIG	Directeur de recherche, Université Toulouse 3 Paul Sabatier	Président du jury

Thèse dirigée par MEROUA BOUZID (Groupe de recherche en informatique, image, automatique et instrumentation)

UNIVERSITÉ CAEN NORMANDIE

GREYC
Laboratoire de recherche en sciences du numérique

Résumé

La logique épistémique dynamique (DEL) a été étudiée pour sa capacité à représenter les connaissances épistémiques (i.e. sur les connaissances des agents à propos des connaissances des autres agents) et exploitée récemment dans le domaine de la planification épistémique.

La logique épistémique dynamique probabiliste (PDEL) qui étend ce formalisme permet quant à elle de raisonner sur les connaissances probabilistes de haut niveau des agents.

DEL a été étudiée de manière théorique en terme de complexité, néanmoins l'utiliser s'avère ardu en pratique avec une implémentation « explicite » (i.e. qui suit la définition mathématique) à cause de l'explosion combinatoire des structures de Kripke, composante de ce formalisme.

C'est pourquoi des représentations dites *symboliques* ont été proposées, permettant de casser cette explosion combinatoire. L'idée est de ne plus avoir besoin de représenter explicitement et complètement les structures afin de faire de la vérification de modèle.

PDEL souffre aussi de ce problème et n'est donc utilisée que dans le cadre d'exemples jouets, sans pouvoir être exploitée dans des cas plus concrets et applicatifs. Pourtant, il serait intéressant de modéliser des jeux combinatoires, comme les jeux de cartes, dont le problème de planification épistémique nécessite la manipulation des probabilités intrinsèques du jeu. Malheureusement, contrairement à DEL, PDEL ne connaît à ce jour pas de représentation symbolique permettant une mise en pratique.

Tout l'objet de cette thèse est d'utiliser les concepts des représentations symboliques et de les adapter à un contexte probabiliste afin de pouvoir utiliser PDEL en pratique. Dans la continuité des travaux récents utilisant la vérification de modèle symbolique pour DEL, nous proposons une représentation symbolique des structures de Kripke probabilistes en tant que fonctions pseudo-booléennes, qui peuvent être représentées en mémoire par différentes structures de données de la famille des diagrammes de décision, en particulier les *Algebraic Decision Diagrams* (ADDs) qui connaissent des algorithmes de manipulation polynomiaux. Grâce à différentes expériences comparant les temps de génération, de vérification de modèles et de mise à jour des modèles, nous montrons que la représentation symbolique des structures de Kripke via les ADDs passe à l'échelle contrairement aux structures explicites même dans le cas de l'exemple réaliste du jeu de cartes Hanabi, ce qui pourrait ouvrir la voie à l'utilisation en pratique de techniques de planification épistémique dans un cadre probabiliste.

Mots-clefs : Intelligence artificielle ; Représentation des connaissances ; Logique épistémique dynamique probabiliste ; Vérification de modèles symbolique ; Diagrammes de Décision Valués ; Hanabi.

Remerciements

Tout d'abord, je souhaite remercier l'université de Caen Normandie où je suis inscrit depuis maintenant 11 ans et la région Normandie qui a financé mes travaux de thèse. J'y ai trouvé un plaisir certain à découvrir l'informatique en licence et master. À force de patience et de détermination, je peux maintenant rédiger ces quelques mots, des années après avoir mis les pieds à la fac, pour finaliser cette thèse, qui n'a pas été de tout repos. Sans compter des problèmes personnels déstabilisants, il y a eu deux longues années de COVID, qui ont rendus les travaux compliqués : de nombreux groupes de travail, écoles d'été, ou conférences ont été annulés, tout du moins en présentiel. Je n'ai pas pu profiter d'une école d'été au Portugal ni présenter mes travaux en Nouvelle-Zélande, c'est dommage, j'y aurai bien fait un petit tour... Cela ne m'a néanmoins pas empêché de faire de très nombreuses rencontres enrichissantes.

En premier lieu, je souhaite remercier profondément les rapporteurs et les autres membres du jury pour avoir accepté ce rôle, et pour toutes leurs remarques pertinentes.

Évidemment, je souhaite remercier l'équipe MAD dans laquelle j'ai pu trouver des collègues et amis, et surtout qui m'ont donné goût à la discipline. Premièrement, je remercie Alexandre et Maroua qui ont soutenu et encadré cette thèse. Tout ne s'est malheureusement pas passé parfaitement, mais tout a été, je le pense, fait au mieux dans les circonstances données. Merci Alexandre d'avoir fait preuve de patience et d'efforts pour me soutenir, bien que l'oganisation ne soit pas ton fort. Une pensée particulièrement à la personne qui m'a donné mon premier cours d'intelligence artificielle en L3 : Grégory, qui est toujours animé d'une passion communicative. Je me dois de compléter le trio du bureau 384 : un grand merci à Bruno Z. qui est toujours à l'écoute et qui trouve toujours la réflexion pertinente qui permet de mettre en perspective le travail accompli. Je n'oublie pas non plus Laurent, Abdel-Illah, Nadjet, Bruno M. et Gaële. Ils ne sont pas vraiment de l'équipe, mais c'est un peu comme : merci à François, Fred, Tiago, Andreas, avec qui, mine de rien, j'ai pu passer du temps. Un merci spécial à Tristan, toujours dynamique et enclin à nous enfermer dans des salles verouillées.

J'ai pu parler ici des permanents de MAD, mais il y a bien évidemment toutes

les personnes que j'ai pu cotoyer dans le laboratoire : Anaëlle, Gaétan, Mathias, Justine, Lauréline, Pierre, Davy, Renaud, Virginie, Amélie, Céline, Simon... Heureusement que vous étiez là, notamment à toutes les pauses du midi. En outre, grâce à nombre d'entre-vous, je pourrai mettre comme expérience conséquente "tarot" sur mon CV. Il y a aussi mes co-bureaux qui ont pu grandement modifié mon expérience : Christopher, Jacques, Sergej et Mihail. Particulièrement, un grand merci à Josselin qui a participé à enrichir considérablement mes connaissances dans les domaines zythologique et musicologique. Il est aussi sûrement la personne qui m'a le plus entendu râler et soutenu sur le long terme.

Par ailleurs, Josselin m'a permis de rencontrer des amis qui ont été nécessaire à ma bonne santé mentale à Caen : Clém, Pumba, Morgane, Rémi, Joss, Lou. Je pense qu'il est possible de résumer nos interactions par deux mots : "raclettes" et "Hellfest", qui dans les deux cas, sont des expérences inoubliables pleines de vie, de bonne humeur et de metal. Il m'est impossible d'oublier mes amis d'ailleurs, que j'ai malheureusement trop peu vu à cause de la distance mais qui m'ont grandement soutenu. Un grand merci. À Jérémy qui a toujours été là, à l'écoute et avec son humour... Unique. Tu es là depuis de nombreuses années et toujours là. Merci. À Aurélie, que j'ai finalement trop peu vu, mais qui a toujours été bienveillante et soucieuse. À l'équipe de geeks : Géraldine, Bibi, Alec, Anaëlle et Nono, qui m'ont soutenu dans une période sentimentale très difficile et le confinement. Je m'excuse, j'ai beau avoir joué de nombreuses heures avec vous, mon Reinhardt est toujours aussi mauvais. À Cloé, tu es celle que j'ai la moins vue, mais avec qui j'ai le plus discuté. Merci d'avoir pu me rendre le sourire dans les moments les plus difficiles. À Marine, qui m'a le plus soutenu, qui a extériorisé beaucoup de frustration à ma place, qui a été présente pour me changer les idées et qui essaye toujours, bon gré mal gré, d'améliorer ma propre estime. Ta présence a amélioré les deux dernières années de cette thèse, plus que tu ne peux le penser. J'ai cité beaucoup de noms et je terminerai par remercier chaleureusement ma famille : mes parents et à mon frère. Papa, si tu as l'occasion de lire cette thèse, tu comprendra peut-être enfin sur quoi je travaille. Bien évidemment, je ne pourrai pas être où j'en suis aujourd'hui sans eux. Ils ont toujours été là pour moi et m'ont encouragé, peu importe mes choix.

Merci à tous pour votre soutien qui m'a été nécessaire pendant cette longue période difficile.

Table des matières

Introduction

Contexte

Nous nous plaçons dans un système où différentes entités (dites « agents ») vont coexister et interagir ensemble. En intelligence artificielle, nous considérons un « agent » comme *une entité autonome qui agit, en orientant son activité vers la réalisation d'objectifs, sur un environnement en utilisant les observations dont il est capable (capteurs, etc) et des actionneurs conséquents [AA07].* En termes d'intelligence, ces agents peuvent être simples ou très complexes. Ils peuvent apprendre ou utiliser des connaissances pour atteindre leurs objectifs.

Nous considérons en particulier des agents qui participent à des jeux, et plus précisemment des jeux qui requièrent de pouvoir raisonner sur les connaissances. Nous nous intéressons donc à ce que l'on appelle les « connaissances *épistémiques* », le mot « épistémique » venant du terme *épistémè* (du grec ancien *ἐπιστήμη*), qui désigne en philosophie l'ensemble des connaissances scientifiques, du savoir d'une époque. Ainsi, les agents vont devoir ici se servir de leurs connaissances, ainsi que de leurs connaissances sur les connaissances des autres agents (i.e. les connaissances dites « imbriquées » ou « d'ordre supérieur ») pour établir un plan d'actions. Typiquement, l'agent pourra se demander s'il sait qu'un autre agent sait une information et agir en conséquence.

Hanabi est un exemple représentatif d'un type de jeu de société où les connaissances épistémiques sont essentielles. Par cette caractéristique et sa difficulté de résolution, il a récemment éveillé l'interêt de la communauté en intelligence artificielle [BFC+20]. C'est un jeu de cartes multijoueur coopératif à information imparfaite où la connaissance d'ordre supérieur joue un rôle très important. En effet, les joueurs doivent prendre des décisions qui dépendent de leurs connaissances sur les connaissances qu'ont les autres joueurs sur les cartes des autres joueurs. Pour modéliser ces connaissances, nous nous basons sur DEL (*Dynamic Epistemic Logic*), un formalisme permettant de raisonner sur des connaissances qui peuvent changer au cours du temps, après l'apparition d'événements. En particulier, dans DEL se pose la question du *model checking* (vérification de modèle), qui consiste à déterminer si une formule logique (qui peut contenir des opérateurs

dynamiques, simulant des événements) est vraie dans un certain état de connaissance, représenté par ce qu'on appelle une structure de Kripke. La construction de DEL permet notamment une approche élégante pour la planification épistémique, qui consiste à établir un plan d'action afin de réaliser une tâche, potentiellement épistémique ; par exemple « quelle est la suite d'actions à appliquer pour organiser un anniversaire surprise à son père sans qu'il ne le sache ? ». Cependant, dans de nombreux jeux réels, en particulier ceux à information imparfaite tels que Hanabi, les stratégies gagnantes n'existent pas. En effet, il peut être impossible d'atteindre le score maximal à cause d'ordre spécifique des cartes. L'idée est alors de chercher une stratégie qui maximise les chances de victoire, ou tout du moins maximise un score final.

Une piste naturelle est d'étudier une adaptation de la planification épistémique avec la logique épistémique dynamique probabiliste (PDEL pour *Probabilistic Dynamic Epistemic Logic*) qui est une généralisation de DEL. Cette extension utilise des structures de Kripke classiques qui sont couplées à des distributions de probabilités.

Malheureusement, les structures de Kripke, qu'elles soient munies de probabilités ou non, sont soumises à une explosion combinatoire du nombre d'états possibles. De ce fait, implémenter ces structures de manière explicite (en suivant naïvement les définitions mathématiques) devient rapidement ingérable dans un programme, à cause de leurs tailles en mémoire. Pour pallier ce problème, il existe ce qu'on appelle des représentations *symboliques* qui permettent de ne pas avoir à modéliser les structures de manière complète et explicite. Ces représentations permettent dès lors de faire du *model checking* efficacement.

Cependant, alors que des travaux ont été faits récemment pour rendre DEL utilisable en pratique en appliquant les idées du *model checking symbolique*, il n'y a pas eu de travaux de ce type pour PDEL.

C'est pourquoi nous allons nous intéresser à une représentation symbolique de la logique épistémique dynamique probabiliste. Pour cela, il nous faudra utiliser les concepts de représentation symbolique déjà employés pour la représentation symbolique de DEL, et les adapter à un contexte probabiliste.

Les travaux sur lesquels nous nous appuierons sont ceux de Malvin Gattinger et ses co-auteurs, présentés dans différents articles [VBvEGS18, Gat19] mais surtout dans sa thèse [Gat18b]. Sa représentation symbolique de DEL est ce qu'il nomme de manière évocatrice *Symbolic Model Checking for Dynamic Epistemic Logic* (SMCDEL), dont il a proposé une implémentation en Haskell [Gat18c] et une interface web [Gat18d].

Nous allons nous inspirer de cette représentation symbolique de DEL et en proposer une extension, qui permettra la vérification de modèle symbolique pour PDEL.

Dans SMCDEL, la représentation symbolique se base sur des formules booléennes, qui peuvent être vues comme des fonctions booléennes. Cette représentation est implémentable en pratique grâce à une structure de données adaptée (parmi d'autres) : le BDD (*Binary Decision Diagram*).

Dans le cas probabiliste, il nous faudra étendre le domaine des fonctions booléennes pour obtenir des fonctions pseudo-booléennes, qui peuvent être représentées en mémoire par différentes structures de données de la famille des diagrammes de décision, en particulier les *Algebraic Decision Diagrams* (ADDs) qui sont une généralisation des BDDs et qui connaissent des algorithmes de manipulation polynomiaux, tout comme les BDDs.

Le but étant de pouvoir mettre en pratique ces représentations, nous allons procéder à différentes expériences qui vont nous permettre de comparer les temps de génération, de vérification de modèles et de mise à jour de modèles pour les versions explicite et symbolique. Nous allons ainsi montrer que la représentation symbolique via les ADDs des structures de Kripke passe à l'échelle contrairement aux structures dites explicites même dans le cas d'exemples réalistes tels que le jeu de cartes Hanabi, ce qui pourrait ouvrir la voie à l'utilisation en pratique de techniques de planification épistémique dans un cadre probabiliste.

Cas d'application : Hanabi

DEL permet notamment de modéliser des types de jeux précis, que nous pouvons généralement décrire de la sorte ; ces jeux sont en règle générale :

à information imparfaite : Dans le jeu, des informations doivent être cachées, ou partielles, pour des agents. C'est à dire qu'aucun agent ne peut décrire de façon exacte l'état du jeu ;

à mémoire parfaite : Il n'y a pas d'oubli du passé, tous les agents se souviennent de l'historique observé. Ici, un état de connaissance résume l'histoire du jeu. C'est ce que l'on appelle la propriété de Markov ;

en tour par tour : C'est à dire que l'ordre des joueurs est connu et que chacun joue à tour de rôle (i.e. il n'y a pas d'actions simultanées).

Par ailleurs, lorsque nous nous intéressons à la planification épistémique, il est plus simple de manipuler des jeux qui ont une fin. Ces jeux sont alors dits :

bornés : c'est à dire que le jeu doit avoir une fin. C'est ce que l'on appelle communément un jeu à horizon fini. Par exemple, un jeu où des ressources sont dépensées à chaque tour, comme un jeu de cartes, est borné, puisque quand il n'y a plus de cartes, la partie est finie.

Enfin, tout l'interêt de ce type de jeu est de posséder la caractéristique suivante :

complétion des connaissances : les connaissances épistémiques sont primordiales dans ces jeux et leur acquisition durant la partie est importante pour résoudre le jeu. Ces connaissances peuvent être sur l'état du jeu lui-même ou sur les connaissances des autres agents.

Hanabi rentre parfaitement dans ce cadre d'étude.

Nous allons commencer par décrire de manière concise ses règles (règles complètes présentes en annexe page 232). Hanabi est un jeu de cartes coopératif où chaque joueur possède une main, qu'il ne voit pas. Néanmoins, il voit les cartes de ses coéquipiers. Chaque carte est définie par une couleur (rouge, bleu, vert, jaune, blanc) et une valeur (de 1 à 5 : 1 ... 5). Le but est alors de poser ses cartes sur la table afin de constituer des piles de cartes triées par couleur et par valeur à la manière d'une réussite. Néanmoins, si des cartes sont posées alors que l'action n'est pas légale dans la formation d'une des suites de couleurs, un jeton rouge est attribué à l'équipe de joueurs ( ). Au bout de trois erreurs, et donc trois jetons rouges, la partie est terminée. Pour plus de détails, vous pouvez vous référer à l'annexe page 232 qui décrit précisemment les règles d'Hanabi. Ainsi, chaque joueur peut jouer une carte (sans forcément en connaître toutes les informations), donner un indice aux autres joueurs quant à leurs propres cartes contre un jeton indice (symbolisé par), ou défausser une carte pour récupérer un jeton indice. Chaque carte posée correctement sur la table compte pour un point. Le but est de maximiser ce nombre.

La figure 1 présente un exemple d'une situation quelconque lors d'une partie d'Hanabi. Présentement, quatre joueurs possèdent quatre cartes chacun. Au centre des mains des joueurs, des cartes ont été jouées sur la table : une suite rouge avec 1 et 2 et une suite bleue avec 1. La réserve représente les jetons que les joueurs peuvent obtenir hors du jeu. Si un joueur joue une carte qui ne s'insère pas dans les suites (ici, toutes les cartes sauf un 3 ou un 2), ils recoivent un jeton rouge ( ). Au bout de trois jetons rouges, la partie est finie. Si une carte est défaussée, un jeton bleu ( ) passe de la réserve à leur banque de jetons bleus pour faire des annonces supplémentaires. Hanabi se place bien comme un jeu de cartes borné, car si la pioche est vide (ici, il reste 31 cartes dans la pioche , symbolisée en gris : p) la partie est finie.

La même situation est présentée à la figure 2, mais avec la vision du joueur J1, qui ne connait ni la valeur, ni la couleur de ses cartes. Nous voyons donc que les informations du jeu sont bien imparfaites pour les joueurs, car ils ne savent pas l'état réel du jeu : les dispositions de leurs mains et de la pioche leur sont inconnues.

Nous n'avons pas choisi ce jeu par hasard. Il est aussi actuellement étudié dans le domaine de l'intelligence artificielle. En effet, contrairement aux jeux d'Echecs

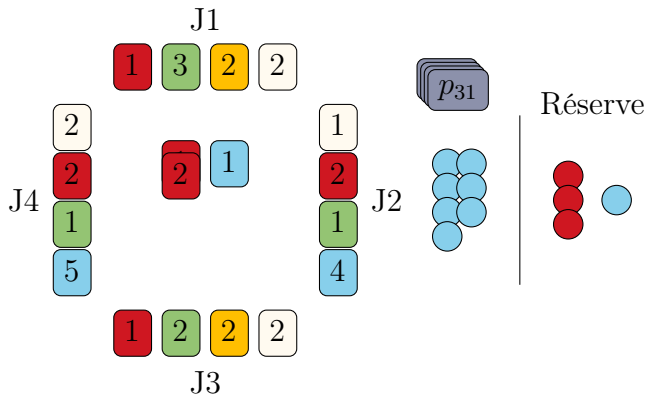


FIGURE 1 – Présentation d'une situation du jeu Hanabi

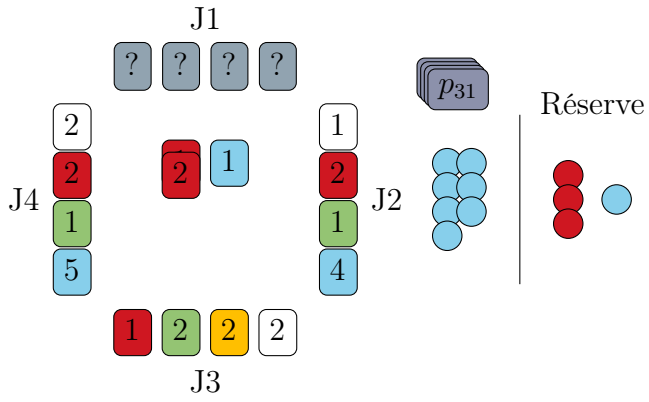


FIGURE 2 – Présentation d'une situation du jeu Hanabi, vue par le joueur J1

[CHhH02] et de Go [SHM+16], les techniques de planification à base de *deep learning* ont du mal à résoudre ce jeu, qui nécessite des connaissances épistémiques. Bien que des stratégies optimales de résolution aient été proposées [CDSD+15], elles n'ont rien d'humain, et seuls des agents artificiels peuvent appliquer les plans d'actions. C'est pourquoi nous souhaitons pouvoir représenter des connaissances probabilistes des agents afin de pouvoir tendre vers la génération de stratégies, qui seront certes moins efficaces, mais plus tangibles pour des êtres humains.

Comme les probabilités prennent une part importante du jeu Hanabi, que celui-ci nécessite la représentation des connaissances, et que ce jeu présente un interêt actuel dans la communauté scientifique, nous allons prendre ce jeu comme appui pour tester notre représentation symbolique probabiliste.

Plan et contribution

Dans la première partie (partie I), nous allons décrire tout le formalisme de la logique épistémique dynamique (chap. 1) puis sa représentation symbolique définie par Gattinger (chap. 2) qui se base sur l'exploitation de la structure de données BDD, que nous présenterons ensuite (sec. 3.1).

À titre de transition, nous présenterons une extension des fonctions booléennes utilisées dans SMCDEL : les fonctions pseudo-booléennes, qui sont représentables en mémoire par une structure similaire aux BDDs, les ADDs (sec. 3.2), ce qui nous permettra de présenter une représentation symbolique probabiliste dans la seconde partie (partie II). Dans celle-ci, nous commencerons par formaliser la logique épistémique dynamique probabiliste (sec. 4.1), avant de modifier quelque peu ses définitions (sec. 4.2) pour pouvoir introduire nos concepts de représentation symbolique probabiliste (chap. 5 et 6), que nous discuterons rapidement (sec. 6.4).

Nous montrerons ensuite (chap. 7) toute la mise en œuvre de notre *framework*, en commencant par présenter des travaux préliminaires qui ont eu lieu au début de la thèse (sec. 7.1), puis l'implémentation des BDDs, des ADDs, de SMCDEL et SMCDEL probabiliste (sec. 7.2), la modélisation du jeu Hanabi pour PDEL (sec. 7.3) et enfin des expérimentations qui comparent les deux représentations symboliques avec une représentation explicite de Hanabi (sec. 8).

L'intégralité de nos travaux présentés principalement dans la seconde partie (partie II) (représentation symbolique probabiliste) est très brièvement résumé dans un article publié dans les actes de la conférence internationale « Autonomous Agents and Multiagent Systems » (AAMAS) en 2022 et dont voici la référence :

[GNB22] *A Symbolic Representation for Probabilistic Dynamic Epistemic Logic*, Sébastien Gamblin, Alexandre Niveau et Maroua Bouzid, AAMAS2022

Nous terminerons enfin notre manuscrit en concluant puis nous évoquerons les perspectives à ces travaux.

Symboles

Nous allons définir quelques notations qui seront récurrentes dans notre rédaction.

$\mathfrak{A}$: Une liste d'agents qui interagissent ensemble et avec l'environnement ;

$\mathcal{L}$: symbole pour un langage, non spécifiés.

PS : Une liste d'atomes propositionnels $\{p,\ q,\ x,\ y\}$, qui peuvent être évalués à vrai ou faux ;

V : un vocabulaire, ensemble de atomes propositionnelles ;

$\mathbb{N}$: entiers naturels, commençant à 0 ;

$\mathbb{Q}$: nombres rationnels ;

$\mathbb{B}$: ensemble booléen : $\{0, 1\}$ ou $\{\bot, \top\}$;

2^X : ensemble des parties d'un ensemble X ;

ϕ, ψ : formules propositionnelles ou épistémiques.

Nous allons nous employer à comptabiliser tous les symboles et notations dans un glossaire présent page 219. Par ailleurs, une liste des acronymes utilisés sera aussi présente page 221.

Première partie

État de l'art du model checking pour DEL

1 | La logique épistémique dynamique

Nous allons commencer par étudier une des manières de représenter la connaissance dite épistémique des agents. Nous allons présenter une technique répandue pour cela : la logique modale (sec. 1.2), qui se base sur la logique propositionnelle, que nous allons présenter de manière préliminaire (sec. 1.1). Ici, nous étudierons spécifiquement la logique K (sec. 1.3) et les logiques nommées S5 et KD45 (sec. 1.4) définies à partir de celle-ci. Nous évoquerons les définitions du *model checking* et de la satisfaisabilité (sec. 1.5) puis, nous présenterons des logiques qui incluent un caractère dynamique : la logique d'annonce publique (sec. 1.6) et la logique épistémique dynamique (sec. 1.7).

Il existe d'autres types de logiques dites *non normales*, comme la logique des conditionnels ou les logiques avec des formalismes non monotones, que nous ne détaillerons pas ici. Pour plus de détails, vous pouvez consulter « Panorama de l'intelligence artificielle, ses bases méthodologiques, ses développements : représentation des connaissances et formalisation des raisonnements »[MPP14] qui effectue un balayage efficace et concis de tous ces formalismes.

Avant de présenter la logique modale, nous allons commencer par décrire la logique propositionnelle, sur laquelle sont basées les logiques modales.

1.1 Logique propositionnelle

Les formules d'un langage logique sont construites à partir d'un ensemble dénombrable d'atomes propositionnels (que l'on symbolisera par PS pour « Propositional Symbols ». PS est l'ensemble de tous les symboles disponibles ; et on y choisit souvent un vocabulaire fini noté V, inclus dans PS : $V \subseteq PS$. Pour la logique propositionnelle (sans modalité), voici comment on décrit le langage :

Définition 1.1 (Langage $\mathcal{L}_{\text{prop}}(V)$)**.** *Étant donné un ensemble d'atomes propositionnels V, le langage $\mathcal{L}_{\text{prop}}(V)$, défini pour le vocabulaire V, de la logique*

propositionnelle est défini par la grammaire suivante (sous forme BNF) :

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi \wedge \phi \text{ avec } p \in V.$$

Les opérateurs logiques utilisés sont ainsi : $\neg$ (négation) et $\wedge$ (conjonction). Les autres symboles que nous allons utiliser sont construits à partir du langage de la manière présentée dans la table 1.1.

Symbole	Construction
$p \vee q$ (ou)	$\neg(\neg p \wedge \neg q)$
$\bot$ (faux)	$\neg\top$
$p \to q$ (implication)	$\neg p \vee q$
$p \leftrightarrow q$ (équivalence)	$(p \to q) \wedge (q \to p)$

TABLE 1.1 – Symboles construits dans le langage $\mathcal{L}_{\text{prop}}(\{p, q\})$

Notons que tous ces opérateurs sont dits « vérifonctionnels », c'est à dire qu'ils obéissent à une table de vérité. En effet, les valeurs des atomes propositionnels présents dans les formules induisent la valeur de la formule.

Une *affectation des variables* est alors définie de la manière ci-dessous et permet de définir une sémantique sur le langage propositionnel $\mathcal{L}_{\text{prop}}$.

Définition 1.2. *Une* affectation *(aussi appelée* interprétation*) pour un vocabulaire V assigne à chaque atome propositionnel une valeur de vérité. Cette fonction est de type $v\colon V \to \{\top, \bot\}$. Avec un vocabulaire V fixé et par abus de notation, on identifie une affectation v à un sous-ensemble d'atomes propositionnels vrais, i.e. $v = \{p \in V\colon v(p) = \top\} \in 2^V$, dans ce cas, nous parlerons de* valuation*. Nous notons $\models$ pour la sémantique propositionnelle usuelle :*

$v \models \top$ *est vrai*

$v \models p$ *ssi* $p \in v$

$v \models \neg\phi$ *ssi* $v \not\models \phi$

$v \models \phi \wedge \psi$ *ssi* $v \models \phi$ *et* $v \models \psi$

Étant donné une valuation v et une formule ϕ, si nous avons $v \models \phi$, nous dirons alors que v est un *modèle* de ϕ ou que v satisfait ϕ.

Une formule ϕ est dite *valide* ssi elle est satisfaite par toutes les valuations possibles et on pourra écrire $\models \phi$. Deux formules ϕ et ψ sont dites *équivalentes sémantiquement* si elles sont satisfaites par exactement les mêmes affectations ; on notera alors $\phi \equiv \psi$.

1.2 Logique modale

Nous pouvons ajouter ce que l'on appelle des « opérateurs modaux » au langage propositionnel pour obtenir la « logique modale », notée $\mathcal{L}_{\text{EL}}$ (« EL »pour « *epistemic logic* »). L'opérateur communément utilisé est l'opérateur $\Box$ (*Box*), qui symbolise la nécessité.

Définition 1.3 (Langage $\mathcal{L}_{\text{EL}}(V)$)**.** *Étant donné un ensemble d'atomes propositionnels V, le langage $\mathcal{L}_{\text{EL}}$, de la logique modale est défini par la grammaire suivante (sous forme BNF) :*

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \Box\phi \text{ avec } p \in V.$$

Ainsi, la formule $\Box\phi$ est lue « ϕ est nécessaire ». Son dual est $\Diamond$ (*Diamond*) qui permet de symboliser la possibilité. Par exemple, lorsqu'il est necessaire que ϕ ($\Box\phi$) alors il est impossible (i.e. non possible) que non ϕ ($\neg\Diamond\neg\phi$). Ainsi, $\Diamond\phi$ peut être réécrit, et est équivalent à $\neg\Box\neg\phi$.

Par ailleurs, nous pouvons noter que : $\neg\Diamond\phi$ est lue : « il est impossible que ϕ »et que $\neg\Box\phi$ est lue : « il est *falsifiable* que ϕ »(i.e. l'existence de ϕ n'est pas nécessaire).

On appelle logique *aléthique* toute logique dont les modalités sont la *possibilité* et la *nécessité*.

Avec le langage $\mathcal{L}_{\text{EL}}$, il nous est maintenant possible d'écrire des formules plus complexes en combinant les opérateurs. Par exemple : $\Box p \rightarrow p$ signifie « si p est nécessaire, alors p est vrai », ou encore $\Box p \rightarrow \Box\Box p$ qui signifie « si p est nécessaire, alors p est nécessairement nécessaire ».

Traditionnellement, on appelle *syntaxe* d'une logique tout ce qui concerne les formules et les preuves et *sémantique* tout ce qui fait appel à des interprétations externes. Il apparaît ainsi une dualité entre une notion interne de *démonstrabilité* ($\vdash \phi$: « la formule ϕ est prouvable ») définie en termes syntaxiques (i.e. par des manipulations sur les mots et des règles de réécriture) et une notion externe de *validité* ($\models \phi$: « la formule ϕ est valide ») définie en termes sémantiques (i.e. en attribuant un sens aux formules par rapport à un référentiel).

De cette dualité résultent alors pour chaque logique deux problèmes de compatibilité, à savoir :

- Le problème de *cohérence* : « toute formule prouvable est-elle valide ? », c'est-à-dire « $\vdash \phi$ entraîne-t-elle $\models \phi$? » ;
- le problème de *complétude* : « toute formule valide est-elle prouvable ? », c'est-à-dire « $\models \phi$ entraîne-t-elle $\vdash \phi$? ».

Nous allons maintenant présenter les systèmes logiques K et ses extensions S5 et KD45, systèmes qui seront suffisants pour représenter la connaissance. Ils

utilisent le même langage, mais nous verrons qu'ils ne s'interprètent pas dans des structures qui vérifient les mêmes contraintes. En effet, par exemple –et nous verrons pourquoi, certaines formules ϕ ne seront pas valides dans K ($\not\models_K \phi$), mais le sont dans S5 ($\models_{S5} \phi$).

Commençons par la logique K.

1.3 Logique K

La logique épistémique, ou logique K (en l'honneur de Saul Kripke) a été imaginée par Hinttikka [Hin64]. Elle a été utilisée et développée à de nombreuses reprises. Nous pouvons trouver de bons résumés de cette logique épistémique dans « *Reasoning About Knowledge* « [FHMV95], dans « *Epistemic logic for computer science and Artificial Intelligence* »[MvdH95] et, encore plus récemment, dans « *Handbook of Epistemic Logic* »[vDHvdHK15], qui recensent toutes les nouveautés dans ce domaine.

Les *modèles* de K sont classiquement représentés par une structure relationnelle $\mathcal{F}$ (pour *frame*) [BMV02] qui relie des *mondes* entre eux grâce à une *relation* d'accessibilité.

Définition 1.4 (Frame)**.** *: Une frame notée $\mathcal{F}$ est un tuple $\langle W, R\rangle$ défini comme suit :*

- *W : un ensemble non vide dont les éléments sont considérés comme des mondes possibles ;*
- *$R \subseteq W \times W$, une relation d'accessibilité qui relie des mondes de W entre eux.*

Si un couple (w, w') est présent dans la structure $\mathcal{F}$, on note alors $(w, w') \in R$ ou wRw'. Basiquement, la relation d'*accessibilité* est appelée ainsi car elle définit quels mondes w' sont accessibles (i.e. être atteints) à partir d'un monde w. Néanmoins, la relation d'accessibilité R est aussi appelée relation d'*indistinguabilité*. La raison est simple : si nous avons wRw', alors il est considéré que le monde w est indistinguable, ou confondu, avec le monde w'.

Définition 1.5 (Frame pointée)**.** *Une frame pointée est un tuple : $\langle \mathcal{F}, w\rangle$ où $\mathcal{F} = \langle W, R\rangle$ est une frame et $w \in W$, qui désigne le monde réel, ou parfois appelé « désigné ».*

En ajoutant une fonction dite de *valuation* à une frame, nous obtenons la définition d'un *modèle de K*, que l'on appelle couramment *modèle de Kripke* dans la littérature. Néanmoins, nous sera pratique d'utiliser le terme *structure de Kripke* pour les logiques modales, au lieu de *modèle de Kripke*[FHMV95].

Définition 1.6. *Une structure de Kripke pour un vocabulaire V est un tuple $\mathcal{M} = \langle \mathcal{F}, \text{val} \rangle$ défini comme suit :*

- $\mathcal{F}$ *: une frame $\langle W, R \rangle$ (définition 1.4) ;*
- val *: fonction de valuation qui attribue à chaque monde de W une valuation (déf. 1.2), i.e.* $\text{val} : w \to 2^V$.

La fonction de valuation que l'on note val est notée généralement V et quelques fois π. Nous utilisons ici la notation val pour éviter la notation V que nous utilisons déjà pour un vocabulaire, ainsi que π que nous utiliserons plus tard dans notre exposé.

Tout l'interêt de la fonction de valuation est de pouvoir attribuer à chaque monde un ensemble de variables propositionnelles vraies, noté v, du vocabulaire donné V. On appelle alors pour un monde $w \in W$, $\text{val}(w) = v$ la valuation du monde w, qui permet de décrire les caractéristiques des mondes possibles.

Par abréviation et commodité, nous noterons une structure de Kripke comme suit : $\mathcal{M} = \langle W, R, \text{val} \rangle$, à la place de l'écriture attendue qui serait $\mathcal{M} = \langle \langle W, R \rangle, \text{val} \rangle$.

A l'instar de la *frame*, nous notons une structure de Kripke pointée de cette manière : $\langle \mathcal{M}, w \rangle$.

Exemple 1.7. *Dans la figure 1.1, nous pouvons voir un exemple de modèle de Kripke représenté de manière graphique, où le vocabulaire $V = \{p, q\}$, l'ensemble des mondes possibles $W = \{w_1, w_2\}$, la relation d'accessibilité $R = \{(w_1, w_1), (w_1, w_2)\}$ et la fonction de valuation* val *est telle que* $\text{val}(w_1) = \{p, q\}$ *et* $\text{val}(w_2) = \{\}$.

La valuation des mondes est notée par la présence et l'absence des variables. Dans les figures et dans les exemples, nous utilisons une notation plus explicite pour les valuations de sorte à ce que nous explicitions quels sont les atomes propositionnels présents ou absents. Ainsi, l'hypothétique valuation $\{q\}$ d'un monde w (sur $V = \{p, q\}$) est notée $\bar{p}q$ car $p \notin \text{val}(w)$ et $q \in \text{val}(w)$. De plus, nous représentons le monde pointé w_1 par une double bordure.

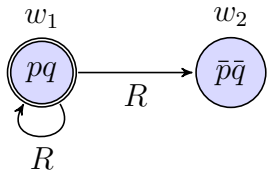


FIGURE 1.1 – Exemple d'un modèle de Kripke $\mathcal{M}$ (ex. 1.7)

Grâce à l'ajout de la fonction de valuation val aux *frames*, il devient alors possible d'interpréter des formules logiques dans ces structures. En effet, la relation de satisfaction détermine si une formule est vraie dans un monde d'un modèle. $\langle \mathcal{M}, w\rangle \models \phi$ se lit « dans $\mathcal{M}$, ϕ est vraie en w » et est définie récursivement dans la définition de la sémantique ci-dessous. La sémantique d'un langage logique permet d'interpréter ce langage dans un modèle de Kripke.

Définition 1.8. *La sémantique du langage $\mathcal{L}_{\mathrm{EL}}(V)$ pour un vocabulaire V est définie inductivement pour une structure de Kripke $\mathcal{M} = \langle W, R, \mathrm{val}\rangle$ définie aussi sur le vocabulaire V, avec $w \in W$ ($p \in V$, $\phi, \psi \in \mathcal{L}_{\mathrm{EL}}(V)$) :*

$$
\begin{array}{lll}
\langle \mathcal{M}, w\rangle \models p & \textit{ssi} & p \in \mathrm{val}(w) \\
\langle \mathcal{M}, w\rangle \models \neg\phi & \textit{ssi} & \langle \mathcal{M}, w\rangle \not\models \phi \\
\langle \mathcal{M}, w\rangle \models \phi \wedge \psi & \textit{ssi} & \langle \mathcal{M}, w\rangle \models \phi \textit{ et } \langle \mathcal{M}, w\rangle \models \psi \\
\langle \mathcal{M}, w\rangle \models \Box\phi & \textit{ssi} & \langle \mathcal{M}, w'\rangle \models \phi \textit{ pour tout } w' \textit{ tel que } (w, w') \in R
\end{array}
$$

Pour compléter cette définition, nous pouvons expliciter la sémantique de l'opérateur $\Diamond$ qui se fait de la sorte : « $\langle \mathcal{M}, w\rangle \models \Diamond\phi$ ssi il existe au moins un monde w' tel que $\langle \mathcal{M}, w'\rangle \models \phi$ avec $(w, w') \in R$ ». C'est à dire qu'il y a au moins un monde w' accessible depuis le monde w qui vérifie ϕ ; ce qui illustre bien la notion de la possibilité portée par l'opérateur $\Diamond$.

Cette sémantique permet une interprétation des formules. Ces formules peuvent alors être caractérisées de *valides* ou *satisfiables*.

Définition 1.9. *Une formule ϕ de $\mathcal{L}_{\mathrm{EL}}(V)$ est dite* valide *si elle est vraie pour tout monde w de toute structure $\mathcal{M}$ définie sur le vocabulaire V.*

Définition 1.10. *Une formule ϕ de $\mathcal{L}_{\mathrm{EL}}(V)$ est dite* satisfiable *s'il existe un modèle $\mathcal{M}$ pour le vocabulaire V qui possède un monde w qui safisfait ϕ, i.e. $\langle \mathcal{M}, w\rangle \models \phi$.*

Dans le modèle $\mathcal{M}$ de la figure 1.1, nous avons par exemple $\langle \mathcal{M}, w_1\rangle \models p \wedge \neg\Box q$, puisque p est vrai dans le monde w_1 et que q n'est pas vrai dans tous les mondes successeurs de w_1 par la relation R, c'est-à-dire w_1 et w_2. C'est ce que représente la nécessitation induite par l'opérateur $\Box$: il est nécessaire que q, i.e. dans tous les mondes w' indistingués de w, il doit y avoir aussi $q \in \mathrm{val}(w')$.

La méthode axiomatique permet de définir l'ensemble des lois logiques à partir d'axiomes logiques et de règles de déduction de telle façon que toutes les lois logiques soient ou bien un axiome ou bien une formule dérivée des axiomes avec un nombre fini d'applications des règles de déduction.

Une *axiomatisation* de l'ensemble des formules valides en logique K est présentée dans la table 1.2 [FHMV95].

Une formule est ainsi *démontrable* dans K si elle est dérivable à partir d'instances des axiomes P et K (numérotés 1 et 2), par des règles d'inférence. Voici comment sont lues les deux règles de réécriture R1 et R2 :

Axiome	Schéma
A1 | P	Toutes les tautologies propositionnelles
A2 | K	Axiome de distribution $\Box(\phi \to \psi) \to (\Box\phi \to \Box\psi)$

Symbole de la règle	Règle
R1 | MP	Modus ponens ϕ et $\phi \to \psi$ implique ψ
R2 | N | RN	Règle de généralisation ϕ implique $\Box\phi$

TABLE 1.2 – Axiomatisation et règles d'inférence des formules valides dans la logique K

- **MP :** on peut déduire toutes les conséquences logiques d'hypothèses ;
- **RN :** si ϕ est un théorème, alors il est nécessaire que ϕ.

Toute formule démontrable de K est ainsi valide (axiomatisation correcte) et toute formule valide de K est démontrable (axiomatisation complète).

Voici comment fonctionne la logique K. Elle nous permet effectivement d'écrire des formules logiques et de les interpréter dans une structure de Kripke et dans des mondes. Néanmoins les notions de connaissance ou de croyance n'y sont pas présentes. C'est ce que nous allons étudier avec des variantes de la logique K : les logiques S5 et KD45.

1.4 Logiques S5 et KD45

En ajoutant des contraintes sur les relations d'accessibilités des structures de Kripke, il est possible de considérer de nouvelles formules valides qui permettent de considérer d'autres systèmes logiques corrects. Dans la table 1.3, nous présentons cinq axiomes numérotés historiquement de A3 à A7 avec le schéma rendant la formule valide et la contrainte que cela implique sur la relation R. Les règles de réécriture quant à elles restent les règles R1 et R2.

Précisons que P (A1) et K (A2) venant du système K sont vrais dans tous les modèles. Ensuite, il est possible d'ajouter une sous-partie des axiomes pour former d'autres modèles logiques. Voici comment peuvent être lus les axiomes si l'on lit $\Box$ comme un opérateur de connaissance :

- **K :** un agent connaît toutes les conséquences logiques de sa connaissance ;

Axiome	Schéma	Contraintes sur R
A3 | T	$\Box\phi \to \phi$	Réflexif
	Axiome de vérité	$\forall x, xRx$
A4 | 4	$\Box\phi \to \Box\Box\phi$	Transitif
	Axiome d'introspection positive	$\forall x, y, z$ si xRy et yRz alors xRz
A5 | 5	$\neg\Box\phi \to \Box\neg\Box\phi$	Euclidien
	Axiome d'introspection négative	$\forall x, y, z$ si xRy et xRz alors yRz
A6 | D	$\Box\phi \to \Diamond\phi$	Sériel
	Axiome de consistance	$\forall x \exists y, xRy$
A7 | B	$\phi \to \Box\Diamond\phi$	Symétrique
	Axiome de Brouwer	$\forall x, y$ si xRy alors yRx

TABLE 1.3 – Axiomes permettant de constuire les systèmes axiomatiques usuels à partir de la logique K

- **T :** ce qu'un agent sait est vrai ;
- **4 :** un agent sait ce qu'il sait ;
- **5 :** un agent sait ce qu'il ne sait pas ;
- **D :** si un agent sait que ϕ, alors il est possible pour l'agent que ϕ ;
- **B :** ϕ implique que tout agent sait qu'il est possible que ϕ.

On nomme parfois l'axiome S comme étant l'ensemble des axiomes $\{K, T\}$, ce qui donne les logiques modales S4 et S5 combinant respectivement les ensembles d'axiomes $\{K, T, 4\}$ et $\{K, T, 5\}$.

Les logiques *aléthiques* sont représentées dans la figure 1.2 [Sch02]. Les logiques qui y sont présentées de sorte à ce que les logiques qui succèdent à une autre dans le graphe soient des sous-logiques de leurs parents, c'est à dire que tout théorème valable dans une logique est aussi un théorème dans les logiques successeures.

Chacun de ces axiomes peut être traduit par une contrainte sur les relations. T, 4, 5, D et B imposent respectivement les contraintes suivantes dans les relations : elles doivent être réflexives, transitives, euclidiennes, sériales, et symétriques. Nous illustrons via la figure 1.3 les relations induites par les axiomes. La relation de sérialité (axiome D) n'est pas présentée. Elle est en effet induite par la relation de réflexivité : la réflexivité impose à chaque monde d'avoir lui-même comme successeur, or la sérialité impose chaque monde d'avoir au moins un successeur. C'est pourquoi l'axiome T implique l'axiome D.

L'axiome 4 est redondant en présence des axiomes T et 5 –réflexivité et caractère euclidien impliquent la transitivité. Par ailleurs, l'axiome B n'est pas toujours explicité car elle résulte de la combinaison des axiomes T et 5.

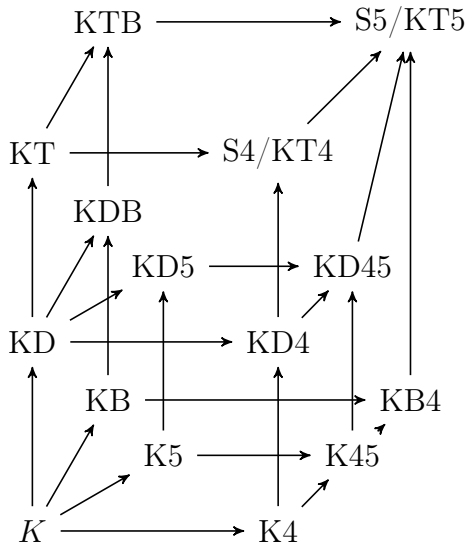


FIGURE 1.2 – Ensemble des logiques aléthiques et leurs relations d'inclusions.

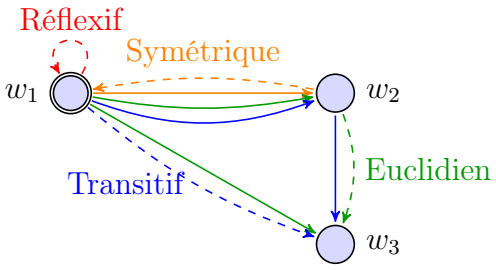


FIGURE 1.3 – Représentation des relations impliquées par les axiomes T (rouge), 4 (bleu), 5 (vert) et B (orange). Les arcs pleins impliquent les arcs pointillés de la même couleur.

Nous pouvons ainsi voir que les différentes combinaisons d'axiomes sont équivalents :

- La relation d'équivalence –à la fois réflexif, symétrique et transitif ($\{T, 4, B\}$) ;
- transitif, sériel et symétrique ($\{4, D, B\}$) ;
- réflexif et euclidien ($\{T, 5\}$).

La logique S5 (ou nommée KT45) est une extension de la logique K. Elle est considérée comme la logique représentant la connaissance, via sa modalité que l'on appelera *épistémique*. La classe des modèles de S5 est alors une sous-classe des modèles de K qui sont bisimilaires à un modèle dont la relation d'accessibilité est

une relation d'équivalence : symétrie, transitivité et réflexivité des relations (pour plus de détail sur cette bisimilarité, voir [FHMV95]).

En philosophie, la logique S5 est considérée comme la logique de la nécessité, en IA, c'est la logique de la connaissance. Une des critiques souvent émise sur S5 est qu'elle possède des propriétés trop fortes d'omniscience. En effet, ici, un agent connait toutes les conséquences logiques de ses connaissances. Un agent sait ce qu'il sait (introspection positive, axiome 4) et sait ce qu'il ne sait pas (introspection négative, axiome 5). Ainsi, le concept de vérité est binaire : soit l'agent sait, soit l'agent ne sait pas. Il n'y a pas de notion de croyance, par exemple.

Dans S5, $\Box$ est noté K (pour *Know*). Ainsi, la formule $\mathrm{K}p \wedge \neg \mathrm{K}q$ exprime le fait que l'agent sait que p et ne sait pas que q.

Exemple 1.11. *La formule* $(\mathrm{K}p \vee \neg \mathrm{K}p)$ *est lue « un agent sait p ou il ne sait pas que p ». Ainsi, il ne connait pas la valeur de p.*

Exemple 1.12. *Néanmoins, la formule* $\mathrm{K}p \vee \mathrm{K}\neg p$ *est lue « un agent sait p ou il sait non p ». Ainsi, l'agent sait si p est vrai –i.e. il connait la valeur de p.*

La logique modale peut traiter également de la croyance, et pas seulement de la connaissance. L'opérateur modal de base est généralement écrit B au lieu de K. Dans ce cas, cependant, l'axiome de vérité (A3 ou T) n'est plus correct -les agents ne croient pas tout le temps la vérité- et il est généralement remplacé par l'axiome de cohérence D. Les modalités sont alors appelées *doxastiques* et représentent alors la croyance via la combinaison d'axiomes $\{K, D, 4, 5\}$.

Pour la suite, nous utiliserons au maximum l'opérateur $\Box$ au lieu des opérateurs de connaissance K ou de croyance B pour rester dans les cadres de structures de Kripke les plus générales possibles. Néanmoins, il nous arrivera d'utiliser des exemples avec K.

Maintenant, nous souhaitons rendre la structure *multi-agents*. Pour cela, nous rendons la structure de Kripke multimodale, c'est-à-dire en y ajoutant plusieurs modalités $\Box$ (ou K, ou B) au lieu d'une seule. Ainsi, nous allons définir une modalité $\Box$ pour chaque agent $a \in \mathfrak{A}$. Ainsi, $\mathcal{M} = \langle W, R, \mathrm{val}\rangle$ devient $\mathcal{M} = \langle W, R_1 \dots R_n, \mathrm{val}\rangle$, où $n = |\mathfrak{A}|$. Pour tout $a \in \mathfrak{A}$, nous aurons donc $R_i \subseteq W \times W$. Néanmoins, pour plus de commodité, nous allons définir de la sorte : $R \subseteq (\mathfrak{A} \times W \times W)$. Ainsi, on écrira R indicé de l'agent en question, de sorte que : $R_a = \{(w, w') \in W^2 \mid (a, w, w') \in R\}$ et garder $\mathcal{M} = \langle W, R, \mathrm{val}\rangle$, qui sera implicitement toujours multiagent.

Grâce à cela, nous pouvons ainsi écrire des formules de connaissances avec plusieurs agents, par exemple :

Exemple 1.13. *Dans un cadre de connaisance (S5, avec l'opérateur* K*), la formule* $\mathrm{K}_a(\mathrm{K}_b p \vee \mathrm{K}_c p)$ *est lue « L'agent a sait que l'agent b ou l'agent c sait p » (Ou non exclusif : les deux agents b et c peuvent savoir).*

Avec les opérateurs K_a avec $a \in \mathfrak{A}$, nous obtenons le langage $\mathcal{L}_{\mathrm{S5}}$ suivant :

Définition 1.14 (Langage $\mathcal{L}_{\mathrm{S5}}$)**.** *$\mathcal{L}_{\mathrm{S5}}$ est le langage défini comme suit :*

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \mathrm{K}_a\phi, \text{ avec } p \in PS \text{ et } a \in \mathfrak{A}.$$

Nous noterons la grande similarité avec le langage $\mathcal{L}_{\mathrm{EL}}$. Ici, il s'agit simplement d'ajouter une modalité pour chaque agent $a \in \mathfrak{A}$ et donc d'obtenir un langage multimodal en restant dans la logique S5.

Les notations suivantes seront utilisées : $\mathrm{Kw}_a\phi := \mathrm{K}_a\phi \vee \mathrm{K}_a\neg\phi$ (« Kw »pour « *know whether* »), c'est à dire, l'agent a connait la valeur de ϕ, ou encore l'opérateur dual de K, qui est une réécriture de $\Diamond$, noté $\hat{\mathrm{K}}$ et prononcé « K chapeau » : $\hat{\mathrm{K}}_a := \neg\mathrm{K}_a\neg\phi$.

Exemple 1.15. *Pour coller aux caractéristiques d'une structure de Kripke dans S5, reprenons l'exemple de la figure 1.1, mais avec les arcs modifiés de sorte à avoir des arcs réflexifs à cause de l'axiome T, ainsi que transitifs par l'axiome 4 et symétrique par l'axiome B. Faisons cela pour deux agents :* $\mathfrak{A} = \{b,\ r\}$ *(bleu et rouge).*

Nous avons donc une nouvelle structure de Kripke $\mathcal{M}$ *avec de nouvelles relations telles que* $R_b = \{(w_1, w_1), (w_2, w_2)\}$ *et* $R_r = E \times E$ *(l'ensemble des arcs possibles).*

Nous obtenons ainsi la figure 1.4. Seulement dans cet exemple, nous avons choisi de différencier les connaissances des agents b et r. En effet, ici, l'agent r ne différencie pas les mondes w_1 *et* w_2 *car ces deux mondes sont reliés entre eux. Par contre, l'agent b distingue bien ces deux mondes. Ainsi, il est clair que la formule suivante est satisfaite :* $\langle \mathcal{M}, w_1 \rangle \models \neg\mathrm{K}_r p \wedge \mathrm{K}_b p \wedge \mathrm{K}_b\neg\mathrm{K}_r p$ *(il en va de même pour q).*

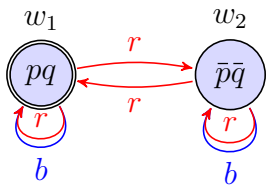


FIGURE 1.4 – Exemple d'un modèle de Kripke $\mathcal{M}$ dans S5 (ex. 1.15)

En outre, des opérateurs peuvent étendre le langage, permettant de modéliser la connaissance commune de la sorte :

Définition 1.16 (Tout le monde sait (*Everyone*))**.**

$$E_A(\phi) = \bigwedge_{a \in A} \mathrm{K}_a\phi,\ A \subseteq \mathfrak{A}$$

Définition 1.17 (Connaissance commune)**.** $C_A(\phi) = \bigwedge_{k \in \mathbb{N}} E_A^k\phi$*, avec* $E_A^0\phi = \phi$*,*

$$E_A^{k+1}\phi = E_A E_A^k \phi$$

En ajoutant cet opérateur modal au langage $\mathcal{L}_{\mathrm{EL}}$, nous obtenons le langage $\mathcal{L}_{\mathrm{S5CK}}$ suivant :

Définition 1.18. $\mathcal{L}_{\mathrm{S5CK}}$ *est le langage :*

$$\phi \coloneqq \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \mathrm{K}_a\phi \mid C_A\phi,\ p \in PS,\ a \in \mathfrak{A}$$

L'opérateur C est souvent employé dans les langages dérivant de la logique épistémique. Il peut néanmoins être enlevé pour simplifier le langage et ses propriétés.

Maintenant que nous avons vu ce que sont les structures de Kripke et comment elles peuvent nous servir, nous allons définir trois problèmes qui nous permettent de caractériser les propriétés computationnelles d'un langage.

1.5 Model checking, satisfaisabilité et validité

Voici trois problèmes fondamentaux : la vérification de modèle (*model checking* ou MC), la satisfaisabilité (SAT) et la validité. Ces trois questions sont longuement étudiées dans le domaine des automates ou structures de Kripke.

Définition 1.19 (Problème de vérification de modèle)**.**

Entrées : une structure de Kripke pointée $\langle \mathcal{M}, w \rangle$ *et une formule* $\phi \in \mathcal{L}$

Sortie : oui si $\langle \mathcal{M}, w \rangle \models \phi$*, sinon non.*

Définition 1.20 (Problème de satisfiabilité, noté SAT)**.**

Entrées : $\phi \in \mathcal{L}$

Sortie : oui s'il existe un modèle de Kripke $\mathcal{M}$ *et un monde* $w \in W$ *de sorte que* $\langle \mathcal{M}, w \rangle \models \phi$*, sinon non.*

Définition 1.21 (Problème de validité)**.**

Entrées : $\phi \in \mathcal{L}$

Sortie : oui si pour tout $w \in W$ *de toute structure de Kripke* $\mathcal{M}$*, on a* $\langle \mathcal{M}, w \rangle \models \phi$*, sinon non.*

Logique	$n = 1$	$n \geqslant 2$	avec la connaissance commune C
K	PSPACE-c	PSPACE-c	EXPTIME-c
S5 / KD45	NP-c	PSPACE-c	EXPTIME-c
			"-c" pour "complet"

TABLE 1.4 – Complexité du problème SAT pour la logique K et S5 en fonction du nombre d'agents (n) et de l'opérateur C[HM92, FHMV95]

Avec ces définitions, il est possible de comparer les logiques que nous allons étudier par la suite.

Pour les deux logiques que nous avons vues précédemment, voici les complexités associées pour le problème SAT présentées dans la table 1.4. Les colonnes présentées dépendent de n, le nombre d'agents, peu importe la profondeur des formules. La complexité du problème de vérification de modèle est quant à elle polynomiale, si le modèle est donné en entrée de façon explicite.

Maintenant, nous allons voir comment il est possible d'ajouter des événements à une logique modale. En effet, nous aimerions pouvoir, par exemple dans Hanabi, annoncer que le joueur 2 possède un 2 rouge. Une logique qui permet cela -en partie- est la logique nommée PAL.

1.6 Logique d'annonces publiques

Brièvement, nous pouvons résumer la logique PAL pour la logique épistémique, avec l'ajout d'annonces. Ainsi, l'opérateur $[\phi!]\psi$ est ajouté au langage de $\mathcal{L}_{\text{EL}}$ et signifie : « si ϕ est publiquement annoncé, alors ψ sera vrai après cette annonce ». On obtient ainsi le langage suivant :

Définition 1.22 (Langage $\mathcal{L}_{\text{PAL}}$)**.** *Étant donné un vocabulaire V, le langage $\mathcal{L}_{\text{PAL}}(V)$ de la logique d'annonce publique est défini par la grammaire suivante (sous forme BNF) :*

$$\phi \coloneqq p \mid \neg\phi \mid \phi \wedge \phi \mid \Box_a\phi \mid [\phi!]\phi, \text{ avec } p \in V \text{ et } a \in \mathfrak{A}$$

Notons que l'opérateur $[\phi!]$ possède aussi son dual : $\langle\phi!\rangle\psi \coloneqq \neg[\phi!]\neg\psi$.

Définition 1.23. *La notion de satisfaction de formule est aussi identique à celle de la logique épistémique (vu 1.8), en y précisant le nouvel opérateur :*

$$\langle\mathcal{M}, w\rangle \models [\phi!]\psi \text{ ssi } \langle\mathcal{M}, w\rangle \not\models \phi \text{ ou si } \langle\mathcal{M}[\phi!], w\rangle \models \psi$$

où l'application d'une annonce sur une structure de Kripke est notée : $\mathcal{M}[\phi!]$ et est définie par le tuple $\langle W[\phi!], R[\phi!], \text{val}[\phi!]\rangle$ avec :

- $W[\phi!] \coloneqq \{w' \in W \mid \langle \mathcal{M}, w' \rangle \models \phi\}$;
- $\forall w_1, w_2 \in W[\phi!]^2,\ w_1 R[\phi!] w_2$ *ssi* $w_1 R w_2$;
- val$[\phi!]$ *conserve les valuations des mondes.*

Cette notion d'annonce permet en fait de conserver tous les mondes où ϕ est vraie, agissant tel un filtre sur les mondes initiaux.

L'utilisation de PAL nous permet donc de raisonner sur des annonces en plus d'exploiter la logique épistémique. Néanmoins, elle ne permet pas de modifier les mondes d'une structure de Kripke, en appliquant des actions dites *ontiques* (i.e. qui agissent concrètement, en influençant le monde).

Par exemple, soit p le fait qu'une pièce soit sur le côte pile. Un exemple d'action qui permettait de modifier l'état du monde, par le lancer d'une pièce serait de pouvoir lancer de nouveau la pièce pour changer son côté actuel.

Notons que la formule $[\phi!]\psi$ est forcément vraie si ϕ est fausse : l'annonce d'une formule fausse est incompatible avec l'hypothèse d'annonces véridiques, et donc chaque formule est vraie après l'annonce d'un mensonge (*principe d'explosion* ou *ex falso quodlibet*).

Néanmoins, la complexité de résolution des problèmes de décision pour PAL sont intéressants puisque SAT est NP-complet en mono-agent et PSPACE-complet en multiagent [Lut06]. Quant au *model checking* il est dans P [KvB04]. Par ailleurs, nous pouvons noter qu'il existe de nombreuses extensions de PAL : PAL+C qui intègre un opérateur de connaissance commune ou EAL (*Epistemic Action Logic*) qui intègre des actions avec précondition.

Voyons maintenant une extension plus poussée de PAL qui permet de modéliser des événements complexes et d'ainsi modifier une structure de Kripke en conséquence. Notamment, avec PAL, il était impossible de modéliser des actions du type « l'agent pioche une carte mais ne la voit pas », mais ce sera typiquement le cas avec cette extension nommée *logique épistémique dynamique*, ou DEL.

1.7 Logique épistémique dynamique

DEL, ou *Dynamic Epistemic Logic*, utilise $\mathcal{L}_{\text{EL}}$ et est plus complète que $\mathcal{L}_{\text{PAL}}$ par l'ajout d'événements qui permettent de mettre à jour les structures de Kripke en modifiant les valuations des mondes. DEL a été défini dans l'article « *The logic of public announcements, common knowledge and private Suspicions* » [BMS98] puis ensuite longuement développé dans le livre « *Dynamic Epistemic Logic* » [vDvdHK07].

Avant de définir le langage utilisé, nous allons décrire comment sont modélisés les actions, ou modèles d'événements.

Un modèle d'événement est défini à l'instar d'une structure de Kripke, avec une structure relationnelle : la frame $\mathcal{F}$. Seulement, cette structure ne portera pas sur l'indistinguabilité de mondes possibles, mais sur des événements possibles. Chaque événement constitutif du modèle aura des conditions d'applications et des capacités à modifier les valeurs de vérité d'atomes propositionnels.

Décrivons cela plus formellement. C'est donc une frame $\mathcal{F}$ qui relie des événements « élémentaires » ou « atomiques » $e \in E$ grâce à $R^{\mathcal{E}}$ qui lie deux événements indistinguables.

Définition 1.24 (Modèle d'événements)**.** *Un modèle d'événements $\mathcal{E}$ défini pour un vocabulaire V est un tuple $\langle \mathcal{F}, \text{pre}, \text{post} \rangle$ avec :*

- *$\mathcal{F}$ une frame $\mathcal{F} = (E, R^{\mathcal{E}})$;*
- *pre : fonction de precondition , $E \to \mathcal{L}_{\text{EL}}(V)$;*
- *post : fonction de postcondition, $V \times E \to \mathcal{L}_{\text{prop}}(V)$.*

La fonction de précondition pre associe à chaque événement $e \in E$ une formule épistémique dans $\mathcal{L}_{\text{EL}}$ qui permet de savoir quel est l'ensemble des mondes dans lesquels l'événement peut se produire. Autrement dit, pour s'appliquer à un monde w, la formule $\text{pre}(e)$ liée à l'événement $e \in E$ devra être vraie dans ce monde. Cette précondition est similaire à la formule annoncée dans les événements de PAL (déf. 1.22), dans la mesure où la précondition d'un événement atomique agit comme un filtre sur les mondes.

post est quant à elle une fonction représentant les postconditions, c'est-à-dire qu'elle décrit pour chaque action $e \in E$, les conséquences de l'événement e sur le monde w, ce qui se fait par un changement de valuations des variables propositionnelles du monde en question.

À la place d'écrire $\mathcal{E} = \langle \mathcal{F}, \text{pre}, \text{post} \rangle$, nous écrirons $\mathcal{E} = \langle E, R^{\mathcal{E}}, \text{pre}, \text{post} \rangle$. Par ailleurs, de la même manière que la structure de Kripke, nous écrirons un modèle d'événement pointé de la sorte : $\langle \mathcal{E}, e \rangle$, $e \in E$.

Par commodité, il nous arrivera d'abréger "$post(e)(p) = \phi$" par "$p \leftarrow \phi$", notamment dans les représentations graphiques des modèles d'événements, comme dans la figure 1.5.

L'événement atomique pointé indique quel événement se produira réellement lors de l'application du modèle d'événements. Dans l'exemple ci-dessous, cet événement est représenté avec un double cadre.

Exemple 1.25. *Voici un exemple de modèle d'événement sur la figure 1.5. Ici, $\mathcal{E} = \langle E, R^{\mathcal{E}}, \text{pre}, \text{post} \rangle$, avec $E = \{e_1, e_2\}$, $R_b^{\mathcal{E}} = E \times E$ (i.e. tous les arcs possibles), $R_r^{\mathcal{E}} = \{(e_1, e_1), (e_2, e_2)\}$ puis $\text{pre}(e_1) = \text{pre}(e_2) = \top$ et $\text{post}(p)(e_1) = \top$, $\text{post}(p)(e_2) = \bot$. Par défaut, toute variable propositionnelle p non explicitée pour un événement atomique e est définie de la sorte : $\text{post}(p)(e) = p$. Ainsi, la valeur de la variable n'est pas modifiée.*

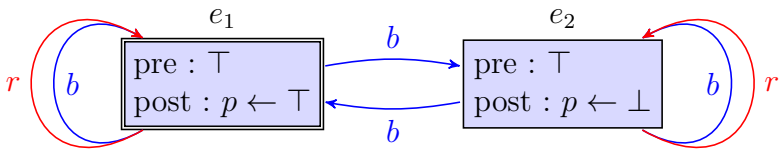


FIGURE 1.5 – Exemple de modèle d'événement (ex. 1.25)

On emploie ces modèles d'événements via un opérateur similaire à celui de PAL. Le langage correspondant s'appelle $\mathcal{L}_{\text{DEL}}$.

Définition 1.26 (Langage $\mathcal{L}_{\text{DEL}}$)**.** *Le langage $\mathcal{L}_{\text{DEL}}(V)$ de la logique épistémique dynamique sur un vocabulaire V est définie par la grammaire suivante sous forme de BNF :*

$$\Phi \coloneqq p \mid \neg\Phi \mid \Phi \wedge \Phi \mid \Box_a\Phi \mid [\mathcal{E}, e]\Phi,\ p \in PS,\ a \in \mathfrak{A},\ e \in E$$

On appellera ainsi $\mathcal{E}$ un événement et $e \in E$ un événement atomique, constituant $\mathcal{E}$. Notons que le dual de $[\mathcal{E}, e]\phi$ est $\langle\mathcal{E}, e\rangle]\phi = \neg[\mathcal{E}, e]\neg\phi$ (comme pour $\Diamond$ et $\Box$)

Ce langage peut être étendu, à l'instar du langage $\mathcal{L}_{\text{EL}}$, avec la connaissance commune :

Définition 1.27 (Langage $\mathcal{L}_{\text{DELCK}}$)**.** $\mathcal{L}_{\text{DELCK}}$ *est le langage défini comme suit :*

$$\Phi \coloneqq p \mid \neg\Phi \mid \Phi \wedge \Phi \mid \Box_a\Phi \mid C_A\Phi \mid [E, e]\Phi,\ p \in X,\ a \in \mathfrak{A},\ e \in E$$

La modification des connaissances d'une structure de Kripke via un modèle d'événement est faite par le « *product update* », ou « mise à jour des connaissances/croyances » qui est basé sur le produit cartésien des mondes/événements des différentes structures $\mathcal{M}$ et $\mathcal{E}$.

Définition 1.28 (Product update)**.** *Soit $\langle\mathcal{M}, w\rangle$ une struture de Kripke pointée avec $\mathcal{M} = \langle W, R, \text{val}\rangle$ définie sur le vocabulaire V et $\langle\mathcal{E}, e\rangle$ un modèle d'événement pointé avec $\mathcal{E} = \langle E, R^{\mathcal{E}}, \text{pre}, \text{post}\rangle$. Le product update de $\langle\mathcal{M}, w\rangle$ et $\langle\mathcal{E}, e\rangle$ est la structure de Kripke pointée $\langle\mathcal{M} \otimes \mathcal{E}, (w, e)\rangle$, avec $\mathcal{M} \otimes \mathcal{E} = \langle W^{\otimes}, R^{\otimes}, \text{val}^{\otimes}\rangle$, défini comme suit :*

- $W^{\otimes} = \{(w, e) \in W \times E \mid \langle\mathcal{M}, w\rangle \models \text{pre}(e)\}$;
- $(w, e)\ R_i^{\otimes}\ (w', e')$ ssi wR_iw' et $eR_i^{\mathcal{E}}e'$, *pour tout* $(w, w') \in W^2$ *et* $(e, e') \in E^2$;

- $\text{val}^{\otimes}((w,e)) = \{p \in V \mid \langle \mathcal{M}, w\rangle \models \text{post}(p)(e)\}$ *pour tout* $(w,e) \in W^{\otimes}$.

Dans le langage $\mathcal{L}_{\text{DEL}}$, la formule $[\mathcal{E}, e]\ \phi$ est lue « après l'exécution du modèle d'événement pointé $\langle \mathcal{E}, e\rangle$, ϕ est vraie ». Ce nouvel opérateur est formalisé ainsi :

Définition 1.29. *La sémantique du langage* $\mathcal{L}_{\text{DEL}}$ *est identique à celle de* $\mathcal{L}_{\text{EL}}$ *(déf. 1.3), en y ajoutant la sémantique de l'opérateur* $[\mathcal{E}, e]$ *:*

$$\langle \mathcal{M}, w\rangle \models [\mathcal{E}, e]\phi \text{ ssi } \langle \mathcal{M}, w\rangle \models \text{pre}(e) \implies \langle \mathcal{M} \otimes \mathcal{E}, (w,e)\rangle \models \phi$$

De fait, de par cette définition du *product update* (déf. 1.28), nous pouvons voir que l'exemple 1.5 modélise un pile ou face. En effet, les préconditions de e_1 et e_2 étant $\top$, les deux événements seront appliqués à tous les mondes. Puis la variable p, qui représente le « pile » de la pièce, sera soit vrai, soit faux. Nous pouvons également voir que l'agent b ne différencie pas les deux actions, mais ce n'est pas le cas de l'agent r, qui lui, saura quel est le résultat du lancer de pièce.

Exemple 1.30. *La figure 1.6 reprend les figures 1.4 et 1.5 (ex. 1.15, page 21) afin de faire le* product update *de* $\langle \mathcal{M}, w_1\rangle$ *avec* $\langle \mathcal{E}, e_1\rangle$*.*

De cette manière, l'agent t *qui ne différenciait pas les mondes* w_1 *et* w_2*, ne les différenciera toujours pas, car il confondra* (w_1, e_1) *et* (w_2, e_1) *ainsi que* (w_1, e_2) *et* (w_2, e_2)*, mais il saura quelle est la valeur de* p*.*

À l'inverse, l'agent b *qui connaissait la valeur de* p*, ne la connaîtra plus. En effet, il ne différencie pas les événements atomiques* e_1 *et* e_2*, ainsi, il ne différenciera pas les conséquences de ces événements.*

Les arcs sont répercutés dans la structure résultante $\mathcal{M} \otimes \mathcal{E}$ *si et seulement si leurs deux arcs parents sont bel et bien présents dans chacun des* $\mathcal{M}$ *et* $\mathcal{E}$*. Par exemple, l'arc* $(w_1, e_1)R_b^{\mathcal{E}}(w_1, e_2)$ *est présent parce que* $w_1 R_b w_1$ *et aussi* $e_1 R_b^{\mathcal{E}} e_2$*.*

Notons que la structure de Kripke $\mathcal{M}$ *et le modèle d'événements* $\mathcal{E}$ *ont tous deux des relations d'accessibilité soumises aux contraintes de S5 et par conséquent, leur product update crée aussi une structure S5 –cela fonctionne pour K et S5, mais pas KD, KD45 ou d'autres logiques.*

Notons un problème qui apparaît lors de la mise à jour des connaissances présentée ici. Sur la figure 1.6, nous avons deux mondes reliés par six arcs qui sont mis à jour par deux événements atomiques, ce qui nous conduit à quatre mondes reliés par seize arcs. Imaginons maintenant mettre à jour la structure de Kripke obtenue $\mathcal{M} \otimes \mathcal{E}$ plusieurs fois par l'événement $\mathcal{E}$; nous aurons alors huit mondes, puis seize, puis trente-deux, etc. Nous voyons bien que le produit est combinatoire. Pour représenter le fait de lancer un dé, il faudrait six événements atomiques, ou pour piocher une carte, il en faudrait cinquante-deux. Le nombre de mondes ne serait potentiellement plus multiplié par deux, mais par six ou cinquante-deux !

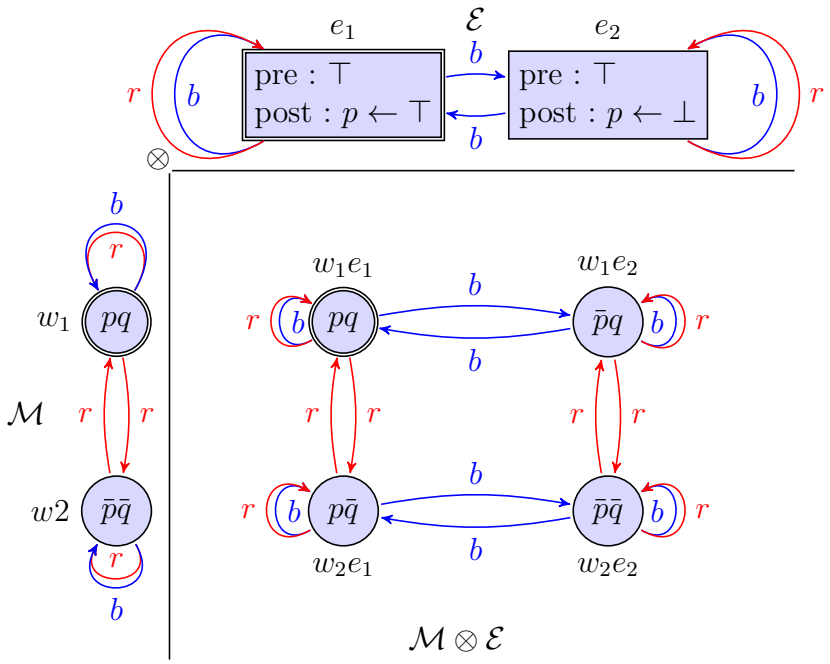


FIGURE 1.6 – Exemple de *product update* (ex. 1.30)

Néanmoins, peu importe le facteur, s'il est supérieur à un, il apparaît une *explosion combinatoire* du nombre de mondes.

Il n'est même pas obligatoire d'appliquer des modèles d'événements pour que cette explosion combinatoire ait lieu. En effet, c'est le nombre de valuations possibles des mondes qui permet de modéliser des mondes possibles différents. De cela découle la création de structures de Kripke énormes qu'il est difficile de représenter informatiquement.

Exemple 1.31. *Si l'on veut représenter les connaissances d'un joueur au jeu du "Démineur", dans une grille de seulement* 9×9 *cases avec* 10 *bombes, cela revient à calculer le coefficient binomial* $\binom{81}{10}$ *qui est égal à seulement* $\frac{81}{10!(81-10)!} = 6.81631e^{18}$. *Serait-il souhaitable de modéliser tous les mondes possibles pour* 30×16 *cases avec* 99 *bombes ? Cela ferait seulement* $5,22833e^{260}$ *mondes possibles. Nous pourrions aussi compter le nombre d'arêtes ; il suffira de mettre ces nombres au carré.*

Dans le cadre d'Hanabi, nous avons besoin de représenter la connaissance et la connaissance sur les connaissances de autres. Dans cette optique, la logique S5 -ou ses variantes- semble toute trouvée pour modéliser cela. D'autant plus que DEL est très élégant et expressif pour modéliser des problèmes en modèles de Kripke

et leurs modifications dans le temps par des modèles d'événements. Par ailleurs, il semble particulièrement intéressant d'imaginer utiliser les extensions probabilisées de DEL pour modéliser des probabilités dans un jeu de cartes tel que Hanabi, et c'est ce que nous allons faire par la suite.

Avant de pouvoir modéliser ce genre de jeux probabilistes grâce à PDEL, nous allons devoir pouvoir étudier des manières de représenter les structures de Kripke (de DEL) de manière efficace, i.e. pouvoir contourner l'explosion combinatoire.

L'explosion combinatoire empêche une représentation efficace en mémoire, il est donc complexe d'implémenter et manipuler les structures de Kripke de manière explicite (i.e. en l'implémentant tel quel) dans un programme.

C'est ce que nous allons voir dans la prochaine partie.

2 | Model checking symbolique pour DEL : SMCDEL

2.1 Avant-propos : état de l'art

DEL est un *framework* datant de 1998 [BMS98] et a été généralisé en 2007 [vDvdHK07]. Le logiciel DEMO [vE08] (2008) et son successeur optimisé DEMO-S5 [vE14] (2014) sont conçus spécifiquement pour utiliser ce formalisme. Ces implémentations, dites *explicites*, qui suivent les définitions mathématiques et n'apportent aucune optimisation calculatoire, offrent une maniabilité accrue car elles permettent la spécification directe dans des langages dynamiques, mais possèdent des performances réduites. Cela illustre le fait que DEL a été créé dans l'esprit de la logique et la philosophie, mais n'a pas été étudié pour être appliqué directement sur un ordinateur : les modèles sont ici des objets mathématiques abstraits dont la taille n'a pas d'importance.

L'objectif de la représentation symbolique est de décrire un modèle de manière la plus compacte possible pour ne pas avoir à le représenter en entier, contrairement à une représentation explicite, qui suit précisémment les définitions mathématiques.

Certains problèmes modélisables avec DEL ont été représentés et résolus de manière symbolique :

- le problème du « dîner des cryptographes » [vdMS04] (2004) ;
- le problème des « cartes russes » [vDvdHvdMR06] (2006) ;
- le problème « somme et produit » [LSSC08] (2008) ;

Dans les trois cas, les techniques utilisées ne sont pas générales et répondent spécifiquement aux besoins du problème. Néamoins, elles utilisent toutes des représentations symboliques exploitant des structures de données adaptées et couramment utilisées dans le *model checking* symbolique : les OBDDs (plus de détails section 3.1).

Dans la foulée, de même types de vérificateurs de modèles apparaissent utilisant la représentation symbolique, notamment pour la logique temporelle [SSL07a, LR06], toujours en utilisant des OBDDs.

Des parallèles entre la logique épistémique et la logique épistemique temporelle ont été établis : comme des *model checkers* existaient pour la logique épistémique temporelle, il a alors été possible d'effectuer du *model checking* sur DEL [vDvdHR13] (2013). Ce n'était néanmoins pas une représentation symbolique de DEL à proprement parler et aucune implémentation de ce parallèle n'a été, à notre connaissance, proposée.

Ce sont Malvin Gattinger, Johan Van Benthem, Jan Van Eijck et Kaile Su, qui ont introduit la première représentation symbolique généralisée de DEL, nommée *SMCDEL* (*Symbolic Model Checking for Dynamic Epistemic Logic*), en utilisant d'abord des variables d'observations pour modéliser la logique S5 [vBvEGS15], puis en élargissant la modélisation à KD45 [VBvEGS18], et enfin en ajoutant les postconditions au modèle symbolique [Gat19]. Tous ces travaux sont présentés dans la thèse de Gattinger [Gat18b]. De manière générale, le but est de représenter les mondes et les relations d'une structure de Kripke grâce à ce qui est appelé des *lois*, qui sont des formules propositionnelles.

Alors que les approches symboliques s'intéressent en général à représenter des exemples classiques de la littérature de DEL (les enfants sales, le dîner des cryptographes, etc) et implémenter le vérificateur de modèle [VBvEGS18, Gat18c], une autre approche a été proposée en 2015, basée sur les *programmes mentaux* [CS15]. Cette alternative s'interesse plus aux aspects calculatoires en étudiant les classes de complexité des opérations.

Cette approche est basée sur les programmes mentaux utilisés dans DL-PA (*Dynamic Logic of Propositional Assignments*) [BHT13], une version de PDL (*Propositional Dynamic Logic*). Les modèles de PDL sont des systèmes de transition : chaque transition est étiquetée avec le nom d'un programme et indique l'exécution possible d'un programme d'un état à un autre. Un programme π est défini inductivement ainsi :

$$\pi \coloneqq p \leftarrow \bot \mid p \leftarrow \top \mid \beta? \mid \pi;\pi' \mid \pi \cup \pi'$$

où $p \leftarrow \bot$ et $p \leftarrow \top$ effectent une valeur à une variable propositionnelle p, $\beta?$ est un test pour savoir si β, une formule propositionnelle, est vraie, $\pi;\pi'$ est l'exécution successive de deux programmes et $\pi \cup \pi'$ l'exécution non-déterministe d'un des deux programmes π et π'. Les mondes sont représentés par leurs valuations ; π est alors un programme qui décrit toutes les accessibilités entre les mondes possibles en modifiant les valuations de ceux-ci. Par exemple : $p \leftarrow \top$ est le programme qui signifie qu'il y a une relation d'accessibilité entre deux mondes w_1 et w_2 si et seulement si $w_2 \models p$ et w_1 et w_2 s'accordent sur toute autre variable q différente

de p. Ce type de modélisation des transitions pour les relations d'accessibilité est alors assez différente de la représentation symbolique classique.

Par ailleurs, ces recherches incluent des résultats théoriques sur la complexité de calculs. Notamment, le *model checking* est PSPACE-complet lors de l'utilisation de programmes mentaux n'utilisant pas l'étoile de Kleene. Enfin, leur *framework* rend le problème de satisfaisabilité dans APAL (*Arbitrary Public Announcement Logic*) décidable alors qu'il ne l'est pas en général [FvD08]).

Par la suite, nous nous intéresserons à la modélisation symbolique de Gattinger. En effet, ses travaux sont simplement adaptables pour obtenir un *framework* exploitant les probabilités, comme peut le faire la littérature des *model checkers* probabilistes. En outre, les travaux de Charrier ne se prêtent a priori pas à adaptation probabiliste.

SMCDEL est présenté illustrativement sur un site internet [Gat18d] et le code complet en Haskel est accessible [Gat18c].

Nous commencerons par présenter les définitions basiques des représentations symboliques (sec. 2.2), permettant de faire du *model checking* de manière statique (sec. 2.3)–i.e. sans événements ou opérateur dynamique, puis présenterons SMCDEL, dans un premier temps sans postconditions (sec. 2.4), puis avec postconditions (sec. 2.5).

Nous l'avons déjà mentionné : une partie des travaux de M. Gattinger permet de représenter la logique S5 via des « variables d'observations ». En effet, il est possible de représenter une structure de Kripke S5 symboliquement simplement avec une liste de variables que les agents peuvent observer. Cette représentation compacte permet de faire du *model checking* de manière efficace en terme de représentation et en temps. Cette idée est présente dans de nombreux travaux [Maf16, CHL$^+$16].

Néanmoins, nous n'allons pas présenter ces simplifications pour pouvoir nous focaliser sur la représentation de modèles KD45, nommés *modèles de croyances*. Nous pouvons remarquer que c'est en fait plus général que cela, car il est possible de représenter n'importe quel type de relation entre les mondes.

2.2 Représentation symbolique de DEL

Dans la prochaine partie, nous présenterons les manières générales de représenter des ensembles de mondes ou de relations de manière symbolique, ce qui servira de base par la suite pour nos définitions. Toutes ces définitions nous serviront aussi de support pour nos propres définitions dans le chapitre 5, afin de représenter PDEL symboliquement.

Remarque 2.1. *Notons que nous ne faisons aucune hypothèse particulière sur les relations d'accessibilité des structures de Kripke – contrairement à une grande*

partie de la littérature DEL, qui se concentre généralement sur la logique modale S5, et donc sur les relations d'équivalence *pour R. Bien que pour des raisons de simplicité nos exemples soient tous S5, notre approche ne repose pas sur cette condition, et peut être directement appliquée à des logiques plus fortes telles que KD45.*

L'idée principale de ce qui va suivre est la suivante : il est possible de représenter un monde comme étant une affectation des variables propositionnelles d'un vocabulaire V, que l'on appelle *valuation.*

Définition 2.2. *Suivant la définition et la notation de Gattinger, pour la valuation $v \in 2^V$, nous notons $v \sqsubseteq V$ la formule sur un ensemble fini de propositions atomiques V qui assigne à toute variable présente dans v la valeur $\top$ et à toutes les autres dans $V \backslash v$ la valeur $\bot$, i.e.*

$$v \sqsubseteq V \coloneqq \bigwedge_{p \in v} p \wedge \bigwedge_{p \in V \backslash v} \neg p$$

Pour un ensemble de variables propositionnelles V et par la définition de la fonction val (déf. 1.6, page 15), qui affecte à chaque monde w d'un ensemble W une valuation $v \in 2^V$, il est alors possible de représenter cette valuation v par une formule booléenne avec un seul modèle notée $v \sqsubseteq V$.

Exemple 2.3. *Si l'on reprend l'exemple de la structure de Kripke 1.4, page 21, nous avons $V = \{p, q\}$ et $\mathrm{val}(w_2) = \{q\}$. Il est alors possible d'écrire la valuation de w_2 comme la formule booléenne $\mathrm{val}(w_2) \sqsubseteq V = \{q\} \sqsubseteq \{p, q\} = q \wedge \neg p$.*

Comme dit précédemment, il est possible d'encoder un monde w de W par sa valuation. Néanmoins, si deux mondes w_1 et w_2 de W possèdent la même valuation, de sorte que $\mathrm{val}(w_1) = \mathrm{val}(w_2)$, alors les encodages symboliques des deux mondes seront identiques : $\mathrm{val}(w_1) \sqsubseteq V$. Il nous faut empêcher ce cas et c'est pourquoi nous souhaitons que la fonction de valuation val soit injective.

Définition 2.4 (Structure de Kripke valuation-injective)**.** *Soit $\mathcal{M}$ une structure de Kripke pour un vocabulaire V et son ensemble de monde possibles W et sa fonction de valuation* val.

$\mathrm{val} \colon W \to 2^V$ *est dite injective si et seulement si $\forall w_1, w_2 \in W \colon w_1 \neq w_2 \to \mathrm{val}(w_1) \neq \mathrm{val}(w_2)$ (i.e. toutes les valuations des mondes sont différentes).*

Avec ce type de fonction de valuation, on parle de structure de Kripke valuation-injective.

Grâce à une fonction de valuation val que l'on impose injective, il est alors possible de représenter un encodage unique de chaque monde w d'un ensemble de

mondes W, qui est $\text{val}(w)$. Notons que cette propriété d'injectivité de val peut être garantie grâce à l'ajout de variables propositionnelles dans le vocabulaire.

La notation $\sqsubseteq$ permet simplement de passer d'une valuation d'un monde (i.e. sa représentation symbolique) à une formule qui n'a qu'un seul et unique modèle. La combinaison de différentes de ces formules permettra de représenter des ensembles des mondes.

Ainsi, il est possible de représenter un ensemble de mondes W de manière symbolique grâce à une formule booléenne, que l'on note θ, dont chaque modèle, serait la représentation symbolique d'un monde, de par sa valuation.

Soit W un ensemble de mondes fini et val une fonction de valuation injective, il est alors possible d'encoder symboliquement l'ensemble de mondes W par une formule booléenne que l'on note ici $\theta \in \mathcal{L}_{\text{prop}}(V)$. Cette formule sera alors alors *un encodage symbolique* de W si et seulement si :

$$\forall v \in 2^V,\ v \models \theta \text{ ssi } \exists w \in W : v = \text{val}(w)$$

Ainsi, étant donné un vocabulaire V, un ensemble de mondes W et une fonction de valuation val injective (déf. 2.4), la *représentation symbolique de* W est la formule :

$$\bigvee_{w \in W} (\text{val}(w) \sqsubseteq V)$$

Autrement dit, il est possible d'obtenir la représentation symbolique de W comme la disjonction de l'ensemble des représentations symboliques des mondes de W.

Exemple 2.5. *Reprenons l'exemple 1.15. Nous avions* $W = \{w_1, w_2\}$ *avec* $\text{val}(w_1) = \{p, q\}$ *et* $\text{val}(w_2) = \emptyset$. *La formule* θ *représentant* W *est* $(p \wedge q) \vee (\neg p \wedge \neg q)$. *En effet* $\{p, q\} \models \theta$ *et* $\emptyset \models \theta$, *ce qui implique qu'il existe des mondes (ici* w_1 *et* w_2*) dont les valuations sont* $\{p, q\}$ *et* $\emptyset$ *sont dans* θ. *De la même manière,* $\{p\}$ *et* $\{q\}$ *ne sont pas des modèles de* θ, *d'où il n'existe pas de mondes dans* W *qui ont ces valuations.*

Néanmoins, toutes les fonctions de valuations dans les structures de Kripke ne sont pas injectives. Cependant, il existe une astuce simple afin d'obtenir un encodage symbolique pour des ensembles de mondes avec des valuations non-uniques : il suffit d'ajouter des propositions atomiques supplémentaires pour distinguer les mondes.

Pour cela, il suffit d'avoir une fonction dite d'*étiquetage*, qui permet de d'obtenir une valuation unique pour chaque élément d'un ensemble A, ou ici W.

Comme nous venons de la mentionner, pour représenter symboliquement W, il nous suffit d'étendre val. Nous avons déjà une fonction val qui sert au moins en partie d'étiquetage. Dans le cas de la représentation d'éléments qui n'ont pas

de fonction de valuation, une fonction d'étiquetage est obligatoire ; comme nous le verrons plus tard, c'est le cas pour des événements (déf. 1.24).

Définition 2.6 (Fonction d'étiquetage)**.** *Une* fonction d'étiquetage λ *pour un ensemble A via un vocabulaire V, avec $|2^V| \geq |A|$ est une fonction injective $\lambda\colon A \to 2^V$.*

Lorsqu'il y aura ambuiguïté sur l'ensemble A et le vocabulaire V utilisés, on notera la fonction d'étiquetage λ ainsi : λ_A^V.

En fait, quand val est injective, val est une fonction d'étiquetage avec pour domaine W, qui peut être alors vue comme fonction d'étiquetage des mondes W sur un vocabulaire V, notée λ_W^V. Quand la fonction val est injective, le vocabulaire V est suffisant pour distinguer les mondes. Dans le cas contraire, il faut étendre le vocabulaire V pour rendre la fonction d'étiquetage injective.

Exemple 2.7. *Soit une structure de Kripke avec deux mondes w_1 et w_2 de sorte à ce que la fonction de valuation* val *soit définie sur le vocabulaire $V = \{p\}$ et que* $\mathrm{val}(w_1) = \mathrm{val}(w_2) = \{p\}$*. Pour définir θ, il nous faudra étendre le vocabulaire V avec une nouvelle variable non utilisée (*fresh variable *en anglais). Par exemple, on pourra alors construire λ avec $V_2 = \{p, q\}$, de sorte à ce que $\lambda_W^{V_2}(w_1) = \{p\}$ et $\lambda_W^{V_2}(w_2) = \{p, q\}$. Précisions que la variable q sera utilisée symboliquement pour distinguer les deux mondes, mais cela n'aura aucun sens de faire de la vérification de modèle avec cette variable dans une formule.*

La représentation symbolique des mondes W précédente, illustrative, était dans le cas d'une structure de Kripke valuation-injective. Pour s'extraire de ce cas, nous pouvons alors utiliser une fonction d'étiquetage.

Définition 2.8. *Étant donné un vocabulaire V, un ensemble fini W et une fonction d'étiquetage λ_W^V, la* représentation symbolique de W *via λ_W^V est définie ainsi :*

$$\mathrm{symb}_{\lambda_W^V}(W) \coloneqq \bigvee_{w \in W} (\lambda_W^V(w) \sqsubseteq V)$$

Nous pouvons voir que la symbolicisation des mondes d'une structure de Kripke valuation-injective est alors $\mathrm{symb}_{\mathrm{val}}(W)$.

Maintenant que nous avons présenté une représentation symbolique d'un monde et d'un ensemble de mondes, voyons comme nous pouvons faire de même avec une relation et un ensemble de relations.

L'idée est de dupliquer le vocabulaire V. Pour cela, nous utiliserons le symbole *prime*. Basiquement, p' sera la variable primée correspondant à p. Il en va de même pour des ensembles. Soit X un ensemble, sous-ensemble de PS, X' sera alors l'ensemble des variables de X primées : $X' = \{p' \mid p \in X\}$. Ici, typiqmenent, nous emploierons le vocabulaire V', vocabulaire primé de V.

Définition 2.9 (Prime)**.** *Si v est une valuation de V, alors v' correspond à la valuation de v sur le vocabulaire V', de sorte à ce qu'il y ait une bijection b entre V et V' ($V \cap V' = \emptyset$) : $\forall p \in V,\ b(p) = p'$.*

Notons qu'obtenir la variante primée d'une variable propositionnelle de V est toujours possible car V est fini et est issu du vocabulaire PS qui est un réservoir infini de variables propositionnelles.

L'idée primordiale derrière cette duplication de variables est la suivante : pour représenter une relation R entre les mondes W d'une structure de Kripke, il est nécessaire de représenter les mondes prédécesseurs et les mondes successeurs. Le vocabulaire V représentera les mondes prédécesseurs et le vocabulaire V' représentera leurs mondes successeurs. Ainsi, tous les modèles d'une formule propositionnelle sur un vocabulaire $V \cup V'$ représenteront les relations entre les mondes.

Une telle formule sera alors un encodage symbolique noté $\text{symb}(R_a)$ d'une relation R_a entre 2^V et 2^V (primé symboliquemnet), défini sur $V \cup V'$, de sorte à ce que $\forall (s,t) \in 2^V \times 2^V$, $s \cup t' \models \text{symb}(R_a)$. De la même manière que pour la représentation symbolique des mondes, nous allons utiliser une fonction d'étiquetage.

Pour cet encodage, nous allons définir la renommage de variable, qui nécessite la définition de la substitution. Nous utilisons la définition utilisée par Gattinger.

Définition 2.10 (Substitution)**.** *Soit ϕ et ψ deux formules et p un atome propositionnel. $[p \mapsto \psi]\phi$ est le remplacement de chaque p dans ϕ par ψ. Pour un ensemble fini d'atomes propositionnels $P = \{p_1, \ldots, p_n\}$, $[P \mapsto \psi]\phi$ est le résultat de la substitution simultanée pour chaque element de P par ψ dans ϕ.*

Pour deux ensembles finis de même taille $P = \{p_1, \ldots, p_n\}$ et $Q = \{q_1, \ldots, q_n\}$, $[P \mapsto Q]\phi$ est la substitution simultanée de chaque p_k en q_k dans ϕ pour $k \in \{1, \ldots, n\}$.

Notons que P et Q doivent être des listes ordonnées et qu'une bijection implicite est utilisée entre elles.

Un cas particulier de la substitution est *le renommage de variables* ; que nous avons déjà utilisé, mais que nous pouvons réécrire $[X \mapsto X']f$ pour la substitution de variables dans X par les variables X' quand la transformation entre X et X' est claire dans le contexte.

Définition 2.11 (Renommage d'une formule)**.** *Si ϕ est une formule définie sur un vocabulaire V, on note ϕ' la formule dont tous les atomes propositionnels seront remplacés par leur équivalent primé : $[V \mapsto V']\phi$.*

Notons en outre que la formule $[p \to q]p'$ est la formule p' et non pas q'. En effet, p' ici est une variable à part entière et n'est pas la formule p primée.

Définissons maintenant l'encodage symbolique de R_a,noté $\text{symb}(R_a)$.

Définition 2.12. *Étant donné un vocabulaire V, un ensemble de mondes W, une relation R_a sur W, et une fonction d'étiquetage λ_W^V, la représentation symbolique de R_a via λ_W^V est définie ainsi :*

$$\mathrm{symb}_{\lambda_W^V}(R_a) \coloneqq \bigvee_{(w_1,w_2)\in R_a} (\lambda_W^V(w_1) \sqsubseteq V) \wedge (\lambda_W^V(w_2) \sqsubseteq V)'$$

Remarque 2.13. *Notons qu'une relation symbolique ne doit pas nécessairement strictement représenter toutes les relations symboliques entre les mondes symboliques. En effet, il est possible de relâcher les contraintes sur les relations symboliques car les mondes symboliques pourront être exploités dans le* model checking *afin de récupérer les mondes réélement existants. De ce fait, une relation totale entre les mondes pourra être modélisée par la formule $\top$ sur le vocabulaire $V \cup V'$. À l'inverse, le relation vide sera transcrite par la formule $\bot$.*

Si nous considérons R_a^{-1} la relation inverse à R_a, alors nous pouvons voir que R_a est symétrique si et seulement si $\mathrm{symb}(R_a^{-1}) \equiv \mathrm{symb}(R_a)$.

Voyons comment représenter le pendant symbolique d'une structure de Kripke : la *structure de croyances*. Trois élements suffisent donc à représenter une structure de croyances, notée $\mathfrak{F}$: un vocabulaire V, une loi représentant les mondes légaux : θ, et un ensemble de lois représentant les relations entre les mondes pour chaque agent $a \in \mathfrak{A} : \Omega = \{\Omega_a \mid a \in \mathfrak{A}\}$.

Définition 2.14. *Une* structure de croyances *est un tuple $\mathfrak{F} = \langle V, \theta, \Omega\rangle$ où :*

- *V est un ensemble fini de variables propositionnelles appelé* vocabulaire *;*
- *$\theta \in \mathcal{L}_{\mathrm{prop}}(V)$ est une formule booléenne sur V appelée la* loi des mondes *;*
- *Ω est un ensemble de formules indexé par les agents tel que pour chaque agent $a \in \mathfrak{A}$, $\Omega_a \in \mathcal{L}_{\mathrm{prop}}(V \cup V')$ est une formule booléenne sur le vocabulaire double $V \cup V'$. Cette formule est appelée la* loi des observations *de a.*

Chaque $s \in 2^V$ tel que $s \models \theta$ est appelé un état *de $\mathfrak{F}$. Un couple $\langle \mathfrak{F}, s\rangle$ où s est un état de $\mathfrak{F}$ est appelé une* scène.

Il est alors possible de transformer une structure de Kripke $\mathcal{M}$ (déf. 1.6, page 15) dite « explicite », où tous les mondes et toutes les relations sont définies et explicitées dans des ensembles, en une structure symbolique $\mathfrak{F}$ appelée structure de croyances.

Nous commençons par définir l'opération sur une *frame*, ce que Gattinger ne fait pas, puis sur une structure de Kripke, ce qu'il présente directement. Ces définitions sont correspondantes à celles de Gattinger. Néanmoins, nous les décorrélons pour pouvoir simplifier l'exposé plus tard. Nous nous servons d'une fonction d'étiquetage adéquate pour représenter une *frame*, alors que pour une structure de Kripke, il s'agira d'utiliser la fonction de valuation val comme fonction d'étiquetage.

Cette décomposition nous permettra d'utiliser la représentation d'une *frame* plus tard dans un autre contexte.

Ainsi, grâce à une fonction d'étiquetage non déterminée, il est aisé de représenter n'importe quelle *frame*, grâce à la prochaine définition.

Définition 2.15. *La représentation symbolique d'une frame $\mathcal{F} = \langle W, R\rangle$ via une fonction d'étiquetage λ_W^V (déf. 2.6) sur le vocabulaire V, notée* $\mathrm{symb}_{\lambda_W^V}(\mathcal{F})$ *est la structure de croyances*

$$\langle V,\ \mathrm{symb}_{\lambda_W^V}(W),\ (\mathrm{symb}_{\lambda_W^V}(R_a))_{a\in\mathfrak{A}}\rangle.$$

Comme nous l'avons vu, dans le cas d'une structure de Kripke valuation-injective, la fonction val sert exactement de fonction d'étiquetage. De cette manière, il est possible de représenter symboliquement une structure de Kripke $\mathcal{M} = \langle W, R, \mathrm{val}\rangle$ avec val une fonction injective, en utilisant la *frame* $\langle W, R\rangle : \mathrm{symb}_{\mathrm{val}}(\mathcal{M}) = \mathrm{symb}_{\mathrm{val}}(\langle W, R\rangle)$, ce qui revient à $\langle V,\ \mathrm{symb}_{\mathrm{val}}(W),\ (\mathrm{symb}_{\mathrm{val}}(R_a))_{a\in\mathfrak{A}}\rangle$.

Exemple 2.16. *Par exemple, représentons symboliquement la structure de Kripke de l'exemple 1.15 (fig. 1.4, page 21) de sorte à avoir* $\mathrm{symb}_{\mathrm{val}}(\mathcal{M}) = \langle V,\ \mathrm{symb}_{\mathrm{val}}(W),\ (\mathrm{symb}_{\mathrm{val}}(R_a))_{a\in\mathfrak{A}}\rangle$ *:*

$$\begin{aligned}
&V = \{p, q\}\ ;\\
&\mathrm{symb}_{\mathrm{val}}(W) = (p \wedge q) \vee (\neg p \wedge \neg q)\ ;\\
&\mathrm{symb}_{\mathrm{val}}(R_b) = \big((p \wedge q) \wedge (p \wedge q)'\big)\ \vee\\
&\qquad\qquad \big((\neg p \wedge \neg q) \wedge (\neg p \wedge \neg q)'\big)\\
&\mathrm{symb}_{\mathrm{val}}(R_r) = \big((p \wedge q) \wedge (p \wedge q)'\big)\ \vee\\
&\qquad\qquad \big((p \wedge q) \wedge (\neg p \wedge \neg q)'\big)\ \vee\\
&\qquad\qquad \big((\neg p \wedge \neg q) \wedge (\neg p \wedge \neg q)'\big)\ \vee\\
&\qquad\qquad \big((\neg p \wedge \neg q) \wedge (p \wedge q)'\big)
\end{aligned}$$

Ainsi, la simple formule $(p \wedge q)$ de la formule $\mathrm{symb}_{\mathrm{val}}(W)$ *(qui a un seul modèle) permet de représenter symboliquement le monde w_1 par sa valuation* $\mathrm{val}(w_1) = \{p, q\}$*. Par ailleurs, la formule $(p \wedge q) \wedge (p \wedge q)'$ permet de représenter symboliquement l'arc qui relie le monde w_1, de valuation* $\mathrm{val}(w_1) = \{p, q\}$*, au monde w_2, de valuation* $\mathrm{val}(w_2) = \emptyset$*, grâce au dédoublement de variable $V \cup V'$.*

Il est aussi possible de faire l'opération inverse, que l'on notera expl.

Nous appelons ici $\mathrm{supp}(\phi)$ l'ensemble des modèles de $\phi \in \mathcal{L}_{\mathrm{prop}}(V)$, avec V un vocabulaire fini, de sorte que $\mathrm{supp}(\phi) \coloneqq \{v \in 2^V \mid v \models \phi\}$.

Définition 2.17. *La représentation explicite* $\mathrm{expl}(\theta)$ *d'une loi des mondes* θ *est l'ensemble de mondes*

$$W \coloneqq \mathrm{supp}(\theta)$$

Remarque 2.18. *Les mondes* W *étant créés à partir du support de* θ*, ils sont littéralement des valuations de monde.*

Définition 2.19. *Soit* $V \subseteq PS$*, un vocabulaire fini. La représentation explicite* $\mathrm{expl}_Z(\Omega_a)$ *d'une loi d'observations* Ω_a *définie sur* $V \cup V'$ *via un ensemble de valuations* $X \subseteq 2^V$ *est une relation d'accessibilité (*$\mathrm{expl}_Z(\Omega_a) \subseteq Z \times Z$*) tel que :*

$$\forall z_1, z_2 \in Z,\ (w_1, w_2) \in \mathrm{expl}_Z(\Omega_a) \text{ ssi } z_1 \cup z_2' \models \Omega_a$$

Définition 2.20. *La représentation explicite d'une structure de croyances* $\mathfrak{F} = \langle V, \theta, \Omega \rangle$*, notée* $\mathrm{expl}(\mathfrak{F})$*, est la structure de Kripke définie avec* $W = \mathrm{expl}(\theta)$ *et l'application identité* $\mathsf{id}_W \colon W \mapsto W,\ w \mapsto w$ *:*

$$\langle W, (\mathrm{expl}_W(\Omega_a))_{a \in \mathfrak{A}}, \mathsf{id}_W \rangle$$

Remarque 2.21. *Notons que la transformation* expl *fonctionne sur toute structure de croyances, même celles qui ne résultent pas d'une transformation* symb*. Par ailleurs, il n'est plus question d'adapter le vocabulaire utilisé, parce qu'une structure de croyances possède de par* θ *une loi des mondes que l'on pourrait déjà qualifier d'injective.*

Enfin, notons qu'en général, $\mathfrak{F} \neq \mathrm{symb}(\mathrm{expl}(\mathfrak{F}))$ *: elles sont équivalentes mais pas identiques.*

Exemple 2.22. *Soit la structure de croyances* $\mathfrak{F} = \langle V, \theta, \Omega \rangle$ *définie comme suit :*

- $V = \{p, q\}$;
- $\theta = (p \wedge q) \vee (\neg p \wedge \neg q)$;
- $\Omega_1 = p \leftrightarrow p'$;
- $\Omega_2 = \top$.

À partir de θ*, il est possible d'obtenir* $W = \mathrm{expl}(\theta)$*, l'ensemble des modèles de* θ *qui est* $\{\{p, q\}, \emptyset\}$ *et* $\mathrm{val}(\{p, q\}) = \{p, q\}$ *et* $\mathrm{val}(\emptyset) = \emptyset$*. Pour ce qui est des relations, nous présentons l'ensemble des modèles dans le tableau 2.1. Les modèles des formules* Ω_1 *et* Ω_2 *sont nombreux. 8 pour la première et 16 pour la seconde. Néanmoins, cela ne signifie pas qu'il y a autant de relations entre les mondes. En effet, la définition de* $\mathrm{expl}_W(\Omega_a)$ *est claire :* $\forall (w_1, w_2) \in W \times W$*,* $(w_1, w_2) \in \mathrm{expl}_W(\Omega_a)$ *ssi* $w_1 \cup w_2' \models \Omega_a$*. Il faut que les modèles restreints au vocabulaire* V *représentent des mondes légaux dans* W *et que les modèles sur* V' *aussi. Ainsi, ce « filtre » sur les mondes, nous permettent d'obtenir les modèles mis en* ***gras*** *dans la table 2.1 et d'avoir :*

p	q	p'	q'	$\Omega_1 = p \leftrightarrow p'$	$\Omega_2 = \top$
0	**0**	**0**	**0**	**1**	**1**
0	0	0	1	1	1
0	0	1	0	0	1
0	**0**	**1**	**1**	**0**	**1**
0	1	0	0	1	1
0	1	0	1	1	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	0	1
1	0	0	1	0	1
1	0	1	0	1	1
1	0	1	1	1	1
1	**1**	**0**	**0**	**0**	**1**
1	1	0	1	0	1
1	1	1	0	1	1
1	**1**	**1**	**1**	**1**	**1**

TABLE 2.1 – Table de vérité des formules Ω_1 et Ω_2 de l'exemple 2.22

- $\text{expl}_W(\Omega_1) = \{(\{p,q\},\{p,q\}),(\emptyset,\emptyset)\}$;
- $\text{expl}_W(\Omega_2) = \{(\{p,q\},\{p,q\}),(\emptyset,\emptyset),(\{p,q\},\emptyset),(\emptyset,\{p,q\})\}$.

Cette structure de Kripke est donc équivalente à celle présentée sur l'exemple 1.15 (l'agent b étant ici l'agent 1 *et l'agent r étant 2). Remarquons simplement que la dénomination des mondes a disparu pour laisser place à leur valuation.*

Nous avons donc terminé de décrire le fonctionnement de la représentation symbolique, avec la représentation des mondes et des relations.

2.3 Model checking sur une structure de croyances

Maintenant qu'il nous est possible de représenter une structure de Kripke de manière symbolique, le but est de pouvoir faire du *model checking* qui soit équivalent au *model checking* sur les structures de Kripke.

Gattinger définit une sémantique du langage $\mathcal{L}_{\text{EL}}$ sur les structures de croyances (i.e. structures de Kripke symbolique), à l'instar d'une structure de Kripke explicite sur la langage $\mathcal{L}_{\text{EL}}$ (déf. 1.8, page 16).

Définition 2.23. *La sémantique de $\mathcal{L}_{\text{EL}}(V)$ sur une structure de croyances pointée est définie inductivement comme suit :*

- $\langle \mathfrak{F}, s \rangle \models \top$ *est vraie ;*
- $\langle \mathfrak{F}, s \rangle \models p$ *ssi* $s \models p$, $p \in V$;
- $\langle \mathfrak{F}, s \rangle \models \neg\phi$ *ssi* $\langle \mathfrak{F}, s \rangle \not\models \phi$;
- $\langle \mathfrak{F}, s \rangle \models \phi \wedge \psi$ *ssi* $\langle \mathfrak{F}, s \rangle \models \phi$ *et* $\langle \mathfrak{F}, s \rangle \models \psi$;
- $\langle \mathfrak{F}, s \rangle \models \Box_i \phi$ *ssi pour tout état* t *de* $\mathfrak{F}$: $s \cup t' \models \Omega_i$ *implique* $\langle \mathfrak{F}, t \rangle \models \phi$.

Pour effectuer du *model checking* symbolique, l'idée clef est d'avoir une fonction permettant de retourner l'ensemble des mondes dans lesquels une formule propositionnelle ou épistémique est vérifiée. Le pendant symbolique de cette idée, est d'avoir une fonction qui permet de retourner une formule propositionnelle dont les modèles représentent les mondes (toujours de par leurs valuations) dans lesquels une formule est vraie.

Cette fonction est appelée par Gattinger « traduction locale », notée $||\phi||_{\mathfrak{F}}$ pour une formule épistémique ϕ et une structure de croyances $\mathfrak{F}$.

Pour la définition de la traduction (déf. 2.26), nous aurons besoin de définir le *quantificateur booléen* $\forall$ (déf. 2.24). Cette définition nécessite la définition de la *substitution* (déf. 2.10), qui permet de remplacer des variables par des formules propositionnelles dans une formule.

Définition 2.24 (Quantificateur booléen)**.** *Le quantificateur booléen* $\forall p\ \phi$ *(défini grâce à la substitution 2.10) est l'abréviation de* $[p \mapsto \top]\phi \wedge [p \mapsto \bot]\phi$. *Son dual est aussi abrégé de la sorte :* $\exists p\ \phi \coloneqq [p \mapsto \top]\phi \vee [p \mapsto \bot]\phi$.

Les quantificateurs booléens permettent de faire « disparaître » une variable p d'une formule ϕ. Cette notation est étendue pour des ensembles, de sorte à ce que nous puissions écrire pour un ensemble X, $\forall X\ \phi$. Par exemple, si $X = \{p, q, r\}$, $\forall X\ \phi = \forall p \forall q \forall r\ \phi$.

Nous en profitons par ailleurs pour définir le *conditionnement* qui utilise la substitution. Le conditionnement est une substitution simplifiée qui remplace des variables par des formules propositionnelles qui sont simplement les constantes $\top$ et $\bot$.

Définition 2.25 (Conditionnement)**.** *Pour un vocabulaire* X*, une valuation* $v \in 2^X$ *et une formule épistémique* $\phi \in \mathcal{L}_{\mathrm{EL}}(X)$*, nous écrivons le* conditionnement *de* ϕ *par la valuation* $v \in 2^X$ *ainsi en reprenant les notations des définitions 2.2 et 2.10 :*

$$[v \sqsubseteq X]\phi = [v \mapsto \top][(X \setminus v) \mapsto \bot]\phi$$

Nous pouvons maintenant décrire la traduction locale d'une formule épistémique.

Définition 2.26. *Pour toute structure de croyances* $\mathfrak{F} = \langle V, \theta, \Omega \rangle$ *et pour toute formule* $\phi \in \mathcal{L}_{\text{EL}}(V)$*, la* traduction locale *de* ϕ *vers* $\mathcal{L}_{\text{prop}}(V)$ *pour* $\mathfrak{F}$ *est construite inductivement ainsi :*

- $||\top||_{\mathfrak{F}} \coloneqq \top$,
- $||p||_{\mathfrak{F}} \coloneqq p$,
- $||\neg\phi||_{\mathfrak{F}} = \neg||\phi||_{\mathfrak{F}}$,
- $||\phi \wedge \psi||_{\mathfrak{F}} \coloneqq ||\phi||_{\mathfrak{F}} \wedge ||\psi||_{\mathfrak{F}}$,
- $||\Box_a\phi||_{\mathfrak{F}} \coloneqq \forall V'((\theta' \wedge \Omega_a) \rightarrow ||\phi||'_{\mathfrak{F}}))$.

Cette fonction permet de construire de manière récursive une formule propositionnelle qui représentera toutes les contraintes induites par la formule ϕ. Pour tous les opérateurs propositionnels, la traduction est simple. Néanmoins, l'opérateur $\Box$ nécessite de plus amples explications :

1. $||\phi||'_{\mathfrak{F}}$ permet de récupérer une formule propositionnelle représentant l'ensemble des mondes satisfaisant ϕ. Celle-ci est primée de manière à ce qu'elle s'applique sur les mondes successeurs, également primés, présents dans la formule propositionnelle Ω_a.
2. $(\theta' \wedge \Omega_a)$ représente tous les arcs représentés de manière symbolique dans Ω_a dont les successeurs sont des mondes légaux (grâce à θ').
3. $(\theta' \wedge \Omega_a) \rightarrow ||\phi||'_{\mathfrak{F}}$ permet d'obtenir une formule propositionnelle sur le vocabulaire dédoublé $V \cup V'$, représentant tous les arcs dont les mondes successeurs satisfont la formule ϕ.
4. L'opérateur $\forall$ permet de retrouver une formule propositionnelle sur le vocabulaire simple V tout en ne conservant que les mondes prédécesseurs dont tous les mondes successeurs satisfont ϕ.

Exemple 2.27. *Prenons un exemple de model checking sur la structure de croyances définie dans l'exemple 2.22.*

Considérons la formule $\Box_2 q$*. Nous pouvons calculer* $||\Box_2 q||_{\mathfrak{F}}$ *de la sorte :*

$$\begin{aligned}
||\Box_2 q||_{\mathfrak{F}} &= \forall V'((\theta' \wedge \Omega_2) \rightarrow ||q||'_{\mathfrak{F}}) \\
&= \forall p' \forall q'((((p \wedge q) \vee (\neg p \wedge \neg q))' \wedge \top) \rightarrow ||q||'_{\mathfrak{F}}) \\
&= \forall p' \forall q'(((p \wedge q) \vee (\neg p \wedge \neg q)') \rightarrow q') \\
&= \forall p' \forall \neg p' \wedge \neg q' \\
&= \bot
\end{aligned}$$

Ainsi, nous avons bien qu'aucune valuation ne vérifie « l'agent 2 sait que q », ce qui est tout à fait normal, la loi d'observation de l'agent 2 est $\top$ *et il existe un monde où q est faux. Ainsi, depuis aucun monde il n'est nécéssaire que q soit vrai.*

Considérons maintenant la formule $\Box_1 p$.

$$\begin{aligned}
||\Box_1 p||_{\mathfrak{F}} &= \forall V'((\theta' \wedge \Omega_1) \rightarrow ||p||'_{\mathfrak{F}})) \\
&= \forall p' \forall q'((((p \wedge q) \vee (\neg p \wedge \neg q))' \wedge p \leftrightarrow p') \rightarrow ||p||'_{\mathfrak{F}}) \\
&= \forall p' \forall q'((((p \wedge q) \vee (\neg p \wedge \neg q))' \wedge p \leftrightarrow p') \rightarrow p') \\
&= p
\end{aligned}$$

Nous avons donc la formule p qui décrit l'ensemble des modèles où $\Box_1 p$ est vrai. Sur le vocabulaire $V = \{p, q\}$, ces modèles sont $\{p\}$ et $\{p, q\}$. Ici, seule la valuation $\{p, q\}$ nous intéresse, car faisant partie de la loi des mondes.

Prenons la temps de comparer deux exemples que nous avons détaillé. Dans l'exemple de structure de croyances présenté 2.16, page 39 et dans l'exemple 2.22, page 40, nous représentons les deux mêmes structures de croyances.

La différence principale est d'avoir modélisé Ω de manière « économique ». En effet, comme nous l'avions déjà fait remarqué (rq. 2.13), il est possible d'encoder plus de mondes successeurs dans un Ω_a, tant que cela permet de répondre à la définition du *model checking* de $\Box$ (i.e. il faut que tous les mondes réels successeurs soient présents mais il peut y en avoir plus). Il n'est pas non plus du tout nécessaire d'encoder les mondes prédécesseurs. En effet, il n'y a pas d'opérateurs qui le nécessite, qui serait une sorte de $\Box$ inversé qui signifirait « tous les mondes qui m'atteignent doivent avoir telle propriété ».

L'astuce est la suivante : grâce à la modélisation de θ, il est possible de reconstituer les arcs « légaux ».

Exemple 2.28. *Par exemple, si nous avons une structure de Kripke $\mathcal{M} = \langle W, R, \mathrm{val}\rangle$ avec $V = \{p, q\}$, $W = \{w_1, w_2\}$, $R_1 = \langle (w_1, w_2), (w_2, w_2)\rangle$, $\mathrm{val}(w_1) = \emptyset$ et $\mathrm{val}(w_2) = \{p\}$, alors nous avons $\theta = \neg q$, mais il suffira d'avoir $\Omega_1 = p'$, qui contiendra les modèles de la formule attendue : $(\neg p \wedge p') \vee (p \wedge p')$, mais aussi d'autres modèles avec la variable q à vraie (alors que θ l'interdit).*

Exemple 2.29. *Un exemple plus intéressant est le fait de pouvoir modéliser une clique complète entre les mondes symboliques grâce à la constante $\top$. $\top$ signifie qu'il y a une relation d'accessibilité entre tous les mondes possibles et imaginables. Si la loi des mondes est aussi $\top$, peu importe le vocabulaire V utilisé, ce sera vraiment le cas. Néanmoins, si la loi des mondes est plus stricte, par exemple $\theta = (p \wedge q) \vee r$, il n'y aura que 5 mondes légaux : $\{r\}$, $\{q, r\}$, $\{p, r\}$, $\{p, q\}$ et $\{p, q, r\}$. Ainsi, il sera inutile définir la relation de clique comme étant la formule $\Omega_a = \theta \wedge \theta'$.*

La relation est alors beaucoup plus simple à exprimer et beaucoup moins contraignante à représenter en mémoire et à manipuler par la suite dans un programme.

Combinant la sémantique précédemment décrite et la définition de la traduction, Gattinger prouve le théorème suivant (théorème 2.6.4 p.58 de son manuscrit de thèse) :

Théorème 2.30. *La traduction locale (déf. 2.26) préserve et reflète la vérité. C'est à dire, pour toute formule $\phi \in \mathcal{L}_{\mathrm{EL}}$ et pour toute scène $\langle \mathfrak{F}, s \rangle$, on a $\langle \mathfrak{F}, s \rangle \models \phi$ ssi $s \models ||\phi||_{\mathfrak{F}}$.*

Nous ajoutons la preuve du théorème ici dans la présentation du *framework* SMCDEL, car elle sera étendue plus tard dans le cas probabiliste.

Démonstration. Par induction sur la formule ϕ. Les cas pour les propositions atomiques sont immédiats. Dans les étapes d'induction, la conjonction et la négation sont standards. Pour le cas de la croyance, faisons la chaîne d'équivalence suivante :

$$
\begin{aligned}
\langle \mathfrak{F}, s \rangle \models \Box_a \phi \iff & \text{ pour tout } t \in \mathfrak{F} : \text{si } s \cup t' \models \Omega_a \text{ alors } \langle \mathfrak{F}, t \rangle \models \phi \\
\iff & \text{ pour tout } t \in \mathfrak{F} : \text{si } s \cup t' \models \Omega_a \text{ alors } t \models ||\phi||_{\mathfrak{F}} \\
\iff & \text{ pour tout } t : \text{si } t \in \mathfrak{F} \text{ et } s \cup t' \models \Omega_a \text{ alors } t \models ||\phi||_{\mathfrak{F}} \\
\iff & \text{ pour tout } t : \text{si } t \models \theta \text{ et } s \cup t' \models \Omega_a \text{ alors } t \models ||\phi||_{\mathfrak{F}} \\
\iff & \text{ pour tout } t : \text{si } t' \models \theta' \text{ et } s \cup t' \models \Omega_a \text{ alors } t' \models ||\phi||'_{\mathfrak{F}} \\
\iff & \text{ pour tout } t : \text{si } s \cup t' \models \theta' \text{ et } s \cup t' \models \Omega_a \text{ alors } s \cup t' \models ||\phi||'_{\mathfrak{F}} \\
\iff & \text{ pour tout } t : \text{si } s \cup t' \models \theta' \to (\Omega_a \to ||\phi||'_{\mathfrak{F}}) \\
\iff & s \models \forall V' (\theta' \to (\Omega_a \to ||\phi||'_{\mathfrak{F}}))
\end{aligned}
$$

□

L'idée derrière ce théorème est simple : pour vérifier qu'une formule ϕ est vraie dans un état (une valuation) s, il suffit de vérifier que l'état est dans les modèles de la traduction de ϕ, qui représente l'ensemble des états dans lequel la formule est vraie.

Exemple 2.31. *Continuons l'exemple 2.27. Nous avions vu que $||\Box_1 p||_{\mathfrak{F}} = p$. Ainsi, dans les mondes pointés possibles de la structure de croyances définie dans l'exemple 2.22 ($\theta = (p \wedge q) \vee (p' \wedge q')$, avec pour modèles $\{p, q\}$ et $\emptyset$), nous avons simplement $\langle \mathfrak{F}, \{p, q\} \rangle \models \Box_1 p$ car $\{p, q\} \models p$. À l'inverse, nous avons $\langle \mathfrak{F}, \emptyset \rangle \not\models \Box_1 p$ car $\emptyset \not\models p$.*

Gattinger montre une dernière chose concernant le *model checking* : les résultats du *model checking* sur une structure de Kripke et sa version symbolique (via la définition 2.15) sont identiques étant donné un monde pointé donné et une formule épistémique quelconque. Il en va de même pour le *model checking* sur une structure de croyances et sa version explicite (via la définition 2.20).

C'est ce que nous illustrons dans notre exemple qu'est Hanabi. Nous présentons une structure de Kripke sur la figure 1.4, page 21, que nous encodons symboliquement dans l'exemple 2.16, page 39. Nous présentons une structure de croyances

dans l'exemple 2.22, page 40, que nous traduisons en version explicite. Ces deux structures de Kripke et de croyances représentent les mêmes états de croyances des agents. Dans les deux cas, l'agent 1 distingue les deux mondes et connait la valeur de la variable p. Quant à lui, l'agent 2 ne connaît pas la valeur de p.

Ainsi, en utilisant la représentation symbolique toute l'information est conservée tout en utilisant et manipulant des formules propositionnelles à la place des structures de Kripke habituelles.

Ces propriétés sont présentées par les deux théorèmes, issus de la thèse de Gattinger (théorèmes 2.6.10 et 2.6.11 p.61 de son manuscrit), qui vont suivre et qui vont nécessiter un lemme (2.6.7 p.60) pour faciliter leurs preuves. Nous conservons les notations de Gattinger et utilisons deux vocabulaires V et U : $U \subseteq V$ Le vocabulaire V ici permet de rendre une structure de Kripke (définie sur V) qui ne l'est pas, injective. Il suffit alors d'étendre la fonction de valuation val (définie sur le vocabulaire U) avec le vocabulaire V de sorte à obtenir λ^V_W.

Lemme 2.32. *Soit une structure de croyances $\mathfrak{F} = \langle V, \theta, \Omega\rangle$ et une structure de Kripke $\mathcal{M} = \langle W, R, \mathrm{val}\rangle$ sur le vocabulaire V, et soit un ensemble d'atomes propositionnels $U \subseteq V$. Si nous avons une fonction d'étiquetage $\lambda\colon W \to 2^V$ telle que :*

C1 *pour tous $w_1, w_2 \in W$ et $a \in \mathfrak{A}$, on a $\lambda(w_1) \cup \lambda(w_2)' \models \Omega_a$ ssi $w_1 R_a w_2$.*

C2 *pour tous $w \in W$ et $p \in U$, nous avons $p \in \lambda(w)$ ssi $p \in \mathrm{val}(w)$.*

C3 *pour chaque $s \in 2^V$, s est un état de $\mathfrak{F}$ ssi $s = \lambda(w)$ pour certains $w \in W$ alors, pour toute formule ϕ sur le vocabulaire U, on a $\langle \mathfrak{F}, \lambda(w)\rangle \models \phi$ ssi $\langle \mathcal{M}, w\rangle \models \phi$.*

Démonstration. Nous allons procéder par induction sur ϕ. Avant tout, supposons que ϕ est un atome propositionnel, disons p. Alors par la condition C2, nous avons $\langle \mathfrak{F}, \lambda(w)\rangle \models p$ ssi $p \in \lambda(w)$ ssi $p \in \mathrm{val}(w)$ ssi $\langle \mathcal{M}, w\rangle \models p$.

Maintenant, supposons que ϕ n'est pas un atome propositionnel et que l'hypothèse d'induction est vraie.

Pour les cas $\neg\phi$ ou $\phi \wedge \psi$, les sémantiques pour une structure de Kripke (déf. 1.8) fonctionnent de la même manière pour la négation et la conjonction, ainsi les deux cas suivent l'hypothèse d'induction.

Pour les croyances : si ϕ est de la forme $\Box_a\phi$ alors par la définition de la sémantique 2.23, on a $\langle \mathfrak{F}, \lambda(w)\rangle \models \Box_a\phi$ ssi $\langle \mathfrak{F}, s\rangle \models \phi$ pour tout état s de $\mathfrak{F}$ avec $\lambda(w) \cup s' \models \Omega_a$. Par C3, c'est équivalent à avoir $\langle \mathfrak{F}, \lambda(w_2)\rangle \models \phi$ pour tout $w_2 \in W$ avec $\lambda(w) \cup \lambda(w_2)' \models \Omega_a$, qui par l'hypothèse C1 est équivalent à $\langle \mathfrak{F}, \lambda(w_2)\rangle \models \phi$ pour tout $w_2 \in W$ avec $(w, w_2) \in R_a$.

Maintenant, par l'hypothèse d'induction, c'est équivalent à $\langle \mathcal{M}, w_2\rangle \models \phi$ pour tout $w_2 \in W$ avec $(w, w_2) \in R_a$, ce qui est exactement $\langle \mathcal{M}, w\rangle \models \Box_a\phi$ par la définition 1.8.

□

Théorème 2.33. *Pour toute structure de croyances $\mathfrak{F}$, tout état s de $\mathfrak{F}$ et toute formule épistémique ϕ, nous avons $\langle \mathfrak{F}, s\rangle \models \phi$ ssi $\langle \text{expl}(\mathfrak{F}), s\rangle \models \phi$.*

Démonstration. On note $\mathcal{M} = \text{expl}(\mathfrak{F}) = \langle W, (\text{expl}_W(\Omega_a))_{a\in\mathfrak{A}}, \mathsf{id}_W\rangle$ avec $\text{expl}(\theta) = W$ (via déf. 2.20). Prouvons les conditions du lemme 2.32 en utilisant la fonction d'identité pour λ et $U = V$:

- C1 est vérifiée par la définition de $\text{expl}_W(R)$;
- les conditions C2 et C3 sont vérifiées par la définition de $\text{expl}(\theta)$.

□

Théorème 2.34. *Pour toute structure de Kripke $\mathcal{M}$ avec une fonction de valuation* val *injective, tout monde pointé w de $\mathcal{M}$ et toute formule épistémique ϕ, nous avons*

$$\langle \mathcal{M}, w\rangle \models \phi \text{ ssi } \langle \text{symb}_{val}(\mathcal{M}), \text{val}(w)\rangle \models \phi$$

Démonstration. Rappelons que $\text{symb}(\mathcal{M})$ se construit comme dans la définition 2.15 en prenant $\lambda = \text{val}$. Nous devons montrer que les hypothèses du lemme 2.32 s'appliquent à val. Ici $U = V$.

Pour montrer C1, prenons deux mondes quelconques $w_1, w_2 \in W$ et $a \in \mathfrak{A}$ et notons que nous avons $\lambda(w_1) \cup \lambda(w_2)' \models \Omega_a$ ssi $\text{val}(w_1) \cup \text{val}(w_2)' \models \text{symb}(R_a)$ ssi $(w_1, w_2) \in R_a$.

Pour C2, prenons un $w \in W$ et un $v \in U$. Par définition de λ, nous avons $v \in \lambda(w)$ ssi $v \in \text{val}(w)$.

Pour le sens droite-gauche de C3 : si $s = \lambda(w)$ pour un $w \in W$, alors par la définition de $\text{symb}(W)$, nous avons $\lambda(w) \models \text{symb}(W)$ et par conséquent $\lambda(w)$ est un état de $\text{symb}(\mathcal{M})$. Pour le sens gauche-droite, supposons que s est un état de $\text{symb}(\mathcal{M})$. Ainsi, $s \models \text{symb}(W)$, par conséquent il doit satisfaire un des modèles de la disjonction et il doit y avoir un monde $w \in W$ tel que $s \models \lambda(w) \sqsubseteq V$. Maintenant, par la définition de $\sqsubseteq$, on a $s = \lambda(w) = \text{val}(w)$. □

Nous en avons terminé avec la présentation des concepts et des définitions pour la vérification de modèle sur les représentations symboliques des structures de Kripke.

2.4 Mise à jour des croyances sans postconditions

Maintenant que nous avons vu comment fonctionnait une structure de croyances, nous allons pouvoir nous intéresser au fait de modifier les croyances des agents sur

des structures symboliques, comme peut le faire DEL de manière explicite (déf. 1.28, page 26).

Commençons dans le cas où il n'y a pas de postconditions au modèle d'événements (déf. 1.24, page 25), c'est à dire que le modèle d'événements $\mathcal{E} = \langle E, R^{\mathcal{E}}, \text{pre}, \text{post}\rangle$ se verrait attribuer une définition simplifiée de la fonction post. Pour rappel, la fonction post attribue à chaque événement atomique $e \in E$ et à chaque variable propositionnelle $p \in V$ une formule propositionnelle $\phi \in \mathcal{L}_{\text{prop}}$ de sorte que post soit définie ainsi $PS \times E \to \mathcal{L}_{\text{prop}}$. Dans le cas présent, il suffira simplement d'avoir une fonction de postcondition qui conserve l'état des variables tels qu'il est, i.e. pour un modèle d'événements donné $\mathcal{E} = \langle E, R^{\mathcal{E}}, \text{pre}, \text{post}\rangle$, on aura $\forall p \in PS,\ \forall e \in E,\ \text{post}(p)(e) = p$.

Nous allons réutiliser les mêmes élements que nous utilisons pour représenter une structure de Kripke symbolique :

- un vocabulaire V^+ strictement distinct du vocabulaire V de la structure de croyances ;
- une loi θ^+ similaire à la *loi des mondes* θ encodant par ailleurs les préconditions ;
- ainsi que des relations Ω^+ similaire à Ω, encodant les relations entre les différents événements atomiques.

Plus formellement, voici comment est défini un modèle d'événements symbolique, que l'on appelera *un modificateur de croyances* :

Définition 2.35. *Un* modificateur de croyances χ *sur un vocabulaire* V *est un tuple* $\langle V^+, \theta^+, \Omega^+\rangle$ *où :*

- V^+ *est un ensemble de propositions atomiques tel que* $V \cap V^+ = \varnothing$,
- $\theta^+ \in \mathcal{L}_{\text{EL}}(V \cup V^+)$ *est une formule potentiellement épistémique appelée la* loi des événements,
- $\Omega^+ \in \mathcal{L}_{\text{prop}}(V^+ \cup V^{+'})$ *est un ensemble de formules booléennes pour chaque agent* $a \in \mathfrak{A}$, *appelées* lois d'observations d'événements.

Un événement de croyances *est une modificateur de croyances associé à un sous-ensemble* $x \in 2^{V^+}$, *écrit* $\langle \chi, x\rangle$.

À l'instar de V qui permettait de décrire les différents mondes (par leurs valuations si val est injective), le vocabulaire V^+ permet de différencier les événements atomiques. Néanmoins, cette fois-ci, il n'y a pas de notion de valuation, alors V^+ servira simplement de vocabulaire d'étiquetage pour différencier les événements atomiques.

θ^+ sert à discriminer les événements légaux, comme le fait θ pour les mondes. Néanmoins, nous pouvons remarquer que θ^+ est une formule épistémique sur $V \cup V^+$,

et non pas seulement sur V^+. Ainsi, il est possible d'ajouter des contraintes à θ^+ grâce au vocabulaire V, ce qui permettra de représenter les conditions d'application des événements atomiques sur les mondes –décrits sur le vocabulaire V : ce sont les préconditions.

Enfin, Ω^+ sert à représenter les relations entre les événements atomiques, comme le faisait Ω grâce au vocabulaire V^+ augmenté par le vocabulaire $V^{+'}$.

Exemple 2.36. *Par exemple, avec un vocabulaire $V^+ = \{r, s\}$, il est possible de définir 4 identifiants différents : $\emptyset$, $\{r\}$, $\{s\}$ et $\{r, s\}$.*

Si nous avons $\theta^+ = r \vee s$, l'identifiant $\emptyset$ est impossible, car $\emptyset \not\models \theta^+$. Contraignons ensuite θ^+ avec un vocabulaire $V = \{p\}$ de sorte à avoir $\theta^+ = (r \wedge \neg s \wedge p) \vee (\neg r \wedge s \wedge \neg p)$: cela signifiera que l'événement identifié par $\{r\}$ ne pourra s'appliquer que si p est vrai et l'événement identifié par $\{s\}$ ne pourra s'appliquer que si p est faux. De ce fait, $\{r, s\}$ pourra s'appliquer dans tous les cas, car aucune contrainte n'est formulée : sa précondition est $\top$.

Enfin, tous les modèles de la formule Ω_a^+ sur le vocabulaire $V^+ \cup V^{+'}$ représenteront les liens symboliques entre les événements atomiques. Si nous prenons $\Omega_a^+ = s \leftrightarrow s'$, nous aurons alors 5 modèles pour la formule (en omettant ce qui est illégal à cause de θ^+) : $\{s, s'\}$, $\{s, r', s'\}$, $\{r, r'\}$, $\{r, s, s'\}$, $\{r, s, r', s'\}$. Ces 5 modèles représentent alors une clique entre les événements atomiques $\{r, s\}$ et $\{s\}$, et un arc réfléxif pour l'événement atomique $\{r\}$. Nous pouvons imaginer $\Omega_b^+ = r \leftrightarrow r'$ qui se comportera de la même manière mais avec une clique entre $\{r, s\}$ et $\{r\}$.

Pour représenter une structure de Kripke de manière symbolique, il nous fallait utiliser une fonction d'étiquetage permettant d'encoder la loi des mondes dans θ : λ_W^V. Il s'avère que dans le cas d'une structure de Kripke où la fonction de valuation val est injective, val servait exactement de fonction d'étiquetage. Pour ce qui est des modèles d'événements, nous n'avons pas de telles fonctions à disposition, alors il faut simplement la donner en paramètre de la transformation.

Définition 2.37. *La représentation symbolique d'une fonction de précondition* $\mathrm{pre} : E \to \mathcal{L}_{\mathrm{EL}}(V)$ *via une fonction d'étiquetage $\lambda_E^{V^+}$, notée* $\mathrm{symb}_{\lambda_E^{V^+}}(\mathrm{pre})$ *est la loi des événements, formule potentiellement épistémique sur $V \cup V^+$,* $\mathrm{symb}_{\lambda_E^{V^+}}(\mathrm{pre})$ *telle que :*

$$\mathrm{symb}_{\lambda_E^{V^+}}(\mathrm{pre}) \coloneqq \bigvee_{e \in E} (\lambda_E^{V^+}(e) \sqsubseteq V^+ \wedge \mathrm{pre}(e))$$

La définition de la représentation symbolique d'un modèle d'événements sans postconditions utilise les définitions des représentations symboliques d'une fonction de précondition (déf. 2.37) et d'une relation d'accessibilité (déf. 2.12).

Définition 2.38. *La représentation symbolique d'un modèle d'événements sans postconditions* $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mathrm{pre}\rangle$ *par une fonction d'étiquetage* $\lambda_E^{V^+}$ *(déf. 2.6), notée* $\mathrm{symb}_{\lambda_E^{V^+}}(\mathcal{E})$ *est le modificateur de croyances*

$$\langle V^+,\ \mathrm{symb}_{\lambda_E^{V^+}}(\mathrm{pre}),\ (\mathrm{symb}_{\lambda_E^{V^+}}(R_a^{\mathcal{E}}))_{a\in\mathfrak{A}}\rangle$$

Une façon de voir un modèle de la nouvelle loi des mondes est la suivante : il faut associer chaque étiquetage $\lambda_E^{V^+}(e)$ d'un événement $e \in E$ à sa précondition.

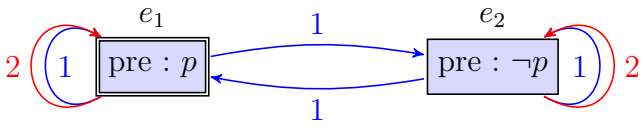


FIGURE 2.1 – Exemple de modèle d'événements

Exemple 2.39. *Soit le modèle d'événements* $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mathrm{pre}\rangle$ *défini ainsi :* $E = \{e_1, e_2\}$, $R_1^{\mathcal{E}} = E^2$, $R_2^{\mathcal{E}} = \{(e_1, e_1), (e_2, e_2)\}$, $\mathrm{pre}(e_1) = p$ *et* $\mathrm{pre}(e_2) = \neg p$, *représenté graphiquement sur le figure 2.1.*

Étant donné que $|E|$ *est égal à 2, nous prendrons* $V^+ = \{e\}$ *et nous pouvons créer* $\lambda_E^{V^+}$ *de telle sorte que* $\lambda_E^{V^+}(e_1) = \emptyset$ *et* $\lambda_{V^+}^{E}(e_2) = \{e\}$.

Par la définition 2.38, nous pouvons construire $\mathrm{symb}_{\lambda_E^{V^+}}(\mathrm{pre}) = (\neg e \wedge p) \vee (e \wedge \neg p)$.

La construction de $\mathrm{symb}_{\lambda_E^{V^+}}(R_r^{\mathcal{E}})$ *(agent rouge) et* $\mathrm{symb}_{\lambda_E^{V^+}}(R_b^{\mathcal{E}})$ *(bleu) se fait de la manière suivante :*

- $\mathrm{symb}_{\lambda_E^{V^+}}(R_r^{\mathcal{E}}) = (\neg e \wedge \neg e') \vee (e \wedge e') \equiv e \leftrightarrow e'$;
- $\mathrm{symb}_{\lambda_E^{V^+}}(R_b^{\mathcal{E}}) = (\neg e \wedge \neg e') \vee (\neg e \wedge e') \vee (e \wedge e') \vee (e \wedge \neg e') \equiv \top$.

L'agent bleu ne différencie donc pas les deux événements atomiques, la formule $\top$ *signifiant toutes les relations symboliques possibles entre les événements. Quant à lui, l'agent rouge les différencie, ce qui est visible par la formule* $e \leftrightarrow e'$.

Nous allons aussi définir la fonction expl. Pour cela nous définissons la représentation symbolique d'une loi des événements θ^+.

Définition 2.40. *Soit* $V \subseteq PS$. *La représentation explicite d'une loi des événements* θ^+ *définie sur le vocabulaire* $V \cup V^+$, *notée* $\mathrm{expl}(\theta^+)$, *est la fonction de précondition* $\mathrm{pre}\colon 2^{V^+} \to \mathcal{L}_{\mathrm{EL}}(V)$ *telle que*

$$\forall e \in 2^{V^+},\ \mathrm{pre}(e) = [e \sqsubseteq V^+]\theta^+$$

Grâce aux définitions des représentations explicites de θ^+ (déf. 2.40) et d'une loi d'observations (déf. 2.19), nous définissons la représentation explicite d'un modificateur de croyances.

Définition 2.41. *La représentation explicite d'un modificateur de croyances* $\chi = \langle V^+, \theta^+, \Omega^+ \rangle$ *sans postconditions, notée* $\mathrm{expl}(\chi)$*, avec* $E = 2^{V^+}$*, est le modèle d'événements avec*

$$\langle E,\ (\mathrm{expl}_E(\Omega_a^+))_{a \in \mathfrak{A}},\ \mathrm{expl}(\theta^+) \rangle$$

Notons que l'on ne peut pas facilement reconstruire E à partir de θ^+, comme c'était fait dans la définition de la dé-symbolicisation d'une structure de croyances 2.20, page 40. En effet, θ^+ n'est pas défini que sur V^+, mais aussi sur V. Pour ce faire, chaque précondition $\mathrm{pre}(e)$ d'un événement e est alors récupéré grâce à l'identifiant $([e \sqsubseteq V^+])$ dans la formule θ^+.

Remarque 2.42. *E pourrait éventuellement être recrée en définissant E comme étant la formule $\forall V \theta^+$. On obtiendra alors toutes les valuations sur V^+ qui représentent des événements atomiques, tout en enlevant le vocabulaire V.*

Notons que cette généralisation avec $E \coloneqq 2^{V^+}$ n'est pas un problème, car pour ce qui est de la création de pre*, on obtient* $\mathrm{pre}(e) = \bot$ *si* $e \not\models \theta^+$*. L'événement sera alors ignoré.*

Nous conservons néanmoins la définition de Gattinger tel quel.

Exemple 2.43. *Faisons la transformation inverse à celle que nous avons présenté dans l'exemple précédent (ex. 2.39). Nous avons donc* $\chi = \langle V^+ = \{e\}, \theta^+ = (\neg e \wedge p) \vee (e \wedge \neg p), \Omega_r = e \leftrightarrow e', \Omega_b = \top \rangle$.

Nous pouvons donc recréer $\mathcal{E}$ de la sorte, avec :

- $E = \{\emptyset, \{e\}\}$;
- $\mathrm{expl}(\theta^+)(\emptyset) = \neg e \wedge p$;
- $\mathrm{expl}(\theta^+)(\{e\}) = e \wedge \neg p$;
- $\mathrm{expl}_E(\Omega_r^+) = \{(\emptyset, \emptyset), (\{e\}, \{e\})\}$;
- $\mathrm{expl}_E(\Omega_b^+) = E^2$.

Une fois le modificateur de croyances défini, il est possible de mettre à jour les croyances des agents via l'équivalent symbolique du *product update* (déf. 1.28).

Définition 2.44. *La* modification des croyances *d'une structure de croyances* $\langle \mathfrak{F}, s \rangle$ *(déf. 2.14, page 38) par un modificateur de croyances* $\langle \chi, x \rangle$ *est défini de sorte à ce que* $\langle \mathfrak{F}, s \rangle \otimes \langle \chi, x \rangle \coloneqq \langle \langle V^\otimes, \theta^\otimes, \Omega^\otimes \rangle, s \cup x \rangle$ *où :*

- $V^\otimes \coloneqq V \cup V^+$;
- $\theta^\otimes \coloneqq \theta \wedge ||\theta^+||_{\mathfrak{F}}$;

- $\Omega_a^{\otimes} := \Omega_a \wedge \Omega_a^{+}$.

L'idée est simple :

- le vocabulaire de la structure résultante est simplement l'union des différents vocabulaires V et V^+ ;
- la nouvelle loi des mondes est la conjonction de la loi initiale θ avec la traduction locale de la loi des événements $\theta^+ : ||\theta^+||_{\mathfrak{F}}$, de sorte à ce qu'une contrainte, ou « filtre », s'applique à θ au moment de la mise à jour des croyances qui se manifeste simplement par la conjonction de θ et de θ^+. Notons que l'utilisation de la traduction $||\theta^+||_{\mathfrak{F}}$ permet d'appliquer des préconditions épistémiques qui seront interprétées dans la structure de croyances $\mathfrak{F}$. Néanmoins, l'idée principale est de pouvoir appliquer un événement symbolique atomique uniquement aux mondes symboliques correspondants.
- les nouvelles lois d'observations sont quant à elles aussi la conjonction des lois d'observations et des lois d'observations d'événements.

Exemple 2.45. *Reprenons la structure de croyances présentée dans l'exemple 2.22. Nous y avions* $\mathfrak{F} = \langle V = \langle p, q\rangle, \theta = q, \Omega = \{p \leftrightarrow p', \top\}\rangle$. *Notons que* V *est ici* $\{p, q\}$, *et que l'on a bien* $V \cap V^+ = \emptyset$.

Nous reprenons aussi le modificateur de croyances crée dans l'exemple 2.39 et mis en entrée de l'exemple 2.43.

θ *étant* q, *nous pouvons construire la nouvelle loi des états :* $\theta^{\otimes} = q \wedge ||(\neg e \wedge p) \vee (e \wedge \neg p)||_{\mathfrak{F}} = q \wedge ((\neg e \wedge p) \vee (e \wedge \neg p))$. *Les deux modèles de la formule* $\theta^{\otimes}$ *sont ainsi :* $\{p, q\}$ *(sans* e*) (qui est moralement le résultat de l'application de* e_1 *sur le modèle* $\{p, q\}$ *de* θ*) et* $\{q, e\}$ *(sans* p*) (qui est le résultat de* e_2 *sur* $\{q\}$*).*

Nous pouvons aussi noter que les deux modèles de la formule $\theta^{\otimes}$ *sont véritablement la représentation symbolique des mondes (sur le vocabulaire double* $V \cup V^+$*) que nous souhaitons. Par exemple, l'événement* e_1 *ne s'est pas appliqué au monde* w_2, *car* $(\neg e \wedge p)$ *est incompatible avec* $\neg pq$.

Pour ce qui est de $\Omega^{\otimes}$, *avec* $\Omega_1^+ = \top$ *et* $\Omega_2^+ = e \leftrightarrow e'$, *nous avons ainsi :*

- $\Omega_1^{\otimes} = \Omega_1 \wedge \Omega_1^+ = p \leftrightarrow p' \wedge \top$;
- $\Omega_2^{\otimes} = \Omega_2 \wedge \Omega_2^+ = \top \wedge e \leftrightarrow e'$.

Il y a deux modèles à $\theta^{\otimes}$ *(*$pq\bar{e}$ *et* $\bar{p}qe$*). Comme l'agent* 1 *différencie les mondes qui ont des valeurs de l'atome* p *différents, alors l'agent* 1 *différencie ces deux mondes ; il en va de même avec l'agent* 2 *qui différencie les événements atomiques étiquetés par* $\emptyset$ *et* $\{e\}$, *alors qu'il ne différenciaient à la base pas les deux mondes valués* $\{p, q\}$ *et* $\{q\}$.

Nous avons vu la définition d'un modificateur de croyances sans postconditions. Nous allons maintenant voir une version plus simple de cette définition, quand il n'y

a qu'un seulement événement atomique et que V^+ peut être réduit à un ensemble vide.

Définition 2.46 (Annonces)**.** *Une annonce d'une formule épistémique ϕ à tous les agents $a \in \mathfrak{A}$ est un modificateur de croyances défini de la sorte :*

$$\chi = \langle V^+, \theta^+, \Omega^+ \rangle \text{ où } V^+ = \emptyset,\ \theta^+ = \phi,\ \Omega_a^+ = (\top)_{a \in \mathfrak{A}}$$

Par ailleurs, une annonce (dite semi-publique *ou* semi-privée*) d'une formule ϕ faite à un sous-groupe d'agents $A \subseteq \mathfrak{A}$ (avec les autres agents $a \subseteq A \backslash \mathfrak{A}$ qui sont conscients qu'une annonce a été faite) est :*

$$\chi = \{V^+, \theta^+, \Omega^+\} \text{ où } V^+ = \{e_\phi\},\ \theta^+ = e_\phi \leftrightarrow ||\phi||_{\mathfrak{F}},\ \Omega^+ = (\Omega_a^+)_{a \in \mathfrak{A}},$$
$$\text{avec } \Omega_a^+ = e_\phi \leftrightarrow e'_\phi \text{ si } a \in A,\ \top \text{ sinon.}$$

De plus, une annonce privée (les agents non-inclus dans le sous-groupe ne sont pas conscients qu'une annonce a été faite) d'une formule ϕ faite à un sous-groupe d'agents $A \subseteq \mathfrak{A}$ est :

$$\chi = \{V^+, \theta^+, \Omega^+\} \text{ où } V^+ = \{e_\phi\},\ \theta^+ = e_\phi \rightarrow ||\phi||_{\mathfrak{F}},\ \Omega^+ = (\Omega_a^+)_{a \in \mathfrak{A}},$$
$$\text{avec } \Omega^+ = e_\phi \leftrightarrow e'_\phi \text{ si } a \in A,\ \neg e'_\phi \text{ sinon.}$$

L'idée est la suivante : avec une annonce d'une formule épistémique ϕ, il faut simplement filtrer pour tous les agents tous les mondes qu'ils ne considèrent plus comme possibles. Dans le cas d'une annonce publique, il suffit d'avoir $\theta^+ = \phi$ et aucune relation supplémentaire n'est nécéssaire dans Ω^+. En effet, le *product update* imposera la nouvelle loi des mondes comme étant $\theta^\otimes = \theta \wedge ||\phi||_{\mathfrak{F}}$. Ainsi, aucun agent ne pourra considérer un monde qui n'est pas compatible avec ϕ.

Pour ce qui est dans des annonces semi-publiques, il s'agit de créer deux événements atomiques discernables par un identifiant noté e_ϕ et d'utiliser cet identifiant pour que les agents les discernent ou non grâce à Ω^+.

Enfin, l'annonce privée fonctionne de la même manière. Néanmoins la loi des événements n'est pas si catégorique car elle permet aux agents de considérer des mondes où l'événement atomique n'a pas lieu et où ϕ est vrai ($\neg e_\phi \wedge \phi$). Enfin, si les agents reçoivent l'annonce ($a \in A$), alors ils distinguent la valeur de e_ϕ et sinon, il est impossible pour eux d'envisager que l'action atomique étiquetée par $\{e_\phi\}$ ait lieu.

Maintenant que nous avons décrit le fonctionnement des modificateurs de croyances, une définition permet d'étendre les traductions locales des formules épistémiques du langage $\mathcal{L}_{\text{EL}}$ (déf. 2.26) en formule propositionnelle $\mathcal{L}_{\text{prop}}$, en y ajoutant l'opérateur $[\mathcal{E}, e]$ du langage $\mathcal{L}_{\text{DEL}}$.

Définition 2.47. *Supposons un modificateur de croyances $\chi = \langle V^+, \theta^+, \Omega^+ \rangle$. L'opérateur dynamique est interprété comme suit :*

$$\langle \mathfrak{F}, s \rangle \models [\chi, x]\phi \text{ ssi } \langle \mathfrak{F}, s \rangle \models [x \sqsubseteq V^+]\theta^+ \implies \langle \mathfrak{F} \otimes \chi, s^x \rangle \models \phi$$

où s^x est la définition du nouvel état (déf. 2.51).

On peut alors étendre la définition des traductions locales (déf. 2.26), avec le cas dynamique.

Définition 2.48. *Pour toute structure de croyances $\mathfrak{F} = \langle V, \theta, \Omega \rangle$ et toute formule $\phi \in \mathcal{L}_{\text{DEL}}(V)$ (sans postconditions), la* traduction locale $||\phi||_{\mathfrak{F}}$ *sur $\mathcal{L}_{\text{prop}}(V)$ par l'opérateur $[\chi, x]$ via χ sans postconditions (déf. 2.35) est définie comme suit, en complément de la traduction locale précédemment définie 2.26, page 43 :*

$$||[\chi, x]\phi||_{\mathfrak{F}} \coloneqq |||[x \sqsubseteq V^+]\theta^+||_{\mathfrak{F}} \to [x \sqsubseteq V^+]||\phi||_{\mathfrak{F}\otimes\chi}$$

Cette traduction fonctionne de la manière suivante :

- nous récupérons la traduction locale de ϕ par rapport à la structure mise à jour par le *product update* $\mathfrak{F} \otimes \chi$: $||\phi||_{\mathfrak{F}\otimes\chi}$;
- la formule ainsi obtenue est conditionnée par l'événement pointé x : $[x \sqsubseteq V^+]||\phi||_{\mathfrak{F}\otimes\chi}$;
- il faut récupérer l'ensemble des modèles de mondes possibles sur $\mathfrak{F}$ par les préconditions modélisées dans θ^+, toujours conditionné par l'événement pointé x : $|||[x \sqsubseteq V^+]\theta^+||_{\mathfrak{F}}$;
- il faut que l'ensemble des mondes avant mise à jour et vérifiant les préconditions impliquent les mondes filtrés par ϕ après mise à jour.

Nous rédigerons les théorèmes et les preuves associées à ces définitions dans la section suivante, qui est un cas avec postconditions qui englobe celui-ci.

2.5 Mise à jour des croyances avec postconditions

Après avoir vu la mise à jour des croyances sans postconditions, nous allons décrire la mise à jour des croyances avec des postconditions.

Bien évidemment, tout ce qui a été expliqué dans la sous-partie précédente reste valable, et nous allons ici nous attacher à la description de l'utilisation des postconditions.

De nouvelles notations apparaissent ici : V_- et θ_-. Elles permettent respectivement de représenter la liste des variables propositionnelles de V qui seront modifiées et la manière dont ces variables vont être modifiées.

Par ailleurs, un autre ensemble de variables disjoints de V et V^+ (et de V_-, car $V_- \subseteq V$) va être introduit : V_-°. Le vocabulaire V_-° fonctionne de la même manière que l'utilisation des variables primes de la définition 2.9, page 37, qui utilise une bijection pour dupliquer le vocabulaire.

Définition 2.49. *Un modificateur de croyances avec postconditions pour le vocabulaire V est un tuple $\chi = \langle V^+, \theta^+, \Omega^+, V_-, \theta_- \rangle$ où :*

- *V^+, θ^+ et Ω^+ sont définis comme précédemment dans la définition 2.35, page 48 ;*
- *$V_- \subseteq V$ est un sous-ensemble du vocabulaire original appelé le* vocabulaire modifié *;*
- *$\theta_- : V_- \to \mathcal{L}_{\text{prop}}(V \cup V^+)$ est une fonction assignant à chaque proposition atomique modifiée une formule booléenne. Cette fonction est appelée le* loi de modification*.*

V_-° est le vocabulaire qui va nous permettre d'historiciser les valeurs des variables qui vont être modifiées, c'est à dire de conserver une trace de leurs anciennes valeurs.

En effet, si V_-, qui est inclus dans V, est la liste des variables modifiées, alors V_-° est alors défini de la sorte : $V_-^\circ = \{p^\circ \mid p \in V_-\}$, avec $V \cap V^+ = \emptyset$, $V \cap V_-^\circ = \emptyset$ et $V^+ \cap V_-^\circ = \emptyset$.

Exemple 2.50. *Reprenons notre exemple 2.45, page 52. Nous avions $\chi = \langle V^+, \theta^+, \Omega^+, V_- \rangle$ avec $V^+ = \{e\}$, $\theta^+ = (\neg e \wedge p) \vee (e \wedge \neg p)$, $\Omega_1^+ = \top$ et $\Omega_2^+ = e \leftrightarrow e'$.*

Ajoutons V_- et θ_- de sorte à avoir $\chi = \langle V^+, \theta^+, \Omega^+, V_-, \theta_- \rangle$. Par exemple, faisons en sorte que l'événement identifié par $\emptyset$ modifie la variable p en $\neg p$, et que l'événement identifié par $\{e\}$ ne modifie rien.

Pour cela, on aura le vocabulaire V_- qui décrira le vocabulaire modifié : ici $V_- = \{p\}$.

Quant à θ_-, on aura $\theta_-(p) = (\neg e \wedge \neg p) \vee (e \wedge p)$, qui signifie que quand $\{e\}$ s'appliquera, le valeur de p sera conservée et dans l'autre événement atomique, la valeur de p sera inversée.

De la même manière que nous modifions les croyances des agents sans les postconditions, nous allons pouvoir le faire avec des événements avec postconditions, seulement il faut ajouter des éléments V_- et θ_-.

Définition 2.51. *Le résultat du* product update *d'une structure de croyances $\mathfrak{F} = \langle V, \theta, \Omega \rangle$ via un modificateur de croyances $\chi = \langle V^+, \theta^+, \Omega^+, V_-, \theta_- \rangle$ est défini ainsi : $\mathfrak{F} \otimes \chi \coloneqq \langle V^\otimes, \theta^\otimes, \Omega^\otimes \rangle$ où*

- *$V^\otimes \coloneqq V \cup V^+ \cup V_-^\circ$ où V_-° est le vocabulaire V_- renommé avec le symbole $\circ$: $V_-^\circ = \{p^\circ \mid p \in V_-\}$,*

- $\theta^{\otimes} \coloneqq [V_{-} \mapsto V_{-}^{\circ}](\theta \wedge ||\theta^{+}||_{\mathfrak{F}}) \wedge \bigwedge_{p \in V_{-}} (p \leftrightarrow [V_{-} \mapsto V_{-}^{\circ}](\theta_{-}(p)))$
- $\Omega_{a}^{\otimes} \coloneqq ([V_{-} \mapsto V_{-}^{\circ}][V'_{-} \mapsto (V_{-}^{\circ})']\Omega_{a}) \wedge \Omega_{a}^{+}$

Un événement *est un couple* $\langle \chi, x \rangle$ *où* $x \in 2^{V^{+}}$. *Étant donné une scène* $\langle \mathfrak{F}, s \rangle$ *et un événement* $\langle \chi, x \rangle$, *nous avons* $\langle \mathfrak{F}, s \rangle \otimes \langle \chi, x \rangle \coloneqq (\mathfrak{F} \otimes \chi, s^{x})$ *où le nouvel état est donné par :*

$$s^{x} \coloneqq (s \backslash V_{-}) \cup (s \cap V_{-})^{\circ} \cup x \cup \{p \in V_{-} \mid s \cup x \models \theta_{-}(p)\}$$

Voilà comment cela fonctionne :

- Le vocabulaire $V^{\otimes}$ est simplement l'union des vocabulaires décrivant $\mathfrak{F}$ et χ, ainsi que V_{-}°, décrivant l'ancien état des variables modifiées.
- Sans postconditions, nous avons $\theta^{\otimes} = \theta \wedge ||\theta^{+}||_{\mathfrak{F}}$. Néanmoins, il faut pouvoir modifier les variables. Pour cela, le vocabulaire modifié $V_{-} \subseteq V$ est renommé en V_{-}° sur les lois des mondes et lois des événements. Enfin, il faut préciser quelles sont les nouvelles valeurs des variables de V_{-} ; pour cela, on leur assigne, grâce à l'équivalence, la valeur de la postcondition définie dans θ_{-} qui est historicisée. De cette manière, si nous avons par exemple $\theta_{-}(p) = \neg p$, nous aurons alors $p \leftrightarrow \neg p^{\circ}$.
- Pour ce qui est des relations, il suffit de transférer les croyances qu'ont les agents sur les variables V_{-} sur les nouvelles variables V_{-}°. Pour cela, un renommage a lieu sur les variables non primées, mais aussi primées.
- Enfin, le nouveau monde pointé est l'union des variables non modifiées, des variables modifiées dans s historicisées, de l'événement pointé x, et des variables de V_{-} qui seront vraies par les postconditions définies par θ_{-}.

Exemple 2.52. *Gardons notre structure de croyances de l'exemple 2.22 (* $V = \{p, q\}$, $\theta = q$, $\Omega_{1} = p \leftrightarrow p'$, $\Omega_{2} = \top$*) et modifions-le avec le modificateur de croyances de l'exemple 2.50 (* $V^{+} = \{e\}$, $\theta^{+} = (\neg e \wedge p) \vee (e \wedge \neg p)$, $\Omega_{1}^{+} = \top$ *et* $\Omega_{2}^{+} = e \leftrightarrow e'$, $V_{-} = \{p\}$ *et* $\theta_{-}(p) = (\neg e \wedge \neg p) \vee (e \wedge p)$. *).*

En suivant la définition 2.51, nous obtenons $V^{\otimes} = \{p, q, e, p^{\circ}\}$.

Avec $[V_{-} \mapsto V_{-}^{\circ}]$ *signifiant ici « remplacer* p *par* p° *», nous avons :*

$$\begin{aligned}\theta^{\otimes} &= [V_{-} \mapsto V_{-}^{\circ}](q \wedge ((\neg e \wedge p) \vee (e \wedge \neg p))) \wedge (p \leftrightarrow [V_{-} \mapsto V_{-}^{\circ}]((\neg e \wedge \neg p) \vee (e \wedge p))) \\ &= (q \wedge (\neg e \wedge p^{\circ}) \vee (e \wedge \neg p^{\circ})) \wedge p \leftrightarrow ((\neg e \wedge \neg p^{\circ}) \vee (e \wedge p^{\circ}))\end{aligned}$$

Les modèles de cette formule sont au nombre de deux : $\{p^{\circ}, q\}$ *et* $\{q, e\}$. *Ces deux modèles sont respectivement la modification des valuations* $\{p, q\}$ *par l'identifiant* $\emptyset$ *et* $\{q\}$ *par* $\{e\}$. *Nous pouvons voir que la valeur de* p *a bien été inversée dans le premier cas, et conservée dans le second.*

Enfin, nous pouvons simplement construire les nouvelles relations de la sorte :

- $\Omega_{1}^{\otimes} = ([V_{-} \mapsto V_{-}^{\circ}][V'_{-} \mapsto (V_{-}^{\circ})']\Omega_{1}) \wedge \Omega_{1}^{+} = p^{\circ} \leftrightarrow p^{\circ'}$;
- $\Omega_{2}^{\otimes} = ([V_{-} \mapsto V_{-}^{\circ}][V'_{-} \mapsto (V_{-}^{\circ})']\Omega_{2}) \wedge \Omega_{2}^{+} = e \leftrightarrow e'$.

Les agents conservent alors leurs relations précédentes sur les variables V_-, renommées en V_-° et ajoutent des relations sur les variables de Ω^+.

Quant au nouveau monde pointé noté s^x, si nous avons $s = \{p, q\}$ et $x = \emptyset$, il est défini ainsi :

$$\begin{aligned} s^x &= \{p,q\}\backslash\{p\} \cup (\{p,q\} \cap \{p\})^\circ \cup \emptyset \cup \emptyset \\ &\qquad\qquad (\textit{car } \{p,q\} \cup \emptyset \not\models (\neg e \wedge \neg p) \vee (e \wedge p)) \\ &= \{q\} \cup \{p^\circ\} \cup \emptyset \cup \emptyset \\ &= \{q, p^\circ\} \end{aligned}$$

Nous pouvons voir que le monde initial $\{p, q\}$ a bien été modifié de sorte à ce que p soit remplacé par $\neg p$, tout en gardant un historique de l'ancienne valeur de p : p°.

Il est possible de transformer un modèle d'événements classique de DEL en un modificateur de croyances grâce à la transformation symb. Comme précédemment, nous allons commencer définir la représentation symbolique d'une partie du tuple : la fonction post.

Définition 2.53. *La représentation symbolique d'une fonction de postconditions* $\mathrm{post} : V \times E \to \mathcal{L}_{\mathrm{prop}}(V)$ *via une fonction d'étiquetage* $\lambda_E^{V^+}$*, notée* $\mathrm{symb}_{\lambda_E^{V^+}}(\mathrm{post})$ *est le tuple* $\langle V_-, \theta_-\rangle$ *où* $V_- \subseteq V$ *et* θ_- *est une loi de modification tels que :*

- $V_- \coloneqq \{p \in V \mid \exists e \in E : \mathrm{post}(p)(e) \neq p\}$ *;*
- *pour* $p \in V_-$, $\theta_-(p) \coloneqq \bigvee_{e \in E}(\lambda_E^{V^+}(e) \sqsubseteq V^+ \wedge \mathrm{post}(p)(e))$

Définition 2.54. *La représentation symbolique d'un modèle d'événements* $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mathrm{pre}, \mathrm{post}\rangle$ *défini sur un vocabulaire* V *via une fonction d'étiquetage* $\lambda_E^{V^+}$*, noté* $\mathrm{symb}_{\lambda_E^{V^+}}(\mathcal{E})$*, est le modificateur de croyances*

$$\langle V^+, \mathrm{symb}_{\lambda_E^{V^+}}(\mathrm{pre}),\ (\mathrm{symb}_{\lambda_E^{V^+}}(R_a^{\mathcal{E}}))_{a\in\mathfrak{A}},\ V_-,\ \theta_-\rangle \ \textit{où}\ \mathrm{symb}_{\lambda_E^{V^+}}(\mathrm{post}) = \langle V_-, \theta_-\rangle.$$

Dans cette définition, nous exploitons la représentation d'un modèle d'événements sans postconditions (déf. 2.38). Le vocabulaire V_- est simple à obtenir, car il faut simplement repérer quelles variables sont modifiées par la post. Enfin, il faut construire une formule propositionnelle θ_- pour chaque p, pour cela, il faut considérer toutes les modifications possibles de p pour chaque événement $e \in E$ et en faire la disjonction. Notons qu'aucun modèle des sous-formules $\lambda_E^{V^+}(e) \sqsubseteq V^+ \wedge \mathrm{post}(p)(e)$ ne peuvent être en commun, car l'injectivité de la fonction d'étiquetage $\lambda_E^{V^+}$ l'impose.

Exemple 2.55. *Nous avons déjà illustré la représentation symbolique d'un modèle d'événements sans postconditions dans l'exemple 2.45. Nous avions un modèle d'événements* $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mathrm{pre}, \mathrm{post}\rangle$ *auquel nous ajoutons maintenant* $\mathrm{post}(p)(e_1) = \neg p$. *Pour construire les derniers éléments* V_- *et* θ_-, *nous pouvons simplement dire ici que* $V_- = \{p\}$ *car seul* p *est modifié et que* $\theta_-(p) = (\neg e \wedge \neg p) \vee (e \wedge p)$.

Notons que cet exemple est le même que celui construit à l'exemple 2.50 pour illustrer un modificateur de croyances avec postconditions.

La fonction inverse est aussi définie en augmentant la définition de la transformation d'un modificateur de croyances en modèle d'événements sans postconditions (déf. 2.41, page 51).

Définition 2.56. *La représentation explicite d'une loi de modification* $\theta_- : V_- \to \mathcal{L}_{\mathrm{prop}}(V \cup V^+)$ *notée* $\mathrm{expl}_{V,V^+}(\theta_-)$ *est la fonction de postcondition* post *telle que :*

$$\forall p \in V,\ \forall e \in 2^{V^+},\ \mathrm{post}(p)(e) \coloneqq \begin{cases} [e \sqsubseteq V^+]\theta_-(p) & \text{si } \theta_- \text{ est définie sur } p \\ p & \text{sinon.} \end{cases}$$

Définition 2.57. *La représentation explicite d'un modificateur de croyances* $\chi = \langle V^+, \theta^+, \Omega^+, V_-, \theta_-\rangle$ *pour le vocabulaire* V *notée* $\mathrm{expl}(\mathcal{E})$ *est le modèle d'événements*

$$\langle 2^{V^+},\ (\mathrm{expl}_{2^{V^+}}(\Omega_a^+))_{a\in\mathfrak{A}},\ \mathrm{expl}(\theta^+),\ \mathrm{expl}_{V,V^+}(\theta_-)\rangle$$

Exemple 2.58. *Conservons le résultat de notre exemple 2.55 et essayons de reconstituer le modèle d'événements initial grâce à la définition 6.10.*

Ainsi nous avons $\chi = \langle V^+, \theta^+, \Omega^+, V_-, \theta_-\rangle$ *avec* $V^+ = \{e\}$, $\theta^+ = (\neg e \wedge p) \vee (e \wedge \neg p)$, $\Omega_1^+ = \top$ *et* $\Omega_2^+ = e \leftrightarrow e'$, $V_- = \{p\}$ *et* $\theta_-(p) = (\neg e \wedge \neg p) \vee (e \wedge p)$.

E est alors construit comme étant 2^{V^+}, *soit* $\{\emptyset, \{e\}\}$. *Nous avons, en écrivant* $\mathrm{pre} = \mathrm{expl}(\theta^+)$, $R_a^{\mathcal{E}} = \mathrm{expl}_{2^{V^+}}(\Omega_a^+)$ *et* $\mathrm{post} = \mathrm{expl}_{V,V^+}(\theta_-)$ *aussi :*

- $\mathrm{pre}(\emptyset) = [\emptyset \sqsubseteq V^+]((\neg e \wedge \neg p) \vee (e \wedge p) = \neg p)$;
- $\mathrm{pre}(\{e\}) = [\{e\} \sqsubseteq V^+]((\neg e \wedge \neg p) \vee (e \wedge p) = p)$;
- $R_1^{\mathcal{E}} = \{(e_1, e_2) \mid e_1 \cup e_2{}' \models \Omega_1^+\} = E^2$ *(car* $\top$ *permet tous les modèles)* ;
- $R_2^{\mathcal{E}} = \{(e_1, e_2) \mid e_1 \cup e_2{}' \models \Omega_2^+\} = \{(\emptyset, \emptyset), (\{e\}, \{e\})\}$, *car* $\emptyset$ *et* $\{e, e'\}$ *sont les seuls modèles de* $e \leftrightarrow e'$;
- $\mathrm{post}(p)(\emptyset)[\emptyset \sqsubseteq V^+]((\neg e \wedge \neg p) \vee (e \wedge p)) = \neg p$;
- $\mathrm{post}(p)(\{e\}) = [\{e\} \sqsubseteq V^+]((\neg e \wedge \neg p) \vee (e \wedge p)) = p$;

De fait, nous retrouvons bien le modèle d'événements initial.

Néanmoins, il faut préciser une chose : la désymbolisation du modificateur de croyances rend un modèle équivalent au modèle d'événements initial, mais pas identique. En effet, on voit bien que le fait de créer E comme étant 2^{V^+} fait exploser le nombre d'événements explicites.

L'interprétation de l'opérateur dynamique sur une structure symbolique est défini identiquement au cas sans postconditions (déf. 2.47).

Remarque 2.59. *Comme pour le langage $\mathcal{L}_{\mathrm{DEL}}$, qui contient des opérateurs dynamiques pour les modèles d'événements, nous pouvons également définir un langage avec des opérateurs dynamiques pour les modificateurs de croyances. Le langage que nous obtiendrons sera plus succinct, mais pas plus expressif. Par abus de notation et simplicité, nous ne différencierons pas les deux langages ; nous ne parlerons que de $\mathcal{L}_{\mathrm{DEL}}$, qui implicitement utilisera $[\mathcal{E}, e]$ dans le cas explicite et $[\chi, x]$ dans le cas symbolique.*

Définition 2.60. *Supposons un modificateur de croyances $\chi = \langle V^+, \theta^+, V_-, \theta_-, \Omega^+\rangle$. L'opérateur dynamique est interprété comme suit :*

$$\langle \mathfrak{F}, s\rangle \models [\chi, x]\phi \text{ ssi } \langle \mathfrak{F}, s\rangle \models [x \sqsubseteq V^+]\theta^+ \text{ implique } \langle \mathfrak{F} \otimes \chi, s^x\rangle \models \phi$$

où s^x est la définition du nouvel état (déf. 2.51).

En outre, il y a le pendant de la définition 2.48 pour la traduction locale d'une formule épistémique dynamique avec postconditions, qui est définie ainsi :

Définition 2.61. *Soit $\chi = \langle V^+, \theta^+, \Omega^+, V_-, \theta_-\rangle$ un modificateur de croyances avec postcondition. Pour toute structure de croyances $\mathfrak{F} = \langle V, \theta, \Omega\rangle$ et toute formule $\phi \in \mathcal{L}_{\mathrm{DEL}}(V)$, en complément de la définition de la traduction locale (déf. 2.26), la* traduction locale *de $|||[\chi, x]||_{\mathfrak{F}}$ sur $\mathcal{L}_{\mathrm{prop}}(V)$ via χ est définie comme suit : :*

$$|||[\chi, x]\phi||_{\mathfrak{F}} \coloneqq |||[x \sqsubseteq V^+]\theta^+||_{\mathfrak{F}} \to [V_-^\circ \mapsto V_-][x \sqsubseteq V^+][V_- \mapsto \theta_-(V_-)]||\phi||_{\mathfrak{F}\otimes\chi}$$

L'idée derrière cette traduction est similaire à celle présentée pour la définition 2.48, qui est :

$$|||[\chi, x]\phi||_{\mathfrak{F}} \coloneqq |||[x \sqsubseteq V^+]\theta^+||_{\mathfrak{F}} \to [x \sqsubseteq V^+]||\phi||_{\mathfrak{F}\otimes\chi}$$

Néanmoins, cette fois-ci, il y a l'utilisation d'une déshistoricisation avec le renommage des variables historiques V_-° en variable V_- (qui est un sous-ensemble de V). En quelques sortes, on annule l'application de l'événement afin de revenir à l'état précédent. $[V_-^\circ \mapsto V_-]$ permet ainsi de revenir aux variables précédentes après avoir appliqué le *product update*. Le renommage $[V_- \mapsto \theta_-(V_-)]$ remplace

quant à lui toutes variables de V_- par leurs postconditions de sorte à leur redonner leurs valeurs précédentes. Cela est la subtitution simultanée de chaque $q \in V_-$ par $\theta_-(q)$.

Les théorèmes suivants, prouvés par Gattinger dans sa thèse [Gat18b], montrent l'équivalence des *product updates* explicites et symboliques (avec ou sans postconditions). Nous reprenons ici les démonstrations de Gattinger (en adaptant simplement les notations) dans l'objectif que la thèse soit auto-contenue, les preuves du chapitre 5 s'appuyant en effet sur ces théorèmes. Les théorèmes et les preuves considèrent le cas général, i.e. même sans les postconditions.

Le théorème qui en découle est aussi véridique :

Théorème 2.62. *La traduction donnée par la définition 2.61 est vraie : pour toute structure de croyances $\mathfrak{F}$, tout état s, tout événement $\langle \chi, x \rangle$ et toute formule ϕ, nous avons :*

$$\langle \mathfrak{F}, s \rangle \models [\chi, x]\phi \iff s \models ||[\chi, x]\phi||_{\mathfrak{F}}$$

Démonstration. Nous commençons par considérer qu'un événement x est applicable sur un monde s, de sorte que $\langle \mathfrak{F}, s \rangle \models [x \sqsubseteq V^+]\theta^+$. Nous avons la chaîne d'équivalences suivante :

$$\begin{aligned}
& s^x \models ||\phi||_{\mathfrak{F}\otimes\chi} && (1)\\
\iff & (s\backslash V_-) \cup (s \cap V_-)^\circ \cup x \cup \{p \in V_- \mid s \cup x \models \theta_-(p)\} \models ||\phi||_{\mathfrak{F}\otimes\chi} && (2)\\
\iff & (s\backslash V_-) \cup (s \cap V_-)^\circ \cup x \models [V_- \mapsto [V_- \mapsto V_-^\circ]\theta_-(V_-)]||\phi||_{\mathfrak{F}\otimes\chi} && (3)\\
\iff & (s\backslash V_-) \cup (s \cap V_-)^\circ \models [x \sqsubseteq V^+][V_- \mapsto [V_- \mapsto V_-^\circ]\theta_-(V_-)]||\phi||_{\mathfrak{F}\otimes\chi} && (4)\\
\iff & s \models [V_-^\circ \mapsto V_-][x \sqsubseteq V^+][V_- \mapsto [V_- \mapsto V_-^\circ]\theta_-(V_-)]||\phi||_{\mathfrak{F}\otimes\chi} && (5)\\
\iff & s \models [V_-^\circ \mapsto V_-][x \sqsubseteq V^+][V_- \mapsto \theta_-(V_-)]||\phi||_{\mathfrak{F}\otimes\chi} && (6)
\end{aligned}$$

Par la sémantique de $[\chi, x]$, nous pouvons écrire : $\langle \mathfrak{F}, s \rangle \models [\chi, x]\phi \iff \langle \mathfrak{F}, s \rangle \models [x \sqsubseteq V^+]\theta^+$ impl. $\langle \mathfrak{F} \otimes \chi, s^x \rangle \models \phi$

déf. 2.60

$\iff s \models ||[x \sqsubseteq V^+]\theta^+||_{\mathfrak{F}}$ impl. $s^x \models ||\phi||_{\mathfrak{F}\otimes\chi}$

thm. 2.30

$\iff s \models ||[x \sqsubseteq V^+]\theta^+||_{\mathfrak{F}}$ impl. $s \models [V_-^\circ \mapsto V_-][x \sqsubseteq V^+][V_- \mapsto \theta_-(V_-)]||\phi||_{\mathfrak{F}\otimes\chi}$

(1) $\iff$ (6) ci-dessus

$\iff s \models ||[x \sqsubseteq V^+]\theta^+||_{\mathfrak{F}} \to [V_-^\circ \mapsto V_-][x \sqsubseteq V^+][V_- \mapsto \theta_-(V_-)]||\phi||_{\mathfrak{F}\otimes\chi}$
$\iff s \models ||[\chi, x]\phi||_{\mathfrak{F}}$

(déf. 2.61)

□

Montrons maintenant l'équivalence des *product updates* symboliques et explicites.

Théorème 2.63.
La fonction expl *(déf. 6.10) préserve la vérité : pour toute scène $\langle \mathfrak{F}, s\rangle$, tout événement $\langle \chi, x\rangle$ et toute formule ϕ sur le vocabulaire V de $\mathfrak{F}$, on a :*

$$\langle \mathfrak{F}, s\rangle \otimes \langle \chi, x\rangle \models \phi \iff \langle \mathrm{expl}(\mathfrak{F}), s\rangle \otimes \langle \mathrm{expl}(\chi), x\rangle \models \phi$$

Démonstration. Prenons une fonction d'étiquetage λ qui associe à chaque monde de $\mathrm{expl}(\mathfrak{F})\otimes\mathrm{expl}(\chi)$ (i.e. $(s,x) \in 2^V \times 2^{V^+}$) un monde de $\mathfrak{F}\otimes\chi$ (i.e. un sous-ensemble de $V \cup V^+ \cup V_-^\circ$) :

$$\lambda(s,x) \coloneqq (s\backslash V_-) \cup (s \cap V_-)^\circ \cup x \cup \{p \in V_- \mid s \cup x \models \theta_-(p)\}$$

qui est la définition de s^x (déf. 2.51). Montrons que λ satisfait les trois conditions du lemme 2.32. $*$ Pour C1, prenons deux mondes (s,x) et (t,y). Nous devons montrer que $\lambda(s,x) \cup \lambda(t,y)' \models \Omega_a^\otimes$ si et seulement si $(s,x)R_a^\otimes(t,y)$. Pour cela, montrons l'équivalence suivante : $\lambda(s,x) \cup \lambda(t,y)' \models \Omega_a^\otimes$ ssi

$$\begin{gathered}(s\backslash V_-) \cup (s \cap V_-)^\circ \cup x \cup \{p \in V_- \mid s \cup x \models \theta_-(p)\} \cup \\ ((t\backslash V_-) \cup (t \cap V_-)^\circ \cup y \cup \{p \in V_- \mid t \cup y \models \theta_-(p)\})' \\ \models [V_- \mapsto V_-^\circ][V_-' \mapsto (V_-^\circ)']\Omega_a \wedge \Omega_a^+\end{gathered}$$

V_- et V_-' n'apparaissent pas dans la formule ci-dessus, comme les anciennes relations épistémiques ne dépendent pas des nouvelles valeurs des atomes propositionnels modifiés. Ainsi, on peut simplifier la valuation pour avoir une condition équivalente :

$$(s\backslash V_-) \cup (s \cap V_-)^\circ \cup x \cup (t\backslash V_-)' \cup (t^\circ \cap V_-^\circ)' \cup y' \models [V_- \mapsto V_-^\circ][V_-' \mapsto (V_-^\circ)']\Omega_a \wedge \Omega_a^+$$

dans laquelle on peut séparer les vocabulaires utilisés en deux conditions, portant respectivement sur V et V^+ :

$$\begin{aligned}(s\backslash V_-) \cup (s \cap V_-)^\circ \cup (t\backslash V_-)' \cup (t^\circ \cap V_-^\circ)') \models [V_- \mapsto V_-^\circ][V_-' \mapsto (V_-^\circ)']\Omega_a \\ \text{et } x \cup y' \models \Omega_a^+ \quad (2.1)\end{aligned}$$

Enlevons dans 2.1 la substitution par $\circ$ de chaque côté pour voir que cette formule est équivalente à $s \cup t' \models \Omega_a$. Par conséquent, $s \cup t'$ est sRt ce qui revient à $(s,x)R_a^\otimes(t,y)$ par les définitions de $\mathrm{expl}(\mathcal{M})$ (déf. 2.20) et $\mathrm{expl}(\mathcal{E})$ (déf. 6.10).

$*$ Pour C2, prenons un (s,x) quelconque et $p \in V$. Nous devons montrer que $p \in \lambda(s,x)$ ssi $p \in \mathrm{val}^\otimes(s,x) = \{p \in V \mid \langle \mathcal{M}, s\rangle \models \mathrm{post}(p)(x)\}$. Il y a deux cas. Premièrement, si $p \notin V_-$, alors $\mathrm{post}(p)(x) = p$ par la définition 6.10 et nous avons directement $p \in \lambda(s,x)$ ssi $p \in s$ ssi $\langle \mathcal{M}, s\rangle \models p$ ssi $p \in \mathrm{val}^\otimes(s,x)$. Deuxièmement,

si $p \in V_-$, alors $p \in \lambda(s,x)$ ssi $s \cup x \models \theta_-(p)$ par la définition de λ et par l'obtention de $\text{post}(p)(x) = [x \sqsubseteq V_-]\theta_-(p)$ via la définition 6.10. Par conséquent, nous avons une équivalence : $p \in \lambda(s,x)$ ssi $s \cup x \models \theta_-(p)$ ssi $s \models [x \sqsubseteq V_-]\theta_-(p)$ ssi $\langle \mathcal{M}, s\rangle \models [x \sqsubseteq V_-]\theta_-(p)$ ssi $p \in \text{val}^{\otimes}(s,x)$.

* Pour C3, prenons un $s^{\otimes} \in 2^{V \cup V^+ \cup V_-^\circ}$ quelconque. Nous voulons montrer que $s^{\otimes} \models \theta^{\otimes}$ ssi il y a un monde (s,x) tel que $\lambda(s,x) = s^{\otimes}$. De gauche à droite, nous supposons $s^{\otimes} \models \theta^{\otimes}$, i.e. :

$$s^{\otimes} \models [V_- \mapsto V_-^\circ](\theta \wedge ||\theta^+||_{\mathfrak{F}}) \wedge \bigwedge_{p \in V_-} (p \leftrightarrow [V_- \mapsto V_-^\circ](\theta_-(p))) \tag{2.2}$$

Soit $s \coloneqq (s^{\otimes} \cup (V \backslash V_-)) \cup \{p \in V_- \mid p^\circ \in s^{\otimes}\}$. De 2.2 ci-dessus, on a $s \models \theta$, ce qui signifie que s est un état de $\mathfrak{F}$ et aussi un monde de $\text{expl}(\mathfrak{F})$. Ensuite, soit $x \coloneqq s^{\otimes} \cap V^+$. Maintenant par la définition de λ, on a : $\lambda(s,x) = s^x = s^{\otimes}$.

De droite à gauche, supposons que nous avons un monde (s,x) tel que $\lambda(s,x) = s^x = s^{\otimes}$. Maintenant nous devons montrer 2.2 pour $s^{\otimes} = (s \backslash V_-) \cup (s \cap V_-)^\circ \cup x \cup \{p \in V_- \mid s \cup x \models \theta_-(p)\}$.

Premièrement, notons que s est un monde de $\text{expl}(\mathfrak{F})$ et aussi un état de $\mathfrak{F}$, i.e. on a $s \models \theta$. Deuxièmement, (s,x) est un monde de $\text{expl}(\mathfrak{F}) \times \text{expl}(\chi)$, c'est pourquoi nous avons $s \cup x \models ||\theta^+||_{\mathfrak{F}}$. Troisièmement, nous avons par définition de λ que pour tout $q \in V_-$:

- $q^\circ \in s^{\otimes}$ ssi $q \in s$;
- $q \in s^{\otimes}$ ssi $s \models \theta_-(q)$.

Ces trois cas impliquent 2.2.

□

Théorème 2.64. *La fonction* symb *(de la définition 2.54) préserve la vérité : pour toute structure de Kripke pointée $\langle \mathcal{M}, w\rangle$, tout modèle d'événements pointé $\langle \mathcal{E}, e\rangle$ et toute formule ϕ sur le vocabulaire de $\mathcal{M}$, on a :*

$$\langle \mathcal{M} \otimes \mathcal{E}, (w,e)\rangle \models \phi \iff \langle \text{symb}_{\lambda_W^U}(\mathcal{M}), \lambda_W^U(w)\rangle \otimes \langle \text{symb}_{\lambda_E^{V^+}}(\mathcal{E}), \lambda_E^{V^+}(e)\rangle \models \phi$$

où λ_W^U est une fonction d'étiquetage avec $V \subseteq U$ qui étend val *et $\lambda_E^{V^+}$ est une fonction d'étiquetage pour E.*

Démonstration. De la même manière que la preuve précédente, nous allons prouver les conditions C1, C2 et C3 du lemme 2.32.

λ doit ici associer les mondes du modèle $\mathcal{M} \otimes \mathcal{E}$ aux états de la structure $\text{symb}(\mathcal{M}) \otimes \text{symb}(\mathcal{E})$. De nouveau, s^x est utilisé mais s et x sont donnés par λ_V^W et $\lambda_{V^+}^{\mathcal{E}}$. Pour plus de lisibilité, nous noterons $\widehat{w}$ pour $\lambda_W^V(w)$ et $\widehat{e}$ pour $\lambda_E^{V^+}(e)$. Soit λ définie ainsi :

$$\lambda(w,e) \coloneqq (\widehat{w}\backslash V_-) \cup (\widehat{w} \cap V_-)^\circ \cup \widehat{w} \cup \{p \in V_- \mid \widehat{w} \cup \widehat{e} \models \theta_-(p)\}$$

La preuve suit la preuve précédente et $\widehat{w}$ et $\widehat{e}$ prennent les rôles de s et x.

$*$ Pour C1, prenons deux mondes quelconques (w_1, e_1) et (w_2, e_2) dans le modèle mis à jour. Nous devons montrer que $\lambda(w_1, e_1) \cup \lambda(w_2, e_2)' \models \Omega_a^\otimes$ ssi $(s, x)R_a(t, y)$. Pour cela, écrivons l'équivalence suivante. Nous avons $\lambda(w_1, e_1) \cup \lambda(w_2, e_2)' \models \Omega_a^\otimes$ ssi

$$\begin{gathered}(\widehat{w_1}\backslash V_-) \cup (\widehat{w_1} \cap V_-)^\circ \cup \widehat{e_1} \cup \{p \in V_- \mid \widehat{w_1} \cup \widehat{e_1} \models \theta_-(p)\} \cup \\ ((\widehat{w_2}\backslash V_-) \cup (\widehat{w_2} \cap V_-)^\circ \cup \widehat{e_2} \cup \{p \in V_- \mid \widehat{w_2} \cup \widehat{e_2} \models \theta_-(p)\})' \\ \models [V_- \mapsto V_-^\circ][V_-' \mapsto (V_-^\circ)']\Omega_a \wedge \Omega_a^+\end{gathered}$$

De nouveau, V_- et V_-' n'apparaissent pas dans la formule, alors ils peuvent être enlevés de la valuation pour obtenir :

$$\begin{gathered}(\widehat{w_1}\backslash V_-) \cup (\widehat{w_1} \cap V_-)^\circ \cup \widehat{e_1} \cup \\ (\widehat{w_2}\backslash V_-)' \cup (\widehat{w_2}^\circ \cap V_-^\circ)' \cup \widehat{e_2}') \\ \models [V_- \mapsto V_-^\circ][V_-' \mapsto (V_-^\circ)']\Omega_a \wedge \Omega_a^+\end{gathered}$$

ce qui se sépare en :

$$\begin{gathered}(\widehat{w_1}\backslash V_-) \cup (\widehat{w_1} \cap V_-)^\circ \\ \cup(\widehat{e_1}\backslash V_-)' \cup (\widehat{e_1}^\circ \cap V_-^\circ)') \\ \models [V_- \mapsto V_-^\circ][V_-' \mapsto (V_-^\circ)']\Omega_a \\ \text{et } \widehat{w_2} \cup \widehat{e_2}' \models \Omega_a^+\end{gathered}$$

Maintenant il est possible d'enlever la substitution par $\circ$ pour montrer l'équivalence à $\widehat{w_1} \cup \widehat{w_2}' \models \Omega_a$. Dans la preuve du théorème 2.34 il a été montré que les conditions du lemme 2.32 s'applique à λ_W^V dans la construction de symb($\mathcal{M}$). En particulier, par la condition C1, nous avons $\widehat{w_1} \cup \widehat{w_2}' \models \Omega_a$ ssi $w_1 R_a w_2$. De plus, par la définition 2.54, nous avons $\widehat{e_1} \cup \widehat{e_2}' \models \Omega_a^+$ ssi $e_1 R^{\mathcal{E}} e_2$. Par conséquent, la condition est équivalente à $w_1 R_a w_2$ et $e_1 R_a^{\mathcal{E}} e_2$. Par définition 1.28 du *product update*, c'est $(w_1, e_1)R_a^\otimes(w_2, e_2)$.

$*$ Pour montrer C2, prenons un monde quelconque (w, e) de $\mathcal{M} \otimes \mathcal{E}$ et un $p \in V$. Nous devons montrer que $p \in \lambda(w, e)$ ssi $p \in \mathrm{val}^\otimes(w)(e) = \{p \in V \mid \langle \mathcal{M}, s\rangle \models \mathrm{post}(p)(e)\}$. Considérons l'ensemble $V_- \coloneqq \{p \in V \mid \exists e \in E\colon \mathrm{post}(p)(e) \neq p\}$ de la définition 2.54 et distinguons deux cas.

Premièrement, si $p \notin V_-$, alors post$(p)(e)$ et nous avons directement $p \in \lambda(w, e)$ ssi $p \in \widehat{w}$ ssi $\langle \mathcal{M}, w\rangle \models p$ ssi $p \in \mathrm{val}^\otimes(w)(e)$.

Deuxièmement, si $p \in V_-$, alors $p \in \lambda(w)(e)$ ssi $\widehat{w} \cup \widehat{e}' \models \theta_-(p)$ par la définition de λ. Notons que $\theta_-(p)$ dans la définition 2.54 est la disjonction sur toutes les valuations issues des événements de E qui sont mutuellement distinctes par la fonction d'étiquetage. Par conséquent, on a les équivalences suivantes : $p \in \lambda(w)(e)$

ssi $\widehat{w} \cup \widehat{e} \models \theta_{-}(p)$ ssi $\widehat{w} \cup \widehat{e} \models \bigvee_{e \in E}(\widehat{e} \sqsubseteq V^{+} \wedge \mathrm{post}(p)(e)$ ssi $\widehat{w} \models \mathrm{post}(p)(e)$ ssi $p \in \mathrm{val}^{\otimes}((w, e))$.

$*$ Montrons C3. Prenons un état quelconque $s^{\otimes} \in 2^{V \cup V^{+} \cup V_{-}^{\circ}}$. Nous voulons montrer que $s^{\otimes} \models \theta^{\otimes}$ ssi il y a un monde (w, e) tel que $\lambda(w, e) = s^{\otimes}$. Prenons aussi :

$s \coloneqq (s^{\otimes} \cap (V \backslash V_{-})) \cup \{p \in V \mid p^{\circ} \in s^{\otimes}\}$ et $x \coloneqq s^{\otimes} \cap V^{+}$.

De gauche à droite, supposons $s^{\otimes} \models \theta^{\otimes}$, i.e.

$$s^{\otimes} \models [V_{-} \mapsto V_{-}^{\circ}](\theta \wedge ||\theta^{+}||_{\mathfrak{F}}) \wedge \bigwedge_{p \in V_{-}} (p \leftrightarrow [V_{-} \mapsto V_{-}^{\circ}](\theta_{-}(p))) \tag{2.3}$$

De l'équation 2.3, on a $s \models \theta$, ce qui signifie que s est un état de $\mathrm{symb}(\mathcal{M})$. Par la condition C3, il y a un monde w de $\mathcal{M}$ tel que $\lambda_{V}^{W}(w) = s$. Nous avons aussi $s \cup x \models ||\theta^{+}||_{\mathfrak{F}}$. Par la définition 2.54, on a $\theta^{+} \coloneqq \bigvee_{e \in E}(\mathrm{pre}(e) \wedge \lambda(e) \sqsubseteq V^{+})$. Par conséquent, il y a un événement atomique $e \in E$ et une formule ϕ tels que $\lambda_{V^{+}}^{E} = x$ et $s \cup x \models ||\,\mathrm{pre}(e)||_{\mathfrak{F}}$, ce qui implique que $\langle \mathcal{M}, w \rangle \models \mathrm{pre}(e)$. Ainsi, (w, e) est un monde de $\mathcal{M} \otimes \mathcal{E}$. De plus, par 2.3 on a pour tout $q \in V_{-}$ que $q \in s^{\otimes}$ ssi $s^{\otimes} \models \mathrm{post}(q)(e)$. Par la définition de λ, on a $\lambda(w, e) = s^{\otimes}$.

De droite à gauche, supposons que nous avons un monde (w, e) dans $\mathcal{M} \times \mathcal{E}$ tel que $\lambda(w, e) = s^{\otimes}$. Nous avons maintenant à montrer $s^{\otimes} \models \theta^{\otimes}$, i.e. 2.3 ci-dessus. Par définition de s et x, nous avons $\widehat{w} = s$ et $\lambda_{V^{+}}^{E} = x$. w est un monde de $\mathcal{M}$ ainsi $s = \widehat{w}$ est un état de $\mathfrak{F}$, i.e. on a $s \models \theta$. (w, e) est un monde de $\mathcal{M} \otimes \mathcal{E}$, par conséquent $\langle \mathcal{M}, w \rangle \models \mathrm{pre}(e)$. Avec $\widehat{w} \cup \lambda_{V^{+}}^{E}(e) = s \cup x$, on a $s \cup x \models ||\theta^{+}||_{\mathfrak{F}}$ et $s \models \phi$ avec $\phi \in \Phi$. De ce fait, $s \cup x \models ||\theta^{+}||_{\mathfrak{F}}$ par la définition de 2.54. Ensuite, nous avons par la définition de λ que pour chaque $q \in V_{-}$, $q^{\circ} \in s^{\otimes}$ ssi $q \in s$ et $q \in s^{\otimes}$ ssi $s \models \theta_{-}(q)$. Tout ceci implique 2.3.

□

Nous avons maintenant terminé de décrire le fonctionnement du *model checking* symbolique pour la logique épistémique dynamique. Cette représentation symbolique est intéressante car elle permet de représenter de manière compacte les relations grâces à des formules propositionnelles. Néanmoins, ce qui est attrayant dans cette approche, c'est aussi la mise en pratique. En effet, il faut pouvoir représenter ces structures, et donc ces formules, informatiquement.

Pour cela, il existe une structure de données largement utilisée qui est le *Binary Decision Diagram* (BDD). L'avantage est que toutes les opérations utilisées lors des manipulations des formules qui représentent les structures de croyances et modificateurs de croyances existent et sont efficaces sur les BDDs. Nous allons détailler cela dans le chapitre suivant.

3 | Structures de données : BDDs et ADDs

Comme nous l'avons vu, Gattinger propose une représentation symbolique des structures de Kripke via des formules propositionnelles ($\mathcal{L}_{PROP}$, définition 1.1), qui ne présuppose rien sur la manière d'implémenter ces formules. Cette représentation théorique nécessite alors d'être mise en pratique à proprement parler. Pour cette mise en pratique, Gattinger propose l'utilisation de structures de données adaptées, les BDDs ; c'est ce qu'il fait dans son programme SMCDEL [Gat18c].

Comme nous allons le rappeler, les formules propositionnelles peuvent être vues comme des fonctions booléennes. Nous allons montrer comment ces fonctions peuvent être représentées par la structure de données qu'est le *Binary Decision Diagram (BDD)* (sec. 3.1). Celui-ci permet ainsi la manipulation de ces fonctions booléennes en pratique. Néanmoins, cette structure n'est pas faite pour représenter des fonctions à valeur non booléene et ainsi ne permet pas de représenter des concepts plus poussés, comme des distributions de probabilités.

Nous allons donc par la suite, présenter les fonctions pseudo-booléennes qui sont une génération des fonctions booléennes et qui permettent de manipuler des valeurs dans un domaine plus vaste qui est $\mathbb{Q}$. Il se trouve qu'il existe une structure de données qui est une généralisation du BDD : l'ADD, permettant de représenter et de manipuler des fonctions pseudo-booléennes. Nous présenterons cela dans la section qui suivra (sec. 3.2).

3.1 Diagrammes de décision binaires

Qu'entendons-nous par *fonction booléenne* et quel est le lien avec une formule propositionnelle ?

Définition 3.1 (Fonction booléenne)**.** *Soit $X = \{x_1, \ldots, x_n\} \subseteq PS$ un ensemble fini de variables propositionnelles ; nous appelons* fonction booléenne (BF) sur X *une fonction totale du type $f \colon 2^X \to \mathbb{B}$.*

Une fonction booléenne, tout comme une formule propositionnelle, peut être représentée par une table de vérité qui donne la valeur de vérité pour chaque affectation possible des variables.

Ainsi, nous pouvons voir une formule propositionnelle $\phi \in \mathcal{L}_{\text{prop}}(V)$ comme une fonction booléenne dont les paramètres sont toutes les valuations possibles de son vocabulaire V, de sorte que :

$$\forall v \in 2^V,\ \phi(v) = \begin{cases} 1, & \text{si } v \models \phi \text{ (déf. 1.2)} \\ 0, & \text{sinon} \end{cases}$$

Exemple 3.2. *Soit la formule propositionnelle ϕ définie sur 4 variables $\{x_1, x_2, x_3, x_4\}$ de sorte que $\phi = (x_1 \wedge x_2) \vee (x_3 \wedge x_4)$. Nous pouvons écrire f comme la fonction booléenne équivalente à ϕ telle que $f(x_1, x_2, x_3, x_4) = (x_1 \wedge x_2) \vee (x_3 \wedge x_4)$. Nous pouvons aussi écrire sa table de vérité, présentée dans la table 3.1. Nous y lisons les valeurs de la fonction f pour chaque affectation possible de ses variables ; par exemple $f(\{x_3, x_4\}) = 1$ pour l'affectation $\{x_1 : 0,\ x_2 : 0,\ x_3 : 1,\ x_4 : 1\}$.*

Il est possible de représenter des fonctions booléennes sous forme d'un graphe, et plus précisément d'un diagramme de décision (DD).

Grâce à de nombreuses implémentations proposées pour les BDDs, une littérature abondante et des algorithmes permettant de manipuler ces structures en temps polynomial, les BDDs sont des structures qui se sont imposées pour l'implémentation de fonctions booléennes.

Dans la littérature, les BDDs sont utilisés pour représenter des ensembles ou des relations de manière compacte. Ces BDDs sont en fait des circuits logiques qui servent dans la vérification formelle, comme par exemple dans la vérification de modèle symbolique de CTL [1] [SSL07b]. Dans la continuité de ce type de travaux, il est naturel que Gattinger ait utilisé les BDDs dans sa vérification de modèle symbolique via ses structures de croyances vues précédemment.

Définissons maintenant formellement la structure BDD [Bry86, BRB90].

Définition 3.3 (BDD (*Binary Decision Diagram*))**.** *Un BDD est un graphe orienté acyclique (DAG) possédant un nœud racine. Les nœuds terminaux sont soit* Vrai *(1 ou $\top$) soit* Faux *(0 ou $\bot$). Chaque nœud non-terminal est associé à une variable propositionnelle ($p \in PS$) et à deux sous-BDD appelés les* fils gauche *et* droit.

Exemple 3.4. *Par exemple, il est possible de représenter la fonction booléenne de l'exemple 3.2, par le BDD de la figure 3.1.*

La lecture graphique d'un BDD se fait ainsi :

1. Computation Tree Logic

x_1	x_2	x_3	x_4	f	x_1	x_2	x_3	x_4	f
0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	1	0
0	0	1	0	0	1	0	1	0	0
0	0	1	1	1	1	0	1	1	1
0	1	0	0	0	1	1	0	0	1
0	1	0	1	0	1	1	0	1	1
0	1	1	0	0	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1

TABLE 3.1 – Table de vérité de f (ex. 3.2).

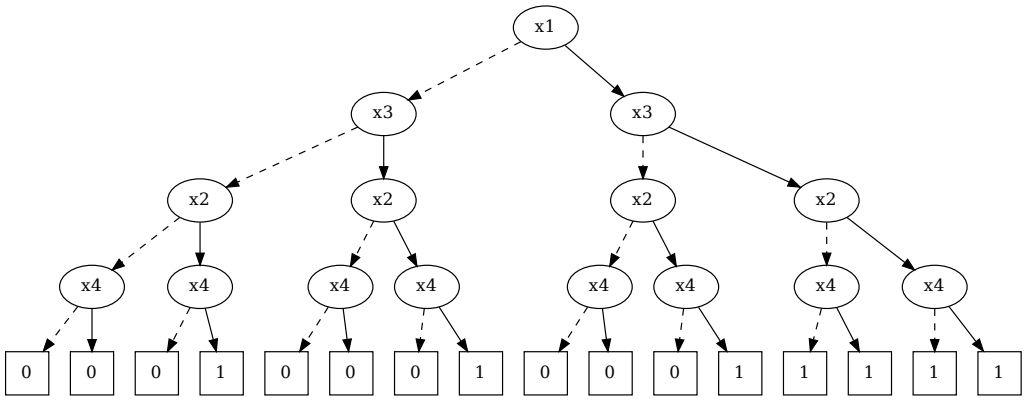


FIGURE 3.1 – BDD représentant la fonction f de l'exemple 3.2.

- les nœuds entourés sont les nœuds non terminaux, qui représentent des variables propositionnelles ;
- les nœuds encadrés sont les feuilles à valeur dans $\mathbb{B}$;
- si l'arc sortant d'un nœud est en pointillé, alors la variable de ce nœud est de valeur *faux*, sinon (arc plein), alors la valeur de la variable est *vrai*.

Néanmoins, comme leurs noms l'indiquent, il est possible de réduire les BDDs pour obtenir des *Reduced Binary Decision Diagram (RBDD)*, qui seront canoniques, en appliquant deux règles :

1. supprimer un noeud non terminal si les deux arcs sortants pointent vers le même sous-graphe ;

2. si deux sous-graphes sont isomorphes (identiques), alors supprimer l'un des deux et rediriger ses arcs entrants vers le sous-graphe restant.

Cette technique de simplification des noeuds permet d'obtenir le BDD réduit de la figure 3.2a, beaucoup plus compact, car les noeuds ne sont pas répétés de manière inutile. Ainsi, dans l'exemple, 0 et 1 n'apparaissent qu'une seule fois (ce qui est obligatoire dans un RBDD), et chaque variable n'apparaît quant à elles qu'une ou deux fois.

Une autre contrainte peut être appliquée aux diagrammes : l'ordre d'apparition des variables. C'est-à-dire qu'à chaque chemin de la racine du diagramme aux feuilles, les variables propositionnelles doivent être rencontrées systématiquement dans un ordre unique. C'est ce qu'on appelle un *Reduced Ordered Binary Decision Diagram (ROBDD).*

Exemple 3.5. *Enfin, en fonction de cet ordre d'apparition des variables dans le BDD, celui-ci n'est pas le même. En effet, alors que le diagramme 3.2a était un RBDD avec pour ordre* $\langle x_1, x_3, x_2, x_4 \rangle$*, nous avons effectué un réordonnancement avec l'ordre* $\langle x_1, x_2, x_3, x_4 \rangle$ *pour obtenir le diagramme 3.2b. Nous noterons que cette contrainte permet de parler « d'étages » ou de « niveaux » dans le diagramme. En effet, à chaque variable est attribué son niveau, qui est son numéro d'apparition dans l'ordre.*

Le ROBDD ainsi obtenu est le BDD minimal pour représenter la formule f pour l'ordre donné ; il n'est pas possible de faire plus conçis : chaque variable n'apparaît qu'une seule fois et unique fois dans le diagramme et chaque nœud est nécessaire.

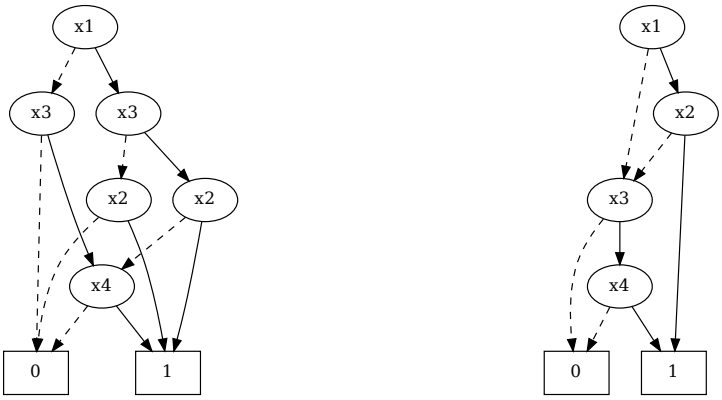
Théorème 3.6. *Étant donné un ordre total sur les variables propositionnelles, il y a exactement un seul ROBDD respectant cet ordre pour chaque formule propositionnelle.*

C'est ce qui permet d'obtenir un ROBDD canonique pour un ordre donné pour chaque formule [Bry86].

Dorénavant, nous allons supposer tout BDD comme étant un ROBDD pour qu'il soit le plus petit possible. Notons que cette procédure de réduction d'un BDD est polynomiale, il est donc peu coûteux de toujours réduire le BDD. Par ailleurs, cette utilisation d'ROBDD à la place de BDD permet de réduire les temps de calculs des algorithmes qui manipulent les BDDs. En effet, moins il y a de nœuds à traiter, plus les algorithmes sont efficaces.

Nous allons maintenant présenter les structures qui permettent d'implémenter ces objets et les algorithmes dont nous avons besoin pour les manipuler.

Voici l'objet clef nécessaire à l'implémentation d'une structure de BDD : un *Nœud* (algo. 3.1), qui est présenté comme tel dans « *Graph-Based Algorithms for Boolean Function Manipulation* » [Bry86]. Une abstraction est faite des variables,



(a) RBDD (b) ROBDD

FIGURE 3.2 – $f(x_1, \ldots, x_4) = (x_1 \wedge x_2) \vee (x_3 \wedge x_4)$ (ex. 3.5).

elles sont simplement représentées par un entier qui est leur position dans l'ordre des variables.

Algorithme 3.1: Structure d'un Nœud

```
1 Objet Nœud contient
2 |   entier indice;
3 |   Nœud* gauche, droite;
4 |   booléen valeur;
5 fin
```

Cette structure de Nœud contient 4 informations :

- son `indice` qui est simplement la position de sa variable dans l'ordre des variables. Si le nœud est une feuille, alors `indice` $= \infty$, sinon `indice` $\in \mathbb{N}$;
- deux pointeurs `gauche` et `droite` qui sont respectivement les deux fils du nœud en question, représentant « si la variable est considérée comme *faux* » et « comme *vrai* ». Si le nœud est une feuille, alors ces deux pointeurs sont Null ;
- la valeur du nœud qui n'a un interêt qu'aux feuilles de diagramme. Si le nœud est une feuille, alors `valeur` $\in \mathbb{B}$, sinon `valeur` = Null. Nous illustrons cela dans l'algorithme 3.2.

Un outil important lors de la représentation des nœuds en mémoire et la *table des nœuds uniques*. C'est l'algorithme de construction des nœuds (algo.

3.2) qui exploite cette particularité grâce à deux variables partagées aux Nœuds : `table_terminaux` et `table_nœuds`. Ce sont deux tables de hachages qui prennent respectivement comme clés les valeurs des nœuds et le tuple ⟨`indice`, `gauche`, `droite`⟩. Les valeurs liées à ces clés sont des Nœuds. Cela permet de d'éviter de reconstruire des Nœuds constammant et de créer des nœuds canoniques.

Algorithme 3.2: Constructeurs des Nœuds feuilles ou internes

Entrée: `valeur` $\in \mathbb{B}$, `indice` $\in \mathbb{N}$, `gauche` et `droite` sont des objets Nœud
Sortie : Un Nœud

```
/* Utilise les tables de nœuds uniques table_terminaux et table_nœuds
   sont sont des variables communes aux Nœuds afin de retourner des
   pointeurs sur des nœuds déjà existants à la place de créer des
   nouvelles instances.                                              */
```

```
1  Fonction construct_feuille(valeur):
2      si valeur est dans table_terminaux alors
3          retourner table_terminaux[valeur] ;
4      sinon
5          table_terminaux[valeur] ← Nœud(∞, Null, Null, valeur);
6      fin
7      retourner table_terminaux[valeur] ;

8  Fonction construct_interne(indice, gauche, droite):
9      k ← ⟨indice, gauche, droite⟩ ;
10     si k est dans table_nœuds alors
11         retourner table_nœuds[k] ;
12     sinon
13         table_nœuds[k] ← Nœud(indice, gauche, droite, Null);
14     fin
15     retourner table_nœuds[k] ;
```

Nous allons maintenant présenter quelques algorithmes qui permettent de manipuler les BDDs.

Bien que cela ne soit pas explicité dans les algorithmes rédigés (algo. 3.3, 3.4, 3.3, 3.5, 3.6), des *caches* sont toujours utilisés.

Lorsque que l'algorithme récursif s'applique, il parcourt les nœuds du diagramme. Plusieurs chemins peuvent mener au même nœud dans le diagramme. Il est alors important de stocker les résultats des calculs pour ne pas avoir à réeffectuer des calculs déjà fait.

Ainsi, à chaque fois qu'un Nœud est crée, il est enregistré dans le cache. Par la suite, à chaque fois qu'un calcul est fait, on vérifie au préalable s'il n'aurait pas été déjà fait. C'est ce qui permet de garantir le caractère polynomial des algorithmes.

Par ailleurs, à la construction de chaque Nœud, il est vérifié que les `indices` des Nœuds fils sont bien supérieurs au Nœud courant. Cela est nécessaire afin de conserver l'ordre du ROBDD.

De plus, notons que ces algorithmes ici présentés ne modifient pas les BDD en place. En effet, ils retournent le résultat de l'application de l'algorithme, tout en ne modifiant pas les entrées.

La complexité des algorithmes que nous présentons sont décrites et expliquées dans « *Graph-Based Algorithms for Boolean Function Manipulation* » [Bry86] et « *A Knowledge Compilation Map* » [DM02]. Nous réutilisons les notations des algorithmes en gras (ex : $\neg\mathbf{C}$, $\vee\mathbf{BC}$...) à titre de repère vis-à-vis de ces articles pour toute éventuelle lecture et comparaison.

Voici la liste des algorithmes présentés :

Opération unaire l'algorithme 3.3, page 72 permet une opération unaire sur un BDD, typiquement, l'opération de négation (cette opération est notée $\neg\mathbf{C}$ dans [DM02]). L'algorithme est récursif et applique une modification du diagramme seulement aux feuilles ;

Opération binaire l'algorithme 3.4, page 73 permet l'agrégation de deux BDDs par un opérateur binaire, typiquement la disjonction ou la conjonction. Ces opérations sont respectivement notées $\vee\mathbf{BC}$ et $\wedge\mathbf{BC}$;

Conditionnement l'algorithme de conditionnement 3.5, page 74 (*restriction* en anglais, **CD**) permet de restreindre la valeur de certaines variables dans le BDD à vrai ou à faux ;

Quantificateur L'algorithme de quantification booléen 3.6, page 75 (**SFO**) permet d'*oublier* un variable dans un BDD en choisissant un opérateur. En fonction de l'opérateur `op` utilisé, son comportement est différent. L'algorithme que nous présentons est en réalité **FO**, qui consiste à oublier une liste de variables plutôt qu'une seule variable.

Pour un nœud `n` et une liste d'indices `indices`, en utilisant quantification(`n`, `indices`, `op`) avec l'opérateur $op(x, y) = x \wedge y$, il s'agira d'une *quantification universelle* (voir déf. 2.24), alors qu'avec l'opérateur $op(x, y) = x \vee y$ il s'agira d'une *quantification existentielle*. Notons que cet algorithme est aussi appelé « oubli de variables » et en anglais *« forgetting »*.

Hormis **FO** qui est conceptuellement l'application multiple de **SFO**, toutes ces transformations des BDDs sont polynomiales en la taille du diagramme de décision (i.e. le nombre de nœuds dans la graphe) grâce à l'utilisation du cache.

La complexité pour des algorithmes de *questionnement* des structures, tels que la consistance (souvent abrégée et notée **CO**), la validité (**VA**), l'implication (**IM**), le comptage (**CT**) et l'énumération (**ME**) des modèles, est quant à elle polynomiale.

Notons qu'en terme d'implémentation, Gattinger utilise ces algorithmiques via le package Haskell, binding de C [Gat18a], pour son *framework* SMCDEL [Gat18b], qui utilise de la vérification de modèle avec du conditionnement (notée $\sqsubseteq$), des quantificateurs (notée $\forall V'$), de la composition (notée $[p \mapsto \phi]$), ou du renommage (notée $[V' \mapsto V]$). Ce *framework* exploite donc les algorithmes polynomiaux sur les BDDs.

Algorithme 3.3: Opération unaire sur un Nœud

Entrée: `n` un Nœud et `op` un opérateur unaire tel que `op` : $\mathbb{B} \to \mathbb{B}$
Sortie : Un Nœud

```
/* Utilise un cache afin d'effectuer les opérations une seule fois, si
   elles se présentent plusieurs fois.                                  */
1 Fonction application_unaire(n, op):
2     si n est une feuille alors
3         retourner construct_feuille(op(n.valeur)) ;
4     sinon
          /* On exploite la récursivité.                                */
5         l ← application_unaire(n.gauche, op);
6         d ← application_unaire(n.droite, op);
7         retourner construct_interne(n.indice, l, d) ;
8     fin
```

Algorithme 3.4: Opération binaire sur deux Nœuds

Entrée: `n1` et `n2` deux Nœuds et `op` un opérateur binaire tel que `op` : $\mathbb{B}, \mathbb{B} \to \mathbb{B}$

Sortie : Un Nœud

```
/* Utilise un cache afin d'effectuer les opérations une seule fois, si
   elles se présentent plusieurs fois.                                  */
```

```
1  Fonction application_binaire(n1, n2, op):
2      si n1 et n2 sont des feuilles alors
3          retourner construct_feuille(op(n1.valeur, n2.valeur)) ;
       /* On détermine lequel des deux nœuds est le plus haut dans l'arbre
          pour pouvoir "descendre" par le côté le plus haut afin de
          respecter l'ordre des variables.                               */
4      indice_min ← min(n1.indice, n2.indice) ;
5      si n1.indice = indice_min alors
6          gauche1 ← n1.gauche ;
7          droite1 ← n1.droite ;
8      sinon
9          gauche1 ← droite1 ← n1 ;
10     fin
11     si n2.indice = indice_min alors
12         gauche2 ← n2.gauche ;
13         droite2 ← n2.droite ;
14     sinon
15         gauche2 ← droite2 ← n2 ;
16     fin
17     g ← application_binaire(gauche1, gauche2, op);
18     d ← application_binaire(droite1, droite2, op);
19     retourner construct_interne(indice_min, g, d) ;
```

Algorithme 3.5: Conditionnement

Entrée: Un nœud `n` et une liste `L` de couples (i, v_i) d'indices $(i \in \mathbb{N})$/valeur de vérité $(v_i \in \mathbb{B})$, i étant dans l'ordre croissant.

Sortie : Un Nœud

```
/* Utilise un cache afin d'effectuer les opérations une seule fois, si
   elles se présentent plusieurs fois.                                  */
```

```
1  Fonction conditionnement(n, m):
2      pour (i, valeur) dans L faire
3          Lc ← liste des couples (j, valeur_j) se trouvant après (i, valeur)
            dans L;
4          si n.indice = i alors
               /* L'indice est trouvé, on peut passer au suivant dans
                  l'affectation. On selectionne un fils en fonction de la
                  valeur de l'affectation.                              */
5              si valeur est vraie alors
6                  retourner conditionnement(n.droite, Lc) ;
7              sinon
8                  retourner conditionnement(n.gauche, Lc) ;
9              fin
10         sinon si n.indice < i alors
               /* On exploite la récursivité en descendant dans le diagramme.
                  */
11             g ← conditionnement(n.gauche, Lc) ;
12             d ← conditionnement(n.droite, Lc) ;
13             retourner construct_interne(n.indice, g, d);
14     fin
```

Algorithme 3.6: Quantificateur booléen ou *forgetting*

Entrée: `n`, un Nœud dont on veut supprimer les indices présents dans la liste d'entiers `indices` (triée croissante) et une opération d'agrégation binaire `op` : $\mathbb{B}, \mathbb{B} \to \mathbb{B}$

Sortie : Un Nœud

```
/* Utilise un cache afin d'effectuer les opérations une seule fois, si
   elles se présentent plusieurs fois.                                  */
```

1 **Fonction** `quantification(`*`n, indices, op`*`):`
2 | **si** *`n`* *est une feuille ou* $|$*`indices`*$| = 0$ *ou* *`n`*.*indice* $>$ max(*`indices`*) **alors**
| | `/* On a déjà fait ce qu'il y avait à faire.                      */`
3 | | **retourner** `n` ;
4 | **sinon si** *`n`*.*indice est dans* *`indices`* **alors**
| | `/* Un nœud à oublier a été trouvé. On applique l'opérateur d'agrégation aux successeurs. */`
5 | | $n\_indices \leftarrow \langle i \mid i \in$ `indices` tel que $i >$ `n`.*indice*$\rangle$;
6 | | $g \leftarrow$ quantification(`n`.*gauche*, $n\_indices$, `op`) ;
7 | | $d \leftarrow$ quantification(`n`.*droite*, $n\_indices$, `op`) ;
8 | | **retourner** application_binaire(g, d, `op`)
9 | **sinon**
| | `/* L'indice ne correspond pas. On exploite la récursivité en descendant dans le diagramme. */`
10 | | $g \leftarrow$ quantification(`n`.*gauche*, `indices`, `op`) ;
11 | | $d \leftarrow$ quantification(`n`.*droite*, `indices`, `op`) ;
12 | | **retourner** contruct_interne(n.indice, g, d, `op`) ;
13 | **fin**

3.2 Diagrammes de décision algébriques

Nous avons donc vu que pour représenter des fonctions booléennes, Gattinger a utilisé des BDDs, ce qui lui a permis de mettre en pratique sa représentation symbolique des structures de Kripke (sec. 3.1). La structure de donnée BDD est très répandue et utilisée, néanmoins il existe des extensions moins utilisées qui vont nous permettre d'appliquer les concepts du *model checking* symbolique à un cadre plus riche que DEL, permettant de manipuler des probabilités. Ces extensions des BDDs permettent de représenter une généralisation des fonctions booléennes, les *fonctions pseudo-booléennes* (PBF, pour « Pseudo-Boolean function »), qui associent à toute valuation de 2^V une valeur rationnelle plutôt qu'une valeur booléenne. Pour simplifier certaines définitions, nous considérerons qu'elles peuvent aussi prendre des valeurs infinies. Nous définissons dans ce but l'ensemble $\overline{\mathbb{Q}}$.

Définition 3.7. *L'*ensemble $\overline{\mathbb{Q}}$ des rationnels augmenté *est défini comme* $\overline{\mathbb{Q}} \coloneqq \mathbb{Q} \cup \{-\infty,\ +\infty\}$.

Les infinis se comportent de la manière attendue intuitivement : en particulier, pour tout $x \in \mathbb{Q}$, *on a* $-\infty < x < +\infty$ *et* $x/\pm\infty = 0$ *; les opérations suivantes* $\pm\infty/\pm\infty$, $+\infty + -\infty$ *et* $-\infty + +\infty$ *sont interdites ; et on choisit la convention* $0 \times \pm\infty = 0$.

On note $\overline{\mathbb{Q}}_+$ *(resp.* $\overline{\mathbb{Q}}_+^*$*) l' ensemble* $\overline{\mathbb{Q}}$ *réduit aux valeurs positives (resp. strictement positives).*

Définition 3.8. *Fonction pseudo-booléen Soit* $X = \{x_1, \ldots, x_n\} \subseteq PS$ *un ensemble fini de variables propositionnelles, nous appelons* fonction pseudo-booléenne (PBF) sur X *une fonction totale du type :* $f\colon 2^X \to \overline{\mathbb{Q}}$.

Une PBF sur un vocabulaire V est donc une généralisation d'une fonction booléenne sur V. En effet, une fonction booléenne peut être vue comme une fonction du type $f\colon 2^X \to \mathbb{B}$ (déf. 3.1). Pour rappel, nous voyons une formule propositionnelle $\phi \in V$ comme une fonction booléenne dont les paramètres sont toutes les valuations possibles de son vocabulaire V, de sorte que $\forall v \in 2^V\colon\ \phi(v) = 1$ si $v \models \phi$, 0 sinon. Nous passons simplement d'une fonction qui a pour domaine des images $\mathbb{B}$ ($f\colon 2^X \to \mathbb{B}$) à $\overline{\mathbb{Q}}$ ($f\colon 2^X \to \overline{\mathbb{Q}}$).

Comme simplification, nous considérons ici $\mathbb{B} \subset \overline{\mathbb{Q}}$ avec $\mathbb{B}$ l'ensemble à deux nombres rationnels $\{0,\ 1\}$.

Remarque 3.9. *Nous utiliserons une notation simple pour les PBFs dans nos exemples : étant donné un ensemble* Φ *de* n *formules propositionnelles mutuellement incohérentes sur une vocabulaire* X *et* $\alpha_i, \ldots, \alpha_n \in \overline{\mathbb{Q}}$, *nous notons*

$$\{\phi_1\colon \alpha_1, \ldots, \phi_n\colon \alpha_n\}$$

la PBF associant la valeur α_i à chaque modèle de la formule ϕ_i, et implicitement 0 pour tous les modèles qui ne seraient pas dans l'union des modèles des formules ϕ_i. Nous utiliserons également une version étendue de cette notation : $\{\phi\colon f(\phi) \mid \phi \in \Phi\}$ où $f\colon \Phi \to \overline{\mathbb{Q}}$.

Exemple 3.10. *Par exemple, soit un vocabulaire donné $X = \{x_1, x_2, x_3, x_4\}$ et une PBF $f = \{(x_1 \wedge x_2) \oplus (x_3 \wedge x_4)\colon 1,\ x_1 \wedge x_2 \wedge x_3 \wedge x_4\colon 2\}$ (avec $\oplus$ l'opération binaire de disjonction exclusive). Chaque formule est compatible avec un ensemble disjoint de valuations et chaque formule associe une valeur à chacun de leurs modèles (ici, 1 et 2). La valeur 0 sera associée implicement à tous les modèles qui n'appartiennent pas aux formules énumérées. La table de vérité de f ici est identique à celle présentée à la table 3.1, page 67, sauf que le dernier modèle $\{x_1, x_2, x_3, x_4\}$ a pour valeur 2 et non 1 (vrai).*

Les fonctions pseudo-booléennes, ne se contentant plus de rester sur le domaine booléen, il est possible de les manipuler grâce à des opérateurs plus complexes. En effet, il est possible d'en faire la somme ou de les multiplier (a contrario, les soustraire ou les diviser).

Exemple 3.11. *Ainsi, nous pouvons voir que dans l'exemple précédent la PBF f est équivalente à une formule f définie de la manière suivante : $f(x_1, x_2, x_3, x_4) = x_1 \times x_2 + x_3 \times x_4$ où $\times$ est l'opérateur multiplicatif et $+$ est l'opérateur additif (respectivement équivalent à la conjonction et la disjonction si les deux termes sont des fonctions booléennes). Néanmoins, ici, l'addition permet le résultat suivant : $f(\{x_1, x_2, x_3, x_4\}) = 2$.*

Nous avons vu que les BDDs étaient particulièrement bien adaptés pour représenter des fonctions booléennes. Malheureusement, elles ne permettent pas de représenter des fonctions pseudo-booléennes. Or, les PBFs permettent de représenter des informations plus riches que les fonctions booléennes, notamment des probabilités ou des fonctions de masses.

Néanmoins, il existe des structures de données permettant de généraliser la définition des BDDs pour que les valeurs du domaine des fonctions représentées ne soient pas que booléens. Nous allons voir brièvement quelques structures qui sont des extensions des BDDs, sous-catégorie des VDD (Diagramme de Décision Valués). Ces structures ont été expliquées récemment dans la thèse de Nicolas Schmidt [Sch15]. La complexité des algorithmes de vérification et de transformation de ces structures sont étudiées dans « *A Knowledge Compilation Map for Ordered Real-Valued Decision Diagrams* » [FMNS14] et la compacité dans « Compacité pratique des diagrammes de décision valués » [FMS14, FMS13a].

Par exemple, la structure Algebraic Decision Diagram (ADD) [BFG+97] est une généralisation aux valuations non booléennes du langage OBDD, les deux

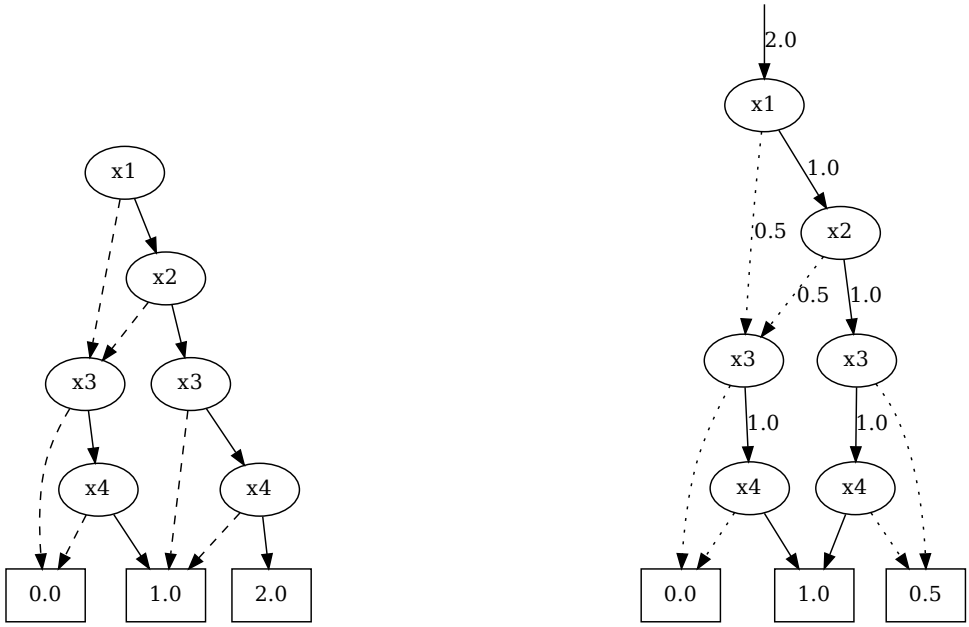
nœuds terminaux 0 et 1 des OBDDs (voir section 3.1) étant remplacés par autant de nœuds que de valeurs du domaine associées à une affectation au moins. Comme illustration, la figure 3.3a représente l'ADD de la fonction f présentée dans l'exemple 3.10. Cette représentation a le mérite d'avoir beaucoup de similitude en terme de structures, d'algorithmique, et donc de complexité, avec les OBDDs. Malheureusement, le problème de cette représentation est l'explosion combinatoire du nombre de nœuds terminaux. En effet, plus il y a de valeurs différentes associées à des valuations, plus l'ADD sera potentiellement grand.

Une autre structure permet de contourner ce problème d'augmentation de taille du diagramme de decision valué. Il s'agit du SLDD (*Semiring Labeled Decision Diagram (SLDD)* [FMS13b]) Ici, ce sont les arcs, et non les nœuds terminaux, qui sont étiquettés par des valeurs. La structure de valuation est un semi-anneau noté $A = \langle \mathcal{D}, \times, +, 1, 0 \rangle$, $\mathcal{D}$ étant le domaine des valeurs des valuations de la structure, 1 dénotant ici l'élément neutre de l'opérateur $\times$ et 0 dénotant l'élément neutre de l'opérateur $+$, absorbant pour $\times$. L'agrégation des valuations d'un chemin (définissant alors le domaine de la fonction) est faite via l'opérateur $\times$ de notre semi-anneau. Cette définition permet de conserver les informations numéraires aux arcs et non aux feuilles, permettant une plus grande mutualisation des arcs du diagrammes ; c'est pourquoi cette représentation est plus compacte que les ADDs. Nous illustrons aussi cette structure par le figure 3.3b. Nous pouvons voir que les valeurs des feuilles sont comprises entre 0 et 1, car tous les arcs de cet SLDD sont normalisés. Un seul arc ici est supérieur à 1, c'est celui de l'arc entrant du nœud étiquetté par x_1. Cet arc est une compensation à toutes les normalisations et représente, de fait, la valeur maximum présente dans le diagramme.

Un autre langage permet de généraliser les BDDs : l'*Affine Algebraic Decision Diagram (AADD)* introduit dans [TP97, SM05] permet d'utiliser conjointement les opérateurs $\times$ et $+$ sur $\mathbb{R}^+$ pour agréger les valeurs des arcs. Dans un SLDD, un arc porte une valuation simple alors que dans un AADD, les arcs sont étiquetés par des couples de valeurs : un facteur multiplicatif et un facteur additif.

Ces trois types de structures permettent de modéliser des fonctions non plus booléennes, mais pseudo-booléennes (déf. 3.8), dans les cas où les valeurs des variables propositionnelles sont binaires.

Exemple 3.12. *Prenons un exemple concret montrant la différence de compacité entre un ADD et un SLDD. Soient n variables $\{x_1, \ldots, x_n\}$ et une fonction $f : 2^n \to \overline{\mathbb{Q}}$ tel que $f(x_1, \ldots, x_n) = 2^{(\sum_{n=1}^{n} 2^{i-1} \times x_i)}$. L'exposant $\sum_{n=1}^{n} 2^{i-1} \times x_i$ permet de calculer la valeur du nombre binaire écrit par les variables x_i (bit de poids faible étant x_0). Par exemple, pour la valuation $v = \{x_3\}$ (nombre binaire « 100 », qui est le nombre 4), nous avons $f(v) = 2^{2^2} = 2^4 = 16$. De cette manière, à chaque valuation sur $\{x_1, \ldots, x_n\}$ est associé la puissance de 2 du nombre binaire des variables. Pour 4 variables, nous avons donc 16 valeurs possibles, allant de* 1 *à*



(a) ADD de f

(b) SLDD de f

FIGURE 3.3 – Exemple d'SLDD et d'ADD pour la même PBF f (ex. 3.10) et l'ordre des variables $\langle x_1, x_2, x_3, x_4 \rangle$

32768 *(*2^{15}*).*

Avec une telle fonction, nous avons autant de valeurs possibles que de valuations. Ainsi, si l'on souhaite représenter f avec un ADD, cet ADD aura autant de feuilles. C'est ce que l'on présente sur la figure 3.4.

Par ailleurs, cette fonction est facilement représentable par un SLDD, ce que nous représentons sur la figure 3.5. En effet, en suivant les chemins vers les feuilles et en partant du maximum (32768), il est possible de récupérer toutes les puissances de 2 en multipliant potentiellement successivement par $\frac{1}{256}$*,* $\frac{1}{16}$*,* $\frac{1}{4}$ *ou* $\frac{1}{2}$*. Par exemple, en suivant toutes les flèches pointillées, on obtient 1, qui est bien* 2^0*, quand toutes les variables* x_i *sont fausses.*

Notons que ce cas est vraiment très particulier, où il s'agit du pire cas pour un ADD, car il y a autant de valeurs que de valuations et du meilleur cas pour un SLDD, car un seul nœud par variable suffit pour obtenir le SLDD. Ça n'aurait pas été le cas si on avait voulu avoir les valeurs de 1 à 16, par exemple.

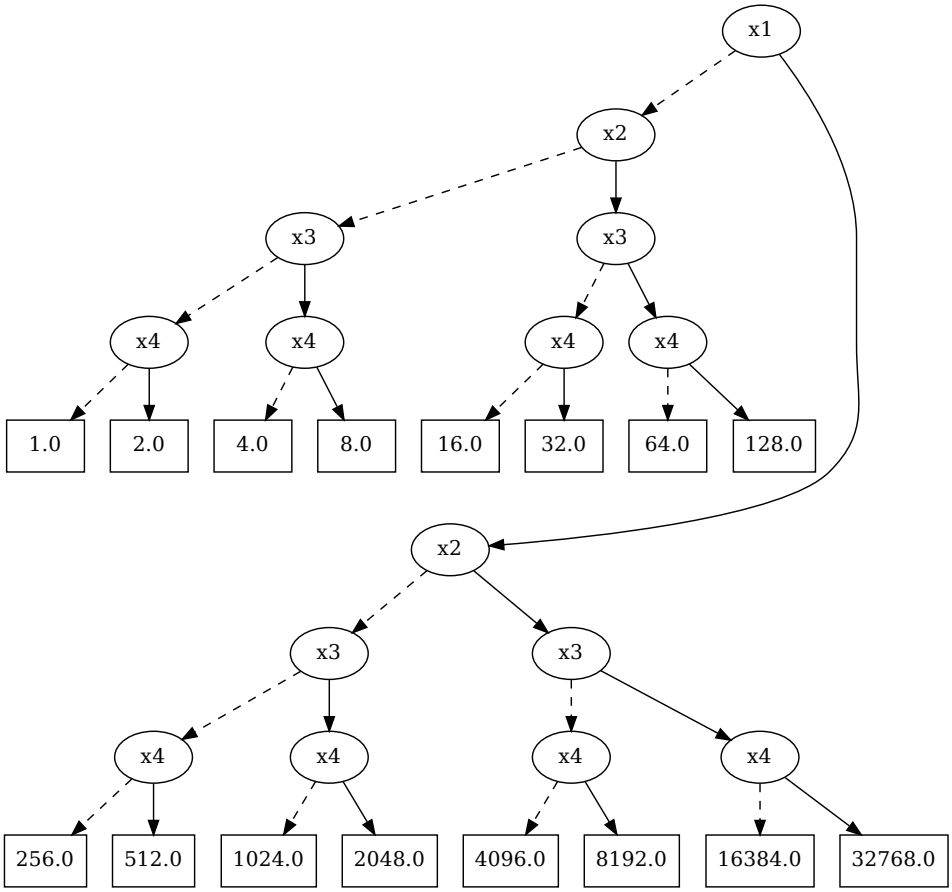


FIGURE 3.4 – ADD "au pire", exemple 3.12.

Par ailleurs, d'autres structures existent, comme les PSDDs (*Probabilistic Sentential Decision Diagrams*), où les nœuds sont probabilistes [KdBCD14]. Néamoins cette structure ne représente que des distributions de probabilités.

Enfin, notons que d'autres structures ont été étudiées pour envisager de représenter des formules différentes des formules de $\mathcal{L}_{\text{prop}}$. En effet, dans « *Efficient Representations for the Modal Logic S5* » [NZ16] (qui s'appuie sur [BFM10]), les auteurs envisagent des langages, nommés EDNF ou EBDD et ESD (*Epistemic Splitting Diagrams*) pour représenter les formules subjectives du type $\mathrm{K}p \wedge \neg \mathrm{K}q$. Ce sont des variantes naturelles des BDDs. Malheureusement, ces structures n'ont pas fait leurs preuves pour des applications utilisables en pratique.

Comme nous l'avons vu, les PBFs ont des valeurs dans $\overline{\mathbb{Q}}$ et non pas dans $\mathbb{B}$ comme lesquels fonctions booléennes. C'est pourquoi, nous ne pouvons utiliser les opérateurs binaires booléens. De nouveaux opérateurs binaires adaptés aux valeurs de $\overline{\mathbb{Q}}$ s'offrent à nous : la multiplication et l'addition.

la multiplication « remplace » la conjonction : multiplier des fonctions booléennes revient strictement à en faire la conjonction dans le cas booléen, car les valeurs obtenues seront 0 ou 1,

l'addition « remplace » la disjonction : additionner des fonctions booléennes ne donne pas strictement des valeurs 0 ou 1, car il est possible d'obtenir la valeur 2 en additionnant deux 1.

Ainsi, si nous avons deux fonctions booléennes f et g, les fonctions $f \wedge g$ et $f \times g$ vont avoir de grandes similitudes, néanmoins elles auront deux valeurs respectivement dans $\{0,\ 1\}$ et $\{0,\ 1,\ 2\}$. Ce que nous pouvons constater en commun entre ces deux fonctions, ce sont leurs *supports*.

Définition 3.13 (Support)**.** *Soit f une fonction allant d'un domaine fini X à l'ensemble des rationnels. Nous nommons* support *de f l'ensemble des valuations auquel f assigne une valeur non nulle, i.e.* $\mathrm{supp}(f) = \{x \in X \mid f(x) > 0\}$.

Exemple 3.14. *Par exemple, si nous reprenons nos exemples de fonction booléenne f (ex. 3.2) et fonction pseudo-booléenne g (ex. 3.10), avec $X = \{x_1, \ldots, x_4\}$:*

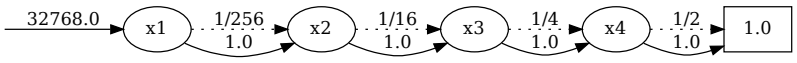


FIGURE 3.5 – SLDD "optimisé", exemple 3.12.

- $f(x_1, \ldots, x_4) = (x_1 \wedge x_2) \vee (x_3 \wedge x_4)$;
- $g(x_1, \ldots, x_4) = (x_1 \times x_2) + (x_3 \times x_4)$.

Nous avons bien des valeurs assignées aux valuations différentes, mais un support identique. Le support est un ensemble de 7 valuations, que nous présentons dans la table 3.2.

x_1	x_2	x_3	x_4	f	g	x_1	x_2	x_3	x_4	f	g
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	1	0	0	1	0	0	1	0	0
0	0	1	0	0	0	1	0	1	0	0	0
0	**0**	**1**	**1**	1	1	**1**	**0**	**1**	**1**	1	1
0	1	0	0	0	0	**1**	**1**	**0**	**0**	1	1
0	1	0	1	0	0	**1**	**1**	**0**	**1**	1	1
0	1	1	0	0	0	**1**	**1**	**1**	**0**	1	1
0	**1**	**1**	**1**	1	1	**1**	**1**	**1**	**1**	1	2

TABLE 3.2 – Tableau des valeurs des fonctions f et g de l'exemple 3.14 en fonction des valuations de $\{x_1, x_2, x_3, x_4\}$. Leurs supports (identiques) sont représentés en gras.

Cette notion de support sera très utile pour pouvoir manipuler une PBF comme un ensemble de valuations.

Nous avons vu que les opérations n'étaient plus les mêmes que dans le cadre booléen. Notamment, il n'est plus possible de considérer la quantification booléenne (déf. 2.24) avec des valeurs rationnelles. Dans le cas des fonctions pseudo-booléennes, il est possible de définir un opérateur, plus général, permettant de supprimer les variables d'une fonction. C'est ce qu'on appelle la *marginalisation.*

Définition 3.15 (Marginalisation)**.** *Soit* $\odot \colon \overline{\mathbb{Q}} \times \overline{\mathbb{Q}} \to \overline{\mathbb{Q}}$ *une opération associative et commutative avec un élément neutre, soit* X *et* Y *deux ensembles de variables propositionnelles tel que* $X \subseteq Y$ *et* f *étant une PBF sur* Y*.*

La $\odot$-marginalisation *de* X *dans* f*, notée* $\mathrm{Marg}_X^{\odot}(f)$*, est la fonction* $g \colon 2^{Y \setminus X} \to \overline{\mathbb{Q}}$ *définie par :*

$$g(v) \coloneqq \bigodot_{v' \in 2^X} f(v \cup v')$$

Notons qu'ici, $v \in 2^{Y \setminus X}$ et $v' \in 2^X$, ainsi $v \cup v' \in 2^Y$, car $X \subseteq Y$.

Voici quelques exemples classiques d'opérations associatives et commutatives avec un élement neutre : l'addition, la multiplication, le minimum, le maximum, ou encore les opérateurs booléens $\vee$ et $\wedge$.

La marginalisation de f définie sur Y, des variables de l'ensemble X par l'opérateur $\odot$ permet donc d'obtenir la fonction g, définie sur $Y \setminus X$, de sorte à ce que les valeurs du support de f, identiques une fois restreints de Y à $Y \setminus X$, se voient agrégés par l'opérateur $\odot$.

Exemple 3.16. *Par exemple, les marginalisations de* $\{x_2, x_4\}$ *pour les fonctions* f *et* g *(ex. 3.14) par l'opération somme, notées* $\mathrm{Marg}^{+}_{\{x_2,x_4\}}(f)$ *et* $\mathrm{Marg}^{+}_{\{x_2,x_4\}}(g)$*, permettent d'obtenir les fonctions* f' *et* g' *sur le vocabulaire* $\{x_1, x_3\}$*, dont les tables des valeurs sont présentées dans la table 3.3.*

En marginalisant les deux variables x_2 *et* x_4*, il se produit pour chaque nouvelle valuation sur* $\{x_1, x_3\}$ *l'agrégation de 4 valuations sur* $\{x_1, x_2, x_3, x_4\}$*. Par exemple, pour le nouveau modèle* $\{x_1, x_3\}$*, il faut faire la somme des 4 modèles où* x_1 *et* x_3 *sont vrais, i.e.* $\{x_1, x_2, x_3, x_4\}$*,* $\{x_1, x_2, x_3\}$*,* $\{x_1, x_3, x_4\}$*,* $\{x_1, x_3\}$*. Nous avons ainsi, par exemple :* $f'(\{x_1, x_3\}) = f(\{x_1, x_2, x_3, x_4\}) + f(\{x_1, x_2, x_3\}) + f(x_1, x_3, x_4) + f(\{x_1, x_3\}) = 1 + 1 + 1 + 0 = 3$.

x1	x3	f'	g'
0	0	0	0
0	1	2	2
1	0	2	2
1	1	3	4

TABLE 3.3 – Tables des valeurs des fonctions f' et g' (ex. 3.16).

Notons que si f est une fonction booléenne, alors la $\vee$-marginalisation (resp. $\wedge$-marginalisation) de X correspond à la *quantification existentielle (resp. universelle)* des variables de X (i.e. $\exists$ (resp. $\forall$)). Nous notons ainsi $\mathrm{Forget}^{\exists}_X(f) = \mathrm{Marg}^{\vee}_X(f)$ et $\mathrm{Forget}^{\forall}_X(f) = \mathrm{Marg}^{\wedge}_X(f)$.

Par ailleurs, nous allons avoir besoin d'une autre opération sur les fonctions pseudo-booléennes : il s'agit de la *coupe*, notée Cut.

Définition 3.17. *Pour une PBF* f *sur un ensemble de variables* $X \subseteq PS$ *et un nombre rationnel* β*, nous notons* $\mathrm{Cut}_{\geq\beta}(f)$ *la* coupe *de la fonction* f *telle que :*

$$\forall v \in 2^X,\ \mathrm{Cut}_{\geq\beta}(f)(v) = \begin{cases} 1, & \textit{si } f(v) \geq \beta \\ 0, & \textit{sinon} \end{cases}$$

Grâce à la définition du support (déf. 3.13), il est aussi possible de voir l'opérateur Cut comme l'opérateur qui renvoie l'ensemble des valuations dont la valeur satisfait l'inégalité vis à vis de β, i.e. :

$$\mathrm{supp}(\mathrm{Cut}_{\geq\beta}(f)) = \{v \mid f(v) \geq \beta\}$$

Exemple 3.18. *Si nous reprenons nos PBFs de l'exemple 3.16, nous aurons par exemple* $\mathrm{Cut}_{\geq 2}(f') = \{\{x_3\}, \{x_1\}, \{x_1, x_3\}\}$*, ou encore* $\mathrm{Cut}_{\geq 3}(g') = \{\{x_1, x_3\}\}$*. Cela se traduit par la table des valeurs 3.4.*

x1	x3	$\mathrm{Cut}_{\geq 2}(f')$	$\mathrm{Cut}_{\geq 3}(g')$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

TABLE 3.4 – Tables des valeurs des fonctions $\mathrm{Cut}_{\geq 2}(f')$ et $\mathrm{Cut}_{\geq 3}(g')$ (ex. 3.18).

Nous verrons dans le prochain chapitre comment cet opération pourra nous servir. Avant de passer au prochain chapitre, nous allons décrire les aspects algorithmiques des opérations que nous avons décrites sur les PBFs, sur les structures ADDs.

Tout d'abord, la gestion d'une structure de donnée pour les ADDs est très similiaire à celle des BDDs, représentant des fonctions booléennes. La construction d'un Nœud (algo. 3.1) est strictement identique, modulo le fait que l'attribut *valeur* n'est pas un booléen, mais bien une valeur rationnelle. À partir de là, l'algorithme de construction de nœuds et de feuilles est identique à celui présenté pour les BDDs (algo. 3.2), toujours, modulo la valeur de l'attribut *valeur*. Ensuite, nous avons vu que nous pouvons appliquer les opérateurs unaires (négation) ou binaires (disjonction et disjonction) sur les BDDs qui représentent les fonctions booléennes (algo. 3.3 et 3.4). Il en va de même pour les opérateurs unaires et binaires des ADDs, il suffit simplement que les opérateurs ne soient plus de la forme $\texttt{op} : \mathbb{B} \to \mathbb{B}$ ou $\texttt{op} : \mathbb{B}, \mathbb{B} \to \mathbb{B}$, mais $\mathbb{B} \to \overline{\mathbb{Q}}$ ou $\texttt{op} : \mathbb{B}, \mathbb{B} \to \overline{\mathbb{Q}}$.

Quant à l'algorithme de conditionnement (algo. 3.5), il est strictement identique entre les BDDs et ADDs.

Par contre, pour ce qui est de l'algorithme de quantification (algo. 3.6), il ne suffit pas de simplement remplacer l'opérateur `op` comme nous l'avons présenté pour application_binaire pour obtenir l'algorithme de marginalisation, qui est présenté dans l'algorithme 3.7, page 85.

Bien que la structure de l'algorithme soit très similaire, la modification de la structure du diagramme de décision doit prendre en compte, qu'à chaque saut d'indice dans les nœuds, il sera peut-être nécéssaire de modifier en conséquence les valeurs de la fonction. C'est tout l'utilité des boucles ligne 6 et 15 de l'algorithme.

Cette présentation de l'algorithme pourrait être adaptée à chaque opérateur. Par exemple, si l'opérateur est $+$, alors cela revient à remplacer les lignes 5 à 8 ou 15 à 17 (i est le nombre d'indices passés) par :

Algorithme 3.7: Marginalisation

Entrée: `n`, un Nœud dont on veut supprimer les indices présents dans la liste d'entiers `indices` (triée croissante) et une opération d'agrégation binaire `op` : $\mathbb{Q}, \mathbb{Q} \to \mathbb{Q}$
Sortie : Un Nœud

```
/* Utilise un cache afin d'effectuer les opérations une seule fois, si
   elles se présentent plusieurs fois.                                  */
```

```
1  Fonction marginalisation(n, indices, op):
2      si |indices| = 0 alors
           /* Tous les indices à marginaliser sont marginalisés.       */
3          retourner n ;
4      sinon si n.indice > max(indices) alors
           /* On est plus bas que tous les indices à supprimer, que l'on n'a
              pas croisé. Il faut appliquer l'opérateur en fonction du nombre
              d'indices que nous souhaitions croiser.                   */
5          res ← n ;
6          pour i de 0 à nombre d'indices passés faire
7              res ← application_binaire(res, res, op) ;
8          fin
9          retourner res ;
10     sinon si n.indice est dans indices alors
           /* Un nœud à oublier a été trouvé. On applique l'opérateur
              d'agrégation aux successeurs et on ajuste en fonction du nombre
              d'indices non vus.                                        */
11         n_indices ← ⟨i | i ∈ indices tel que i > n.indice⟩ ;
12         g ← marginalisation(n.gauche, n_indices, op) ;
13         d ← marginalisation(n.droite, n_indices, op) ;
14         res ← application_binaire(g, d, op) ;
15         pour i de 0 à nombre d'indices passés faire
16             res ← application_binaire(res, res, op) ;
17         fin
18         retourner res ;
19     sinon
           /* On reconstruit un nœud et on utilise la récursivité.     */
20         g ← marginalisation(n.gauche, indices, op) ;
21         d ← marginalisation(n.droite, indices, op) ;
22         retourner construct_interne(n.indice, g, d) ;
23     fin
```

$$res \leftarrow \text{application\_binaire}(\text{res}, \text{construct\_feuille}(2^{\text{i}}, \times)).$$

Pour la multiplication, l'itération est exactement ce que nous souhaitons et revient moralement à calculer le nœud `n` à la puissance i ($\texttt{n} \times \cdots \times \texttt{n}$).

Pour ce qui est des opérateurs minimum ou maximum, par exemple, les boucles ne servent à rien et il suffit de retourner le res déjà calculé.

Notons, que comme le *forgetting*, les opérateurs binaires doivent être associatifs et commutatifs. Dans le cas booléen, cela s'avère pour les opérateurs $\vee$ et $\wedge$. Pour ce qui est de la marginalisation, elle est alors possible avec l'addition, la multiplication, le minimum ou le maximum, mais ça ne sera pas le cas avec la division (sans considérer qu'il faudrait gérer la division par 0) ou encore avec la soustraction, qui est elle, anti-commutative ($\forall x, y,\ x \odot y = -y \odot x$).

Notons que l'algorithme existe pour la marginalisation d'un seul indice. Ici, il s'agit d'en oublier plusieurs en un seul passage descendant dans le diagramme.

Tous les algorithmes présentés ici sur les ADDs possèdent la même complexité que ceux des BDDs. Simplement, notons que comme il peut y avoir de plus nombreuses feuilles contrairement aux BDDs, il y a potentiellement moins de simplification et de mise en cache lors des calculs.

Conclusion de la première partie

Nous avons vu dans le premier chapitre qu'il était possible de représenter les connaissances épistémiques des agents et leurs modifications, grâce à la logique épistémique dynamique (DEL).

Nous avons par la suite vu que Gattinger et ses co-auteurs avaient défini des structures symboliques (i.e. structures de croyances et modificateurs de croyances) dans son *framework* SMCDEL, permettant de faire du *model checking* grâce à des fonctions booléennes (sec. 2.2 et 2.5). Ces structures sont représentées de manière informatique par des structures de données adaptées et efficaces : les BDDs (sec. 3.1).

Nous en avons profité pour présenter une extension des fonctions booléennes, les fonctions pseudo-booléennes (sec. 3.2), nous permettant de manipuler des valeurs non plus dans le domaine des booléens, mais des rationnels. Cela nous permet, par exemple, de représenter naturellement des fonctions de probabilités. Par la suite et dans la même section, nous avons présenté des structures de données (SLDDs et ADDs), qui sont une généralisation des BDDs, qui permettent de représenter des fonctions pseudo-booléennes.

Les PBFs, implémentées avec des ADDs, vont nous servir dans la deuxième partie de briques élémentaires pour créer un *framework* utilisant le modèle de SMCDEL mais pouvant manipuler des probabilités.

Deuxième partie

Model checking symbolique pour PDEL

Introduction de la seconde partie

Une des motivations principale de la création d'un *framework* représentant symboliquement DEL est d'éviter l'explosion combinatoire du nombre des mondes des structures de Kripke. Cette représentation symbolique rend alors possible l'utilisation en pratique de DEL, contournant ce phénomène combinatoire.

Malheureusement, DEL ne permet pas de représenter des informations probabilistes. Pour cela, il existe une extension de DEL, nommée naturellement PDEL, pour *Probabilistic Dynamic Epistemic Logic*, qui, pour résumer, couple l'utilisation des structures de Kripke et des modèles d'événements avec des lois de probabilité discrètes.

PDEL a été étudié théoriquement, mais à l'instar de DEL, le *framework* apparaît rapidement comme impossible à exploiter en pratique, pour exactement les mêmes raisons. PDEL utilisant des fonctions de probabilités, nous nous sommes demandés s'il était possible d'adapter les définitions de SMCDEL de sorte à pouvoir manipuler des distributions de probabilités.

Pour cela, nous aurons besoin d'envisager quelques modifications des définitions classiques de PDEL (notamment via une version dénormalisée) ; c'est ce que nous verrons dans le chapitre 4.

Ensuite, comme Gattinger a pu le proposer avec une représentation symbolique de DEL, nous allons proposer une représentation symbolique permettant d'utiliser des probabilités, via les PBFs (chap. 5 et 6).

Par ailleurs, SMCDEL étant implémenté avec des BDDs, nous nous sommes demandé s'il était possible d'implémenter notre variante avec une extension des BDDs, les ADDs, qui peuvent représenter des distributions de probabilités. En outre, les ADDs, comme les BDDs, permettent des vérifications de modèles efficaces en pratique grâce à des algorithmes polynomiaux ; nous allons mettre en pratique cela dans le cadre de la représentation symbolique de PDEL. Nous présenterons la mise en œuvre de cette représentation en terme de programmes et de fonctionnalités dans le chapitre 7. Dans ce même chapitre nous illustrerons un cas applicatif de notre *framework* en présentant la représentation symbolique du jeu Hanabi.

Enfin, nous mettrons en pratique cette représentation symbolique grâce à des expérimentations basée sur notre représentation d'Hanabi (chap. 8), où nous comparerons SMCDEL à notre version probabiliste, en discutant de l'interêt de notre modélisation.

4 | Logique épistémique dynamique probabiliste

Nous avons défini le fonctionnement de la logique épistémique dynamique (sec. 1.7) qui permet de modéliser l'évolution des connaissances épistémiques des agents dans le temps, grâce l'application de modèles d'événements. Nous allons présenter une extension de celle-ci qui permet de raisonner sur les connaissances à propos les probabilités : *Probabilistic Dynamic Epistemic Logic (PDEL)*. Avec cette extension, il est possible pour les agents de raisonner sur des connaissances évidemment épistémiques, mais aussi de raisonner avec des probabilités, de sorte à quantifier leurs certitudes sur leurs connaissances.

Dans ce chapitre, nous allons tout d'abord présenter cette extension probabiliste de DEL (sec. 4.1) telle qu'elle existe dans la littérature. Puis, nous présenterons une variante de celle-ci, que l'on qualifiera de *dénormalisée* (sec. 4.2) ; les différences entre les deux nous permettront d'établir des définitions simplifiées dans le cas symbolique. Ensuite, nous présenterons un modèle événementiel simplifié pour PDEL qui n'est pas exploité dans la littérature, le modèle observationnel (sec. 4.3), et nous le comparerons au modèle événementiel standard (sec. 4.5 et 4.5). Pour conclure ce chapitre, nous présenterons des exemples d'utilisation des deux types d'événements introduits (sec. 4.4).

4.1 État de l'art : PEL et PDEL

Ce sont les travaux de Fagin et Halpern en 1994 [FH94] qui définissent ce que sont des structures de Kripke avec des probabilités. Ensuite, Barteld P. Kooi en 2003 définit PDEL qui permet de représenter des probabilités sur les mondes de la structure de Kripke et de pouvoir faire des mises à jour des connaissances en fonction de probabilités sur les événements. Enfin, van Benthem et al. en 2009 [vBGK09] définissent un système global permettant de gérer les probabilités dans une structure de Kripke, en distinguant les probabilités pré-établies sur les mondes,

les probabilités sur les événements, mais aussi les incertitudes conditionnelles quant aux événements grâce à des distributions de probabilités sur les événements atomiques.

Comme nous l'avons fait précédemment en présentant la logique épistémique (sec. 1.4) avant la logique épistémique dynamique (sec. 1.7), nous allons voir dans un premier temps la logique épistémique probabiliste (sec. 4.1.1) puis nous ajoutons un opérateur dynamique (sec. 4.1.2).

4.1.1 Logique épistémique probabiliste

Celles-ci se basent logiquement sur les définitions que nous avons déjà vues, sans probabilités, des *frames* (déf. 1.4) et structures de Kripke (déf. 1.6).

Définition 4.1. *Une* frame *probabiliste est une structure* $\mathcal{F} = \langle W, R, \mu \rangle$ *où :*

- W*,* R *sont définis comme précédemment pour une frame classique (déf. 1.4),*
- $\mu : \mathfrak{A} \to (W \to (W \to [0,1]))$ *est un ensemble de fonctions, dites* de probabilités*, qui assigne une loi de probabilité discrète sur* W *à chaque agent* $a \in \mathfrak{A}$ *et à chaque monde* $w \in W$*. La probabilité assignée à* w' *via la loi de probabilité discrète pour l'agent* a *au monde* w *est noté* $\mu_a(w)(w')$*.*

Remarque 4.2. *Par commodité et abus de langage, nous appelerons* μ_a *loi de probabilité discrète alors qu'en réalité, c'est un ensemble de lois de probabilité discrètes. Notons que par définition des lois de probabilité discrètes, nous avons :*

$$\forall a \in \mathfrak{A},\ \forall w \in W,\ \sum_{w' \in W} \mu_a(w)(w') = 1$$

Dans une *frame* probabiliste, il y a bien deux types d'informations distinctes : R représente les informations non-probabilistes et μ les informations probabilistes.

Comme le faisait une structure de Kripke classique, une structure de Kripke probabiliste ajoute la notion de valuation aux mondes de la *frame* grâce à la fonction val.

Définition 4.3. *Une structure de Kripke probabiliste pour un vocabulaire* V *est une structure* $\mathcal{M} = \langle \mathcal{F}, \mathrm{val} \rangle$ *où :*

- $\mathcal{F} = \langle W, R, \mu \rangle$ *est une frame probabiliste (déf. 4.1),*
- val *est une fonction de valuation* $W \to 2^V$*, comme défini déf. 1.6.*

Intuitivement et comme pour DEL, la relation d'accessibilité R_a relie des mondes que l'agent a considère comme *indistinguables* et $\mu_a(w_1)(w_2)$ indique la probabilité que l'agent a attribue au fait d'être dans le monde w_2 alors qu'il est en réalité dans w_1.

Avoir à la fois une relation d'accessibilité et des lois de probabilité discrètes peut sembler redondant, mais cela présente deux avantages :

- cela permet de modéliser une incertitude non quantifiée entre les mondes possibles, ce qui n'est pas la même chose que de leur assigner une probabilité uniforme ;
- cela permet de faire la distinction entre une probabilité nulle d'être vrai et une impossibilité réelle. Une discussion plus approfondie est présente dans l'article « *Reasoning About Knowledge and Probability* » [FH94].

Néanmoins, il est tout à fait naturel et intuitif de ne considérer que ce que Demey et Kooi dans l'article « *Logic and probabilistic update* »[DK14] appellent les structures *a-consistantes*, dans lesquelles $\mu_a(w)$ donne une probabilité non nulle uniquement aux états accessibles via la relation R_a. Cela permet ainsi aux agents d'attribuer des probabilités nulles aux mondes qu'ils considèrent comme impossibles, ce qui empêche un agent de considérer ces deux propositions comme étant possibles simultanement : « l'agent a assigne une probabilité non nulle à ϕ » et « l'agent sait que $\neg\phi$ ». Si c'était le cas, l'agent en question saurait que ϕ est impossible de par la relation R_a mais considérerait aussi que la probabilité de ϕ est supérieure à 0. Pour pallier ce problème, voici comment Demey et Kooi définissent la a-consistance :

Définition 4.4 (Consistance)**.** *Soit une* frame *probabiliste et un agent* $a \in \mathfrak{A}$*. Une* frame $\mathcal{F}$ *est dite a-consistante* ssi *pour tous états* w, w' *:* $(w, w') \notin R_a \implies \mu_a(w)(w') = 0$*.*

La consistance permet de faire correspondre les relations et les probabilités, de sorte que s'il n'y a pas de relation, il y a une probabilité nulle entre les mondes.

L'uniformité est aussi définie et permet de considérer des lois de probabilité discrètes identiques pour des mondes reliés.

Définition 4.5 (Uniformité)**.** *Soit une* frame *probabiliste et un agent* $a \in \mathfrak{A}$*. Une* frame $\mathcal{F}$ *est dite a-uniforme si pour tous états* w, w' *:* $(w, w') \in R_a \implies \mu_a(w) = \mu_a(w')$*.*

Remarque 4.6. *Notons que l'a-uniformité correspond au principe d'introspection*

Nous ne faisons aucune hypothèse sur la a-consistance ou la a-uniformité des structures de Kripke probabilistes. Nos définitions porteront sur des structures générales, néanmoins nos exemples auront ces propriétés par soucis de simplicité. Comme précédemment, nous ne faisons aucune hypothèse sur les relations d'accessibilités.

Afin de pouvoir manipuler des formules contenant des probabilités, la logique $\mathcal{L}_{\text{EL}}$ (déf. 1.3, page 13) est étendue avec l'opérateur Pr de sorte à obtenir le langage $\mathcal{L}_{\text{PEL}}$.

Définition 4.7 (Langage épistémique probabiliste $\mathcal{L}_{\text{PEL}}$)**.** *Pour un vocabulaire donné V, le langage $\mathcal{L}_{\text{PEL}}(V)$ de la logique épistémique probabiliste est défini par la grammaire suivante sous forme Backus-Naur :*

$$\phi ::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \Box_a\phi \mid \alpha_1\text{Pr}_a(\phi_1) + \cdots + \alpha_n\text{Pr}_a(\phi_n) \geq \beta$$

où $p \in V$, $a \in \mathfrak{A}$, $\alpha_1, \ldots, \alpha_k, \beta$ sont des nombres rationnels, et $n \geq 1$.

Ce langage permet d'écrire des formules mixtes telles que : $\text{K}_a(\text{Pr}_b(\phi) \geq \beta)$ ou $\text{Pr}_a(\text{K}_b\phi) \geq \beta$. De cette manière, on peut parler de connaissances des agents à propos des probabilités que les autres agents assignent à une proposition ϕ, ou des probabilités qu'ils assignent au fait qu'un autre agent sache une proposition. Notons que la formule $\text{Pr}_a(\phi) + \text{Pr}_b(\psi) \geq \beta$ n'est pas bien formée. En effet, il n'est pas possible de combiner les opérateurs Pr_a et Pr_b, qui ne portent pas sur les mêmes agents.

Intuitivement, $\text{Pr}_a\phi \geq \beta$ signifie que l'agent a assigne une probabilité d'au moins β à ce que la formule ϕ soit vraie. Les combinaisons linéaires permettent d'exprimer des jugements tel que : « l'agent a considère ϕ au moins deux fois plus probable que ψ » : $\text{Pr}(\phi) \geq 2 \times \text{Pr}(\psi)$ (cette formule étant une abréviation de cette formule-ci : $\text{Pr}(\phi) - 2 \times \text{Pr}(\psi) \geq 0$).

Remarque 4.8. *Par commodité, il nous arrivera d'abréger la combinaison linéaire d'une formule probabiliste, pour $a \in \mathfrak{A}$, $\alpha_1, \ldots, \alpha_n, \beta \in \mathbb{Q}$ et $n \in \mathbb{N}(n \geq 1)$, de la sorte :*

$$\alpha_1\text{Pr}_a(\phi_1) + \cdots + \alpha_n\text{Pr}_a(\phi_n) \geq \beta = \sum_{i=1}^{n} \alpha_i\text{Pr}_a(\phi_i) \geq \beta$$

De la même manière qu'il existe des axiomes pour les formules et les relations R (table 1.3), il en existe sur les probabilités. Ceux-ci permettent une cohérence naturelle à l'interprétation des formules. Ils sont d'abord présentés par Fagin [FHM88] puis repris par Kooi [Koo03]. Nous les présentons dans la table 4.1.

Tous ces axiomes semblent très naturels quant à la manipulation des probabilités. Décrivons brièvement ces propriétés dans le même ordre que la table 4.1 :

Non négativité : il ne peut pas y avoir de probabilités négatives ;

Probabilité de la vérité : la probabilité de la vérité ($\top$) se doit d'être égale à 1 ;

Additivité : l'additivité est préservée : l'addition de la probabilité d'une formule ϕ en conjonction avec ψ et de la même formule avec $\neg\psi$ se doit d'être la probabilité de ϕ. En effet, les deux formules sont incompatibles mais ont bien ϕ en commun, ce qui fait que la probabilité résultante est bien celle de ϕ ;

Axiome	Schéma
Logique probabiliste	
Non négativité	$\Pr_a(\phi) \geq 0$
Probabilité de la vérité	$\Pr_a(\top) = 1$
Additivité	$\Pr_a(\phi \wedge \psi) + \Pr_a(\phi \wedge \neg\psi) = \Pr_a(\phi)$
Équivalence	si $\phi \leftrightarrow \psi$ alors $\Pr_a(\phi) = \Pr_a(\psi)$
Inégalités linéaires	
Terme 0	$\sum_{i=1}^{n} \alpha_i \Pr(\phi_i) \geq \beta \leftrightarrow (\sum_{i=1}^{n} \alpha_i \Pr(\phi_i)) + 0 \cdot \Pr_a(\phi_{n+1}) \geq \beta$
Permutation	$\sum_{i=1}^{n} \alpha_i \Pr(\phi_i) \geq \beta \rightarrow \sum_{i=1}^{n} \alpha_{j_i} \Pr(\phi_{j_i}) \geq \beta$ où $j_1, \ldots, j_n$ est une permutation de $i, \ldots, n$
Addition	$\sum_{i=1}^{n} \alpha_i \Pr(\phi_i) \geq \beta \wedge \sum_{i=1}^{n} \alpha'_i \Pr(\phi_i) \geq \beta' \rightarrow \sum_{i=1}^{n} (\alpha_i + \alpha'_i) \Pr(\phi_i) \geq (\beta + \beta')$
Multiplication	$\sum_{i=1}^{n} \alpha_i \Pr(\phi_i) \geq \beta \leftrightarrow \sum_{i=1}^{n} d \times \alpha_i \Pr(\phi_i) \geq d \times \beta$ où $d > 0$
Dichotomie	$(\Pr_a \phi \geq \beta) \vee (\Pr_a \phi \leq \beta)$
Monotonie	$(\Pr_a \phi \geq \beta) \rightarrow (\Pr_a \phi > \beta')$ où $\beta > \beta'$

TABLE 4.1 – Axiomes permettant de constuire le système axiomatique à partir de la logique $\mathcal{L}_{\text{PEL}}$

Équivalence : si deux formules sont équivalentes, elles doivent avoir la même probabilité.

Les probabilités pouvent être manipulées par des inégalités linéaires, d'autres axiomes s'ajoutent.

Terme 0 : L'ajout d'un dernier terme nul à une inégalité linéaire ne doit pas changer le résultat de l'inégalité ;

Permutation : l'ordre des formules ϕ_i ne doit pas avoir d'importance dans la somme ;

Addition : il est possible de factoriser les cœfficients de la conjonction de deux

inégalités portant sur le même ensemble de formules ϕ_i, il en va de même avec les seuils qui peuvent être vus comme un facteur commun ;

Multiplication : multiplier tous les cœfficients et le seuil par un facteur β ne change pas la valeur de vérité de la formule ;

Dichotomie : la probabilité d'une formule est forcèment soit supérieure ou égale à un seuil β soit inférieure ;

Monotonie : si la probabilité d'une formule ϕ est supérieure ou égale à un seuil β, alors elle sera forcément supérieure à un seuil β', qui est inférieur à β ($\Pr_a\phi \geq \beta > \beta'$).

Afin de pouvoir interpréter les formules probabilistes dans le contexte d'une structure de Kripke, il faut définir la sémantique de l'opérateur de probabilité Pr. La sémantique est la même que pour la logique épistémique (déf. 1.8), augmentée de celle du nouvel opérateur, qui utilise les probabilités décrites par les ensembles de lois de probabilité discrètes μ_a.

Définition 4.9 (Sémantique de $\mathcal{L}_{\text{PEL}}$)**.** *La sémantique du langage $\mathcal{L}_{\text{PEL}}(V)$ pour un vocabulaire V est définie inductivement pour une structure de Kripke probabiliste $\mathcal{M} = \langle W, R, \mu, \text{val}\rangle$ définie aussi sur le vocabulaire V, avec $w \in W$ ($n \in \mathbb{N}^*$, $\alpha_1, \ldots, \alpha_n, \beta \in \mathbb{Q}$) et étend le langage $\mathcal{L}_{\text{EL}}(V)$ (déf. 1.8) :*

$$\mathcal{M}, w \models \alpha_1\Pr_a(\phi_1) + \cdots + \alpha_n\Pr_a(\phi_n) \geq \beta \text{ ssi } \sum_{i=1}^{n} \alpha_i \left(\sum_{w' : \mathcal{M}, w' \models \phi_i} \mu_a(w)(w') \right) \geq \beta$$

Expliquons l'idée en prenant un cas simple. En prenant $n = 1$ et $\alpha_1 = 1$, nous avons :

$$\mathcal{M}, w \models \Pr_a(\phi_1) \geq \beta \text{ ssi } \sum_{w' : \mathcal{M}, w' \models \phi_1} \mu_a(w)(w') \geq \beta$$

Il s'agit alors de vérifier que la somme des probabilités sortantes du monde w vers chaque monde w' vérifiant la sous-formule ϕ_1 est bien supérieure ou égale au seuil β. La mécanisme est le même pour une combinaison linéaire de formules, il faut simplement additionner les probabilités résultantes de toutes les sous-formules ϕ_i.

4.1.2 Logique épistémique dynamique probabiliste

Après avoir vu les structures de Kripke probabilistes, nous allons définir des *modèles d'événements* pour PDEL, comme nous avons pu le faire pour DEL. Le

modèle d'événements est basé sur une *frame* probabiliste et ajoute une notion de préconditions conditionnelles via un ensemble de formules Φ et des probabilités sur les préconditions. Nous basons cette définition sur celle de la littérature [vBGK09].

Remarque 4.10. *Afin de facilement faire la distinction entre un modèle d'événements de DEL et un modèle d'événements de PDEL, nous utiliserons le terme* modèle d'événements *seulement pour DEL et utiliserons le terme* modèle conditionnel *pour le modèle d'événements probabiliste de PDEL.*

Définition 4.11. Un modèle conditionnel *est un tuple* $\langle \mathcal{F}^{\mathcal{E}}, \Phi, \mu_{\text{pre}} \rangle$ *où :*

- $\mathcal{F}^{\mathcal{E}}$ *est une frame probabiliste (déf. 4.1), dont on note les élements* E, $R^{\mathcal{E}}$, $\mu^{\mathcal{E}}$ *;*
- Φ *est un ensemble de formules épistémiques mutuellement incohérentes appelées* préconditions conditionnelles *;*
- μ_{pre} *assigne à chaque précondition conditionnelle* $\phi \in \Phi$ *une loi de probabilité discrète sur* E*. On écrit* $\mu_{\text{pre}}(\phi)(e)$ *la probabilité de l'événement* e *étant donné* $\phi \in \Phi$*.*

Remarque 4.12. *Dans la définition classique de PDEL, les préconditions conditionnelles sont notées* pre*, c'est à dire strictement de la même manière que les préconditions de DEL (voir déf. 1.24) alors que ce sont des objets de nature complètement différente. Pour éviter toute confusion avec la définition de DEL, nous noterons donc les préconditions conditionnelles* μ_{pre} *au lieu de* pre*.*

On notera pour plus de simplicité $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mu^{\mathcal{E}}, \Phi, \mu_{\text{pre}} \rangle$, comme nous avons pu le faire précédemment. Notons que, conformément à la définition de la littérature, la notion de postcondition présente dans les modèles d'événements de DEL (déf. 1.24) n'est pas présente dans ce modèle conditionnel.

La définition doit être comprise comme suit. Une partie du modèle consiste en la spécification des « probabilités d'occurrence » d'un processus qui fait que des événements se produisent avec certaines probabilités, en fonction d'un ensemble de conditions Φ. Un tel processus est capturé par la fonction μ_{pre}. Ces processus peuvent avoir la forme : **Si** ϕ, **alors faire** l'événement atomique $e \in E$ avec probabilité β ($\mu_{\text{pre}}(\phi)(e) = \beta$).

Notons que Φ est un ensemble de formules épistémiques mutuellement incohérentes, ce qui signifie que pour un monde w d'une structure $\mathcal{M}$, il ne peut y avoir qu'une seule et unique formule $\phi \in \Phi$ telle que $\langle \mathcal{M}, w \rangle \models \phi$.

La fonction $\mu_a^{\mathcal{E}}$ abstrait les « probabilités d'observation ». La probabilité $\mu_a^{\mathcal{E}}(e)(e')$ est la probabilité qu'attribue l'agent a au fait que l'événement e' puisse s'appliquer à la place de l'événement e.

Il est alors possible, à l'instar des structures non probabilisées, de mettre à jour une structure de Kripke probabilisée via un modèle conditionnel, de la manière suivante :

Définition 4.13 (*Product update* probabiliste)**.** *Soit* $\mathcal{M} = \langle W, R, \mu, \text{val}\rangle$ *une structure de Kripke probabiliste et* $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mu^{\mathcal{E}}, \Phi, \mu_{\text{pre}}\rangle$ *un modèle conditionnel. Pour* $w \in W$ *et* $e \in E$*, nous écrivons* $\mu_{\text{pre}}(w)(e)$ *pour la valeur de* $\mu_{\text{pre}}(\phi)(e)$ *où* ϕ *est l'élément de* Φ *qui est satisfait dans* $\langle \mathcal{M}, w\rangle$*. Si un tel* ϕ *n'existe pas,* $\mu_{\text{pre}}(w)(e) = 0$*. Le* product update $\langle \mathcal{M}, w\rangle \otimes \langle E, e\rangle$ *est la structure de Kripke probabiliste pointée* $\langle \mathcal{M}^{\otimes}, (w, e)\rangle$ *où* $\mathcal{M}^{\otimes} = \langle W^{\otimes}, R^{\otimes}, \mu^{\otimes}, \text{val}^{\otimes}\rangle$ *avec :*

- $W^{\otimes} = \{(w, e) \in W \times E \mid \mu_{\text{pre}}(w)(e) > 0\}$ *;*
- $(w, e)\ R_i^{\otimes}\ (w', e')$ ssi wR_iw' et $eR_i^{\mathcal{E}}e'$*, pour tout* $(w, w') \in W^2$ *et* $(e, e') \in E^2$ *(comme dans la définition du* product update *classique (déf. 1.28) ;*
- $\mu_a^{\otimes}((w, e), (w', e')) = \dfrac{\mu_a(w)(w') \cdot \mu_{\text{pre}}(w')(e') \cdot \mu_a^{\mathcal{E}}(e)(e')}{\sum_{w'' \in W, e'' \in E} \mu_a(w)(w'') \cdot \mu_{\text{pre}}(w'')(e'') \cdot \mu_a^{\mathcal{E}}(e)(e'')}$
  si le dénominateur est non nul et 0 sinon ;
- $\text{val}^{\otimes}((w, e)) = \text{val}(w)$.

Ainsi, le nouvel espace d'états après la mise à jour est l'ensemble des paires (w, e) tel que l'événement atomique e a une probabilité positive d'être appliqué dans le monde w (comme spécifié par μ_{pre}). La relation d'indistinguabilité est définie comme précédemment. La partie la plus intéressante est l'obtention des nouvelles probabilités. La nouvelle probabilité $\mu^{\otimes}((w, e))((w', e'))$ est le produit arithmétique des probabilités pré-établies pour w', la probabilité que e' s'applique à w', et la probabilité que a observe e'. Pour obtenir une probabilité correcte, celle-ci est ensuite normalisée.

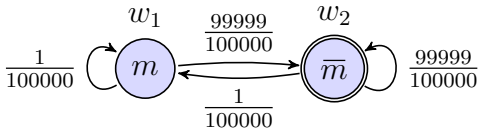


FIGURE 4.1 – Exemple d'une structure de Kripke probabiliste $\mathcal{M}$ (ex. 4.14)

Exemple 4.14. *L'exemple illustratif de PDEL est celui de l'hypocondriaque qui pense être malade [vBGK09]. A priori, il sait qu'il a une chance sur* 100000 *d'être malade (variable propositionnelle* m*). On obtient la structure de Kripke probabilisée (qui par ailleurs est a-uniforme et a-consistante, déf. 4.5 et 4.4) présentée à la figure 4.1.*

Voici l'événement que nous allons modéliser pour modifier les connaissances de l'agent : l'hypocondriaque va s'examiner afin d'en apprendre plus face à sa potentielle maladie.

Objectivement, s'il est vraiment malade, alors il aura les symptômes (événement atomique e_1) avec 97% de chance et s'il ne l'est pas, il n'aura pas les symptômes (événement atomique e_2). Subjectivement, comme il ne croit pas que l'examen va être concluant, dans tous les cas (qu'il ait les symptômes ou non) il considèrera qu'il a 50% de chances d'avoir les symptômes réels (et donc 50% de chances de ne pas les avoir). Ainsi, on obtient le modèle conditionnel présenté à la figure 4.2. Ici, $\mu_{\text{pre}}(m)(e_1) = 0.97$, $\mu_{\text{pre}}(m)(e_2) = 0.03$, $\mu_{\text{pre}}(\overline{m})(e_2) = 1$.

Après l'application du product update*, nous obtenons la structure de Kripke probabiliste (toujours a-uniforme et a-consistante) présentée à la figure 4.3. Les probabilités sont bien mises à jour et nous pouvons voir que les connaissances de l'agent sont bien modifiées.*

L'exemple illustre bien les différentes probabilités utilisées :

- *les probabilités pré-établies, c'est à dire les probabilités présentes sur la structure de Kripke « précédente » (ici, 1 sur 100000 et 99999 sur 100000 sur la figure 4.1 ;*
- *les probabilités que l'agent attribue à l'apparition des événements atomiques (ici, 0.5 sur chaque événement atomique) ;*
- *les probabilités conditionnelles qui sont les probabilités objectives qui s'appliquent sur les événements atomiques en fonction de s'il est malade ou non.*

La probabilité de l'hypocondriaque d'être malade est toujours 1 sur 100000. Depuis cette observation, il ne sait pas plus s'il est malade ou non. Néanmoins, une information a été ajoutée dans les connaissances de l'agent : l'événement e_1 modélisant le fait qu'il ait vraiment des symptômes, il n'a que de très faibles chances d'avoir les symptômes, bien qu'il considère qu'il est encore moins probable qu'il soit malade et qu'il n'ait pas les symptômes (monde $(w_1 e_2)$).

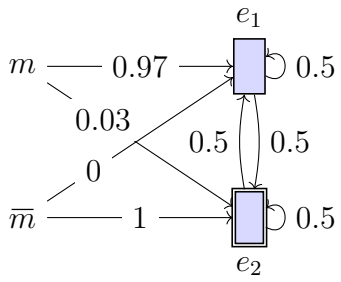


FIGURE 4.2 – Exemple de modèle d'événement probabiliste $\mathcal{E}$ (ex. 4.14)

Ensuite, notons que des formules probabilistes dynamiques sont définies [FH94], de sorte à avoir le langage suivant, extension de $\mathcal{L}_{\text{PEL}}$ (déf. 4.7) :

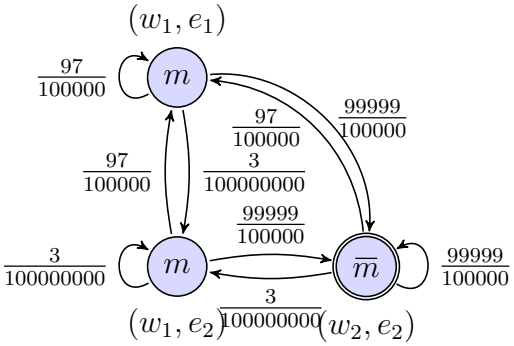


FIGURE 4.3 – Exemple de *product update* entre les structures probabilistes des figures 4.1 et 4.2

Définition 4.15 (Langage $\mathcal{L}_{\text{PDEL}}$)**.** *Étant donné un vocabulaire V, le langage $\mathcal{L}_{\text{PDEL}}$ de la logique épistémique dynamique probabiliste est défini par la grammaire suivante (sous forme BNF) :*

$$\phi \coloneqq p \mid \neg\phi \mid \phi \wedge \phi \mid \Box_a \phi \mid [\mathcal{E}, e]\phi \mid \alpha_1 \text{Pr}_a(\phi_1) + \ldots \alpha_n \text{Pr}_a(\phi_n) \geq \beta$$

où $p \in V$, $a \in \mathfrak{A}$, $\alpha_1, \ldots, \alpha_n, \beta \in \mathbb{Q}$ avec $n \geq 1$ et $\mathcal{E}$ est un modèle conditionnel pointé par $e \in E$, son ensemble d'événements.

La sémantique qui permet d'interpréter ce langage via une structure de Kripke est la suivante :

Définition 4.16. *La sémantique du langage $\mathcal{L}_{\text{PDEL}}$ est identique à celle de $\mathcal{L}_{\text{PEL}}$ (déf. 4.9), en y ajoutant la sémantique de l'opérateur $[\mathcal{E}, e]$:*

$$\langle \mathcal{M}, w \rangle \models [\mathcal{E}, e]\phi \text{ ssi } \mu_{\text{pre}}(w)(e) > 0 \implies \langle \mathcal{M} \otimes \mathcal{E}, (w, e) \rangle \models \phi$$

Reformulons cette cette condition avec des mots : après l'application d'un modèle conditionnel pointé $\langle \mathcal{E}, e \rangle$ sur une structure de Kripke pointée $\langle \mathcal{M}, w \rangle$, ϕ est vraie si et seulement si quand la probabilité conditionnelle d'appliquer e sur w est non nulle (c'est-à-dire que l'événement est applicable), alors le monde résultant (w, e) satisfait ϕ.

Avant de passer à la suite, faisons un point sur le passage d'une structure de Kripke classique à une structure de Kripke probabilisée. En effet, il semble plutôt naturel de pouvoir passer de l'une à l'autre. Évidemment, enlever des probabilités enlèverait de l'information à la structure, tandis qu'ajouter des probabilités à une structure non probabilisée enrichirait celle-ci. Néanmoins, comment décider des

nouvelles probabilités à attribuer ? L'idée la plus simple et naturelle serait de considérer pour un agent $a \in \mathfrak{A}$ et pour tout monde $w \in W$ une distribution de probabilité $\mu_a(w)$ de sorte à ce que pour chaque monde w' tel que wR_aw', les probabilités $\mu(w)(w')$ soient identiques. C'est ce qui correspond à la définition de la a-uniformité (déf. 4.4). Cette définition des probabilités de sorte que chaque agent considère les mondes indistinguables comme équiprobables a le mérite d'être simple et permet de quantifier plus finement la connaissance des agents : l'agent peut ne pas savoir que ϕ ($\neg \mathrm{K}\phi$) mais pour autant le considérer comme très probable : ($\Pr(\phi) \geq 0.8$).

Toutes les structures de Kripke non probabilistes ne sont peut-être pas traduisibles facilement en structure probabiliste avec cette simplification d'a-uniformité. En effet, cela dépend de ce que l'on souhaite représenter comme information.

Néanmoins dans le cadre de jeux stochastiques, comme des jeux de cartes ou de dés, il est tout à fait envisageable de modéliser une structure probabilisée simplement grâce à la notion de a-uniformité.

Exemple 4.17. *Prenons l'exemple d'un lancer de dé avec 6 faces équiprobables. Soit le vocabulaire $V = \{v_1, \ldots, v_6\}$ désignant les valeurs possibles du dé. Un agent a qui ne voit pas ce qui se passe ne saura pas le résultat ($\neg \bigvee_{v \in V} \mathrm{K}_a v$), mais pourra raisonner sur les probabilités d'avoir obtenu le nombre 6 ($\Pr_a(v_6) \geq 1/6$), ou un nombre pair ($\sum_{i \in \{2,4,6\}} \Pr_a(v_i) \geq 1/2$).*

4.2 Variante dénormalisée de PDEL

Dans la suite, nous allons modifier les définitions de la littérature que nous avons présentées. Les quelques différences que nous suggérons nous permettront des manipulations simplifées dans le cadre symbolique. Ensuite, nous allons comparer les anciennes et nouvelles définitions, de sorte à montrer qu'elles sont équivalentes.

Dans cette section, trois modifications principales vont être appliquées à PDEL :

- nous allons dénormaliser toutes les *frames*, qu'elles soient utilisées dans les structures de Kripke ou les modèles conditionnels ;
- nous allons ajouter la notion de postconditions présente dans la définition du modèle d'événements de DEL (déf. 1.24) aux modèles conditionnels (déf. 4.1) présentés dans la section précédente ;
- nous allons modifier la définition des préconditions conditionnelles (notées pour rappel Φ) de la définition des modèles conditionnels (déf. 4.11), de sorte qu'elles ne soient plus nécessairement mutuellement incohérentes.

Ces modifications nécessiteront bien évidemment des définitions différentes des nouvelles structures et modèles, mais aussi des sémantiques et des mises à jour des connaissances.

Tout d'abord, nous allons définir ce que sont des *frames* « probabilistes » dénormalisées. L'idée principale est simple : nous ne souhaitons plus manipuler des lois de probabilité discrètes, dont la somme est égale à 1. Ainsi, à la place d'utiliser des *lois de probabilité discrètes*, nous utiliserons ce que nous nommerons des *loteries*. S'afranchir de cette contrainte dans les *frames* des structures et des modèles permet de considérer la normalisation d'une manière différente. En effet, les *product updates* seront eux aussi dénormalisés et n'auront plus besoin de normaliser les probabilités. C'est néanmoins lors du *model checking* qu'il faudra normaliser les loteries pour pouvoir comparer les informations numériques au seuil β d'une formule probabiliste (voir la définition d'une formule probabiliste, déf. 4.7).

4.2.1 Structure de Kripke dénormalisée

Nous allons commencer par définir une version « dénormalisée » d'une structure de Kripke. Nous allons discuter des avantages de ces définitions plus tard, quand nous l'exploiterons, notamment dans le chapitre 5 traitant de la représentation symbolique de PEL. Comme nous le disions, notre pendant dénormalisé d'une loi de probabilité discrète est ce que l'on nomme une *loterie* définie ci-dessous (terme que l'on peut retrouver dans l'article suivant : [vES14]).

Définition 4.18 (X−*loterie*)**.** *Une X−loterie L est une fonction allant d'un domaine (ensemble dénombrable) X à l'ensemble des rationnels positifs, c'est-à-dire $L\colon X \mapsto \mathbb{Q}_+$.*

Notons que notre définition de support (déf. 3.13) est valable sur toute fonction, et par conséquent aussi sur une loterie, notamment quand l'ensemble X est un ensemble de sous-ensembles (par exemple, $X = 2^V$ avec V un vocabulaire ou ensemble de variables propositionnelles).

Une loterie L sera dite « normalisée » si $\sum_{x\in X} L(x) = 1$. Dans ce cas, une loterie sera alors une loi de probabilité discrète. Spécifions ce que nous appelons normalisation d'une loterie L via la fonction N().

Définition 4.19 (Normalisation)**.** *Soit L une X−loterie. La* normalisation *de L, notée $\mathrm{N}(L)$, est la X-loterie associant à chaque $x \in X$ la valeur $\frac{L(x)}{\sum_{x'\in X} L(x')}$ si $\mathrm{supp}(L) \neq \emptyset$ (i.e. le dénominateur n'est pas nul), et la valeur 0 sinon.*

Grâce à la notion de loterie, nous allons pouvoir définir une notion de *frame* dénormalisée. La définition est identique à la définition d'une *frame* probabiliste (déf. 4.1), modulo la modification de μ_a, qui ne sera non plus un ensemble de loi de probabilité discrète, mais un ensemble de loteries.

Définition 4.20. *Une* frame *probabiliste (dénormalisée) est une structure* $\mathcal{F} = \langle W, R, \mu \rangle$ *où :*

- W, R *sont définis comme précédemment dans une* frame *classique (déf. 1.4) ;*
- μ *associe à chaque* $a \in \mathfrak{A}$ *et chaque monde* $w \in W$ *une* W*-loterie* $\mu_a(w)$, *i.e. une fonction* $W \mapsto \mathbb{Q}_+$.

Remarque 4.21. *Remarquons que naturellement, une structure de Kripke dénormalisée sera alors construite à partir d'une* frame *dénormalisée* $\mathcal{F}$ *et d'une fonction* val *:* $\mathcal{M} = \langle \mathcal{F}, \mathrm{val} \rangle$. *Par la même occasion et par extension, nous utiliserons le terme de* structure de Kripke dénormalisée *quand elle sera composée d'une* frame *dénormalisée.*

Remarquons par ailleurs qu'une structure de Kripke dénormalisée est une généralisation d'une structure de Kripke classique ; en effet, quand les loteries sont normalisées, alors il s'agit d'une structure de Kripke classique.

Bien évidemment la sémantique de $\mathcal{L}_{\mathrm{PEL}}$ (déf. 4.7) change et devient plus complexe. En effet, pour que la vérification de modèle vis-à-vis d'une probabilité β ait bien lieu de sorte à comparer les valeurs présentes dans les loteries de W, il est nécessaire de normaliser la loterie. Afin de différencier les deux sémantiques, nous noterons $\models_d$ dans le cas dénormalisé.

Définition 4.22 (Sémantique de PEL (sur les structures dénormalisées))**.** *Soit* $\langle \mathcal{M}, w \rangle$ *une structure de Kripke dénormalisée pointée définie sur un vocabulaire* V. *Nous définissons la sémantique de PEL sur les structures de Kripke dénormalisées récursivement comme suit, où* $p \in V$, $n \geq 1$, $\phi, \psi, \phi_1, \ldots, \phi_n \in \mathcal{L}_{\mathrm{PEL}}$, $a \in \mathfrak{A}$, *et* $\alpha_1, \ldots, \alpha_n, \beta \in \mathbb{Q}$ *:*

$$
\begin{array}{lll}
\langle \mathcal{M}, w \rangle \models_d p & \textit{ssi} & p \in \mathrm{val}(w) \\
\langle \mathcal{M}, w \rangle \models_d \neg\phi & \textit{ssi} & \langle \mathcal{M}, w \rangle \not\models_d \phi \\
\langle \mathcal{M}, w \rangle \models_d \phi \wedge \psi & \textit{ssi} & \langle \mathcal{M}, w \rangle \models_d \phi \textit{ et } \langle \mathcal{M}, w \rangle \models_d \psi \\
\langle \mathcal{M}, w \rangle \models_d \Box_a \phi & \textit{ssi} & \langle \mathcal{M}, w' \rangle \models_d \phi \textit{ pour tout } w' \textit{ tel que } (w, w') \in R_a
\end{array}
$$

$$
\langle \mathcal{M}, w \rangle \models_d \alpha_1 \mathrm{Pr}_a(\phi_1) + \cdots + \alpha_n \mathrm{Pr}_a(\phi_n) \geq \beta \textit{ ssi}
$$

$$
\sum_{i=1}^{n} \alpha_i \left(\sum_{w' : \langle \mathcal{M}, w' \rangle \models_d \phi_i} \mathrm{N}(\mu_a(w))(w') \right) \geq \beta
$$

Nous notons $\mathrm{N}(\mathcal{M})$ la normalisation d'une structure de Kripke $\mathcal{M} = \langle W, R, \mu, \mathrm{val} \rangle$ (normalisée, ou pas), définie comme $\mathrm{N}(\mathcal{M}) = \langle W, R, \mathrm{N}(\mu), \mathrm{val} \rangle$ où $\mathrm{N}(\mu)$ est la normalisation de chaque loi de probabilité discrète de $\mu : a \in \mathfrak{A} \to (w \in W \to \mathrm{N}(\mu_a(w)))$.

Montrons maintenant l'équivalence du *model checking* sur une structure de Kripke dénormalisée avec le *model checking* sur sa version normalisée.

Proposition 4.23. *Soit $\mathcal{M}$ une structure de Kripke dénormalisée $\langle W, R, \mu, \text{val}\rangle$.*

$$\forall w \in W \text{ et } \forall \phi \in \mathcal{L}_{\text{PEL}}, \text{ nous avons } \langle \mathcal{M}, w\rangle \models_d \phi \iff \langle \text{N}(\mathcal{M}), w\rangle \models \phi$$

Démonstration. Soient $\mathcal{M}^d = \langle W, R, \mu^d, \text{val}\rangle$ une structure de Kripke dénormalisée et $\text{N}(\mathcal{M}^d) = \langle W, R, \text{N}(\mu^d), \text{val}\rangle$.

Si ϕ ne contient pas l'opérateur Pr, il n'est aucunement mention de μ et les sémantiques sont identiques. Ainsi par induction, nous avons aussi $\langle \mathcal{M}^d, w\rangle \models_d \phi \iff \langle \text{N}(\mathcal{M}^d), w\rangle \models \phi$.

Pour le cas probabiliste, soit $\phi = \alpha_1 \text{Pr}_a(\phi_1) + \cdots + \alpha_n \text{Pr}_a(\phi_n) \geq \beta$.

Par la définition de la sémantique de PEL dénormalisée, nous avons :

$$\langle \mathcal{M}^d, w\rangle \models_d \phi \text{ ssi } \sum_{i=1}^{k} \alpha_i \left(\sum_{w' : \langle \mathcal{M}^d, w'\rangle \models_d \phi_i} \text{N}(\mu_a^d(w))(w') \right) \geq \beta$$

Or, par la définition de la sémantique de PEL et pour une structure de Kripke $\mathcal{M}$, nous avons classiquement :

$$\langle \mathcal{M}, w\rangle \models \phi \text{ ssi } \sum_{i=1}^{k} \alpha_i \left(\sum_{w' : \langle \mathcal{M}, w'\rangle \models \phi_i} \mu_a(w)(w') \right) \geq \beta$$

En réécrivant $\mathcal{M}$ comme étant $\text{N}(\mathcal{M}^d)$ avec $\text{N}(\mu^d)$ comme fonction μ, et par induction ($\{w' : \langle \text{N}(\mathcal{M}^d), w'\rangle \models \phi_i\} = \{w' : \langle \mathcal{M}^d, w'\rangle \models_d \phi_i\}$) nous avons :

$$\begin{aligned}
\langle \text{N}(\mathcal{M}^d), w\rangle \models \phi \text{ ssi } & \sum_{i=1}^{k} \alpha_i \left(\sum_{w' : \langle \text{N}(\mathcal{M}^d), w'\rangle \models \phi_i} \text{N}(\mu_a^d(w))(w') \right) \geq \beta \\
\text{ssi } & \sum_{i=1}^{k} \alpha_i \left(\sum_{w' : \langle \mathcal{M}^d, w'\rangle \models_d \phi_i} \mu_a^d(w)(w') \right) \geq \beta \\
\text{ssi } & \langle \mathcal{M}^d, w\rangle \models_d \phi
\end{aligned}$$

D'où $\langle \mathcal{M}^d, w\rangle \models_d \phi \iff \langle \text{N}(\mathcal{M}^d), w\rangle \models \phi$ pour une formule qui contient l'opérateur Pr.

□

En particulier, si $\mathcal{M}$ est une structure de Kripke normalisée, on a $\mathcal{M} = \text{N}(\mathcal{M})$, donc $\langle \mathcal{M}, w\rangle \models_d \phi \iff \langle \mathcal{M}, w\rangle \models \phi$. Il n'y a donc pas d'ambiguïté nécessitant de différencier $\models_d$ et $\models$. Dans la suite, on utilisera toujours $\models$.

4.2.2 Le modèle conditionnel dénormalisé

Après avoir défini la version dénormalisée d'une structure de Kripke, nous allons pouvoir faire de même avec un modèle conditionnel.

Nous reprenons la définition d'un modèle conditionnel (déf.4.11) et dénormalisons les lois de probabilité discrètes ($\mu^{\mathcal{E}}$), mais aussi les probabilités conditionnelles (μ_{pre}). En outre, nous ajoutons une fonction de postcondition post au modèle. Par ailleurs, une modification supplémentaire a lieu : Φ n'est plus « un ensemble de formules épistémiques mutuellement incohérentes », mais simplement « un ensemble de formules épistémiques ».

Définition 4.24. *Un modèle conditionnel (dénormalisé) $\mathcal{E}$ défini pour un vocabulaire V est une structure $\langle \mathcal{F}^{\mathcal{E}}, \Phi, \mu_{\text{pre}}, \text{post} \rangle$ où :*

- $\mathcal{F}^{\mathcal{E}}$ *est une* frame *probabiliste dénormalisée, avec les élements notés E, $R^{\mathcal{E}}$, $\mu^{\mathcal{E}}$ (def 4.20) ;*
- Φ *est un ensemble de formules épistémiques appelé* préconditions conditionnelles *;*
- μ_{pre} *assigne à chaque précondition $\phi \in \Phi$ une $E-$loterie. On écrit $\mu_{\text{pre}}(\phi)(e)$ la probabilité de e étant donné $\phi \in \Phi$;*
- post *: fonction de postcondition, $V \times E \to \mathcal{L}_{\text{prop}}(V)$*

Un modèle conditionnel pointé *est un couple $\langle \mathcal{E}, e \rangle$ d'un modèle conditionnel et d'un événement $e \in E$, considéré comme l'événement qui aura lieu.*

Définition 4.25. *Un modèle conditionnel dont les préconditions conditionnelles sont mutuellement incohérentes entre elles est dit* à préconditions séparées.

Remarque 4.26. *Les modèles conditionnels dénormalisés sont une généralisation des modèles conditionnels classique de PDEL. En effet, un modèle conditionnel classique est un modèle conditionnel dénormalisé à préconditions séparées, dont les loteries sont normalisées et qui n'a pas de postconditions ($\forall (p, e) \in V \times E \colon \text{post}(e)(p) = p$).*

Remarque 4.27. *Nous conservons ici la définition de la fonction de postcondition allant vers des formules propositionnelles, comme dans la définition de DEL que nous avons présenté (identique à celle de Gattinger). Notons que cette formule pourrait être épistémique ou probabiliste. Ici, il est plus simple de conserver cette définition. Néanmoins, il est en effet possible de simuler des postconditions épistémiques en utilisant une succession de modèles d'événements avec postconditions propositionnelles. Pour plus d'information, voir la thèse de Tristan Charrier [Cha18].*

Avec la définition des structures de Kripke et modèles conditionnels dénormalisés, nous pouvons ainsi définir la mise à jour probabiliste dénormalisée des connaissances, basée sur la définition classique du *product update* (déf. 4.13). Pour pallier la modification de Φ, l'interprétation de $\mu_{\text{pre}}(w)(e)$ diffère de la version originale.

Définition 4.28 (*Product update* probabiliste)**.** *Soit $\langle \mathcal{M}, w\rangle$ une structure de Kripke pointée et $\langle \mathcal{E}, e\rangle$ un modèle conditionnel pointé pour un même vocabulaire V. Le* product update *de $\langle \mathcal{M}, w\rangle$ par $\langle \mathcal{E}, e\rangle$ noté $\langle \mathcal{M}, w\rangle \otimes \langle \mathcal{E}, e\rangle$ est la structure de Kripke pointée $\langle\langle W^{\otimes}, R^{\otimes}, \mu^{\otimes}, \text{val}^{\otimes}\rangle, (w, e)\rangle$ avec :*

- $W^{\otimes} = \{(w, e) \in W \times E \mid \mu_{\text{pre}}(w)(e) > 0\}$;
- $(w_1, e_1) R_a^{\otimes} (w_2, e_2)$ *ssi* $(w_1, w_2) \in R_a$ *et* $(e_1, e_2) \in R_a^{\mathcal{E}}$;
- $\mu_a^{\otimes}((w_1, e_1))((w_2, e_2)) = \mu_a(w_1)(w_2) \times \mu_{\text{pre}}(w_2)(e_2) \times \mu_a^{\mathcal{E}}(e_1)(e_2)$;
- $\text{val}^{\otimes}((w, e)) = \{p \in V \mid \langle \mathcal{M}, w\rangle \models \text{post}(p)(e)\}$.

Nous écrivons $\mu_{\text{pre}}(w)(e)$ pour la valeur $\frac{\sum_{\phi \in \Phi \mid \langle \mathcal{M}, w\rangle \models \phi} \mu_{\text{pre}}(\phi)(e)}{\sum_{\phi \in \Phi \mid \langle \mathcal{M}, w\rangle \models \phi} \sum_{e' \in E} \mu_{\text{pre}}(\phi)(e')}$ s'il existe $\phi \in \Phi$ tel que $\langle \mathcal{M}, w\rangle \models \phi$, 0 sinon.

Remarquons une chose qui est importante : contrairement au *product update* classique, il n'y a pas de normalisation après multiplication. Et pour cause, nous souhaitons conserver une structure dénormalisée, il n'y a donc aucune raison de normaliser. Par ailleurs, les calculs des nouveaux ensembles de mondes, des relations et des nouvelles valuations se font strictement de la même manière.

Remarque 4.29. *Explicitons l'utilisation du nouveau calcul de $\mu_{\text{pre}}(w)(e)$. Classiquement, comme les formules sont mutuellement incohérentes il ne pouvait y avoir qu'une seule formule ϕ de Φ telle que $\langle \mathcal{M}, w\rangle \models \phi$. Maintenant, comme il est possible que plusieurs formules de Φ s'appliquent au monde w, il faut pouvoir agréger les valeurs des loteries. Nous sommons simplement toutes ces valeurs. Néanmoins, comme il faut pouvoir garder une cohérence entre les loteries, nous normalisons cette valeur par les sommes des valeurs des loteries de chaque formule.*

Notons que si les loteries sont normalisées, alors nous avons :

$$\frac{\sum_{\phi \in \Phi \mid \langle \mathcal{M}, w\rangle \models \phi} \mu_{\text{pre}}(\phi)(e)}{\sum_{\phi \in \Phi \mid \langle \mathcal{M}, w\rangle \models \phi} \sum_{e' \in E} \mu_{\text{pre}}(\phi)(e')} = \frac{\sum_{\phi \in \Phi \mid \langle \mathcal{M}, w\rangle \models \phi} \mu_{\text{pre}}(\phi)(e)}{\mid \{\phi \in \Phi \mid \langle \mathcal{M}, w\rangle \models \phi\} \mid}$$

En effet, la somme des valeurs d'une loterie normalisée est 1.

Remarque 4.30. *Si Φ est un ensemble de formules épistémiques mutuellement incohérentes, alors nous avons*

$$\frac{\sum\limits_{\phi\in\Phi|\langle\mathcal{M},w\rangle\models\phi}\mu_{\text{pre}}(\phi)(e)}{\sum\limits_{\phi\in\Phi|\langle\mathcal{M},w\rangle\models\phi}\sum\limits_{e'\in E}\mu_{\text{pre}}(\phi)(e')} = \frac{\mu_{\text{pre}}(\psi)(e)}{\sum\limits_{e'\in E}\mu_{\text{pre}}(\psi)(e')}$$

où ψ est l'unique $\psi \in \Phi$ tel que $\langle\mathcal{M}, w\rangle \models \psi$.

De plus, si les loteries sont normalisées, nous avons donc la formule originelle de PDEL : $\frac{\sum\limits_{\phi\in\Phi|\langle\mathcal{M},w\rangle\models\phi}\mu_{\text{pre}}(\phi)(e)}{\sum\limits_{\phi\in\Phi|\langle\mathcal{M},w\rangle\models\phi}\sum\limits_{e'\in E}\mu_{\text{pre}}(\phi)(e')} = \mu_{\text{pre}}(\phi)(e)$.

4.2.3 Équivalence des modèles conditionnels

Nous avons montré que le *model checking* fonctionnait bien avec des structures dénormalisées, mais qu'en est-il pour le *product update* ? Montrons donc que le *product update* dénormalisé est équivalent au *product update* normalisé, suivant les définitions classiques de PDEL.

Définissons la normalisation d'un modèle conditionnel à préconditions séparées en un modèle conditionnel classique. Ainsi, nous avons naturellement des préconditions conditionnelles incohérentes et aucune postcondition, nécessaires à la définition d'un modèle conditionnel classique.

Définition 4.31 (Normalisation d'un modèle conditionnel)**.** *La normalisation d'un modèle conditionnel à préconditions séparées $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mu^{\mathcal{E}}, \Phi, \mu_{\text{pre}}, \text{post}\rangle$ est définie comme $\text{N}(\mathcal{E}) = \langle E, R^{\mathcal{E}}, \text{N}(\mu^{\mathcal{E}}), \Phi, \text{N}(\mu_{\text{pre}})\rangle$ où :*

- $\text{N}(\mu^{\mathcal{E}})$ *est la normalisation de chaque loterie de $\mu^{\mathcal{E}}$: $a \in \mathfrak{A} \to (e \in E \to \text{N}(\mu^{\mathcal{E}}_a(e)))$;*
- $\text{N}(\mu_{\text{pre}})$ *est la normalisation de chaque loterie de μ_{pre} : $\phi \in \Phi \to \text{N}(\mu_{\text{pre}}(\phi))$.*

Comparons tout d'abord des loteries par la somme de leurs valeurs.

Définition 4.32 (Loteries à sommes égales)**.** *Deux X-loteries L_1 et L_2 sont dites à* sommes égales *si :*

$$\sum_{x\in X} L_1(x) = \sum_{x\in X} L_2(x)$$

Notons que cette restriction entre deux loteries normalisées est vérifiée. En effet, la somme d'une loterie normalisée est par définition égale à 1.

Nous pouvons donc comparer les deux *product updates* : normalisé et dénormalisé.

Proposition 4.33. *Soit $\mathcal{M}$ une structure de Kripke dénormalisée et $\mathcal{E}$ un modèle conditionnel à préconditions séparées sans postcondition, dont toutes les $E-$loteries $\mu_{\text{pre}}(\phi)$ pour $\phi \in \Phi$ sont à sommes égales. Nous avons :*

$$\mathrm{N}(\mathcal{M} \otimes_d \mathcal{E}) \text{ est isomorphe à } \mathrm{N}(\mathcal{M}) \otimes \mathrm{N}(\mathcal{E})$$

où $\otimes_d$ réfère au product update *dénormalisé (déf 4.28) et $\otimes$ au* product update *normalisé classique (déf. 4.13).*

Démonstration. Soit $\mathcal{M} = \langle W, R, \mu, \text{val}\rangle$ une structure de Kripke dénormalisée et sa version normalisée notée $\mathrm{N}(\mathcal{M}) = \langle W, R, \mathrm{N}(\mu), \text{val}\rangle$.

Soit un modèle conditionnel à préconditions séparées $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mu^{\mathcal{E}}, \Phi, \mu_{\text{pre}}\rangle$ et sa version normalisée notée $\mathrm{N}(\mathcal{E}) = \langle E, R^{\mathcal{E}}, \mathrm{N}(\mu^{\mathcal{E}}), \Phi, \mathrm{N}(\mu_{\text{pre}})\rangle$, dont les E-loteries $\mu_{\text{pre}}(\phi)$ pour $\phi \in \Phi$ sont à sommes égales.

Remarquons tout d'abord que lors des deux *product updates* (normalisé et dénormalisé), les définitions des mondes $W^{\otimes}$, des relations $R^{\otimes}$ et des valuations val des mondes sont inchangées (la normalisation de la fonction pre ne change pas le fait que valeur de $\mu_{\text{pre}}(\phi)$ soit nulle, donc l'ensemble des mondes est identique).

Il nous reste simplement à montrer que les probabilités résultantes des deux calculs sont identiques.

Notons $\mathrm{N}(\mu^{N\otimes N})$ et μ^N les fonctions de probabilités, respectivement obtenues par le calcul $\mu^{\mathrm{N}(\mathcal{M}\otimes\mathcal{E})}$ et par le calcul $\mu^{\mathrm{N}(\mathcal{M})\otimes\mathrm{N}(\mathcal{E})}$. $\mu^{N\otimes N}$ est ainsi la loi de probabilité discrète de $\mu^{\mathrm{N}(\mathcal{M}\otimes\mathcal{E})}$ avant re-normalisation.

Voici la formule du *product update* normalisé :

$$\mu_a^{\otimes}((w,e),(w',e')) = \frac{\mu_a(w)(w') \cdot \mu_{\text{pre}}(w')(e') \cdot \mu_a(e)(e')}{\sum\limits_{w''\in W, e''\in E} \mu_a(w)(w'') \cdot \mu_{\text{pre}}(w'')(e'') \cdot \mu_a^{\mathcal{E}}(e)(e'')} \tag{4.1}$$

et du *product update* dénormalisé (déf. 4.28) :

$$\mu_a^{\otimes}((w,e))((w',e')) = \mu_a(w)(w') \cdot \mu_{\text{pre}}(w')(e') \cdot \mu_a^{\mathcal{E}}(e)(e') \tag{4.2}$$

Nous avons donc à partir de l'égalité 4.2, l'écriture de μ_a^N :

$$\mu_a^N((w,e))((w',e')) = \mu_a(w)(w') \cdot \mu_{\text{pre}}(w')(e') \cdot \mu_a^{\mathcal{E}}(e)(e') \tag{4.3}$$

qui permet d'obtenir sa version normalisée $\mathrm{N}(\mu_a^N)$:

$$\mathrm{N}(\mu_a^N((w,e)))((w',e')) = \frac{\mu_a(w)(w') \cdot \mu_{\text{pre}}(w')(e') \cdot \mu_a^{\mathcal{E}}(e)(e')}{\sum\limits_{w''\in W, e''\in E} \mu_a(w)(w'') \cdot \mu_{\text{pre}}(w'')(e'') \cdot \mu_a^{\mathcal{E}}(e)(e'')} \tag{4.4}$$

car si nous prenons L une X-loterie avec $X = \{W \times E\}$, nous avons $\mathrm{N}(L)((w,e)) = \frac{L((w,e))}{\sum_{w'' \in W, e'' \in E} L((w'',e''))}$. Prenons maintenant $L = \mu_a^N((w,e))$.

Écrivons maintenant $\mu_a^{N \otimes N}$ à partir de l'égalité 4.1, avec les lois de probabilité discrètes de $\mathrm{N}(\mathcal{M})$ et $\mathrm{N}(\mathcal{E})$ qui sont par définition $\mathrm{N}(\mu)$, $\mathrm{N}(\mu_{\mathrm{pre}})$ et $\mathrm{N}(\mu^{\mathcal{E}})$:

$$\mu_a^{N \otimes N}((w,e),(w',e'))$$

$$= \frac{\mathrm{N}(\mu_a(w))(w') \cdot \mathrm{N}(\mu_{\mathrm{pre}}(w'))(e') \cdot \mathrm{N}(\mu_a^{\mathcal{E}}(e))(e')}{\sum\limits_{w'' \in W, e'' \in E} \mathrm{N}(\mu_a(w))(w'') \cdot \mathrm{N}(\mu_{\mathrm{pre}}(w''))(e'') \cdot \mathrm{N}(\mu_a^{\mathcal{E}}(e))(e'')} \tag{4.5}$$

$$= \frac{\frac{\mu_a(w)(w')}{\sum_{w_1 \in W} \mu_a(w)(w_1)} \cdot \frac{\mu_{\mathrm{pre}}(w')(e')}{\sum_{e_1 \in E} \mu_{\mathrm{pre}}(w')(e_1)} \cdot \frac{\mu_a^{\mathcal{E}}(e)(e')}{\sum_{e_2 \in E} \mu_a^{\mathcal{E}}(e)(e_2)}}{\sum\limits_{w'' \in W, e'' \in E} \frac{\mu_a(w)(w'')}{\sum_{w_2 \in W} \mu_a(w)(w_2)} \cdot \frac{\mu_{\mathrm{pre}}(w'')(e'')}{\sum_{e_3 \in E} \mu_{\mathrm{pre}}(w'')(e_3)} \cdot \frac{\mu_a^{\mathcal{E}}(e)(e'')}{\sum_{e_4 \in E} \mu_a^{\mathcal{E}}(e)(e_4)}} \tag{4.6}$$

Nous notons pour un monde w donné, $\alpha = \sum_{x \in W} \mu_a(w)(x)$, $\beta = \sum_{x \in E} \mu_a^{\mathcal{E}}(e)(x)$ et $\gamma = \sum_{x \in E} \mu_{\mathrm{pre}}(w)(x)$. α et β sont des constantes vis-à-vis de l'argument (w,e) et nous savons les loteries de préconditions μ_{pre} à sommes égales, donc les dénominateurs des fractions centrales sont identiques (γ).

$$= \frac{\frac{\mu_a(w)(w')}{\alpha} \cdot \frac{\mu_{\mathrm{pre}}(w')(e')}{\gamma} \cdot \frac{\mu_a^{\mathcal{E}}(e)(e')}{\beta}}{\sum\limits_{w'' \in W, e'' \in E} \frac{\mu_a(w)(w'')}{\alpha} \cdot \frac{\mu_{\mathrm{pre}}(w'')(e'')}{\gamma} \cdot \frac{\mu_a^{\mathcal{E}}(e)(e'')}{\beta}} \tag{4.7}$$

$$= \frac{\frac{\mu_a(w)(w')}{\alpha} \cdot \frac{\mu_{\mathrm{pre}}(w')(e')}{\gamma} \cdot \frac{\mu_a^{\mathcal{E}}(e)(e')}{\beta}}{\frac{1}{\alpha \cdot \gamma \cdot \beta} \cdot \sum\limits_{w'' \in W, e'' \in E} \mu_a(w)(w'') \cdot \mu_{\mathrm{pre}}(w'')(e'') \cdot \mu_a^{\mathcal{E}}(e)(e'')} \tag{4.8}$$

$$= \frac{\mu_a(w)(w') \cdot \mu_{\mathrm{pre}}(w')(e') \cdot \mu_a^{\mathcal{E}}(e)(e')}{\sum\limits_{w'' \in W, e'' \in E} \mu_a(w)(w'') \cdot \mu_{\mathrm{pre}}(w'')(e'') \cdot \mu_a^{\mathcal{E}}(e)(e'')} \tag{4.9}$$

$$= \mathrm{N}(\mu_a^N((w,e)))((w',e')) \tag{4.10}$$

□

Nous voyons qu'il y a une condition à cette proposition : que les loteries des préconditions soient à sommes égales (déf. 4.32). Évidemment, nous voyons l'importance de celle-ci durant la preuve, sinon la preuve s'avère impossible. Prenons un contre-exemple illustratif.

Par ailleurs, le fait que Φ soit un ensemble de formules mutuellement incohérentes, ou non, ne change pas ce résultat : les *product updates* sont équivalents peu importe les propriétés de Φ.

Exemple 4.34. *Soit la structure de Kripke mono-agent suivante* $\mathcal{M} = \langle \{w_1, w_2\}, \{(w_1,w_1),(w_1,w_2),(w_2,w_2)\}, \mathrm{val}(w_1) = \{p\}, \mathrm{val}(w_2) = \{\}, \mu(w_1)(w_1) =$

$\mu(w_1)(w_2) = \mu(w_2)(w_2) = 1\rangle$. *Soit le modèle conditionnel sans postconditions* $\mathcal{E} = \langle E = \{e_1, e_2\}, 2^E, \forall e, e' \in E^2, \mu^{\mathcal{E}}(e)(e') = 1, \Phi = \{p, \neg p\}, \mu_{\text{pre}}(p)(e_1) = \mu_{\text{pre}}(p)(e_2) = \mu_{\text{pre}}(\neg p)(e_1) = 1, \mu_{\text{pre}}(\neg p)(e_2) = 2.\rangle$. *Ces deux structures sont présentées en haut de la figure 4.4 et leur* product update normalisé *est présenté à leur intersection (*$\text{N}(\mathcal{M} \otimes \mathcal{E})$*).*

En normalisant ces structures, nous obtenons : $\mu(w_1)(w_1) = \mu(w_1)(w_2) = 0.5$, $\mu(w_2)(w_2) = 1$, $\mu_{\text{pre}}(p)(e_1) = \mu_{\text{pre}}(p)(e_2) = 0.5$, $\mu_{\text{pre}}(\neg p)(e_1) = 1/3, \mu_{\text{pre}}(\neg p)(e_2) = 2/3$ *et* $\forall e, e' \in E^2, \mu^{\mathcal{E}}(e)(e') = 0.5$.

Les versions normalisées sont présentées en dessous de la figure ($\text{N}(\mathcal{M})$ *et* $\text{N}(\mathcal{E})$*). Nous voyons le* product update *résultant* $\text{N}(\mathcal{M}) \otimes \text{N}(\mathcal{E})$. *Nous constatons que les deux structures résultantes des product updates sont bien différentes.*

Comparons les arcs sortant des mondes (w_1, e_1). *Tous les numérateurs sont identiques, mais ce n'est pas le cas des dénominateurs. En effet, dans le cas du dessus, la somme des arcs sortants avant normalisation est de* 5 *(*$1+1+1+2$*), il est alors ensuite simple de normaliser en divisant par* 5. *Ce n'est pas le cas pour le calcul en dessous qui se décompose ainsi :*

Avant normalisation	*Après normalisation en divisant par la somme :* 0.5
$\mu^{\otimes}((w_1, e_1))((w_1, e_1)) = 0,5 \times 0,5 \times 0,5 = 1/8$	$\mu^{\otimes}((w_1, e_1))((w_1, e_1)) = 1/4$
$\mu^{\otimes}((w_1, e_1))((w_2, e_1)) = 0,5 \times 0,5 \times 1/3 = 1/12$	$\mu^{\otimes}((w_1, e_1))((w_2, e_1)) = 1/6$
$\mu^{\otimes}((w_1, e_1))((w_2, e_2)) = 0,5 \times 0,5 \times 2/3 = 1/6$	$\mu^{\otimes}((w_1, e_1))((w_2, e_2)) = 1/3$
$\mu^{\otimes}((w_1, e_1))((w_1, e_2)) = 0,5 \times 0,5 \times 0,5 = 1/8$	$\mu^{\otimes}((w_1, e_1))((w_1, e_2)) = 1/4$

En conclusion, la division faite pour la normalisation n'est pas identique dans les deux cas.

Notons que si nous avions eu la E−loterie de telle sorte que $\mu_{\text{pre}}(p)(e_1) = \mu_{\text{pre}}(p)(e_2) = 1.5$, *pour avoir des sommes de loteries égales à* 3, *les deux product updates auraient été identiques.*

Maintenant que nous avons présenté le *product update*, nous pouvons présenter la sémantique dans le cas de formules dynamiques probabilistes avec des structures dénormalisées.

Cette définition de la sémantique de PDEL classique s'écrit de la même manière dans le cas de PDEL dénormalisé (voir déf. 4.16).

Définition 4.35. *La sémantique du langage* $\mathcal{L}_{\text{PDEL}}$ *est identique à celle de* $\mathcal{L}_{\text{PEL}}$ *dénormalisé (déf. 4.22), en y ajoutant la sémantique de l'opérateur* $[\mathcal{E}, e]$ *avec* $\mathcal{E}$ *un modèle conditionnel dénormalisé :*

$$\langle \mathcal{M}, w \rangle \models_d [\mathcal{E}, e]\phi \text{ ssi } \mu_{\text{pre}}(w)(e) > 0 \implies \langle \mathcal{M} \otimes \mathcal{E}, (w, e) \rangle \models_d \phi$$

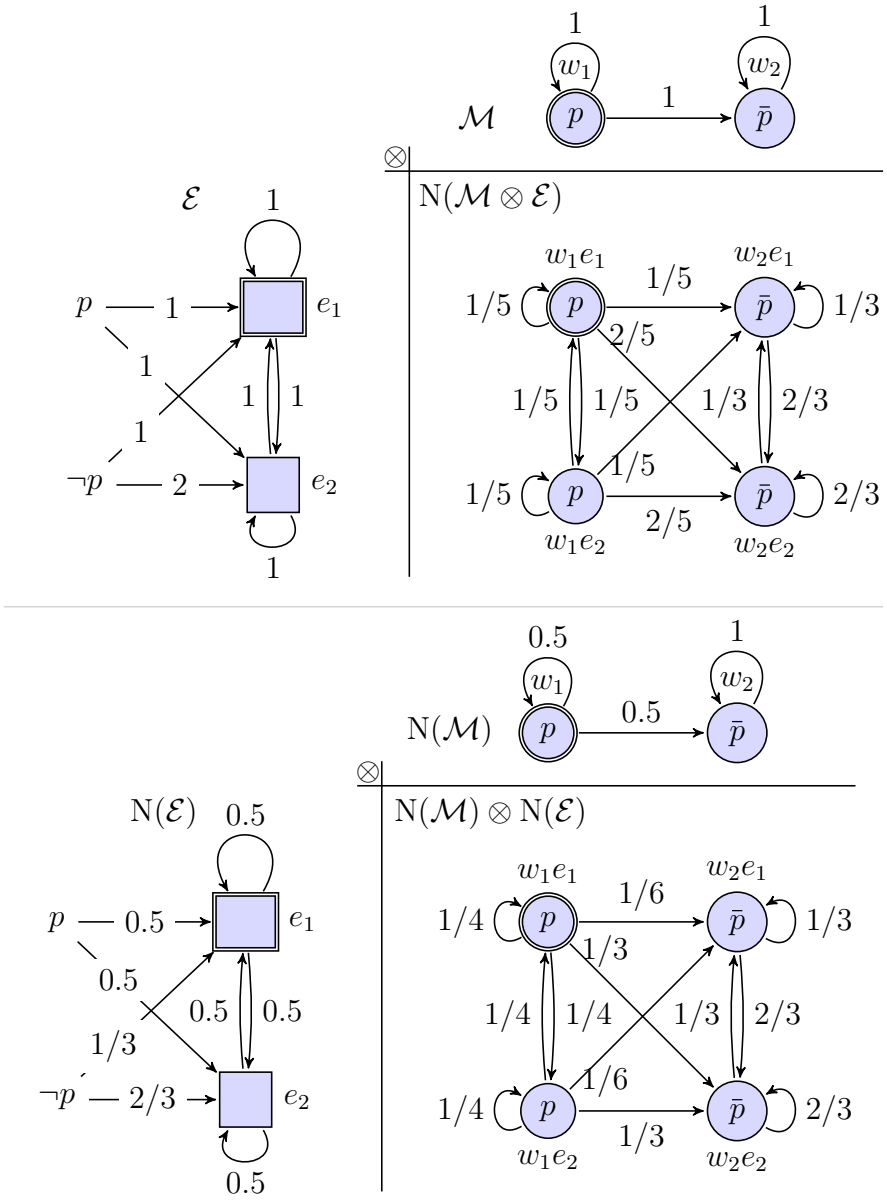


FIGURE 4.4 – Example de loteries à sommes non égales : $\mathrm{N}(\mathcal{M}) \otimes \mathrm{N}(\mathcal{E})$ (ex. 4.34)

De la même manière que dans la remarque pour SMCDEL (rq. 2.59), nous abusons de la notation de $\mathcal{L}_{\mathrm{PDEL}}$, qui contiendra l'opérateur $[\mathcal{E}, e]$ pour le cas normalisé (modèle d'événements classique) et dénormalisé (modèle conditionnel à préconditions séparées).

Proposition 4.36. *Soit $\mathcal{M} = \langle W, R, \mu, \mathrm{val} \rangle$ une structure de Kripke dénormalisée*

et $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mu^{\mathcal{E}}, \Phi, \mu_{\text{pre}} \rangle$ *un modèle conditionnel dénormalisé, dont toutes les* $E-$*loteries* $\mu_{\text{pre}}(\phi)$ *pour* $\phi \in \Phi$ *sont à sommes égales nous avons :*

$$\langle \mathcal{M}, w \rangle \models_d [\mathcal{E}, e]\phi \textit{ ssi } \langle \mathrm{N}(\mathcal{M}), w \rangle \models [\mathrm{N}(\mathcal{E}), e]\phi$$

Démonstration. Nous avons déjà montré tous les cas non dynamiques dans la proposition 4.23.

Nous avons simplement maintenant à montrer que :

$$\langle \mathcal{M}, w \rangle \models_d [\mathcal{E}, e]\phi \text{ ssi } \langle \mathrm{N}(\mathcal{M}), w \rangle \models [\mathrm{N}(\mathcal{E}), e]\phi$$

C'est à dire que nous avons ($\otimes_d$ et $\otimes$ réfèrant respectiement aux *product update* dénormalisé (déf 4.28) et normalisé (déf. 4.13)) :

$\langle \mathcal{M}, w \rangle \models_d [\mathcal{E}, e]\phi$ ssi $\mathrm{N}(\mu_{\text{pre}}(w))(e) > 0 \implies \langle \mathrm{N}(\mathcal{M}) \otimes \mathrm{N}(\mathcal{E}), (w, e) \rangle \models \phi$.

Ensuite, la proposition 4.23 permet de conclure, par les comparaison des semantiques que : $\mathrm{N}(\mu_{\text{pre}}(w))(e) \implies \langle \mathrm{N}(\mathcal{M} \otimes \mathcal{E}), (w, e) \rangle \models \phi$.

Enfin, la proposition 4.33 sur l'isomorphisme, ainsi que le fait que $\forall \mathcal{M}, \forall \mathcal{E},\ \mathrm{N}(\mathcal{M} \otimes_d \mathcal{E}) \iff \mathrm{N}(\mathcal{M}) \otimes \mathrm{N}(\mathcal{E})$ et que la normalisation de $\mu_{\text{pre}}(w)$ ne modifie pas l'inégalité vis à vis de 0, nous pouvons conlure la proposition.

□

Puisque pour un $\mathcal{E}$ normalisé, nous avons $\mathrm{N}(\mathcal{E}) = \mathcal{E}$, comme pour le cas non dynamique, nous allons simplifier l'écriture en utilisant $\models$ à la place de $\models_d$.

La sémantique de $\mathcal{L}_{\text{PDEL}}$ est donc la même que celle présentée à la définition dénormalisée (déf. 4.16), mais avec la sémantique de $\mathcal{L}_{\text{PEL}}$ présentée à la définition 4.22. Notons simplement que le fait d'avoir des loteries de préconditions à sommes égales n'est pas problématique. En effet, il est naturel de vouloir décrire un problème avec des lois de probabilité discrètes au lieu de loteries. Or les lois de probabilité discrètes sont un sous cas des loteries et somment naturellement toutes identiquement à 1. Ainsi, dans ce cas, la condition des loteries à sommes égales est toujours respectée.

Un modèle conditionnel normalisé est un sous-cas d'un modèle conditionnel à préconditions séparées . Comme nous l'avons vu, il est alors très simple d'obtenir le premier à partir du second, grâce à notre définition de la normalisation d'un modèle conditionnel à préconditions séparées (déf. 4.31). L'opération inverse n'est pour autant pas si simple. En effet, tout le problème réside dans la transformation des préconditions conditionnelles Φ, qui n'ont pas de contraintes entre elles, en formules mutuellement incohérentes. Si elles sont de base incohérentes, la transformation est directe ; sinon, il faut créer un nouvel ensemble Φ et modifier en conséquence μ_{pre}, de sorte à ce que les deux modèles conditionnels soient équivalents. Or, la taille de ce nouvel ensemble peut exploser exponentiellement.

Exemple 4.37. *Soit V un vocabulaire et $\Phi = \{p \mid p \in V\}$. Toutes les formules de Φ ont alors des modèles en commun. Une façon simple d'obtenir un Φ' de formules mutuellement incohérentes est alors de créer l'ensemble $\{v \sqsubseteq V \mid v \in 2^V\}$, qui est l'ensemble des modèles possibles. Il ne restera plus qu'à adapter μ_{pre}. On voit alors que la transformation est exponentielle car l'ensemble de formmules passe de la taille $\mid V \mid$ à $2^{|V|}$.*

Bien évidemment, il s'agit du pire cas. Comme nous l'avons dit, si les formules sont déjà incohérentes entre elles, la transformation sera directe. Dans les autres cas, où les formules ont des modèles en commun, il est possible de construire une transformation adaptée qui traite les sous groupes de formules mutuellement incohérentes entre elles.

Exemple 4.38. *Soit $V = \{p, q, r\}$ et $\Phi = \{(a \vee b) \wedge \neg c,\ a \wedge c \wedge \neg c,\ \neg a \wedge \neg b \wedge \neg c\}$. Il est possible de construire l'ensemble $\{(a \oplus b) \wedge \neg c,\ a \wedge b \wedge \neg c,\ \neg a \wedge \neg b \wedge \neg c\}$ dont les formules sont mutuellement incohérentes entre elles. Il faudra bien évidemment reconstruire μ_{pre} en conséquence pour que les deux* product updates *soient équivalents.*

Ainsi, dans le cas à préconditions séparées ou dans des sous cas précis, comme ci-dessus, la taille de l'ensemble des formules mutuellement incohérentes ne change pas. Ainsi, dans le meilleur des cas, la transformation est constante et dans le pire cas, elle semble exponentielle avec un algorithme simple.

Un algorithme généralisé qui construit l'ensemble de formules minimal serait potentiellement conceptuellement intéressant à étudier mais n'a ici pas d'application utile.

Nous venons de présenter le modèle conditionnel dénormalisé et l'avons comparé à sa version normalisée. Nous allons maintenant introduire une version simplifiée de ce type d'événement : le modèle observationnel.

4.3 Le modèle observationnel

Comme l'évoquent van Benthem et al. [vBGK06] dans une prépublication de leur article « *Dynamic Update with Probabilities* », les modèles d'événements probabilistes peuvent être définis de manière plus simples que les modèles vus précédemment (déf. 4.11). Ainsi, nous allons comparer la version que nous avons déjà présentée, avec une nouvelle.

Simplement, nous pouvons voir ce nouveau type de modèle comme un modèle d'événements classique (déf. 1.24) avec l'ajout de l'ensemble des lois de probabilité discrètes μ_a. On y retrouve alors la définition originale des préconditions pre et la notion de probabilités conditionnelles, modélisée par Φ et μ_{pre} (déf. 4.11), disparaît.

Nous appelons ce type de modèle *un modificateur observationnel* et nous le définissons directement comme étant dénormalisé pour éviter l'énumération de définitions très similaires, et avec des postconditions.

Le seul point commun entre un modèle observationnel et un modèle conditionnel est l'utilisation d'une frame dénormalisée (déf. 4.20). Contrairement au modèle conditionnel, le modèle observationnel bénéficie des fonctions pre et post définies par DEL (déf. 1.24).

Définition 4.39. *Un modèle observationnel défini pour un vocabulaire V est un tuple $\langle \mathcal{F}^{\mathcal{E}}, \mathrm{pre}, \mathrm{post} \rangle$ où :*

- $\mathcal{F}^{\mathcal{E}}$ *est une* frame probabiliste dénormalisée*, avec les élements notés E, $R^{\mathcal{E}}$, $\mu^{\mathcal{E}}$ (déf. 4.20) ;*
- pre *: fonction de precondition , $E \to \mathcal{L}_{\mathrm{PEL}}(V)$;*
- post *: fonction de postcondition, $V \times E \to \mathcal{L}_{\mathrm{prop}}(V)$.*

Remarque 4.40. *On peut définir un modèle observationnel normalisé comme 'un modèle observationnel où les loteries de la* frame *probabiliste utilisée sont normalisées.*

Nous notons $\mathrm{N}(\mathcal{E})$ la normalisation d'un modèle observationnel $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mu^{\mathcal{E}}, \mathrm{pre}, \mathrm{post} \rangle$, définie comme $\mathrm{N}(\mathcal{E}) = \langle E, R^{\mathcal{E}}, \mathrm{N}(\mu^{\mathcal{E}}), \mathrm{pre}, \mathrm{post} \rangle$ où $\mathrm{N}(\mu^{\mathcal{E}})$ est la normalisation de chaque loi de probabilité discrète de $\mu^{\mathcal{E}} : a \in \mathfrak{A} \to (e \in E \to \mathrm{N}(\mu^{\mathcal{E}}_a(e)))$.

Remarque 4.41. *À titre purement informatif, notons que les auteurs envisagent dans la prépublication [vBGK06] un autre type de modèle événementiel qu'ils nomment* modèle de protocole*. Celui-ci, à l'inverse du modèle conditionnel, ne considère pas les probabilités que peuvent assigner les agents aux événements atomiques et se concentre sur les probabilités conditionnelles. C'est donc simplement un tuple $\langle E, \Phi, \mu_{\mathrm{pre}} \rangle$.*

Vis-à-vis de notre dénomination des événements nous devrions renommer :

- modèle de protocole *en* modèle "purement conditionnel" *(modèle d'événements probabilistes avec probabilités conditionnelles) ;*
- le modèle conditionnel *en* modèle observationnel et conditionnel *(modèle d'événements probabilistes avec probabilités observationnelles et conditionnelles) ;*
- le modèle observationnel *en* modèle "purement observationnel" *(modèle d'événements probabilistes avec probabilités observationnelles).*

La mise à jour des connaissances peut alors être définie avec ce nouveau type d'événement dénormalisé. Celle-ci est identique aux définitions du *product update* de DEL (déf. 1.28) pour ce qui est de la mise à jour des mondes, des relations et des valuations des mondes.

Définition 4.42. *Soit* $\mathcal{M} = \langle W, R, \mu, \text{val}\rangle$ *une structure de Kripke probabiliste dénormalisée et* $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mu^{\mathcal{E}}, \text{pre}, \text{post}\rangle$ *un modèle observationnel dénormalisé. Pour* $w \in W$ *et* $e \in E$*, le* product update $\langle \mathcal{M}, w\rangle \otimes \langle E, e\rangle$ *est la structure de Kripke probabiliste dénormalisée pointée* $\langle \mathcal{M}^{\otimes}, (w, e)\rangle$ *où* $\mathcal{M}^{\otimes} = \langle W^{\otimes}, R^{\otimes}, \mu^{\otimes}, \text{val}^{\otimes}\rangle$ *avec :*

- $W^{\otimes}$*,* $R^{\otimes}$ *définis identiquement aux définitions de DEL :*
  - $W^{\otimes} = \{(w, e) \in W \times E \mid \langle \mathcal{M}, w\rangle \models \text{pre}(e)\}$ *;*
  - $(w, e)\ R_a^{\otimes}\ (w', e')$ ssi wR_aw' et $eR_a^{\mathcal{E}}e'$*, pour tout* $(w, w') \in W^2$ *et* $(e, e') \in E^2$ *;*
- $\mu_a^{\otimes}((w, e), (w', e')) = \mu_a(w)(w') \times \mu_a^{\mathcal{E}}(e)(e')$ *;*
- $\text{val}^{\otimes}((w, e)) = \{p \in V \mid \langle \mathcal{M}, w\rangle \models \text{post}(p)(e)\}$ *pour tout* $(w, e) \in W^{\otimes}$*.*

Remarque 4.43. *Si on souhaite que le* product update *donne des résultats normalisés, il faut utiliser la formule suivante :*

$$\mu_a^{\otimes}((w, e), (w', e')) = \frac{\mu_a(w)(w') \times \mu_a^{\mathcal{E}}(e)(e')}{\sum\limits_{w'' \in W, e'' \in E} \mu_a(w)(w'') \times \mu_a^{\mathcal{E}}(e)(e'')}$$

Notons que ce $\mu^{\otimes}$ *est une simplification du calcul de la définition du* product update *d'un modèle conditionnel normalisé (déf. 4.13), en enlevant la notion de probabilités conditionnelles.*

Dans le cas d'un modèle observationnel, la sémantique s'écrit de la même manière que la sémantique de DEL (déf. 1.29), seulement elle n'augmente pas $\mathcal{L}_{\text{EL}}$ mais $\mathcal{L}_{\text{PEL}}$. Notons que nous utilisons toujours un abus de notation (c.f. rq. 2.59) et considérons abusivement l'utilisation des deux types de modèles évenementiels dans notre langage.

Définition 4.44. *La sémantique du langage* $\mathcal{L}_{\text{PDEL}}$ *est identique à celle de* $\mathcal{L}_{\text{PEL}}$ *(déf. 4.9), en y ajoutant la sémantique de l'opérateur* $[\mathcal{E}, e]$*, avec* $\langle \mathcal{E}, e\rangle$ *un modèle observationnel pointé :*

$$\langle \mathcal{M}, w\rangle \models [\mathcal{E}, e]\phi \text{ ssi } \langle \mathcal{M}, w\rangle \models \text{pre}(e) \implies \langle \mathcal{M} \otimes \mathcal{E}, (w, e)\rangle \models \phi$$

4.4 Choix du modèle événementiel

La définition d'un modèle observationnel est intéressante. En toute généralité, PDEL impose par sa définition des probabilités conditionnelles. Néanmoins, nous pouvons vouloir modéliser et profiter des probabilités, sans pour autant vouloir expliciter des préconditions conditionnelles dans un modèle conditionnel.

Nous illustrons dans cette section l'interêt spécifique de chaque modèle.

Pour modéliser le lancer d'un dé ou d'une pièce qui s'avèrent truqués, la définition d'un modèle conditionnel est parfaite. Néanmoins, pour modéliser un lancer de dé ou de pièce classique, la définition d'un modèle observationnel est strictement suffisante.

Notons que nous utilisons dans les deux exemples qui vont suivre les définitions classiques de PDEL, avec des lois de probabilité discrètes et avec normalisation après *product update*, de sorte à simplifier la lisibilité des exemples.

Exemple 4.45. *Définissons le modèle observationnel pour le lancer d'une pièce équiprobable. Notons qu'il s'avère que les lois de probabilité discrètes seront ici a-consistantes et a-uniformes. Un agent, nommé r, voit le lancer, l'autre, nommé b, ne le voit pas. Soit $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mu^{\mathcal{E}}, \mathrm{pre}, \mathrm{post}\rangle$ défini sur le vocabulaire $V = \{p\}$ avec :*

- $E = \{e_p, e_f\}$ *(pour pile ou face) ;*
- $R_r^{\mathcal{E}} = \{(e_p, e_p), (e_f, e_f)\}$;
- $R_b^{\mathcal{E}} = E \times E$;
- $\mu_r^{\mathcal{E}}(e_p)(e_p) = \mu_r^{\mathcal{E}}(e_f)(e_f) = 1$;
- $\forall e, e' \in E,\ \mu_b^{\mathcal{E}}(e)(e') = 1/2$;
- $\mathrm{pre}(e_p) = \mathrm{pre}(e_f) = \top$;
- $\mathrm{post}(p)(e_p) = \top$;
- $\mathrm{post}(p)(e_f) = \bot$.

Ici, on peut considérer que la pièce est posée sur la table, du côté pile. Ainsi, nous aurons la structure de Kripke définie sur le vocabulaire $V = \{p\}$ telle que $\mathcal{M} = \langle W, R, \mu, \mathrm{val}\rangle$ avec $W = \{w\}$, $\forall a \in \mathfrak{A},\ R_a = \{(w, w)\}$ et $\mu_a(w)(w) = 1$, puis $\mathrm{val}(w) = \{p\}$.

Nous représentons $\mathcal{M}$, $\mathcal{E}$ et leur product update *$\mathcal{M} \otimes \mathcal{E}$ à la figure 4.45.*

Nous voyons qu'il n'y a besoin d'aucune connaissance a priori sur le fait que la pièce puisse être truquée ou non et que la mise à jour des connaissances est simple et reflète parfaitement ce que nous souhaitons.

Exemple 4.46. *Créons maintenant le modèle conditionnel pour le lancer d'une pièce truquée en gardant les mêmes paramètres que l'exemple avec la pièce non truquée (ex. 4.45). Les lois de probabilité discrètes seront ici toujours a-consistantes et a-uniformes. Considérons que la pièce n'est pas bien équilibrée et qu'elle tombe sur pile 2 fois sur 3. Soit $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mu^{\mathcal{E}}, \Phi, \mu_{\mathrm{pre}}, \mathrm{post}\rangle$ défini sur le vocabulaire $V = \{p\}$ avec :*

- *E, $R^{\mathcal{E}}$, $\mu^{\mathcal{E}}$ et* post *identiques à l'exemple 4.45 ;*
- $\Phi = \{t, \neg t\}$

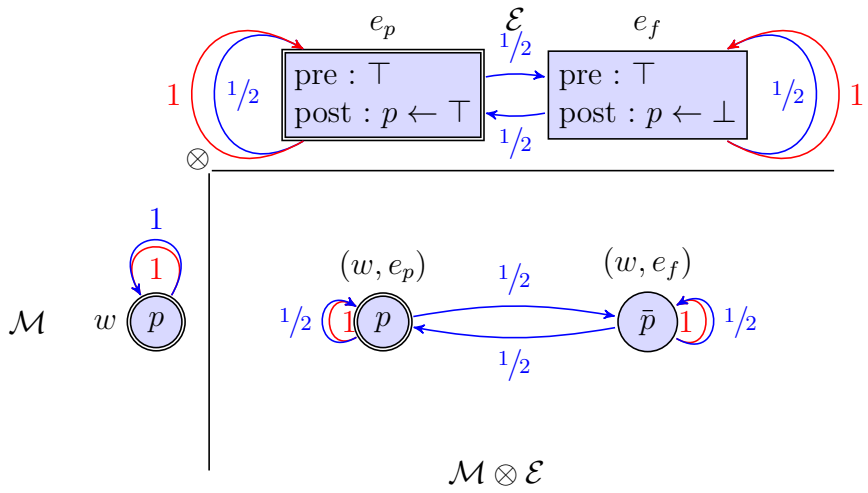


FIGURE 4.5 – Exemple de lancer d'une pièce équiprobable (ex. 4.45)

- $\mu_{\text{pre}}(t)(e_p) = 2/3$ *et* $\mu_{\text{pre}}(t)(e_f) = 1/3$;
- $\mu_{\text{pre}}(\neg t)(e_p) = 1/2$ *et* $\mu_{\text{pre}}(\neg t)(e_f) = 1/2$;
- $\text{post}(p)(e_p) = \top$;
- $\text{post}(p)(e_f) = \bot$.

Cette fois-ci, il faut un contexte pour pouvoir appliquer le modèle conditionnel, à savoir, si la pièce est truquée ou non, représentée par la variable t. Admettons que l'agent r sache que la pièce est truquée : il va donc différencier les deux mondes potentiels. Ainsi, nous aurons la structure de Kripke définie sur le vocabulaire $V = \{p, t\}$ telle que $\mathcal{M} = \langle W, R, \mu, \text{val}\rangle$ avec $W = \{w_1, w_2\}$, $R_r = \{(w_1, w_1), (w_2, w_2)\}$, $R_b = W \times W$, $\mu_r(w_1)(w_1) = \mu_r(w_2)(w_2) = 1$ et $\forall w, w' \in W$, $\mu_b^{\mathcal{E}}(w)(w') = 1/2$ puis $\text{val}(w_1) = \{p, t\}$ et $\text{val}(w_2) = \{p\}$.

La structure $\mathcal{M}$ et le modèle conditionnel $\mathcal{E}$, ainsi que leur product update *est présenté figure 4.6.*

Nous voyons que l'agent b sait que la pièce est truquée et que par conséquent, bien qu'il n'ait pas vu le résultat de la pièce, il estime que $2/3$ du temps, le résultat sera pile. Ce n'est pas le cas pour le joueur a. Néanmoins, il voit la résultat de la pièce, ainsi, il sait que p est vrai. Par ailleurs, il envisageait d'une manière équiprobable que la pièce était truquée, ainsi avec ce lancé de pièce, il envisage la probabilité que la pièce soit truquée à $4/7$.

Remarque 4.47. $\frac{4}{7}$ *est obtenu à cause de la normalisation des deux fractions* $\frac{1}{3}$

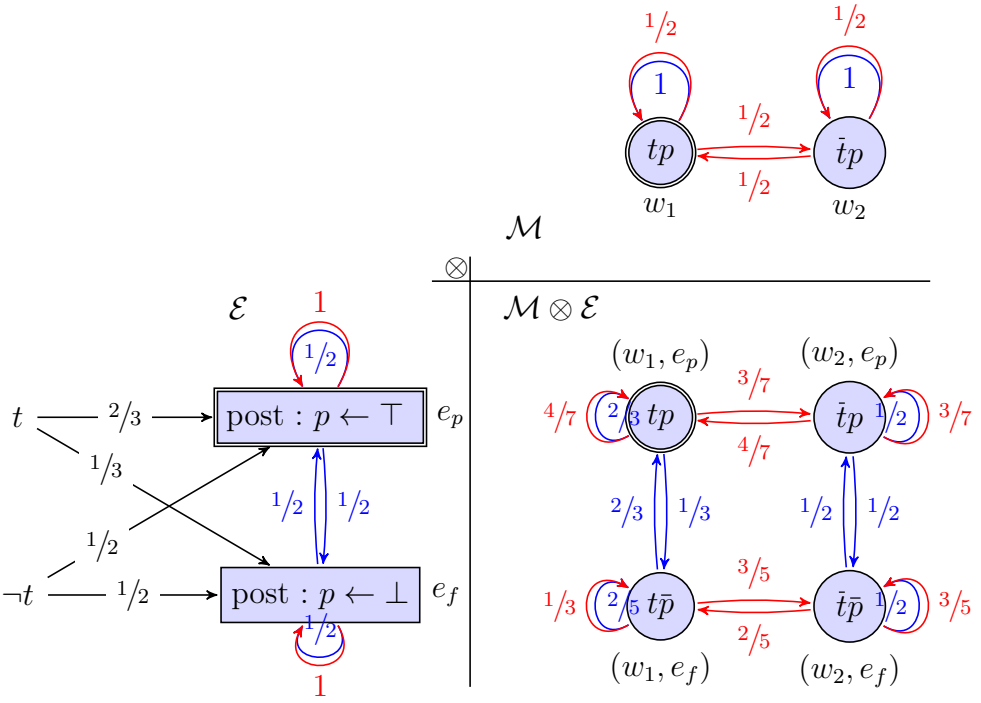


FIGURE 4.6 – Exemple d'un lancé de pièce truquée (ex. 4.46).

et $\frac{1}{4} : \frac{\frac{1}{4}}{\frac{1}{3}+\frac{1}{4}} = \frac{\frac{1}{4}}{\frac{4}{12}+\frac{3}{12}} = \frac{\frac{1}{4}}{\frac{7}{12}} = \frac{4}{7}$. *Il en va de même pour* $\frac{3}{7}$ *qui est son complément à 1.*

Les deux exemples 4.45 et 4.46 présentent des situations similaires puisqu'une pièce est lancée mais avec des concepts très différentes.

Dans la premier cas non truqué, il n'y a aucun prérequis pour lancer la pièce : il n'y a pas de variable t à introduire dans le vocabulaire. Quand bien même nous souhaiterions modéliser une probabilité conditionnelle, il paraît complexe de l'intégrer dans ce genre de modèle d'observations.

Dans le second cas avec la pièce truquée, tous les éléments sont modélisés et il y a des pré-requis. En effet, il faut pouvoir affecter le vocabulaire dédié aux préconditions conditionnelles aux mondes et les agents ont par conséquent des connaissances a propos de ces atomes propositionnels.

Bien évidemment, il est possible de transformer un type de modèle en l'autre, mais la création directe d'un des modèles représentant l'équivalent de l'autre n'est pas intuitive. Par ailleurs, dans le cas du modèle observationnel, une précondition est attribuée à chaque événement atomique. Par exemple, c'est tout à fait adapté pour modéliser une action de type « piocher une carte », où chaque événement

atomique représenterait le fait de piocher une carte précise avec pour précondition le fait que la carte soit dans la pioche. Cela semble plus complexe de modéliser ce type d'action avec un modèle conditionnel et des préconditions conditionnelles.

Nous voyons bien qu'il est plus simple de modéliser un événement stochastique sans préconditions conditionnelles particulières via un modèle observationnel plutôt qu'avec un modèle conditionnel.

Avec ces quelques définitions faites, nous allons pouvoir comparer les deux types de modèles : modèle conditionnel et modèle observationnel.

4.5 Conversions entre modèles conditionnel et observationnel

Nous allons maintenant étudier la transformation d'un modèle conditionnel en un modèle observationnel et vice versa.

Définition 4.48. *La transformation* obs *d'un modèle conditionnel à préconditions séparées* $\mathcal{E}_c = \langle E_c, R_c, \mu_c^{\mathcal{E}}, \Phi_c, \mu_{\text{pre}}, \text{post}_c \rangle$ *défini pour le vocabulaire* V *en un modèle observationnel noté* $\mathcal{E}_o$ *de sorte que* $\text{obs}(\mathcal{E}_c) = \langle E_o, R_o, \mu_o^{\mathcal{E}}, \text{pre}_o, \text{post}_o \rangle$ *est définie ainsi :*

- $E_o = \{(\phi, e) \in \Phi \times E_c \mid \mu_{\text{pre}}(\phi)(e) > 0\}$;
- $\text{pre}_o((\phi, e)) = \phi$;
- $(\phi, e) R_{oa} (\phi', e')$ *ssi* $\exists e, e'$ *tel que* $e R_{ca} e'$;
- $\mu_{oa}^{\mathcal{E}}((\phi, e))((\phi', e')) = \mu_{ca}^{\mathcal{E}}(e)(e') \times \mu_{\text{pre}}(\phi')(e')$
- $\forall (\phi, e) \in E_o, \forall p \in V,\ \text{post}_o((\phi, e))(p) = \text{post}_c(e)(p)$

Nous pouvons voir ici le modèle observationnel $\mathcal{E}_o$ comme une précompilation du modèle $\mathcal{E}_c$, conservant toutes les informations, mais préagrégées.

Proposition 4.49. *Pour tout modèle conditionnel à préconditions séparées* $\mathcal{E}_c = \langle E_c, R_c^{\mathcal{E}}, \mu_c^{\mathcal{E}}, \Phi_c, \mu_{\text{pre}}, \text{post}_c \rangle$ *et pour toute structure de Kripke* $\mathcal{M} = \langle W, R, \mu, \text{val} \rangle$, *nous avons :*

$$\mathcal{M} \otimes \mathcal{E}_c \text{ est isomorphe à } \mathcal{M} \otimes \text{obs}(\mathcal{E}_c)$$

à une bijection près entre leurs ensembles de mondes.

Démonstration. Notons $\text{obs}(\mathcal{E}_c) = \mathcal{E}_o = \langle E_o, R_o, \mu_o^{\mathcal{E}}, \text{pre}_o, \text{post}_o \rangle$, $\mathcal{M}_c = \mathcal{M} \otimes \mathcal{E}_c = \langle W_c, R_c, \mu_c, \text{val}_c \rangle$ et $\mathcal{M}_o = \mathcal{M} \otimes \text{obs}(\mathcal{E}_c) = \langle W_o, R_o, \mu_o^{\mathcal{E}}, \text{val}_o \rangle$.

Nous nous rapportons aux définitions des *product update* pour les modèles observationnels (déf. 4.42) et pour les modèles conditionnels (déf. 4.28) et à la définition de la transformation obs (déf. 4.48).

Soit la bijection b telle que $\forall (w,e) \in W_c,\ b((w,e)) = (w,((\phi,e)))$ avec $\phi \in \Phi_c$ tel que $\langle \mathcal{M}, w\rangle \models \phi$.

Montrons que b est une fonction bijective. Pour tout couple $(w,e) \in W \times E_c$ avec $\mu_{\text{pre}}(w)(e) > 0$ (i.e. il existe un unique ϕ tel que $\mu_{\text{pre}}(\phi)(e) > 0$), on a un unique couple $(w,(\phi,e)) \in (W,(\Phi,E_c))$ car par définition $W_o = \{(w,(\phi,e)) \in (W,(\Phi,E_c)) \mid \langle \mathcal{M}, w\rangle \models \phi\}$. Donc b est bien une fonction. Par ailleurs, b est donc une fonction injective (pour chaque (w,e) on a un unique $(w,(\phi,e))$, mais aussi surjective, car $(w,(\phi,e))$ donne directement un unique (w,e). b est donc une bijection.

Soit $((w,e),(w',e')) \in W_c^2$ et $((w,(\phi,e)),(w',(\phi,e')) \in W_o^2 \mid b((w,e)) = (w,(\phi,e))$ et $b((w',e')) = (w',(\phi',e'))$.

Par définition, $(w,e)R_{ca}(w',e') \iff wR_aw'$ et $eR_{ca}^{\mathcal{E}}e'$ et $(w,(\phi,e))R_{oa}(w',(\phi',e')) \iff wR_aw'$ et $eR_{ca}^{\mathcal{E}}e'$ donc $(w,e)R_{ca}(w',e') \iff (w,(\phi,e))R_{oa}(w',(\phi',e'))$. Puisque nous avons $b((w,e)) = (w,(\phi,e))$ et $b((w',e')) = (w',(\phi',e'))$, on a donc : $(w,e)R_{ca}(w',e') \iff b((w,e))R_{ca}b((w',e'))$.

Par définition, on a : $\mu_{ca}((w,e))((w',e') = \mu_a(w)(w') \times \mu_{ca}^{\mathcal{E}}(e)(e') \times \mu_{\text{pre}}(w')(e')$

Or, on a aussi par définition : $\mu_{oa}((w,(\phi,e))((w',(\phi',e')) = \mu_a(w)(w') \times \mu_{oa}^{\mathcal{E}}(e)(e') = \mu_a(w)(w') \times \mu_{ca}^{\mathcal{E}}(e)(e') \times \mu_{\text{pre}}(w')(e')$ car $\mu_{oa}^{\mathcal{E}}(e)(e') = \mu_{ca}^{\mathcal{E}}(e)(e') \times \mu_{\text{pre}}(w')(e')$. Donc $\mu_{ca}((w,e))((w',e')) = \mu_{oa}((w,(\phi,e))((w',(\phi',e')) = \mu_{ca}(b((w,e)))(b((w',e'))$.

Enfin, $\text{val}_c((w,e)) = \{p \in V \mid \langle \mathcal{M}, w\rangle \models \text{post}_c(p)(e)\}$ et $\text{val}_o((w,(\phi,e))) = \{p \in V \mid \langle \mathcal{M}, w\rangle \models \text{post}_o(p)((\phi,e))\}$. Or $\text{post}_o((\phi,e))(p) = \text{post}_c(e)(p)$, donc $\text{val}_c((w,e)) = \text{val}_o((w,(\phi,e))) = \text{val}_o(b((w,e)))$.

Nous avons donc bien $\mathcal{M}_o$ est isomorphe à $\mathcal{M}_c$ par la bijection b.

□

Remarque 4.50. *La proposition précédente n'est pas valable dans le cas d'un modèle conditionnel général. En effet, il ne s'agira alors pas d'un isomorphisme mais d'une bisimulation ; chaque monde w sera alors multiplié x fois, où x est le nombre de préconditions pouvant s'appliquer à w.*

Généraliser cette proposition n'aurait pas d'utilité en pratique, car il s'avère peu intuitif de modéliser des probabilités conditionnelles « non séparées ».

Nous allons maintenant étudier le caractère calculatoire d'une telle transformation. Nous considérons que la taille d'un ensemble est son nombre d'éléments, que la taille d'une relation est son nombre d'éléments et que la taille d'une loi de probabilité discrète est la taille de son domaine, nous allons décrire les transformations des modèles conditionnels en modèles observationnels et inversement.

Proposition 4.51. *La transformation d'un modèle conditionnel en un modèle observationnel équivalent via* obs *est polynomiale.*

Démonstration. Par la transformation obs, nous avons $E_o : \Phi \times E_c$. R_o et $\mu_o^{\mathcal{E}}$ sont définies en fonction de E_o :

- $R_o : E_o \times E_o$;
- $\mu_o^{\mathcal{E}} : E_o \times E_o \to \mathbb{Q}_+$.

La transformation est donc polynomiale.

□

Remarque 4.52. *Que les structures soient normalisées ou non, la transformation est polynomiale dans les deux cas. Il suffirait de normaliser les probabilités.*

Nous avions déjà vu qu'il n'était pas facile de transformer un ensemble de formules (Φ) sans contraintes spécifiques, en un ensemble de formules mutuellement incohérentes (ex. 4.37) : au pire cas, cela peut s'avérer exponentiel. C'est ce qui rend la transformation d'un modèle observationnel en un modèle conditionnel à préconditions séparées complexe, car l'ensemble des formules de préconditions d'un modèle observationnel (qui n'ont aucune contraintes entre-elles) doivent être retranscrites dans un ensemble Φ, cette fois-ci dont les formules sont mutuellement incohérentes.

Ici, nous utilisons maintenant les définitions dénormalisées, donc sans contraintes sur les préconditions, ce qui nous permettra de montrer que la transformation d'un modèle observationnel en un modèle conditionnel est non plus exponentielle, mais polynomiale.

Définition 4.53. *Soit* $\mathcal{E}_o = \langle E, R, \mu^{\mathcal{E}}, \mathrm{pre}^o, \mathrm{post}\rangle$ *un modèle observationnel et* cond *la transformation de* $\mathcal{E}_o$ *en modèle conditionnel de sorte que* $\mathrm{cond}(\mathcal{E}_o) = \mathcal{E}_c = \langle E, R, \mu^{\mathcal{E}}, \Phi, \mu^c_{\mathrm{pre}}, \mathrm{post}\rangle$*, définie ci-dessous.*

Nous conservons les événements, les relations, les distributions de probabilités d'observations et les postconditions Nous définissons donc μ^c_{pre} *et* Φ *ainsi nous avons :*

- $\Phi = \{\mathrm{pre}^o(e) \mid e \in E\}$ *;*
- $\forall \phi \in \Phi \forall e \in E,\ \mu^c_{\mathrm{pre}}(\phi)(e) = \begin{cases} 1 & si\ \mathrm{pre}^o(e) = \phi \\ 0 & sinon \end{cases}$.

Clairement, nous pouvons voir que la supression du caractère mutuellement incohérent des formules épistémiques permet de créer des conditions préalables facilement manipulables.

Remarque 4.54. *Nous pouvons voir que s'il existe deux événements* e *et* e' *de sorte que* $\mathrm{pre}^o(e) = \mathrm{pre}^o(e')$*, alors ces doublons n'apparaîtront pas dans* Φ *à cause de la définition d'un ensemble. Notons que ce changement de nombre de préconditions (de sorte que* $|\Phi| \neq |E|$*) n'est pas problématique, car cela ne change strictement rien à la création de* μ^c_{pre}*.*

Proposition 4.55. *La transformation* cond *est polynomiale.*

Démonstration. Nous avons simplement $|\Phi| = |E|$ et μ^c_{pre} est la réunion de $|E|$ loteries.

□

Montrons maintenant que nous avons bien une équivalence entre les mises à jour des connaissances.

Proposition 4.56. *Pour toute structure $\mathcal{M}$ et modèle observationnel $\mathcal{E}$, nous avons :*

$$\mathcal{M} \otimes \mathcal{E} = \mathcal{M} \otimes \text{cond}(\mathcal{E})$$

Démonstration. Soit $\mathcal{E}_o = \langle E, R, \mu^{\mathcal{E}}, \text{pre}^o, \text{post}\rangle$ un modèle observationnel et cond la transformation de $\mathcal{E}_o$ en modèle conditionnel de sorte que $\text{cond}(\mathcal{E}_o) = \mathcal{E}_c = \langle E, R, \mu^{\mathcal{E}}, \Phi, \mu^c_{\text{pre}}, \text{post}\rangle$ (déf. 4.53).

Soit $\mathcal{M} = \langle W, R, \mu, \text{val}\rangle$, $\mathcal{M}_o = \mathcal{M} \otimes \mathcal{E}_o = \langle W_o, R_o, \mu^{\mathcal{E}}_o, \text{val}_o\rangle$ et $\mathcal{M}_c = \mathcal{M} \otimes \text{cond}(\mathcal{E}) = \langle W_c, R_c, \mu^{\mathcal{E}}_c, \text{val}_c\rangle$.

Nous pouvons comparer tous les éléments de $\mathcal{M}_o$ et $\mathcal{M}_c$ grâce aux définitions de cond (déf. 4.53), et des *product updates* avec les modèles observationnels (déf. 4.42) et les modèles conditionnels (déf. 4.28).

Pour les mondes, nous avons :

- $W_c = \{(w, e) \in W \times E \mid \mu^c_{\text{pre}}(w)(e) > 0\}$, or par définition de $\mu^c_{\text{pre}}(w)(e)$ et du *product update* (déf. 4.28), on a :

$$\mu^c_{\text{pre}}(w)(e) = \frac{\sum_{\phi \in \Phi \mid \langle \mathcal{M}, w\rangle \models \phi} \mu^c_{\text{pre}}(\phi)(e)}{\sum_{\phi \in \Phi \mid \langle \mathcal{M}, w\rangle \models \phi} \sum_{e' \in E} \mu^c_{\text{pre}}(\phi)(e')}$$

qui est positif si et seulement s'l existe un $\phi \in \Phi$ tel que $\langle \mathcal{M}, w\rangle \models \phi$, i.e. $\text{pre}^o(e) = \phi$, par construction de $\text{cond}(\mathcal{E}_o)$ (sinon, $\mu^c_{\text{pre}}(w)(e) = 0$).

Autrement dit, $\forall \phi \in \Phi$, $\mu^c_{\text{pre}}(w)(e) = 1$ ssi $\langle \mathcal{M}, w\rangle \models \text{pre}^o(e)$, sinon 0, donc $W_c = \{(w, e) \in W \times E \mid \langle \mathcal{M}, w\rangle \models \text{pre}^o(e)\} = W_o$;

Pour ce qui est de relations, comme $R^{\mathcal{E}}$ ne subit pas de transformation, elles sont identiques de sorte que :

- $(w_1, e_1) R_{ca} (w_2, e_2)$ ssi $w_1 R_a w_2$ et $e_1 R^{\mathcal{E}}_a e_2$ ssi $(w_1, e_1) R_{oa} (w_2, e_2)$;

Pour ce qui est des probabilités, voici l'égalité :

- $\mu_{ca}((w_1, e_1))((w_2, e_2))$
  $= \mu_a(w_1)(w_2) \times \mu^c_{\text{pre}}(w_2)(e_2) \times \mu^{\mathcal{E}}_a(e_1)(e_2)$
  $= \mu_a(w_1)(w_2) \cdot \mu^{\mathcal{E}}_a(e_1)(e_2)$

$= \mu_{oa}((w_1, e_1))((w_2, e_2))$

car $\mu^c_{\text{pre}}(w_2)(e_2) = 1$ ssi $\langle \mathcal{M}, w_2 \rangle \models \text{pre}^o(e_2)$, ce qui est la condition pour que (w_2, e_2) existe ;

Enfin, les définitions des fonctions de valuations sont identiques :

- $\text{val}_c((w, e)) = \{p \in PS \mid \langle \mathcal{M}, w \rangle \models \text{post}(p)(e)\} = \text{val}_o((w, e))$

□

Nous pouvons donc voir que la suppression des formules mutuellement incohérentes de Φ nous permet d'obtenir une transformation polynomiale au lieu d'exponentielle.

5 | Model checking symbolique pour PEL

Nous avons précédemment vu comment nous pouvions représenter symboliquement les structures de Kripke et faire du *model checking* dessus grâce aux travaux de Gattinger. Nous avons aussi présenté une variante probabiliste de DEL : PDEL, et nous avons présenté une variante des modèles d'événements de PDEL : les modèles observationnels. Nous avons ensuite présenté des variantes aux définitions de PDEL, en utilisant des loteries au lieu des distributions de probabilités et en s'affranchissant des contraintes imposées par l'ensemble de formules mutuellement incohérentes Φ.

Grâce à ces définitions, nous allons commencer par représenter la logique épistémique probabiliste symbolique de manière statique, sans mise à jour des croyances ou des probabilités (sec. 5.1), et nous y montrerons comment faire du *model checking* (sec. 5.2).

5.1 Représentation symbolique

Rappelons la définition d'une structure de Kripke probabilisée : $\mathcal{M} = \langle W, R, \mu, \text{val}\rangle$. Nous avions décrit une représentation symbolique de W (déf. 2.8, page 36), une représentation symbolique de R (déf. 2.12, page 38), nous allons maintenant définir une représentation symbolique de μ.

Rappelons que la représentation symbolique d'une structure de Kripke est une structure de croyances notée $\mathfrak{F} = \langle V, \theta, \Omega\rangle$, où V est le vocabulaire utilisé pour décrire les mondes, θ est la loi des mondes décrivant tous les mondes légaux et Ω est un ensemble de formules booléennes sur $\mathcal{L}_{\text{prop}}(V \cup V^+)$ décrivant les relations symboliques entre les mondes pour chaque agent.

Ω était défini comme une formule booléenne, mais nous pouvons l'écrire comme étant une *fonction* booléenne.

Voici comment Ω est maintenant utilisé. Grâce à Ω_a nous avions : $\forall(s,t) \in$

$2^V \times 2^V$, si $s \cup t' \models \Omega_a$, alors il y a une relation symbolique entre les valuations s et t. Sous une formulation fonctionnelle, nous obtenons : $\forall (s,t) \in 2^V \times 2^V$, $\Omega_a(s \cup t') = 1$ s'il existe une relation, sinon 0.

Afin de représenter symboliquement les loteries, nous allons donc utiliser la même méthode que pour représenter des relations (i.e. via Ω), non pas avec des fonctions booléennes, mais avec des PBFs. Le domaine des valeurs ne sera bien évidemment plus $\{0,1\}$ mais des valeurs rationnelles $\mathbb{Q}$ (et plus généralement $\overline{\mathbb{Q}}$, voir déf. 3.7) qui seront fournies par les ensembles de loteries μ. Cette représentation symbolique des ensembles de loteries sera notées Π.

Nous obtenons donc naturellement la définition d'une structure de Kripke symbolique probabiliste, basée sur la définition d'une structure de croyances (déf. 2.14, page 38). Nous appellerons cette nouvelle structure *une structure de probabilités* par souci de concision, mais il faut comprendre le terme *structure de croyances probabiliste*.

Définition 5.1 (structure de probabilités)**.** Une structure de probabilités *est un tuple* $\mathfrak{F} = \langle V, \theta, \Omega, \Pi \rangle$ *tel que :*

- $V \subseteq PS$ *est un ensemble fini de variables propositionnelles appelé* vocabulaire*,*
- θ *est une fonction booléenne sur* V*, appelée* la loi des états *;*
- Ω *associe à chaque agent* $a \in \mathfrak{A}$ *une fonction booléenne* Ω_a *sur* $V \cup V'$*, appelée la* loi d'observations *;*
- Π *associe à chaque agent* $a \in \mathfrak{A}$ *une PBF* Π_a *sur* $V \cup V'$*, appelée* la loi de probabilités*.*

Tout $s \in 2^V$ *tel que* $s \models \theta$ *est appelé un* état *de* $\mathfrak{F}$ *; le couple* $\langle \mathfrak{F}, s \rangle$ *est appelé* une structure probabiliste pointée*.*

Voici comment on peut représenter l'ensemble des loteries μ_a, en utilisant la notation des PBFs (voir rq. 3.9).

Définition 5.2. *La représentation symbolique d'une* $X-$*loterie* ℓ *via une fonction d'étiquetage* λ^V_X*, notée* $\mathrm{symb}_{\lambda^V_X}(\ell)$*, est la PBF définie sur* V

$$\{\lambda^V_X(x) \sqsubseteq X : \ell(x) \mid x \in X\}$$

Nous pouvons définir la représentation symbolique d'un ensemble de loteries de la manière ci-dessous, en utilisant la notation d'une PBF vue précédemment (rq. 3.9, page 76) et la définition de $\sqsubseteq$ (déf. 2.2, page 34).

Définition 5.3. *La représentation symbolique d'un ensemble de* $X-$*loteries* L *de la forme* $X \to (X \to \mathbb{Q}_+)$ *via une fonction d'étiquetage* λ^V_X*, notée* $\mathrm{symb}_{\lambda^V_X}(L)$*, est la PBF définie sur* $V \cup V'$

$$\{\lambda^V_X(x_1) \sqsubseteq X \wedge (\lambda^V_X(x_2) \sqsubseteq X)' : L(x_1)(x_2) \mid (x_1, x_2) \in X^2\}$$

Notons que cette notation est possible grâce à la fonction d'étiquetage qui permet que toutes les formules dans la PBF soient mutuellement incohérentes, car par définition, la fonction d'étiquetage est injective.

Avec cette définition, nous pouvons définir une représentation symbolique d'une structure de Kripke probabiliste, en se basant sur la définition précédemment vue d'une représentation symbolique d'une structure de Kripke classique (déf. 2.15).

Définition 5.4 (Représentation symbolique d'une *frame* probabiliste)**.** *La* representation symbolique *d'une* frame *probabiliste (déf. 4.1, page 94),* $\mathcal{F} = \langle W, R, \mu \rangle$ *via une fonction d'étiquetage* λ^V_W*, notée* $\mathrm{symb}_{\lambda^V_W}(\mathcal{F})$*, est la structure de probabilités*

$$\langle V,\ \mathrm{symb}_{\lambda^V_W}(W),\ (\mathrm{symb}_{\lambda^V_W}(R_a))_{a\in\mathfrak{A}},\ (\mathrm{symb}_{\lambda^V_W}(\mu_a))_{a\in\mathfrak{A}}\rangle$$

La représentation symbolique d'une structure de Kripke probabiliste se fait de la même manière, via val *comme fonction d'étiquetage si* val *est valuation-injective, ou* val *modifié sur un vocabulaire étendu pour la rendre valuation-injective, sinon.*

Remarque 5.5. *La définition de la représentation symbolique d'une structure de Kripke valuation-injective est ainsi la structure de probabilités*

$$\mathrm{symb}_{\lambda^V_W}(\mathcal{M}) = \langle V,\ \mathrm{symb}_{\mathrm{val}}(W),\ (\mathrm{symb}_{\mathrm{val}}(R_a))_{a\in\mathfrak{A}},\ (\mathrm{symb}_{\mathrm{val}}(\mu_a))_{a\in\mathfrak{A}}\rangle$$

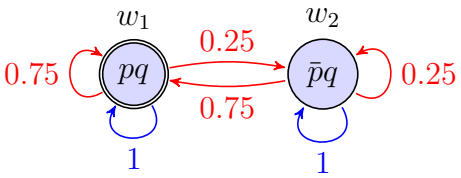


FIGURE 5.1 – Exemple d'un modèle de Kripke $\mathcal{M}$ probabiliste (ex. 5.6)

Exemple 5.6. *Reprenons l'exemple de structure de Kripke non probabiliste (ex. 1.1, page 15) transformée en structure de croyances (ex. 2.16, page 39). Admettons que nous ajoutions des probabilités de sorte que* $\mu_1(w_1)(w_1) = 1$*,* $\mu_1(w_2)(w_2) = 1$*,* $\mu_2(w_1)(w_1) = 0.75$*,* $\mu_2(w_1)(w_2) = 0.25$*,* $\mu_2(w_2)(w_1) = 0.75$ *et* $\mu_2(w_2)(w_2) = 0.25$*, de sorte à obtenir la figure 5.1. Rappelons que* $\mathrm{val}(w_1) = \{p, q\}$ *et* $\mathrm{val}(w_2) = \{q\}$*. Les représentations symboliques des* μ_a *sont :*

- $\mathrm{symb}\,\lambda^V_W(R_1) = \{pqp'q' : 1,\ \neg pq\neg p'q : 1\} = \{pqp'q' \vee \neg pq\neg p'q : 1\}$
- $\mathrm{symb}\,\lambda^V_W(R_2) = \{pqp'q' : 0.75,\ \neg pq\neg p'q' : 0.25,\ pq\neg p'q' : 0.25,\ \neg pqp'q' : 0.75\}$

De manière inverse à la définition ci-dessus, et en appliquant la définition 2.20, page 40 permettant de passer d'une structure de croyances à une structure de Kripke, nous pouvons définir la fonction expl($\mathfrak{F}$). Pour cela, il nous faut définir expl(Π).

Définition 5.7 (Représentation explicite d'une loi de probabilités)**.** *La représentation explicite d'une loi de probabilités* Π_a *définie sur* $V \cup V'$*, via un ensemble* X *est un ensemble de* X*-loteries noté* expl_X *(*$X \to (X \to \mathbb{Q}_+)$*) tel que :*

$$\forall (x_1, x_2) \in X^2,\ \text{expl}_X(\Pi_a)(x_1)(x_2) = \Pi_a(x_1 \cup x_2')$$

La transformation est vraiment symétrique à la précédente, les valeurs présentes dans Π_a sont copiées dans ces loteries $\text{expl}_X(\Pi_a)$.

Il en découle donc la définition de la représentation explicite d'une structure de probabilités, mêlant la définition 2.20, page 40, sur les structures non probabilistes et la définition ci-dessus 5.7.

Définition 5.8 (Représentation explicite d'une structure de probabilités)**.** *La* representation explicite *d'une structure probabiliste* $\mathfrak{F} = \langle V, \theta, \Omega, \Pi \rangle$ *pour un vocabulaire* V*, notée* expl($\mathfrak{F}$)*, avec* $W = \text{expl}(\theta)$ *et l'application identité* $\mathsf{id}_X \colon X \to X,\ x \to x$*, est la structure de Kripke*

$$\langle W,\ (\text{expl}_W(\Omega_a))_{a \in \mathfrak{A}}, (\text{expl}_W(\Pi_a))_{a \in \mathfrak{A}}, \mathsf{id}_W \rangle$$

Remarque 5.9. *La remarque que nous avions faite (rq. 2.18, page 40) est toujours valable : un monde est véritablement une valuation ici.*

Il s'agit ici de simplement recopier toutes les valeurs présentes dans les PBFs $(\Pi_a)_{a \in \mathfrak{A}}$ dans les distributions de probabilités $(\mu_a)_{a \in \mathfrak{A}}$. Notons qu'une distribution de probabilité est bien créée pour chaque monde de W, ce que nous souhaitons. De plus, le domaine de chaque distribution de probabilités $\mu_a(w)$ pour chaque agent a est bien l'ensemble des mondes possibles défini par W, issu de θ, ce qui correspond exactement à nos attentes.

Maintenant que nous avons défini les structures de probabilités, représentation symbolique des structures de Kripke probabilistes, nous allons pouvoir voir comment se passe le *model checking* sur celles-ci de manière symbolique.

5.2 Model checking sur les structures de probabilités

Pour effectuer le *model checking* sur des structures de probabilités nous aurons besoin de la définition de la fonction ci-dessous : nonzero.

Définition 5.10 (Nonzero)**.** *Soit* $f\colon X \to \mathbb{Q}$*. La fonction* $\text{nonzero}(f)\colon X \to \overline{\mathbb{Q}}$ *est la fonction qui associe* v *à* $f(v)$ *si* $f(v) \neq 0$ *et à* ∞ *sinon.*

Autrement dit, $\text{nonzero}(f)$ est identique à la fonction f sauf dans le cas où $f(v)$ est nul : dans ce cas là, $\text{nonzero}(f)(v) = \infty$. La fonction nonzero est une astuce pour pouvoir gérer les fonctions au support vide, notamment lorsqu'elles sont au dénominateur d'une fraction.

Π_a représente un ensemble de W-loteries, une par monde de W. Nous devons introduire la normalisation de Π_a, $\text{N}(\Pi_a)$, qui normalise toutes les loteries de Π_a simultanément grâce aux fonctions nonzero et marginalisation.

Nous savons que Π_a n'est pas exactement la représentation symbolique d'une W-loterie pour un agent a, elle est plutôt la représentation symbolique d'un ensemble de W-loteries. C'est pourquoi, la normalisation de Π_a notée $\text{N}(\Pi_a)$ n'est pas suffisante pour obtenir la représentation symbolique d'une seule distribution de probabilité. Ainsi, nous avons besoin de définir $\text{N}_\theta(\Pi_a)$, la normalisation de la fonction Π_a grâce à la marginalisation des variables V'.

Définition 5.11 (Normalisation de Π_a : $\text{N}_\theta()$)**.** *La normalisation* $\text{N}_\theta(\Pi_a)$ *de la fonction* Π_a *définie sur* $V \cup V'$ *via une loi des mondes* θ *est définie ainsi :*

$$\forall v \in 2^{V \cup V^\star},\ \text{N}_\theta(\Pi_a)(v) = \frac{\Pi_a(v)}{\text{nonzero}(\text{Marg}^{\dagger}_{V'}(\Pi_a \cdot \theta'))(v \setminus V')}$$

Voici comment fonctionne cette normalisation :

- tout d'abord, il faut récupérer les valeurs dans la fonction Π_a qui sont au numérateur ;
- ensuite, il faut pouvoir diviser par la somme des loteries qui succèdent à une valuation précise v, ce qui est spécifiquement la maginalisation sur V'. Comme la dénominateur est alors défini seulement sur l'ensemble V, il faut retirer V' du paramètre de la fonction ;
- l'utilisation de la fonction nonzero permet de diviser par des valeurs qui seraient nulles. De cette manière, en remplaçant les valeurs nulles par l'infini, la valeur globale devient 0. Notons que si le dénominateur est 0, alors le numérateur l'est aussi, ce qui donnerait la fraction 0/0, en effet $\text{Marg}^{\dagger}_{V'}(\Pi_a)(v \setminus V') = 0$ implique que $\Pi_a(v) = 0$.

Remarque 5.12. *Pourquoi la présence de* θ *est-elle nécessaire lors de la normalisation de* Π_a *?*

En effet, pour obtenir des probabilités réelles, il faut bel et bien qu'il y ait θ' *dans le calcul de la normalisation de* Π_a*. Si des arcs symboliques dans* Π_a *sont possibles vers des mondes non existants, les probabilités réelles résultantes après normalisation seraient faussées. En effet, s'il existe d'autres valeurs non pertinantes, elles*

seront quand même sommées alors qu'elles n'ont pas de sens. Ainsi, après normalisation nous obtenons les probabilités réelles sur lesquelles seront appliquées les opérations du model checking.

Exemple 5.13. *Prenons* $V = \{p\}$, $\theta = p$, $\Pi_a = \{\top : 1\}$.

Nous montrons les résultats de la normalisation de Π_a *sans l'utilisation de* θ *et de la normalisation définie ci-dessous dans la table 5.1.*

Variables V		Dénominateur	Normalisation
p	p'	$\mathrm{Marg}^{+}_{V'}(\Pi_a)$	$\frac{\Pi_a}{\mathrm{Marg}^{+}_{V'}(\Pi_a)}$
0	0	2	0.5
0	1	2	0.5
1	0	2	0.5
1	1	2	0.5
p	p'	$\mathrm{Marg}^{+}_{V'}(\Pi_a \cdot \theta')$	$\frac{\Pi_a}{\mathrm{Marg}^{+}_{V'}(\Pi_a \cdot \theta')}$
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

TABLE 5.1 – Comparaison des calculs de création de Π_a à partir de Ω_a

Dans le cas sans θ*, la normalisation se fait comme s'il existait des valeurs entre un état* $\{p\}$ *et état non existant* $\emptyset$*.*

Tout comme les définitions proposées dans la section 2.3 afin de définir une sémantique sur une structure de croyances, nous allons définir une sémantique pour le langage $\mathcal{L}_{\mathrm{PEL}}$ sur une structure de probabilités et ensuite, nous allons définir une traduction locale. Les deux définitions étendent les définitions précédentes pour répondre à la modélisation des probabilités. Nous pouvons faire cela grâce à la normalisation de Π_a qui permet de récupérer les probabilités réelles représentées par la loterie Π_a, afin de pourvoir comparer ces probabilités à la constante β.

Définition 5.14. *La sémantique de* $\mathcal{L}_{\mathrm{PEL}}(V)$ *sur une structure de probabilités pointée* $\langle\langle V, \theta, \Omega, \Pi\rangle, s\rangle$ *(basée sur la définition 2.23, page 41) est définie inductivement comme suit :*

- $\langle \mathfrak{F}, s\rangle \models \top$ *est vraie ;*
- $\langle \mathfrak{F}, s\rangle \models p$ *ssi* $s \models p$ *pour* $p \in V$ *;*
- $\langle \mathfrak{F}, s\rangle \models \neg\phi$ *ssi* $\langle \mathfrak{F}, s\rangle \not\models \phi$ *;*

- $\langle \mathfrak{F}, s\rangle \models \phi \wedge \psi$ *ssi* $\langle \mathfrak{F}, s\rangle \models \phi$ *et* $\langle \mathfrak{F}, s\rangle \models \psi$;
- $\langle \mathfrak{F}, s\rangle \models \Box_i \phi$ *ssi pour tout état* t *de* $\mathfrak{F}$: $s \cup t' \models \Omega_i$ *implique* $\langle \mathfrak{F}, t\rangle \models \phi$;
- $\langle \mathfrak{F}, s\rangle \models \alpha_1 \mathrm{Pr}_a(\phi_1) + \cdots + \alpha_k \mathrm{Pr}_a(\phi_k) \geq \beta$ *ssi*

$$\sum_{i=1}^{k} \alpha_i \left(\sum_{\text{pour tout état } t \text{ de } \mathfrak{F} \text{ tel que } \langle \mathfrak{F}, t\rangle \models \phi_i} \mathrm{N}_\theta(\Pi_a)(s \cup t') \right) \geq \beta$$

Cette sémantique manipule des opérateurs sur des PBFs. Nous en profitons pour rappeler brièvement la notation de la multiplication ou l'addition de deux fonctions qui ont un domaine de définition identique : f et g : $(f + g)(x) = f(x) + g(x)$ et $(f \cdot g)(x) = f(x) \times g(x)$. Pour ce qui est de la division, nous pouvons faire de même, néanmoins, il faut s'assurer que le dénominateur soit non nul.

Nous pouvons éventuellement utiliser des arguments « sur-spécifiés » pour des fonctions. Par exemple, avec $f : 2^X \to \overline{\mathbb{Q}}$ et $g : 2^Y \to \overline{\mathbb{Q}}$ avec $X \subseteq Y$, nous pouvons écrire, pour $v \in 2^Y$ $(f \times g)(v) = f(v) \times g(v)$, où il faudra comprendre $(f \cdot g)(v) = f(v \setminus (Y \setminus X)) \times g(v)$ où $v \setminus (Y \setminus X)$ est simplement la valuation v restreinte aux arguments du domaine de définition de $f : X$.

Comme Gattinger a pu définir la traduction locale des formules épistémiques, nous allons faire de même avec les formules épistémiques probabislites. Pour cela, il nous suffit de reprendre la traduction locale des formules épistémiques (déf. 2.26, page 43) et d'y ajouter le cas de l'opérateur probabiliste Pr.

Définition 5.15 (Traduction locale d'une formule)**.** *Soit* $\mathfrak{F} = \langle V, \theta, \Omega, \Pi\rangle$ *une structure probabiliste et* ϕ *une formule de* $\mathcal{L}_{\mathrm{PEL}}(V)$*. La* traduction locale de ϕ sur $\mathfrak{F}$*, notée* $||\phi||_{\mathfrak{F}}$*, est la fonction booléenne définie inductivement ainsi :*

- $||\top||_{\mathfrak{F}} \coloneqq \top$
- $||p||_{\mathfrak{F}} \coloneqq p$
- $||\neg\phi||_{\mathfrak{F}} \coloneqq \neg||\phi||_{\mathfrak{F}}$
- $||\phi \wedge \psi||_{\mathfrak{F}} \coloneqq ||\phi||_{\mathfrak{F}} \wedge ||\psi||_{\mathfrak{F}}$
- $||\Box_a\phi||_{\mathfrak{F}} \coloneqq \mathrm{Forget}^{\forall}_{V'}((\Omega_a \wedge \theta') \to (||\phi||_{\mathfrak{F}})')$
- $||\alpha_1 \mathrm{Pr}_a(\phi_1) + \cdots + \alpha_k \mathrm{Pr}_a(\phi_k) \geq \beta||_{\mathfrak{F}} \coloneqq$

$$\mathrm{Cut}_{\geq}(\Sigma_{i=1}^{k} \alpha_i \times \mathrm{Marg}^{\star}_{V'}(\Pi_a \cdot \theta' \cdot (||\phi_i||_{\mathfrak{F}})'), \beta \times \mathrm{nonzero}(\mathrm{Marg}^{\star}_{V'}(\Pi_a \cdot \theta')))$$

La traduction locale pour la formule probabiliste utilise les définitions de la marginalisation (déf. 3.15), de la coupe (déf. 3.17) et de la fonction nonzero (déf. 5.10). La traduction locale permet de retrouver une formule propositionnelle définie sur V dont le support est l'ensemble des valuations des mondes qui satisfont la formule.

Voici l'idée derrière cette nouvelle traduction locale. Tout d'abord, la formule $\Pi_a \cdot \theta' \cdot ||\phi_i||'_{\mathfrak{F}}$ permet de récupérer la PBF sur $V \cup V'$ représentant toutes les loteries menant à des mondes légaux (via θ') qui satisfont ϕ (via $||\phi_i||'_{\mathfrak{F}}$). Ensuite, la première marginalisation permet de récupérer la PBF définie sur V qui a pour support l'ensemble des valuations des mondes prédécesseurs dont les successeurs satisfont ϕ et qui associe à chacune de ces valuations la somme des loteries sortantes de cette valuation. Ce processus est itéré pour chaque formule ϕ_i en pondérant les fonctions par les cœfficients α_i.

Une astuce est utilisée pour la normalisation de la fonction Π_a. En effet, nous souhaitons pouvoir passer des loteries à des probabilités, il nous faut donc normaliser Π_a. Pour cela, nous divisons Π_a par la marginalisation de Π_a sur le vocabulaire V' : c'est la définition de la normalisation de Π_a. Nous passons simplement la dénominateur de cette normalisation dans le membre de droite de l'inégalité et l'utilisons comme constante pour n'avoir à faire ce calcul qu'une seule et unique fois au lieu de i fois.

Enfin, la fonction coupe permet de récupérer tous les modèles dont les valeurs sont supérieures au seuil désiré β.

La résultat final est donc une fonction booléenne représentant l'ensemble des modèles qui satisfont la formule probabiliste, et revient à la sémantique voulue pour PDEL dénormalisé. De la même manière que Gattinger le montre dans son théorème (que nous avons repris en th. 2.30 : $\langle \mathfrak{F}, s\rangle \models \phi$ ssi $s \models ||\phi||_{\mathfrak{F}}$), nous allons montrer que la traduction locale d'une formule probabiliste préserve et reflète la vérité. Nous étendons donc le théorème 2.30 précédent, dont nous avons déjà présenté la preuve. Comme tous les cas non probabilistes sont déjà montrés, nous nous contenterons de simplement faire le cas probabiliste ici.

Afin de prouver ce théorème, nous utiliserons ce lemme.

Lemme 5.16. *Soit $\nu \in \mathbb{Q}$, $\beta \in \mathbb{Q}$, $\delta \in \overline{\mathbb{Q}}^*_+$. Si $\nu \neq 0 \implies \delta \neq \infty$, alors nous avons $\nu/\delta \geq \beta \iff \nu \geq \beta \times \delta$.*

Démonstration. Si $\delta \neq \infty$ alors le résultat découle des règles habituelles des inéquations puisque δ est positif.

Supposons donc $\delta = \infty$. Nous avons alors $\nu = 0$ et donc la chaîne d'équivalences suivante, en utilisant le fait que $0/\infty = 0 \times \infty = 0$ (voir remarque 3.7, page 76), $\nu/\delta \geq \beta \iff 0 \geq \beta \iff 0 \times \delta \geq \beta \times \delta \iff 0 \geq \beta \times \delta \iff \nu \geq \beta \times \delta$. □

Théorème 5.17. *La traduction locale (déf. 5.15) préserve et reflète la vérité. C'est à dire, pour toute formule $\phi \in \mathcal{L}_{\text{PEL}}$ et pour structure de probabilités pointée $\langle \mathfrak{F}, s\rangle$, on a*

$$\langle \mathfrak{F}, s\rangle \models \phi \text{ ssi } s \models ||\phi||_{\mathfrak{F}}$$

Démonstration. Prouvons que $\langle \mathfrak{F}, s\rangle \models \alpha_1 \mathrm{Pr}_a(\phi_1) + \cdots + \alpha_k \mathrm{Pr}_a(\phi_k) \geq \beta$ si et seulement si s est une valuation (monde) présent dans les modèles de la traduction locale $||\alpha_1 \mathrm{Pr}_a(\phi_1) + \cdots + \alpha_k \mathrm{Pr}_a(\phi_k) \geq \beta||_{\mathfrak{F}}$.

$\langle \mathfrak{F}, s\rangle \models \alpha_1 \mathrm{Pr}_a(\phi_1) + \cdots + \alpha_k \mathrm{Pr}_a(\phi_k) \geq \beta$

$$\iff \sum_{i=1}^{k} \alpha_i \left(\sum_{\substack{\text{pour tout état } t \text{ de } \mathfrak{F} \\ \text{tel que } \langle \mathfrak{F}, t\rangle \models \phi_i}} \mathrm{N}(\Pi_a)(s \cup t') \right) \geq \beta \text{ (déf. 5.14)} \tag{5.1}$$

par la déf. de $\mathrm{N}(\Pi_a)$ 5.11 :

$$\iff \sum_{i=1}^{k} \alpha_i \left(\sum_{\substack{\text{pour tout état } t \text{ de } \mathfrak{F} \\ \text{tel que } \langle \mathfrak{F}, t\rangle \models \phi_i}} \frac{\Pi_a(s \cup t')}{\mathrm{nonzero}\left(\mathrm{Marg}^{+}_{V'}\left(\Pi_a \cdot \theta'\right)\right)(s)} \right) \geq \beta \tag{5.2}$$

Ici, nous avons $\mathrm{nonzero}\left(\mathrm{Marg}^{+}_{V'}\left(\Pi_a \cdot \theta'\right)\right)(s)$ qui est une constante positive (vis-à-vis de t et des ϕ_i). Nous pouvons donc sortir le dénominateur des sommes, puis le changer de membre en appliquant le lemme 5.16 (le dénominateur est toujours strictement positif et ne peut pas être infini si le numérateur est non nul) :

$$\iff \sum_{i=1}^{k} \alpha_i \left(\sum_{\substack{\text{pour tout état } t \text{ de } \mathfrak{F} \\ \text{tel que } \langle \mathfrak{F}, t\rangle \models \phi_i}} \Pi_a(s \cup t') \right) \geq \beta \times \mathrm{nonzero}\left(\mathrm{Marg}^{+}_{V'}\left(\Pi_a \cdot \theta'\right)\right)(s) \tag{5.3}$$

Par induction sur les formules et par la définition d'un état de $\mathfrak{F}$ (déf. 2.14)

$$\iff \sum_{i=1}^{k} \alpha_i \left(\sum_{t \in 2^V \text{tel que } t \models \theta \text{ et } t \models ||\phi_i||_{\mathfrak{F}}} \Pi_a(s \cup t') \right) \geq \beta \times \mathrm{nonzero}\left(\mathrm{Marg}^{+}_{V'}\left(\Pi_a \cdot \theta'\right)\right)(s) \tag{5.4}$$

0 est l'élement neutre de la somme, or pour une formule ϕ, vue comme une PBF, on a $\phi(t) = 0$ quand $t \not\models \phi$. Nous pouvons donc sortir la condition de la somme pour en faire une multiplication. La somme s'effectue donc sur toutes les valuations possible sur V

$$\iff \sum_{i=1}^{k} \alpha_i \left(\sum_{t \in 2^V} \Pi_a(s \cup t') \times (\theta \cdot ||\phi_i||_{\mathfrak{F}})(t) \right) \geq \beta \times \mathrm{nonzero}\left(\mathrm{Marg}^{+}_{V'}\left(\Pi_a \cdot \theta'\right)\right)(s) \tag{5.5}$$

Nous effectuons un simple renommage de t en t', de θ en θ' et $||\phi_i||_{\mathfrak{F}}$ en $||\phi_i||'_{\mathfrak{F}}$

$$\iff \sum_{i=1}^{k} \alpha_i \left(\sum_{t \in 2^V} \Pi_a(s \cup t') \times (\theta' \cdot ||\phi_i||'_{\mathfrak{F}})(t') \right) \geq \beta \times \text{nonzero}\left(\text{Marg}^{+}_{V'}\left(\Pi_a \cdot \theta'\right)\right)(s) \tag{5.6}$$

$$\iff \sum_{i=1}^{k} \alpha_i \left(\sum_{t \in 2^V} \left(\Pi_a \cdot \theta' \cdot ||\phi_i||'_{\mathfrak{F}}\right)(s \cup t') \right) \geq \beta \times \text{nonzero}\left(\text{Marg}^{+}_{V'}\left(\Pi_a \cdot \theta'\right)\right)(s) \tag{5.7}$$

Par la définition de la marginalisation (déf. 3.15)

$$\iff \sum_{i=1}^{k} \alpha_i \left(\text{Marg}^{+}_{V'}\left(\Pi_a \cdot \theta' \cdot ||\phi_i||'_{\mathfrak{F}}\right)\right)(s) \geq \beta \times \text{nonzero}\left(\text{Marg}^{+}_{V'}\left(\Pi_a \cdot \theta'\right)\right)(s) \tag{5.8}$$

Cette inégalité est vraie si pour un argument donné s, le membre de gauche est supérieur ou égal au membre de droite. C'est la définition de la coupe (déf. 3.17), qui permet de retrouver une fonction booléenne.

$$\iff \left(\text{Cut}_{\geq}\left(\Sigma_{i=1}^{k}\alpha_i \times \text{Marg}^{+}_{V'}\left(\Pi_a \cdot \theta' \cdot ||\phi||'_{\mathfrak{F}}\right), \beta \times \text{nonzero}\left(\text{Marg}^{+}_{V'}\left(\Pi_a\right)\right)\right)\right)(s) \tag{5.9}$$

$$\iff s \models ||\alpha_1 \text{Pr}_a(\phi_1) + \cdots + \alpha_k \text{Pr}_a(\phi_k) \geq \beta||_{\mathfrak{F}} \tag{5.10}$$

□

Nous voulons montrer que le *model checking* sur une structure explicite ou sur une structure symbolique, via les transformations symb et expl, sont équivalentes. Pour cela, nous devons montrer l'égalité des normalisations explicites et symboliques.

Lemme 5.18 (Égalité des normalisations)**.** *Soit un vocabulaire V, W un ensemble de mondes, λ^V_W une fonction d'étiquetage, μ_a une ensemble de $W-$loteries ($W \to (W \to \mathbb{Q}_+)$), Π_a une loi de probabilités définie sur $V \cup V^+$ et $\theta = \text{symb}_{\lambda^V_W}(W)$.*

Si pour tout $w_1, w_2 \in W$ on a $\Pi_a(\lambda^V_W(w_1) \cup \lambda^V_W(w_2)') = \mu_a(w_1)(w_2)$ alors nous avons $\text{N}_\theta(\Pi_a)(\lambda^V_W(w_1) \cup \lambda^V_W(w_2)') = \text{N}(\mu_a(w_1))(w_2)$.

Démonstration. Par la définition de la normalisation de Π_a (déf. 5.11), nous avons $\forall w_1, w_2 \in W$

$$\mathrm{N}_\theta(\Pi_a)(\lambda_W^V(w_1) \cup \lambda_W^V(w_2))$$

$$= \left(\frac{\Pi_a}{\mathrm{nonzero}(\mathrm{Marg}^+_{V'}(\Pi_a \cdot \theta'))} \right) (\lambda_W^V(w_1) \cup \lambda_W^V(w_2)) \tag{5.11}$$

$$= \frac{\Pi_a(\lambda_W^V(w_1) \cup \lambda_W^V(w_2))}{\mathrm{nonzero}(\mathrm{Marg}^+_{V'}(\Pi_a \cdot \theta'))(\lambda_W^V(w_1))} \tag{5.12}$$

Nous abusons de la notation de nonzero en le faisant appliquer sur un rationnel et non une fonction mais son fonctionnement est identique

$$= \frac{\Pi_a(\lambda_W^V(w_1) \cup \lambda_W^V(w_2))}{\mathrm{nonzero}(\mathrm{Marg}^+_{V'}(\Pi_a \cdot \theta')(\lambda_W^V(w_1)))} \tag{5.13}$$

Par la définition de la marginalisation (déf. 3.15)

$$= \frac{\Pi_a(\lambda_W^V(w_1) \cup \lambda_W^V(w_2))}{\mathrm{nonzero}(\sum_{v' \in 2^{V'}}(\Pi_a \cdot \theta')(\lambda_W^V(w_1) \cup v'))} \tag{5.14}$$

$$\tag{5.15}$$

Ici, il y a deux possibilités.

- Si le dénominateur est nul, nonzero impose que la fraction soit nulle. Cela signifie que le support de $\mu_a(w_1)$ est aussi vide par hypothèse, ainsi $\mathrm{N}(\Pi_a)(\lambda_W^V(w_1) \cup \lambda_W^V(w_2)) = 0 = \mathrm{N}_\theta(\mu_a(w_1)(w_2))$.
- Sinon, nous avons : $\mathrm{N}_\theta(\Pi_a)(\lambda_W^V(w_1) \cup \lambda_W^V(w_2))$

$$= \frac{\Pi_a(\lambda_W^V(w_1) \cup \lambda_W^V(w_2))}{\sum_{v' \in 2^{V'}}(\Pi_a \cdot \theta')(\lambda_W^V(w_1) \cup v')} \tag{5.16}$$

or comme θ a des valeurs booléennes, $(\Pi_a \cdot \theta')(\lambda_W^V(w_1) \wedge v') = 0$ si v' n'est pas dans le support de θ', donc la somme sur les élements de 2^V qui ne représentent pas des étiquettes de mondes est nulle. On peut alors remplacer v' par son antécédent dans W via λ_W^V que l'on nomme w_3

$$= \frac{\Pi_a(\lambda_W^V(w_1) \cup \lambda_W^V(w_2))}{\sum_{w_3 \in W} \Pi_a(\lambda_W^V(w_1) \cup \lambda_W^V(w_3))} \tag{5.17}$$

Par hypothèse $\Pi_a(\lambda_W^V(w_1) \cup \lambda_W^V(w_2)) = \mu_a(w_1)(w_2)$

$$= \frac{\mu_a(w_1)(w_2)}{\sum_{w_3 \in W} \mu_a(w_1)(w_3)} \tag{5.18}$$

$$= \mathrm{N}(\mu_a(w_1))(w_2) \text{ déf. de la normalisation d'une loterie 4.19} \tag{5.19}$$

$$\tag{5.20}$$

$$\forall v \in V, \mathrm{Marg}^+_{V'}(\Pi_a)(v) \coloneqq \sum_{v' \in 2^{V'}} \Pi_a(v \cup v')$$

□

Afin de montrer l'équivalence du *model checking* entre structure de Kripke probabiliste et structure de probabilités, nous allons étendre le lemme de Gattinger dans le cas non probabiliste (lemme 2.32) avec une quatrième condition.

Lemme 5.19. *On suppose avoir une structure de probabilités* $\mathfrak{F} = \langle V, \theta, \Omega, \Pi\rangle$ *et une structure de Kripke probabiliste* $\mathcal{M} = \langle W, R, \mu, \mathrm{val}\rangle$ *avec un ensemble d'atomes propositionnels* $U \subseteq V$*. De plus, on suppose avoir une fonction* $\lambda\colon W \to 2^V$ *telle que :*

C1 *pour tout* $w_1, w_2 \in W$ *et* $a \in \mathfrak{A}$*, on a* $\lambda(w_1) \cup \lambda(w_2)' \models \Omega_a$ *ssi* $w_1 R_a w_2$*.*

C2 *pour tout* $w \in W$ *et* $p \in U$*, on a* $p \in \lambda(w)$ *ssi* $p \in \mathrm{val}(w)$*.*

C3 *pour chaque* $s \in 2^V$*,* s *est un état de* $\mathfrak{F}$ *ssi* $s = \lambda(w)$ *pour certains* $w \in W$*.*

C4 *pour tout* $w_1, w_2 \in W$ *et* $a \in \mathfrak{A}$*, on a* $\Pi_a(\lambda(w_1) \cup \lambda(w_2)') = \mu_a(w_1)(w_2)$*.*

alors, pour toute formule ϕ *sur le vocabulaire* U*, on a* $\langle \mathfrak{F}, \lambda(w)\rangle \models \phi$ *ssi* $\langle \mathcal{M}, w\rangle \models \phi$*.*

Démonstration. Nous procédons par induction sur ϕ. Tous les cas non probabilistes découlent du lemme non probabiliste 2.32.

Supposons maintenant que ϕ est de la forme $\alpha_1 \mathrm{Pr}_a(\phi_1) + \cdots + \alpha_k \mathrm{Pr}_a(\phi_k) \geq \beta$. Par la sémantique (déf. 5.14) de la formule, nous avons : $\langle \mathfrak{F}, \lambda(w)\rangle \models \alpha_1 \mathrm{Pr}_a(\phi_1) + \cdots + \alpha_k \mathrm{Pr}_a(\phi_k) \geq \beta$ ssi :

$$\sum_{i=1}^{k} \alpha_i \left(\sum_{\text{pour tout état } t \text{ de } \mathfrak{F} \text{ tel que } \langle \mathfrak{F}, t\rangle \models \phi_i} \mathrm{N}_\theta(\Pi_a)(\lambda(w) \cup t') \right) \geq \beta.$$

Par C3, c'est équivalent à

$$\sum_{i=1}^{k} \alpha_i \left(\sum_{\text{pour tout état } w_2 \text{ de } W \text{ tel que } \langle \mathfrak{F}, \lambda(w_2)\rangle \models \phi_i} \mathrm{N}_\theta(\Pi_a)(\lambda(w) \cup \lambda(w_2)') \right) \geq \beta.$$

Avec C4 et le lemme de l'égalité des normalisations (lemme. 5.18) c'est équivalent à

$$\sum_{i=1}^{k} \alpha_i \left(\sum_{\text{pour tout état } w_2 \text{ de } W \text{ tel que } \langle \mathfrak{F}, \lambda(w_2)\rangle \models \phi_i} \mathrm{N}(\mu_a(w))(w_2) \right) \geq \beta$$

qui, avec l'hypothèse d'induction, est exactement la définition 4.22 :

$\langle \mathcal{M}, w\rangle \models \alpha_1 \mathrm{Pr}_a(\phi_1) + \cdots + \alpha_k \mathrm{Pr}_a(\phi_k) \geq \beta$ ssi

$$\sum_{i=1}^{k} \alpha_i \left(\sum_{w_2 : \mathcal{M}, w_2 \models \phi_i} \mathrm{N}(\mu_a(w))(w_2) \right) \geq \beta.$$

□

Grâce à ce lemme, nous pouvons maintenant démontrer les deux théorèmes suivants.

Théorème 5.20. *Pour toute structure de probabilités $\mathfrak{F}$, tout état s de $\mathfrak{F}$ et toute formule $\phi \in \mathcal{L}_{\text{PEL}}$, on a :*

$$\langle \mathfrak{F}, s \rangle \models \phi \text{ ssi } \langle \text{expl}(\mathfrak{F}), s \rangle \models \phi$$

Démonstration. On note $\mathcal{M} = \text{expl}(\mathfrak{F}) = \langle W, (\text{expl}_W(\Omega_a))_{a \in \mathfrak{A}}, (\text{expl}_W(\Pi_a))_{a \in \mathfrak{A}}, \text{val} \rangle$ avec $\text{expl}(\theta) = \langle W, \text{val} \rangle$ (via déf. 2.20).

nous utilisons le lemme 5.19 et on utilise la fonction d'identité pour λ. Nous devons montrer les quatre conditions. Les conditions C1, C2 et C3 sont prouvées identiquement au théorème 2.33 (version non probabiliste).

Pour prouver C4, nous devons regarder la définition de $\text{expl}_W(\Pi_a)$ (déf. 5.7) qui conserve bien les probabilités. Donc C4 est vérifiée.

□

Théorème 5.21. *Pour toute structure de Kripke pointée $\langle \mathcal{M}, w \rangle$, toute formule $\phi \in \mathcal{L}_{\text{PEL}}$,et pour λ la fonction d'étiquetage appropriée : $\lambda(w) \coloneqq \{p \in V \mid p \in \text{val}(w)\}$ on a :*

$$\langle \mathcal{M}, w \rangle \models \phi \text{ ssi } \langle \text{symb}(\mathcal{M}), \lambda(w) \rangle \models \phi$$

Démonstration. Nous devons montrer que le lemme 5.19 s'applique à la définition 5.4.

C1, C2 et C3 sont montrés par Gattinger (Thm. 2.34).

Comme nous supposons déjà une structure de Kripke valuation-injective, la fonction d'étiquetage appropriée est $\lambda : W \to 2^V$ est la fonction $\lambda(w) \coloneqq \{p \in V \mid p \in \text{val}(w)\}$.

Pour montrer C4, prenons $w_1, w_2 \in W$ et $a \in \mathfrak{A}$ et notons que nous avons $\Pi_a(\lambda(w_1) \cup \lambda(w_2)') = \text{symb}_\lambda(\mu_a)(\text{val}(w_1) \cup \text{val}(w_2)') = \mu_a(w_1)(w_2) = \alpha$.

□

Nous avons montré l'équivalence entre le model checking sur les structures de probabilités et sur leur transformation en modèle explicite, et inversement. Nous allons maintenant voir comment intégrer la modification des probabilités grâce aux modèles d'observations et d'événements symboliques.

6 | Mise à jour symbolique des probabilités

Dans la suite de l'exposé, nous allons présenter deux variantes de ce qui se rapproche des modificateurs de croyances définis par Gattinger et présentés plus tôt (déf. 2.49), que nous appelons toujours des *modificateurs* et qui pourront mettre à jour les probabilités des structures.

L'avantage du modèle observationnel (déf. 4.39) est qu'il est très similaire au modèle d'événements de DEL (déf. 1.24, page 25). En effet, il suffit d'y ajouter une notion de probabilités. Ainsi la représentation symbolique d'un tel modèle sera fortement inspirée de la représentation symbolique d'un modèle d'événements de DEL. Nous appelerons un tel modificateur avec des probabilités *un modificateur observationnel*. Nous traiterons de ce cas dans la section 6.1.

Pour ce qui est de l'équivalent symbolique d'un modèle conditionnel (déf. 4.11), il ne s'adaptera pas aussi simplement que l'équivalent du modèle observationnel. En effet, la présence des préconditions conditionnelles (Φ) et des probabilités conditionnelles (μ_{pre}) introduisent d'autres notions qu'il faudra pouvoir manipuler. Nous nommerons cet équivalent symbolique un *modificateur conditionnel* et traiterons ce cas dans la section 6.2.

Dans les deux cas, le symbole représentant les probabilités symboliques d'un modificateur observationnel ou d'un modificateur conditionnel sera Π^+, de la même manière que Gattinger note une loi d'observations Ω^+.

6.1 Le modificateur observationnel

Dans le cadre de la représentation symbolique d'un modèle observationnel (déf. 4.39), nous définissons l'équivalent d'un modificateur de croyances (déf. 2.49) auquel nous ajoutons une loi de probabilités, ensemble de PBFs, notée Π^+. Nous appelons cette représentation un modificateur observationnel.

Définition 6.1. *Un modificateur observationnel pour le vocabulaire V est un tuple $\chi = \langle V^+, \theta^+, \Omega^+, \Pi^+, V_-, \theta_- \rangle$ où :*

- *V^+ est un ensemble d'atomes propositionnels tel que $V \cap V^+ = \emptyset$;*
- *$\theta^+ \in \mathcal{L}_{\text{PEL}}(V \cup V^+)$ est une formule potentiellement épistémique appelée la* loi des événements *;*
- *Ω^+ associe à chaque agent $a \in \mathfrak{A}$ une fonction booléenne sur $V^+ \cup V^{+'}$ appelée* loi d'observations des événements *de a ;*
- *Π^+ associe à chaque agent $a \in \mathfrak{A}$ une PBF sur $V^+ \cup V^{+'}$ appelée* loi de probabilités des événements *de a ;*
- *$V_- \subseteq V$ est un sous-ensemble du vocabulaire original appelé le* vocabulaire modifié *;*
- *$\theta_- : V_- \to \mathcal{L}_{\text{prop}}(V \cup V^+)$ est une fonction assignant à chaque proposition atomique modifiée une formule booléenne. Cette fonction est appelée la* loi de modification*.*

Un modificateur observationnel pointé est un modificateur observationnel associé à un sous-ensemble $x \in 2^{V^+}$, écrit $\langle \mathfrak{F}, x \rangle$.

Maintenant que nous avons défini un modificateur observationnel, nous allons pouvoir définir le *product update* symbolique d'une structure de probabilités avec celui-ci. Toutes les définitions gérant les variables d'historicisation V_- grâce à θ_- sont basées sur la définition du *product update* d'une structure de croyances avec postconditions (déf. 2.51). Il suffit simplement d'y ajouter la mise à jour des lois de probabilités Π via Π^+.

Définition 6.2 (Mise à jour symbolique des probabilités)**.** *Le* product update *d'une structure de probabilités pointée $\langle \mathfrak{F}, s \rangle$ ($\mathfrak{F} = \langle V, \theta, \Omega, \Pi \rangle$) par un modificateur observationnel pointé $\langle \chi, x \rangle$ pour un vocabulaire V ($\chi = \langle V^+, \theta^+, \Omega^+, \Pi^+, V_-, \theta_- \rangle$) est la structure de probabilités $\mathfrak{F} \otimes \chi = \langle V^\otimes, \theta^\otimes, \Omega^\otimes, \Pi^\otimes \rangle$ avec pour état pointé s^x, telle que :*

- $V^\otimes := V \cup V^+ \cup V_-^\circ$;
- $\theta^\otimes := [V_- \mapsto V_-^\circ](\theta \wedge ||\theta^+||_{\mathfrak{F}}) \wedge \bigwedge_{p \in V_-} (p \leftrightarrow [V_- \mapsto V_-^\circ](\theta_-(p)))$;
- $\Omega_a^\otimes := (([V_- \mapsto V_-^\circ][V_-' \mapsto (V_-^\circ)']\Omega_a) \wedge \Omega_a^+)_{a \in \mathfrak{A}}$;
- $\Pi_a^\otimes := (([V_- \mapsto V_-^\circ][V_-' \mapsto (V_-^\circ)']\Pi_a) \cdot \Pi_a^+)_{a \in \mathfrak{A}}$;

avec $s^x := (s \backslash V_-) \cup (s \cap V_-)^\circ \cup x \cup \{p \in V_- \mid s \cup x \models \theta_-(p)\}$
où V_-° est le vocabulaire V_- renommé avec le symbole $\circ$: $V_-^\circ = \{p^\circ \mid p \in V_-\}$

Pour les explications quant à l'application des postconditions, nous vous invitons à lire la section "Mise à jour des croyances avec postconditions" (sec. 2.5).

Notons que comme précédemment dans l'état de l'art, le cas sans postcondition est inclus dans le cas avec postconditions. En effet, dans le cas où $V_- = \emptyset$, nous obtenons exactement la définition d'une modification de structures de croyances (déf. 2.51) et nous obtenons simplement les nouvelles lois de probabilités suivantes :

$$(\Pi_a \cdot \Pi_a^+)_{a \in \mathfrak{A}}$$

Notons que les loteries résultantes sont simples à obtenir. Il s'agit tout simplement d'un produit, très similaire à la conjonction des relations symboliques Ω et Ω^+. Cela provient directement de notre utilisation d'une version dénormalisée des structures de Kripke et modèle observationnel (déf. 4.20 et 4.39). Sans ces modifications de PDEL, il aurait fallu renormaliser les probabilités.

Exemple 6.3. *Continuons l'exemple non probabilisé 2.45, avec les probabilités présentés dans les exemples 5.6 (structure de probabilités) et 6.8 (modificateur observationnel).*

Les mises à jour étant identiques à celles présentées par Gattinger avec les structures et modificateurs de croyances, nous allons nous concentrer sur la mise à jour des probabilités.

Pour rappel, nous avions :

- $V = \{p, q\}$ *et* $\theta = q$ *;*
- $V^+ = \{e\}$ *et* $\theta^+ = (\neg e \wedge p) \vee (e \wedge \neg p)$.

Pour ce qui est du calcul des probabilités, nous avons :

- $\Pi_b = \{pqp'q' \vee \neg pq\neg p'q : 1\}$ *;*
- $\Pi_r = \{pqp'q' : 0.75,\ \neg pq\neg p'q' : 0.25,\ pq\neg p'q' : 0.25,\ \neg pqp'q' : 0.75\}$ *;*
- $\Pi_b^+ = \{\neg e' : 1,\ e' : 2\}$ *;*
- $\Pi_r^+ = \{e \leftrightarrow e' : 1\}$.

Cela nous permet simplement d'avoir les nouvelles probabilités :

- $\Pi_b^\otimes = \{pqp'q' \vee \neg pq\neg p'q : 1\} \cdot \{\neg e' : 1,\ e' : 2\}$
  $= \{(pqp'q' \vee \neg pq\neg p'q) \wedge \neg e' : 1,\ (pqp'q' \vee \neg pq\neg p'q) \wedge e' : 2\}$
- $\Pi_r^\otimes = \{pqp'q' : 0.75, \neg pq\neg p'q' : 0.25, pq\neg p'q' : 0.25, \neg pqp'q' : 0.75\} \cdot \{e \leftrightarrow e' : 1\}$
  $= \{$
  $(pqp'q') \wedge (e \leftrightarrow e') : 0.75,\ (\neg pq\neg p'q') \wedge (e \leftrightarrow e') : 0.25,$
  $(pq\neg p'q') \wedge (e \leftrightarrow e') : 0.25,\ (\neg pqp'q') \wedge (e \leftrightarrow e') : 0.75$
  $\}$

L'interprétation de l'opérateur dynamique sur une structure de probabilités est identique à celui sur les structures de croyances. Notons que nous utilisons encore un abus de notation (c.f. rq. 2.59, page 59) et considérons abusivement l'utilisation des types de modificateurs dans notre langage, comme le fait Gattinger.

Définition 6.4. *Soit une structure de probabilités pointée* $\langle \mathfrak{F}, s\rangle$ *et un modèle observationnel pointé* $\langle \chi = \langle V^\intercal, \theta^\intercal, V_-, \theta_-, \Omega^\intercal, \Pi^\intercal\rangle, x\rangle$*. L'opérateur dynamique est interprété comme suit :*

$$\langle \mathfrak{F}, s\rangle \models [\chi, x]\phi \text{ ssi } \langle \mathfrak{F}, s\rangle \models [x \sqsubseteq V^\intercal]\theta^\intercal \implies \langle \mathfrak{F} \otimes \chi, s^x\rangle \models \phi$$

où s^x *est la définition du nouvel état (déf. 6.2).*

Par ailleurs, la définition de la traduction locale se formule de la même manière dans le cas non probabiliste (déf. 2.61).

Définition 6.5 (Traduction locale dynamique probabiliste)**.** *Soit* χ *un modèle observationnel. En complément de la traduction locale probabiliste (déf. 5.15, page 133), la traduction locale* $|||[\chi, x]||_{\mathfrak{F}}$ *de l'opérateur dynamique pour toute structure de probabilités* $\mathfrak{F}$ *définie pour un vocabulaire* V *et toute formule épistémique dynamique* $\phi \in \mathcal{L}_{\text{PDEL}}(V)$ *est définie comme suit :*

$$|||[\chi, x]\phi||_{\mathfrak{F}} \coloneqq |||[x \sqsubseteq V^\intercal]\theta^\intercal||_{\mathfrak{F}} \to [V_-^\circ \mapsto V_-][x \sqsubseteq V^\intercal][V_- \mapsto \theta_-(V_-)]||\phi||_{\mathfrak{F}\otimes\chi}$$

Nous prouvons le théorème suivant :

Théorème 6.6. *6.5 Pour toute structure de probabilités pointée* $\langle \mathfrak{F}, s\rangle$*, tout modificateur observationnel pointé* $\langle \chi, x\rangle$ *et toute formule* $\phi \in \mathcal{L}_{\text{PDEL}}$*, nous avons :*

$$\langle \mathfrak{F}, s\rangle \models [\chi, x]\phi \iff s \models |||[\chi, x]\phi||_{\mathfrak{F}}$$

La preuve est identique à la preuve présentée par Gattinger pour le théorème dans le cas de $\mathcal{L}_{\text{DEL}}$ (repris ici en théorème 2.62). Dans cette preuve, il se sert par induction du cas déjà prouvé pour la traduction locale d'une formule de $\mathcal{L}_{\text{EL}}$ (thm. 2.30). Or, les sémantiques dynamiques non probabiliste (déf. 2.60) et probabiliste (déf. 6.4) s'écrivent de la même manière. Ainsi le théorème ci-dessus, qui s'exprime exactement de la même manière que celui de $\mathcal{L}_{\text{DEL}}$ (déf. 2.62), utilise par induction le cas d'une formule de $\mathcal{L}_{\text{PEL}}$ (thm. 5.17) et se prouve identiquement.

De la même manière qu'il était possible de transformer un modèle d'événéments (non probabiliste) en un modificateur de croyances (déf. 2.54, page 57), il est possible de transformer un modèle observationnel en un modificateur observationnel. La transformation est très similaire, seulement il faut y ajouter la définition des $\text{symb}_{\lambda^V_W}(\Pi_a)$ (déf. 5.3).

Définition 6.7. *La représentation symbolique d'un modèle observationnel* $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mu^{\mathcal{E}}, \mathrm{pre}, \mathrm{post}\rangle$ *défini sur un vocabulaire* V *via une fonction d'étiquetage* $\lambda_E^{V^+}$*, notée* $\mathrm{symb}_{\lambda_E^{V^+}}(\mathcal{E})$*, est le modificateur observationnel*

$$\langle V^+,\ (\mathrm{symb}_{\lambda_E^{V^+}}(R_a^{\mathcal{E}}))_{a\in\mathfrak{A}},\ (\mathrm{symb}_{\lambda_E^{V^+}}(\mu_a^{\mathcal{E}}))_{a\in\mathfrak{A}},\ \mathrm{symb}_{\lambda_E^{V^+}}(\mathrm{pre}),\ V_-,\ \theta_-\rangle$$
$$\textit{où } \mathrm{symb}_{\lambda_E^{V^+}}(\mathrm{post}) = \langle V_-, \theta_-\rangle.$$

Avec, pour rappel :

- $\mathrm{symb}_{\lambda_E^{V^+}}(\mathrm{pre}) \coloneqq \bigvee_{e\in E}(\lambda_E^{V^+}(e) \sqsubseteq V^+ \wedge \mathrm{pre}(e))$ (déf. 2.37, page 49) ;
- $\mathrm{symb}_{\lambda_W^V}(R_a^{\mathcal{E}}) \coloneqq \bigvee_{(e_1,e_2)\in R_a}(\lambda_E^{V^+}(e_1) \sqsubseteq V^+) \wedge (\lambda_E^{V^+}(e_2) \sqsubseteq V^+)'$ (déf. 2.12, page 38) ;
- $\mathrm{symb}_{\lambda_W^V}(\mu_a^{\mathcal{E}}) \coloneqq \{\lambda_E^{V^+}(e_1) \sqsubseteq E \wedge (\lambda_E^{V^+}(e_2) \sqsubseteq E)' \colon \mu^{\mathcal{E}}(e_1)(e_2) \mid (e_1, e_2) \in E^2\}$ (déf. 5.3, page 128) ;
- et par la définition de $\mathrm{symb}_{\lambda_E^{V^+}}(\mathrm{post})$ (déf. 2.53, page 57) :
  - $V_- \coloneqq \{p \in V \mid \exists e \in E : \mathrm{post}(p)(e) \neq p\}$;
  - pour $p \in V_-$, $\theta_-(p) \coloneqq \bigvee_{e\in E}(\lambda_E^{V^+}(e) \sqsubseteq V^+ \wedge \mathrm{post}(p)(e))$

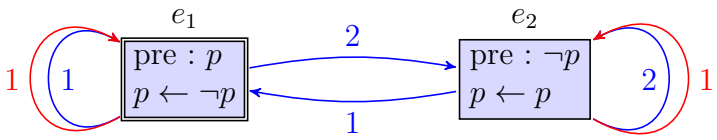


FIGURE 6.1 – Exemple de modèle observationnel

Exemple 6.8. *Prenons l'exemple du modèle observationnel présenté à la figure 6.1. Les chiffres sur les arcs sont les valeurs des loteries et les couleurs représentent les agents.*

Cet exemple est identique à l'exemple (sans postcondition) présenté 2.39, page 50, nous y avons déjà défini θ^+ *et* Ω^+*. L'exemple 2.55, page 58 présente la transformation avec des postconditions pour obtenir* V_- *et* θ_-*. En effet, les définitions de* symb *des éléments d'un modificateur de croyances (déf. 2.54) sont identiques à celles pour une structure de probabilités (déf. 6.7), sauf pour les loteries qui n'y sont pas présentes.*

Nous nous contentons donc ici de définir Π^+*. La technique est identique à celle présentée dans l'exemple 5.6, mais nous utilisons ici des loteries au lieu de distributions de probabilités.*

La construction de Π_r^+ *(agent rouge) et* Π_b^+ *(bleu) se fait de la manière suivante :*

- $\Pi_r^\star = \{\neg e \wedge \neg e' : 1,\ e \wedge e' : 1\} = \{e \leftrightarrow e' : 1\}$;
- $\Pi_b^\star = \{\neg e \wedge \neg e' : 1,\ \neg e \wedge e' : 2,\ e \wedge e' : 2,\ e \wedge \neg e' : 1\} = \{\neg e' : 1,\ e' : 2\}$.

Comme le fait Gattinger (thm. 2.64), nous établissons le théorème sur un vocabulaire U (inclus dans V) qui permet de rendre une structure de Kripke potentiellement non-injective, injective ; c'est ce que l'on entend par « qui étend val ».

Si la structure de Kripke est injective val (défini sur le vocabulaire U) n'a pas besoin d'être étendu par sur le vocabulaire V : nous avons $V = U$ et $\lambda_W^U = \text{val}$.

Théorème 6.9. *Les fonctions* symb *d'une structure de Kripke probabiliste (déf. 5.4) et d'un modificateur observationnel (déf. 6.7) préservent la vérité : pour toute structure de Kripke pointée $\langle \mathcal{M}, w\rangle$ définie sur le vocabulaire V, tout modèle observationnel pointé $\langle \mathcal{E}, e\rangle$ défini pour le vocabulaire V et toute formule ϕ sur le vocabulaire de $\mathcal{M}$, on a, avec λ_W^U une fonction d'étiquetage sur $U \subseteq V$ qui étend* val *et $\lambda_E^{V^\star}$ une fonction d'étiquetage pour $\mathcal{E}$:*

$$\langle \mathcal{M} \otimes \mathcal{E}, (w, e)\rangle \models \phi \iff \langle \text{symb}_{\lambda_W^U}(\mathcal{M}), \lambda_W^U(w)\rangle \otimes \langle \text{symb}_{\lambda_E^{V^\star}}(\mathcal{E}), \lambda_E^{V^\star}(e)\rangle \models \phi$$

Démonstration. De la même manière que la preuve de Gattinger (thm. 2.64), nous allons prouver le théorème grâce au lemme d'équivalence entre structure de probabilités et structure de Kripke (lemme. 5.19). Il faut alors montrer les quatre conditions du lemme. C1, C2 et C3 sont déjà prouvés (preuve du thm. 2.64), il nous reste donc à prouver C4.

λ doit ici associer les mondes du modèle $\mathcal{M} \otimes \mathcal{E}$ aux états de la structures $\text{symb}_{\lambda_W^V}(\mathcal{M}) \times \text{symb}_{\lambda_E^{V^\star}}(\mathcal{E})$. De nouveau, s^x est utilisé mais s et x sont donnés par λ_V^W et $\lambda_{V^\star}^{\mathcal{E}}$.

Pour plus de lisibilité, nous noterons $\widehat{w}$ pour $\lambda_W^V(w)$ et $\widehat{e}$ pour $\lambda_E^{V^\star}(e)$.

Soit λ défini ainsi pour tout $w \in W$ et $e \in E$:

$$\lambda(w, e) \coloneqq (\widehat{w} \backslash V_-) \cup (\widehat{w} \cap V_-)^\circ \cup \widehat{e} \cup \{p \in V_- \mid \widehat{w} \cup \widehat{e} \models \theta_-(p)\}$$

La preuve suit la preuve précédente et $\widehat{w}$ et $\widehat{e}$ prennent les rôles de s et x.

$*$ Prenons deux mondes quelconques (w_1, e_1) et (w_2, e_2) dans le modèle mis à jour. Nous devons montrer que $\Pi_a^\otimes(\lambda(w_1, e_1) \cup \lambda(w_2, e_2)') = \mu_a^\otimes(s, x)(t, y)$. Pour cela, écrivons l'équivalence suivante. Nous avons $\Pi_a^\otimes(\lambda(w_1, e_1) \cup \lambda(w_2, e_2)')$

$= ([V_- \mapsto V_-^\circ][V_-' \mapsto (V_-^\circ)'](\Pi_a \cdot \Pi_a^+)($
$\quad (\widehat{w_1} \backslash V_-) \cup (\widehat{w_1} \cap V_-)^\circ \cup \widehat{e_1} \cup \{p \in V_- \mid \widehat{w_1} \cup \widehat{e_1} \models \theta_-(p)\} \cup$
$\quad ((\widehat{w_2} \backslash V_-) \cup (\widehat{w_2} \cap V_-)^\circ \cup \widehat{e_2} \cup \{p \in V_- \mid \widehat{w_2} \cup \widehat{e_2} \models \theta_-(p)\})'$
)

De nouveau, V_- et V_-' n'apparaissent pas dans la PBF, alors ils peuvent être enlevés des valuations pour obtenir :

$$= ([V_- \mapsto V_-^\circ][V'_- \mapsto (V_-^\circ)']\Pi_a \cdot \Pi_a^+)(\\ (\widehat{w_1}\backslash V_-) \cup (\widehat{w_1} \cap V_-)^\circ \cup \widehat{e_1} \cup \\ (\widehat{w_2}\backslash V_-)' \cup (\widehat{w_2}^\circ \cap V_-^\circ)' \cup \widehat{e_2}')$$

ce qui se sépare en :

$$= ([V_- \mapsto V_-^\circ][V'_- \mapsto (V_-^\circ)']\Pi_a)(\\ (\widehat{w_1}\backslash V_-) \cup (\widehat{w_1} \cap V_-)^\circ \cup (\widehat{e_1}\backslash V_-)' \cup (\widehat{e_1}^\circ \cap V_-^\circ)') \\ \times \\ \Pi_a^+(\widehat{w_2} \cup \widehat{e_2}')$$

Maintenant il est possible d'enlever la substitution par $\circ$ pour montrer l'équivalence à $\Pi_a(\widehat{w_1} \cup \widehat{w_2}')$. Dans la preuve du théorème 5.21 (équivalence du *model checking* non dynamique), nous avons vu que les conditions du lemme 5.19 s'appliquent à λ_V^W dans la construction de $\mathrm{symb}_{\lambda_W^V}(\mathcal{M})$. En particulier, par la condition C4, nous avons $\Pi_a(\widehat{w_1} \cup \widehat{w_2}') = \mu_a(w_1)(w_2)$. De plus, par la définition de $\mathrm{symb}_{\lambda_E^{V^+}}(\mathcal{E})$ 2.54, nous avons $\Pi_a^+(\widehat{e_1} \cup \widehat{e_2}') = \mu_a^{\mathcal{E}}(e_1)(e_2)$. Par conséquent, la condition est équivalente à $\mu_a(w_1)(w_2) \times \mu_a^{\mathcal{E}}(e_1)(e_2)$. Par définition 4.42 du *product update*, c'est $\mu_a^{\otimes}(w_1, e_1)(w_2, e_2)$.

□

Définition 6.10. *La représentation explicite d'un modificateur observationnel* $\chi = \langle V^+, \theta^+, \Omega^+, \Pi^+, V_-, \theta_- \rangle$ *pour le vocabulaire* V *notée* $\mathrm{expl}(\mathcal{E})$ *est le modèle observationnel suivant, où* $E = 2^{V^+}$

$$\langle E, (\mathrm{expl}_E(\Omega_a^+))_{a \in \mathfrak{A}}, (\mathrm{expl}_E(\Pi_a^+))_{a \in \mathfrak{A}},\ \mathrm{expl}(\theta^+),\ \mathrm{expl}_{V,V^+}(\theta_-) \rangle$$

Avec, pour rappel :

- $E \coloneqq 2^{V^+}$;
- $\mathrm{expl}(\theta^+) \coloneqq [e \sqsubseteq V^+]\theta^+$; (déf. 2.40)
- $\forall e_1, e_2 \in E,\ (e_1, e_2) \in \mathrm{expl}_E(\Omega_a^+)$ ssi $e_1 \cup e_2' \models \Omega_a^+$ (déf. 2.19) ;
- $\forall (e_1, e_2) \in E^2,\ \mathrm{expl}_E(\Pi_a^+)(e_1)(e_2) = \Pi_a^+(e_1 \cup e_2')$ (déf. 5.7) ;
- $\forall p \in V,\ \forall e \in E,\ \mathrm{expl}_{V,V^+}(p)(e) \coloneqq [e \sqsubseteq V^+]\theta_-(p)$ si $p \in V_-$, p sinon (déf. 2.56).

La retranscription des valeurs de Π^+ dans $\mathrm{expl}_E(\Pi_a^+)$ se passe exactement de la même manière que pour la définition 5.8 (représentation explicite d'une structure probabiliste) et les remarques qui y étaient faites sont aussi valables ici.

Théorème 6.11. *Les fonctions* expl *pour une structure de probabilités (déf. 5.8) et pour un modificateur observationnel préservent la vérité : pour tout structure de probabilités pointée $\langle \mathfrak{F}, s\rangle$ avec pour vocabulaire V, tout modificateur observationnel pointé $\langle \chi, x\rangle$ défini pour le vocabulaire V et toute formule $\phi \in \mathcal{L}_{\mathrm{PDEL}}(V)$, on a :*

$$\langle \mathfrak{F}, s\rangle \otimes \langle \chi, x\rangle \models \phi \iff \langle \mathrm{expl}(\mathfrak{F}), s\rangle \otimes \langle \mathrm{expl}(\chi), x\rangle \models \phi$$

Démonstration. Nous allons de nouveau nous appuyer sur une preuve de Gattinger (thm. 2.63). Nous allons toujours prouver le théorème grâce au lemme d'équivalence entre structure de probabilités et structure de Kripke (lemme. 5.19). C1, C2 et C3 sont ainsi déjà prouvées, il nous reste alors à prouver C4.

Nous prenons toujours une fonction d'étiquetage λ qui associe chaque monde de $\mathrm{expl}(\mathfrak{F}) \times \mathrm{expl}(\chi)$ (i.e. $(s, x) \in 2^V \times 2^{V^+}$) à un état de $\mathfrak{F} \times \chi$ (i.e. un sous-ensemble de $V \cup V^+ \cup V_-^\circ$) :

$$\lambda(s, x) \coloneqq (s \backslash V_-) \cup (s \cap V_-)^\circ \cup x \cup \{p \in V_- \mid s \cup x \models \theta_-(p)\}$$

qui est la définition de s^x (déf. 2.51).

Nous allons montrer la condition C4 du lemme 5.19 pour cette fonction d'étiquetage λ.

∗ La preuve est très similaire à la preuve de C1 en utilisant Π à la place de Ω et en utilisant les notations des PBFs à la place des formules logiques.

Prenons deux mondes de $\mathrm{expl}(\mathfrak{F})$: (s, x) et (t, y). Nous devons montrer que : $\Pi_a^\otimes(\lambda(s, x) \cup \lambda(t, y)') = \mu_a^\otimes(s, x)(t, y)$. Pour cela, montrons l'égalité :
$\Pi_a^\otimes(\lambda(s, x) \cup \lambda(t, y)')$

$$\begin{aligned} &= ([V_- \mapsto V_-^\circ][V'_- \mapsto (V_-^\circ)']\Pi_a \cdot \Pi_a^+)(\\ &\qquad (s\backslash V_-) \cup (s \cap V_-)^\circ \cup x \cup \{p \in V_- \mid s \cup x \models \theta_-(p)\} \cup \\ &\qquad ((t\backslash V_-) \cup (t \cap V_-)^\circ \cup y \cup \{p \in V_- \mid t \cup y \models \theta_-(p)\})') \end{aligned}$$

V_- et V'_- n'apparaissent pas dans l'argument de la PBF, comme les anciennes relations épistémiques ne dépendent pas des nouvelles valeurs des atomes propositonnels modifiés. Ainsi, on peut simplifier la fonction pour avoir :

$$= ([V_- \mapsto V_-^\circ][V'_- \mapsto (V_-^\circ)']\Pi_a \cdot \Pi_a^+)((s\backslash V_-) \cup (s \cap V_-)^\circ \cup x \cup (t\backslash V_-)' \cup (t^\circ \cap V_-^\circ)' \cup y'$$

dans laquelle on peut séparer les deux lois de probabilités car elles sont définies sur des vocabulaires différents :

$$\begin{aligned} &= ([V_- \mapsto V_-^\circ][V'_- \mapsto (V_-^\circ)']\Pi_a)((s\backslash V_-) \cup (s \cap V_-)^\circ \cup (t\backslash V_-)' \cup (t^\circ \cap V_-^\circ)') \\ &\qquad \times \\ &\qquad \Pi_a^+(x \cup y') \end{aligned}$$

Enlevons la substitution par $\circ$ pour voir que la PBF utilisant Π_a revient à $\Pi_a(s \cup t')$. Par conséquent, tout le produit est aussi égal à $\mu_a(s)(t) \times \mu_a^{\mathcal{E}}(x)(y)$ qui est $\mu_a^{\otimes}((s,x))((t,y))$ par les définitions de expl($\mathcal{M}$) (déf. 5.8) et expl($\mathcal{E}$) (déf. 6.10 et 5.7).

□

Après avoir présenté les modèles observationnels symboliques, nous allons pouvoir présenter les modèles conditionnels symboliques, i.e. les modèles d'événements de PDEL avec des préconditions conditionnelles (déf. 4.24), représentés de manière symbolique.

6.2 Le modificateur conditionnel

La grande différence entre les modèles observationnels et les modèles conditionnels est la gestion des préconditions. Tandis que pour un modèle observationnel, il n'est utilisé qu'une fonction de précondition pre qui assigne à chaque événement atomique $e \in E$ une formule épistémique (identique à DEL, i.e. pre: $E \to \mathcal{L}_{\text{EL}}$ ou ici $\mathcal{L}_{\text{PEL}}$), il faut prévoir pour représenter symboliquement PDEL un ensemble de préconditions conditionnelles notées Φ et pour chaque $\phi \in \Phi$ une E-loterie $\mu_{\text{pre}}(\phi)$ sur les événements $e \in E$, qui associe la « probabilité » d'apparition de l'événement e étant donné ϕ.

Dans la définition classique, ces préconditions conditionnelles doivent être mutuellement incohérentes (déf. 4.11, page 99). Nous rappelons que nous avons relâché cette définition (déf. 4.24, page 107) de sorte que les formules n'aient plus de contraintes.

Voyons comment nous allons pouvoir représenter un modèle conditionnel de manière symbolique. On appelera ce nouveau modificateur *un modificateur conditionnel*. Pour ce faire, nous aurons besoin de retravailler la notion de précondition symbolique. Φ est un ensemble de formules épistémiques probabilistes, que l'on ne pourra pas abstraire de manière symbolique sans un contexte d'application (une structure de probabilités). Ainsi, l'ensemble de formules Φ sera aussi présent dans un modificateur conditionnel. Afin de pouvoir modéliser de manière symbolique une loi de probabilité discrète, nous allons utiliser, comme précédemment, des PBFs, qui seront notées $\theta^{\text{pre}}(\phi)$ pour chaque $\phi \in \Phi$.

Un modificateur conditionnel fonctionne en partie comme un modificateur observationnel, pour ce qui est des éléments V^+, Ω^+, Π^+, θ_- qui auront la même définition. Néanmoins, la notion de loi des événements qu'était θ^+ est maintenant remplacée par les deux éléments Φ et θ^{pre}.

Définition 6.12. Un modificateur conditionnel *défini pour le vocabulaire V est un tuple $\chi = \langle V^+, \Omega^+, \Pi^+, \Phi, \theta^{\text{pre}}, V_-, \theta_- \rangle$ où :*

- V^+, Ω^+, Π^+, V_- *et* θ_- *sont définis identiquement à la définition d'un modèle observationnel 6.12 ;*
  - V^+ *est un ensemble d'atomes propositionnel tel que :* $V \cap V^+ = \emptyset$ *;*
  - Ω^+ *associe à chaque agent* $a \in \mathfrak{A}$ *une fonction booléenne sur* $V^+ \cup V^{+\prime}$ *appelé* loi d'observations des événements *de a ;*
  - Π^+ *associe à chaque agent* $a \in \mathfrak{A}$ *une PBF sur* $V^+ \cup V^{+\prime}$ *appelée* loi de probabilités des événements *de a ;*
  - $V_- \subseteq V$ *est un sous-ensemble du vocabulaire originel* V *appelé* vocabulaire modifié*,*
  - θ_- *associe à chaque atome propositionnel modifié* $p \in V_-$ *une fonction booléenne* $\theta_-(p)$ *sur* $V \cup V^+$ *appelée* la loi de modification *de p ;*
- Φ *est un ensemble de formules de* $\mathcal{L}_{\text{PEL}}(V)$*, appelée* conditions préalables *;*
- θ^{pre} *associe à chaque* $\phi \in \Phi$ *une PBF* $\theta^{\text{pre}}(\phi)$ *sur* V^+ *appelée la* loi de probabilités conditionnelles*.*

Un modificateur conditionnel pointé est un un modificateur conditionnel associé à un sous-ensemble $x \in 2^{V^+}$*, écrit* $\langle \mathfrak{F}, x \rangle$*.*

Illustrons l'utilisation de ces deux éléments grâce à la transformation d'un modèle conditionnel en un modificateur conditionnel.

Afin de définir la symbolicisation d'un modèle conditionnel, nous devons garder en mémoire que nous avons déjà Φ, et qu'il nous faut une fonction d'étiquetage qui nous procure le vocabulaire V^+. Il nous reste simplement à construire θ^{pre}, en utilisant la représentation symbolique d'une X$-$*loterie* (déf. 5.2).

Définition 6.13. *La représentation symbolique de* $\mu_{\text{pre}} \colon \Phi \to (E \to \mathbb{Q}_+)$ *via une fonction d'étiquetage* $\lambda_E^{V^+}$*, notée* $\text{symb}_{\lambda_E^{V^+}}(\mu_{\text{pre}})$ *est la loi de probabilités conditionnelle* $\Phi \to (2^{V^+} \to \mathbb{Q}_+)$

$$\forall \phi \in \Phi,\ \text{symb}_{\lambda_E^{V^+}}(\mu_{\text{pre}})(\phi) = \text{symb}_{\lambda_E^{V^+}}(\mu_{\text{pre}}(\phi))$$

i.e.

$$\forall \phi \in \Phi,\ \text{symb}_{\lambda_E^{V^+}}(\mu_{\text{pre}})(\phi) = \{\lambda_E^{V^+}(e) \sqsubseteq V^+ \colon \mu_{\text{pre}}(\phi)(e) \mid e \in E\}$$

Notons que nous utilisons la version étendue de notre notation des PBFs (rq. 3.9, page 76).

La représentation symbolique d'un modèle conditionnel complet est quant à elle similaire à la représentation symbolique d'un modèle observationnel (déf. 6.7), il faut néanmoins utiliser $\text{symb}_{\lambda_E^{V^+}}(\mu_{\text{pre}})$ au lieu de $\text{symb}_{\lambda_E^{V^+}}(\text{pre})$, utilisé dans les cas d'une structure de croyances (déf. 2.38) et d'une structure de probabilités (déf. 6.7).

Définition 6.14. *La représentation symbolique d'un modèle conditionnel* $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mu^{\mathcal{E}}, \Phi, \mu_{\text{pre}}, \text{post}\rangle$ *défini pour un vocabulaire* V *via une fonction d'étiquetage* $\lambda_E^{V^+}$*, notée* $\text{symb}_{\lambda_E^{V^+}}(\mathcal{E})$*, est le modificateur conditionnel*

$$\langle V^+,\ (\text{symb}_{\lambda_E^{V^+}}(R_a^{\mathcal{E}}))_{a\in\mathfrak{A}},\ (\text{symb}_{\lambda_E^{V^+}}(\mu_a^{\mathcal{E}}))_{a\in\mathfrak{A}},\ \Phi,\ \text{symb}_{\lambda_E^{V^+}}(\mu_{\text{pre}}),\ V_-,\ \theta_-\rangle$$
$$\textit{où } \text{symb}_{\lambda_E^{V^+}}(\text{post}) = \langle V_-, \theta_-\rangle.$$

L'opération inverse est possible et nous pouvons construire le modèle conditionnel à partir d'un modificateur conditionnel. Pour la construction de μ_{pre}, il faut toujours considérer que les événements $e \in E$ reconstruits sont par définition des valuations, ce qui permet très aisément de manipuler θ^{pre}, qui est un ensemble de PBFs.

Définition 6.15. *La représentation explicite de* $\theta^{\text{pre}}\colon \Phi \to (2^{V^+} \to \mathbb{Q}_+)$ *via un ensemble* $X \subseteq 2^{V^+}$*, notée* $\text{expl}_X(\theta^{\text{pre}})$*, est l'ensemble de lois de probabilité discrètes sur* X $\Phi \to (X \to \mathbb{Q}_+)$

$$\forall \phi \in \Phi,\ \forall e \in X,\ \text{expl}_X(\theta^{\text{pre}})(\phi)(e) = \theta^{\text{pre}}(\phi)(e)$$

Remarque 6.16. *A l'instar des représentations explicites des lois d'observations* Ω_a *ou des lois de probabilités* Π_a *via des ensembles* X*, la représentation explicite de* θ^{pre} *récupère simplement les informations de* θ^{pre}*.*

La représentation explicite d'un modificateur conditionnel est similiaire à la représentation explicite d'un modificateur observationnel (déf. 6.10), modulo la gestion des préconditions, où il ne s'agit plus d'utiliser $\text{expl}(\theta^+)$ mais $\text{expl}_E(\theta^{\text{pre}})$.

Définition 6.17. *La représentation explicite d'un modificateur conditionnel* $\chi = \langle V^+, \Omega^+, \Pi^+, \Phi, \theta^{\text{pre}}, V_-, \theta_-\rangle$ *pour le vocabulaire* V *notée* $\text{expl}(\mathcal{E})$ *est le modèle conditionnel suivant avec* $E = 2^{V^+}$

$$\langle E, (\text{expl}_E(\Omega_a^+))_{a\in\mathfrak{A}}, (\text{expl}_E(\Pi_a^+))_{a\in\mathfrak{A}}, \Phi,\ \ \text{expl}_E(\theta^{\text{pre}}),\ \text{expl}_{V,V^+}(\theta_-)\rangle$$

Lors de l'utilisation du modèles observationnels, il était nécessaire de pouvoir calculer la traduction locale de θ^+. θ^+ étant simplement une formule épistémique probabiliste, la traduction locale de $||\theta^+||_{\mathfrak{F}}$ était simplement définie par la définition de la traduction (déf. 5.14). Pour rappel, cette traduction est une formule propositionnelle. Nous n'avons plus une telle notion.

Nous avons Φ qui est un ensemble de formules épistémiques sur V (qui peuvent avoir des modèles en commun puisqu'elles ne sont pas nécessairement mutuellement incohérentes) et θ^{pre} qui définit des PBFs sur V^+.

La notion clé dans cette section est alors la suivante : comment est-il possible de calculer ce qui s'apparenterait à la traduction locale de θ^{pre} ?

Définition 6.18 (Précondition conditionnelle symbolique)**.** *Pour tout modificateur conditionnel $\chi = \langle V^+, \Omega^+, \Pi^+, \Phi, \theta^{\text{pre}}, V_-, \theta_- \rangle$ défini pour le vocabulaire V, sa* précondition conditionnelle symbolique*, notée $\Theta^+_{\mathfrak{F}}$, est la PBF définie sur le vocabulaire $V \cup V^+$:*

$$\Theta^+_{\mathfrak{F}} \coloneqq \sum_{\phi \in \Phi} \theta^{\text{pre}}(\phi) \cdot ||\phi||_{\mathfrak{F}}$$

Pour pouvoir manipuler le support de $\Theta^+_{\mathfrak{F}}$ comme étant une formule propositionnelle, nous définissons la fonction $\text{supp}_{\text{prop}}$ d'une fonction.

Définition 6.19. *Soit une PBF g définie sur un vocabulaire V. Le* support propositionnel *de g, noté $\text{supp}_{\text{prop}}(g)$, est la formule propositionnelle suivante :*

$$\text{supp}_{\text{prop}}(g) \coloneqq \bigvee_{v \in 2^V | g(v) > 0} v \sqsubseteq V^+$$

Cette définition est très similaire à la définition du support d'une fonction (déf. 3.13) mais permet d'obtenir une formule propositionnelle au lieu d'un ensemble de valuations. Elle permet, entre autres, de pouvoir utiliser un conditionnement sur ce qu'aurait était $\text{supp}(\Theta^+_{\mathfrak{F}})$.

Remarque 6.20. *Soit $\langle \mathfrak{F}, s \rangle$ une structure de probabilités pointée et $\langle \chi, x \rangle$ un modificateur conditionnel pointé ($s \in 2^V$ et $x \in 2^{V^+}$). $\Theta^+_{\mathfrak{F}}$ ne peut être créé qu'à partir de $\mathfrak{F}$ puisqu'il faut pouvoir calculer les traductions locales des formules $\phi \in \Phi$. Notons que si toutes les formules ϕ sont propositionnelles, la traduction locale de ϕ ne dépend pas de $\mathfrak{F}$.*

$\theta^{\text{pre}}(\phi)$ représente, via une loterie, une distribution de probabilités sur les événements possibles dans V^+.

La fonction $[x \sqsubseteq V^+]\Theta^+_{\mathfrak{F}}$ est la PBF qui associe tous les modèles des formules $\phi \in \Phi$ à la probabilité conditionnelle de l'événement x. Ainsi, en écrivant $[x \sqsubseteq V^+] \text{supp}_{\text{prop}}(\Theta^+_{\mathfrak{F}})(s)$, on obtient la probabilité que l'événement x survienne dans l'état s (rappelons que $[x \sqsubseteq V^+]$ est la notation du conditionnement –déf. 2.25). Ici, le conditionnement sert à simuler l'événement x. Nous pouvons aussi voir cela ainsi de manière booléenne : x est applicable dans s si $\langle \mathfrak{F}, s \rangle \models [x \sqsubseteq V^+] \text{supp}_{\text{prop}}(\Theta^+_{\mathfrak{F}})$.

Considérons un premier exemple simple.

Exemple 6.21. *Soit $V = \{p\}$ et $V^+ = \{q\}$, $\Phi = \{p, \neg p\}$ avec $\theta^{\text{pre}}(p) = \{q\colon 0.5,\ \neg q\colon 0.5\}$ et $\theta^{\text{pre}}(\neg p) = \{q\colon 0.7,\ \neg q\colon 0.3\}$. Puisque les formules de Φ ne comportent pas d'opérateurs épistémiques et/ou probabilistes, notons que les traductions locales des formules sont elles-mêmes. $\Theta^+_{\mathfrak{F}}$ sera donc la PBF définie sur $V \cup V^+$: $\{p \wedge q\colon 0.5,\ p \wedge \neg q\colon 0.5,\ \neg p \wedge q\colon 0.7,\ \neg p \wedge \neg q : 0.3\}$.*

Affectation du Vocabulaire			Sous-formules et valeurs associées				Valeurs de $\Theta^{+}_{\mathfrak{F}}$
p	q	e	p	$q \wedge e$	$q \wedge \neg e$	$p \wedge e$	
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	4	0	4
0	1	1	0	2	0	0	2
1	0	0	3	0	0	0	3
1	0	1	3	0	0	6	9
1	1	0	3	0	4	0	7
1	1	1	3	2	0	6	11

TABLE 6.1 – Précondition conditionnelle symbolique de l'exemple 6.22

Considérons un second exemple avec des formules qui ne sont pas mutuellement incohérentes dans Φ.

Exemple 6.22. *Soit* $V = \{p, q\}$, $V^{+} = \{e\}$, $\Phi = \{p, q, \Box_a p\}$ *avec* $\theta^{\text{pre}}(p) = \{\top : 3\}$, $\theta^{\text{pre}}(q) = \{e : 2,\ \neg e : 4\}$ *et* $\theta^{\text{pre}}(\Box_a) = \{e : 6\}$. *Admettons ici que* $||\Box_a p||_{\mathfrak{F}} = p$.

$$
\begin{aligned}
\Theta^{+}_{\mathfrak{F}} &= \sum_{\phi \in \{p,q\}} \theta^{\text{pre}}(\phi) \cdot ||\phi||_{\mathfrak{F}} \\
&= \theta^{\text{pre}}(p) \cdot ||p||_{\mathfrak{F}} + \theta^{\text{pre}}(q) \cdot ||q||_{\mathfrak{F}} + \theta^{\text{pre}}(\Box_a p) \cdot ||\Box_a p||_{\mathfrak{F}} \\
&= \{\top : 3\} \cdot p + \{e : 2,\ \neg e : 4\} \cdot q + \{e : 6\} \cdot p \\
&= \{p : 3\} + \{q \wedge e : 2,\ q \wedge \neg e : 4\} + \{p \wedge e : 6\}
\end{aligned}
$$

Nous n'explicitons pas plus ce calcul et illustrons son résultat dans la table 6.1. On voit ici une agrégation des valeurs des fonctions θ^{pre} *sur les modèles en communs des formules de* Φ*, dans le même esprit que pouvait faire l'agrégation des probabilités conditionnelles de PDEL modifié (voir remarque 4.29).*

Pour ce qui est de la mise à jour des probabilités, elle se base bien évidemment sur les définitions précédentes du *product update*. Il s'agit néanmoins ici de modifier les probabilités en conséquence, de sorte à avoir un nouveau Π qui prenne en compte les probabilités conditionnelles, c'est à dire qui utilise $\Theta^{+}_{\mathfrak{F}}$ pour la mise à jour des lois de probabilités et $\text{supp}_{\text{prop}}(\Theta^{+}_{\mathfrak{F}})$ pour la mise à jour de la loi des mondes.

Définition 6.23 (Mise à jour symbolique des probabilités conditionnelles)**.** *La* mise à jour *d'une structure de probabilités pointée* $\langle \mathfrak{F} = \langle V, \theta, \Omega, \Pi \rangle, s \rangle$ *par un modificateur conditionnel pointé* $\langle \chi = \langle V^+, \Omega^+, \Pi^+, \Phi, \theta^{\text{pre}}, V_-, \theta_- \rangle, x \rangle$*, est la structure de probabilités pointée* $\langle \mathfrak{F} \otimes \chi = \langle V^\otimes, \theta^\otimes, \Omega^\otimes, \Pi^\otimes \rangle, s^x \rangle$ *où :*

- V*,* Ω *et* s^x *sont définis identiquement à la définition du* product update *d'une structure de croyances par un modificateur de croyances (déf. 2.51) :*
  - $V^\otimes \coloneqq V \cup V^+ \cup V_-^\circ$ *où* V_-° *est le vocabulaire* V_- *renommé avec le symbole* $\circ$ *:* $V_-^\circ = \{p^\circ \mid p \in V_-\}$ *;*
  - $\Omega^\otimes \coloneqq ([V_- \mapsto V_-^\circ][V'_- \mapsto (V_-^\circ)']\Omega_a) \wedge \Omega_a^+$ *;*
  - $s^x \coloneqq (s \backslash V_-) \cup (s \cap V_-)^\circ \cup x \cup \{p \in V_- \mid s \cup x \models \theta_-(p)\}$ *;*
- $\theta^\otimes \coloneqq [V_- \mapsto V_-^\circ](\theta \wedge \text{supp}_{\text{prop}}(\Theta_\mathfrak{F}^+)) \wedge \bigwedge_{p \in V_-}(p \leftrightarrow [V_- \mapsto V_-^\circ](\theta_-(p)))$ *;*
- $\Pi_a^\otimes \coloneqq ([V_- \mapsto V_-^\circ][V'_- \mapsto (V_-^\circ)'](\Pi_a \cdot (\Theta_\mathfrak{F}^+)')) \cdot \Pi_a^+$.

Remarque 6.24. *La multiplication des trois types de probabilités est identique à la multiplication lors du* product update *d'une structure de Kripke via un modèle observationnel dénormalisé (déf. 4.28, page 108). Elle est similaire à la multiplication* $\Pi_a \cdot \Pi_a^+$ *pour les modèles observationnels (déf. 6.2), mais intègre les probabilités conditionnelles présentes dans* $\Theta_\mathfrak{F}^+$*. Notons que la fonction* $\Theta_\mathfrak{F}^+$ *est primée de sorte à appliquer la probabilité conditionnelle aux valuations sur* V^+ *dans* Π_a^+*, comme le fait le* product update *explicite sur les événements atomiques confondus (déf. 4.28) qui, pour rappel, est défini comme étant* $\mu_a^\otimes((w_1, e_1))((w_2, e_2)) = \mu_a(w_1)(w_2) \times \mu_{\text{pre}}(w_2)(e_2) \times \mu_a^\mathcal{E}(e_1)(e_2)$.

Nous avons donc dans cette définition le comportement classique de la mise à jour du vocabulaire et des lois d'observations, et le comportement que nous avons vu pour la mise à jour des probabilités pour la loi des mondes et la loi des probabilités.

Grâce à ce *product update*, il est possible de définir la sémantique sur une structure de probabilités de l'opérateur dynamique.

De nouveau (c.f. rq. 2.59, page 59), nous ne définissons pas un nouveau langage dynamique contenant un modèle conditionnel et nous considérons par abus qu'il est compris dans le langage $\mathcal{L}_{\text{PDEL}}$.

Définition 6.25. *Soit* $\langle \mathfrak{F} = \langle V, \theta, \Omega, \Pi \rangle, s \rangle$ *une structure de probabilités pointée et* $\langle \chi = \langle V^+, \Omega^+, \Pi^+, \Phi, \theta^{\text{pre}}, V_-, \theta_- \rangle, x \rangle$ *un modificateur conditionnel pointé. L'opérateur dynamique est interprété comme suit :*

$$\langle \mathfrak{F}, s \rangle \models [\chi, x]\phi \text{ ssi } \langle \mathfrak{F}, s \rangle \models [x \sqsubseteq V^+] \text{supp}_{\text{prop}}(\Theta_\mathfrak{F}^+) \implies \langle \mathfrak{F} \times \chi, s^x \rangle \models \phi$$

où s^x *est la définition du nouvel état par le* product update *(déf. 6.23) et* $\Theta_\mathfrak{F}^+$ *est définition de la précondition conditionnelle symbolique (déf. 6.18).*

Pour ce qui est de la traduction locale, il n'est ici plus question d'utiliser θ^+ qui n'est plus présent dans le modificateur conditionnel. À la place, il faut utiliser le support propositionnel de $\Theta^+_{\mathfrak{F}}$.

La traduction locale d'une formule épistémique dynamique par un modificateur de probabilités conditionnelles avec des postconditions est définie ainsi :

Définition 6.26 (Traduction locale pour PDEL)**.** *Soit $\mathfrak{F} = \langle V, \theta, \Omega, \Pi\rangle$ une structure de probabilités, χ un modificateur conditionnel $\langle V^\star, \Omega^\star, \Pi^\star, \Phi, \theta^{\text{pre}}, V_-, \theta_-\rangle$ et ϕ une formule de $\mathcal{L}_{\text{PDEL}}$. En complément de la translation locale non dynamique (déf. 5.15), la* traduction locale booléenne de $[\chi, x]$ de $\mathfrak{F}$*, notée $|||[\chi, x]||_{\mathfrak{F}}$, est la fonction booléenne définie inductivement de la manière suivante :*

$$|||[\chi, x]\phi||_{\mathfrak{F}} \coloneqq [x \sqsubseteq V^\star] \operatorname{supp}_{\text{prop}}(\Theta^\star_{\mathfrak{F}}) \to [V^\circ_- \mapsto V_-][x \sqsubseteq V^\star][V_- \mapsto \theta_-(V_-)]||\phi||_{\mathfrak{F}\otimes\chi}$$

Cette définition est identique à la définition de la traduction locale d'une formule de $\mathcal{L}_{\text{PDEL}}$ dans SMCDEL (déf. 2.61), modulo le fait que c'est $\Theta^+_{\mathfrak{F}}$ qui est utilisé à la place de θ^+.

Exemple 6.27. *Reprenons l'exemple 6.22. Nous avons seulement les valuations $\emptyset$ et $\{e\}$ qui n'appartenait pas au support de $\Theta^\star_{\mathfrak{F}}$. Ainsi, si nous avons $x = \{e\}$, le conditionnement de $\operatorname{supp}_{\text{prop}}(\Theta^\star_{\mathfrak{F}})$ par la valuation $\{e\}$ est la formule $(p \wedge q) \vee (\neg p \wedge q) \vee (p \wedge \neg q) \equiv \neg(\neg p \wedge \neg q)$, qui est ainsi la formule qui représente toutes les valuations possibles de mondes pointés où le modificateur observationnel pointé par e peut être appliqué.*

Théorème 6.28. *La traduction donnée par la définition 6.26 est vraie : pour toute structure de probabilités pointée $\langle \mathfrak{F}, s\rangle$, tout modificateur conditionnel pointé $\langle \chi, x\rangle$ et toute formule ϕ, nous avons : $\langle \mathfrak{F}, s\rangle \models [\chi, x]\phi \iff s \models |||[\chi, x]\phi||_{\mathfrak{F}}$*

La preuve s'inspire de la preuve présentée par Gattinger, que nous avons réécrite preuve 2.62. Nous utilisons simplement une définition modifiée de la sémantique de l'opérateur dynamique qui utilise $\Theta^+_{\mathfrak{F}}$ au lieu de θ^+.

Nous devons néamoins prouver que le support propositionnel de $\Theta^+_{\mathfrak{F}}$ permet bel et bien de tenir le même rôle que θ^+ dans le cas non conditionnel.

Démonstration. La preuve ressemble grandement à la preuve du théorème 2.62.

Nous commençons par considérer qu'un événement x est applicable sur un état s, de sorte que $\langle \mathfrak{F}, s\rangle \models [x \sqsubseteq V^+] \operatorname{supp}_{\text{prop}}(\Theta^+_{\mathfrak{F}})$ (voir remarque 6.20). On a montré que $s^x \models ||\phi||_{\mathfrak{F}\times\chi}$ est équivalent à $s \models [V^\circ_- \mapsto V_-][x \sqsubseteq V^+][V_- \mapsto \theta_-(V_-)]||\phi||_{\mathfrak{F}\times\chi}$ (1) (voir preuve du théorème 2.62).

Par la sémantique de $[\chi, x]$, nous pouvons écrire : $\langle \mathfrak{F}, s\rangle \models [\chi, x]\phi$
$\iff \langle \mathfrak{F}, s\rangle \models [x \sqsubseteq V^+] \operatorname{supp}_{\text{prop}}(\Theta^+_{\mathfrak{F}})$ impl. $\langle \mathfrak{F} \times \chi, s^x\rangle \models \phi$

déf. 6.25

$\iff s \models [x \sqsubseteq V^+]\,\mathrm{supp}_{\mathrm{prop}}(\Theta^+_{\mathfrak{F}})$ impl. $s^x \models ||\phi||_{\mathfrak{F}\times\chi}$

par hypothèse d'induction et la traduction locale (déf. 5.17).

$\iff s \models [x \sqsubseteq V^+]\,\mathrm{supp}_{\mathrm{prop}}(\Theta^+_{\mathfrak{F}})$ impl. $s \models [V^\circ_- \mapsto V_-][x \sqsubseteq V^+][V_- \mapsto \theta_-(V_-)]||\phi||_{\mathfrak{F}\times\chi}$

Voir preuve du théorème 2.62

$\iff s \models [x \sqsubseteq V^+]\,\mathrm{supp}_{\mathrm{prop}}(\Theta^+_{\mathfrak{F}}) \to [V^\circ_- \mapsto V_-][x \sqsubseteq V^+][V_- \mapsto \theta_-(V_-)]||\phi||_{\mathfrak{F}\times\chi}$
$\iff \langle \mathfrak{F}, s\rangle \models [\chi, x]\phi$

(déf. 6.26)

□

Nous voulons maintenant montrer les équivalences entre le *model checking* d'une structure de probabilités et sa version explicite.

Théorème 6.29. *Soit χ un modificateur conditionnel.* $\mathrm{expl}(\chi)$ *(déf. 6.17) préserve la vérité : pour tout structure de probabilités pointé $\langle \mathfrak{F}, s\rangle$, tout modificateur conditionnel $\langle \chi, x\rangle$ et toute formule ϕ sur le vocabulaire de $\mathfrak{F}$, on a :*

$$\langle \mathfrak{F} \otimes \chi, s^x\rangle \models \phi \iff \langle \mathrm{expl}(\mathfrak{F}), s\rangle \otimes \langle \mathrm{expl}(\chi), x\rangle \models \phi$$

Démonstration. Prenons de nouveau la fonction d'étiquetage λ qui assigne à chaque couple monde/événement une valuation :

$$\lambda(s, x) \coloneqq (s \backslash V_-) \cup (s \cap V_-)^\circ \cup x \cup \{p \in V_- \mid s \cup x \models \theta_-(p)\}$$

qui est la définition de s^x 6.23

∗ C1 et C2 ne changent pas et sont identiques à la preuve du théorème 6.11. La preuve pour C3 est très similaire en remplaçant $||\theta^+||_{\mathfrak{F}}$ par $\mathrm{supp}_{\mathrm{prop}}(\Theta^+)$.

∗ Pour montrer C3 du lemme 5.19, nous prenons un quelconque $s^\otimes \in 2^{V \cup V^+ \cup V^\circ_-}$. Nous voulons montrer que $s^\otimes \models \theta^\otimes$ ssi il y a un monde (s, x) tel que $\lambda(s, x) = s^\otimes$.

De gauche à droite : supposons $s^\otimes \models \theta^\otimes$, i.e.

$$s^\otimes \models [V_- \mapsto V^\circ_-](\theta \wedge \mathrm{supp}_{\mathrm{prop}}(\Theta^+_{\mathfrak{F}})) \wedge \bigwedge_{p \in V_-} (p \leftrightarrow [V_- \mapsto V^\circ_-](\theta_-(p))) \tag{6.1}$$

Soit $s \coloneqq (s^\otimes \cup (V \backslash V_-)) \cup \{p \in V_- \mid p^\circ \in s^\otimes\}$. De 6.1, on a $s \models \theta$, ce qui signifie que s est un état de $\mathfrak{F}$ et aussi un monde de $\mathrm{expl}(\mathfrak{F})$. Ensuite, soit $x \coloneqq s^\otimes \cap V^+$. Maintenant par la définition de λ, on a : $\lambda(s, x) = s^x = s^\otimes$.

De droite à gauche, supposons que nous avons un monde (s, x) tel que $\lambda(s, x) = s^x = s^\otimes$. Maintenant nous devons montrer que 6.1 pour $s^\otimes = (s \backslash V_-) \cup (s \cap V_-)^\circ \cup x \cup \{p \in V_- \mid s \cup x \models \theta_-(p)\}$.

Premièrement, notons que s est un monde de expl($\mathfrak{F}$) et aussi un état de $\mathfrak{F}$, i.e. on a $s \models \theta$. Deuxièmement, (s, x) est un monde de expl($\mathfrak{F}$) $\times$ expl(χ), c'est pourquoi nous avons $s \cup x \models \text{supp}_{\text{prop}}(\Theta^{+}_{\mathfrak{F}})$.

Troisièmement, nous avons par définition de λ que pour tout $q \in V_{-}$:

- $q^{\circ} \in s^{\otimes}$ ssi $q \in s$;
- $q \in s^{\otimes}$ ssi $s \models \theta_{-}(q)$.

Ces trois cas impliquent 6.1.

$*$ Montrons maintenant C4.

Prenons deux mondes (s, t) et (t, y). Nous devons montrer que

$\Pi^{\otimes}_{a}(\lambda(s, x) \cup \lambda(t, y)') = \mu(s, x)(t, y)$. Pour cela, montrons la chaîne d'égalité suivante : $\Pi^{\otimes}_{a}(\lambda(s, x) \cup \lambda(t, y)')$

$= ([V_{-} \mapsto V^{\circ}_{-}][V'_{-} \mapsto (V^{\circ}_{-})'](\Pi_{a} \cdot (\Theta^{+}_{\mathfrak{F}})') \cdot \Pi^{+}_{a})($
$(s \backslash V_{-}) \cup (s \cap V_{-})^{\circ} \cup x \cup \{p \in V_{-} \mid s \cup x \models \theta_{-}(p)\} \cup$
$((t \backslash V_{-}) \cup (t \cap V_{-})^{\circ} \cup y \cup \{p \in V_{-} \mid t \cup y \models \theta_{-}(p)\})')$

V_{-} et V'_{-} n'apparaissent pas dans les arguments de la PBF, comme les anciennes relations ne dépendent pas des nouvelles valeurs des atomes propositonnels modifiés. Ainsi, on peut simplifier la fonction pour avoir :

$= ([V_{-} \mapsto V^{\circ}_{-}][V'_{-} \mapsto (V^{\circ}_{-})'](\Pi_{a} \cdot (\Theta^{+}_{\mathfrak{F}})') \cdot \Pi^{+}_{a})((s \backslash V_{-}) \cup (s \cap V_{-})^{\circ} \cup x \cup (t \backslash V_{-})' \cup (t^{\circ} \cap V^{\circ}_{-})' \cup y'$

dans laquelle on peut séparer les trois fonctions par leurs vocabulaires respectifs :

$$
\begin{aligned}
&= ([V_{-} \mapsto V^{\circ}_{-}][V'_{-} \mapsto (V^{\circ}_{-})']\Pi_{a})((s \backslash V_{-}) \cup (s \cap V_{-})^{\circ} \cup (t \backslash V_{-})' \cup (t^{\circ} \cap V^{\circ}_{-})') \\
&\times [V'_{-} \mapsto (V^{\circ}_{-})']((\Theta^{+}_{\mathfrak{F}})')((t \backslash V_{-})' \cup (t^{\circ} \cap V^{\circ}_{-})' \cup y') \\
&\times \Pi^{+}_{a}(x \cup y')
\end{aligned}
$$

Enlevons la substitution par $\circ$ dans l'argument de la première ligne pour voir qu'elle est égale à $\Pi_{a}(s \cup t')$. Par conséquent, toute la condition est égale à $\mu_{a}(s)(t) \times \mu_{\text{pre}}(t)(y) \times \mu^{\mathcal{E}}_{a}(x, y)$ qui est $\mu^{\otimes}_{a}(s, x)(t, y)$ par les définitions de expl($\mathcal{M}$) 5.8 et expl($\mathcal{E}$) 6.17.

□

Théorème 6.30. *Soit $\lambda^{V^{*}}_{E}$ une fonction d'étiquetage pour un ensemble d'événements E.* $\text{symb}_{\lambda^{V^{*}}_{E}}(\mathcal{E})$ *(déf. 6.14) préserve la vérité : pour toute structure de Kripke pointée $\langle \mathcal{M}, w \rangle$, tout modèle conditionnel pointé $\langle \mathcal{E}, e \rangle$ et toute formule ϕ sur le vocabulaire de $\mathcal{M}$, on a :*

$$\langle \mathcal{M} \otimes \mathcal{E}, (w, e) \rangle \models \phi \iff \langle \text{symb}_{\lambda^{U}_{W}}(\mathcal{M}), \lambda^{U}_{W}(w) \rangle \otimes \langle \text{symb}_{\lambda^{V^{*}}_{E}}(\mathcal{E}), \lambda^{V^{*}}_{E}(e) \rangle \models \phi$$

où λ^{U}_{W} est une fonction d'étiquetage avec $V \subseteq U$ qui étend val.

Démonstration. Pour plus de lisibilité, nous noterons $\widehat{w}$ pour $\lambda_W^V(w)$ et $\widehat{e}$ pour $\lambda_E^{V^+}(e)$.

De nouveau, prenons :

$$\lambda(w,a) \coloneqq (\widehat{w} \backslash V_-) \cup (\widehat{w} \cap V_-)^\circ \cup \widehat{e} \cup \{p \in V_- \mid \widehat{w} \cup \widehat{e} \models \theta_-(p)\}$$

∗ C1 et C2 ne changent pas et sont identiques à la preuve du théorème 6.9. Par ailleurs, la preuve pour C3 est très similaire à la preuve 6.9 en remplaçant $||\theta^+||_{\mathfrak{F}}$ par $\mathrm{supp}_{\mathrm{prop}}(\Theta^+_{\mathfrak{F}})$.

∗ Montrons C3. Prenons un état quelconque $s^\otimes \in 2^{V \cup V^+ \cup V_-^\circ}$. Nous voulons montrer que $s^\otimes \models \theta^\otimes$ ssi il y a un monde (w,e) tel que $\lambda(w,e) = s^\otimes$. Prenons aussi : $s \coloneqq (s^\otimes \cap (V \backslash V_-)) \cup \{p \in V \mid p^\circ \in s^\otimes\}$ et $x \coloneqq s^\otimes \cap V^+$.

De gauche à droite, supposons $s^\otimes \models \theta^\otimes$, i.e.

$$s^\otimes \models [V_- \mapsto V_-^\circ](\theta \wedge \mathrm{supp}_{\mathrm{prop}}(\Theta^+_{\mathfrak{F}})) \wedge \bigwedge_{p \in V_-} (p \leftrightarrow [V_- \mapsto V_-^\circ](\theta_-(p))) \tag{6.2}$$

De 6.2, on a $s \models \theta$, ce qui signifie que s est un état de $\mathrm{symb}_{\lambda_W^V}(\mathcal{M})$. Par la condition C3, il y a un monde w de $\mathcal{M}$ tel que $\widehat{w} = s$. Nous avons aussi $s \cup x \models \mathrm{supp}_{\mathrm{prop}}(\Theta^+_{\mathfrak{F}})$. Par la définition de la représentation symbolique de μ_{pre} (déf. 6.13), on a $\forall \phi \in \Phi$, $\mathrm{symb}_{\lambda_E^{V^+}}(\mu_{\mathrm{pre}})(\phi) = \{\lambda_E^{V^+} \sqsubseteq V^+(e) \colon \mu_{\mathrm{pre}}(\phi)(e)\}_{e \in E}$. Par conséquent, il y a une action $e \in E$ et une formule ϕ tel que $\widehat{e} = x$ et $\theta^{\mathrm{pre}}(\phi)(e) > 0$, ce qui implique que $\mu_{\mathrm{pre}}(w)(e) > 0$. Ainsi, (w,e) est un monde de $\mathcal{M} \otimes \mathcal{E}$. De plus, par 6.2 on a pour tout $q \in V_-$ que $q \in s^\otimes$ ssi $s^\otimes \models \mathrm{post}(q)(e)$. Par la définition de λ, on a $\lambda(w,e) = s^\otimes$.

De droite à gauche, supposons que nous avons un monde (w,e) dans $\mathcal{M} \times \mathcal{E}$ tel que $\lambda(w,e) = s^\otimes$. Nous avons maintenant à montrer $s^\otimes \models \theta^\otimes$, i.e. la formule 6.2. Par définition de s et x, nous avons $\widehat{w} = s$ et $\widehat{e} = x$. w est un monde de $\mathcal{M}$ ainsi $s = \widehat{w}$ est un état de $\mathfrak{F}$, i.e. on a $s \models \theta$. (w,e) est un monde de $\mathcal{M} \otimes \mathcal{E}$, par conséquent $\mu_{\mathrm{pre}}(w)(e) > 0$. Avec $\widehat{w} \cup \widehat{e} = s \cup x$, on a $\theta^{\mathrm{pre}}(x) > 0$ et $s \models \phi$ avec $\phi \in \Phi$. De ce fait, $s \cup x \models \mathrm{supp}_{\mathrm{prop}}(\Theta^+_{\mathfrak{F}})$ par définition $\Theta^+_{\mathfrak{F}}$ (déf. 6.18). Ensuite, nous avons par la définition de λ que pour chaque $q \in V_-$, $q^\circ \in s^\otimes$ ssi $q \in s$ et $q \in s^\otimes$ ssi $s \models \theta_-(q)$. Tout ceci implique la formule 6.2.

∗ Montrons maintenant C4.

Prenons deux mondes quelconques (w_1,e_1) et (w_2,e_2) dans le modèle explicite mis à jour. Nous devons montrer que $\Pi_a^\otimes(\lambda(w_1,e_1) \cup \lambda(w_2,e_2)') = \mu_a^\otimes(s,x)(t,y)$. Pour cela, écrivons les égalités suivantes. Nous avons $\Pi_a^\otimes(\lambda(w_1,e_1) \cup \lambda(w_2,e_2)')$

$= ([V_- \mapsto V_-^\circ][V_-' \mapsto (V_-^\circ)'](\Pi_a \cdot (\Theta^+_{\mathfrak{F}})' \cdot \Pi_a^+)($
$(\widehat{w_1} \backslash V_-) \cup (\widehat{w_1} \cap V_-)^\circ \cup \widehat{e_1} \cup \{p \in V_- \mid \widehat{w_1} \cup \widehat{e_1} \models \theta_-(p)\} \cup$
$((\widehat{w_2} \backslash V_-) \cup (\widehat{w_2} \cap V_-)^\circ \cup \widehat{e_2} \cup \{p \in V_- \mid \widehat{w_2} \cup \widehat{e_2} \models \theta_-(p)\})'$

)

De nouveau, V_- et V'_- n'apparaissent pas dans les arguments de la PBF, alors ils peuvent être enlevés de la valuation pour obtenir :

$$= ([V_- \mapsto V_-^\circ][V'_- \mapsto (V_-^\circ)'](\Pi_a(\Theta^+_{\mathfrak{F}})' \cdot \Pi^+_a)(\\ (\widehat{w_1} \backslash V_-) \cup (\widehat{w_1} \cap V_-)^\circ \cup \widehat{e_1} \cup \\ (\widehat{w_2} \backslash V_-)' \cup (\widehat{w_2}^\circ \cap V_-^\circ)' \cup \widehat{e_2}')$$

ce qui se sépare en :

$$= ([V_- \mapsto V_-^\circ][V'_- \mapsto (V_-^\circ)']\Pi_a)((\widehat{w_1} \backslash V_-) \cup (\widehat{w_1} \cap V_-)^\circ \cup (\widehat{e_1} \backslash V_-)' \cup (\widehat{e_1}^\circ \cap V_-^\circ)') \\ \times [V'_- \mapsto (V_-^\circ)'](({\Theta^+_{\mathfrak{F}}})')((\widehat{w_2} \backslash V_-)' \cup (\widehat{w_2}^\circ \cap V_-^\circ)' \cup \widehat{e_2}') \\ \times \Pi^+_a(\widehat{w_2} \cup \widehat{e_2}')$$

Maintenant il est possible d'enlever la substitution par $\circ$ pour montrer l'égalité à $\Pi_a(\widehat{w_1} \cup \widehat{w_2}')$. Dans la preuve du théorème 5.21, il a été montré que les conditions du lemme 5.19 s'applique à λ^V_W dans la construction de $\mathrm{symb}_{\lambda^V_W}(\mathcal{M})$. En particulier, par la condition C4, nous avons $\Pi_a(\widehat{w_1} \cup \widehat{w_2}') = \mu_a(w_1)(w_2)$. De plus, par la définition 6.14, nous avons $\Pi^+_a(\widehat{e_1} \cup \widehat{e_2}') = \mu^{\mathcal{E}}_a(e_1)(e_2)$ et $\theta^{\mathrm{pre}}(\phi)(\widehat{e_2}) = \mu_{\mathrm{pre}}(w_2)(e_2)$ pour $\phi \in \Phi$ tel que $\langle \mathcal{M}, w_2 \rangle \models \phi$, i.e. $(\Theta^+_{\mathfrak{F}})(\widehat{w_2} \cup \widehat{e_2}) = \mu_{\mathrm{pre}}(w_2)(e_2)$. Par conséquent, nous avons :

$\Pi^{\otimes}_a(\lambda(w_1, e_1) \cup \lambda(w_2, e_2)') = \mu_a(w_1)(w_2) \times \mu_{\mathrm{pre}}(w_2)(e_2) \times \mu^{\mathcal{E}}_a(e_1)(e_2)$.

Par définition du *product update* explicite (déf. 6.23), c'est $\mu^{\otimes}_a(w_1, e_1)(w_2, e_2)$. □

Comme nous venons de la voir, le *model checking* incluant le *product update* est bien équivalent entre les structures de probabilités et leurs versions explicites, et inversement.

Nous allons maintenant discuter de quelques points de réflexion et de modélisation avant de passer au chapitre suivant.

6.3 Relation entre loi des mondes θ^+ et précondition conditionnelle symbolique Θ^+

Nous avions vu qu'il était possible de passer d'un modèle observationnel à un modèle conditionnel et inversement dans la section 4.5. Qu'en est-il pour les modèles symboliques ? Peut-on à partir d'un modèle observationnel $\mathcal{E}$ créer un modificateur conditionnel qui soit équivalent au modificateur observationnel $\mathrm{symb}(\mathcal{E})$?

Nous avons vu que dans le cas des modificateur observationnel, θ^+ est défini sur $V \cup V^+$ et représente la loi des événements comme pouvait le faire θ dans une structure de croyances ou de probabilités. Néanmoins θ^+ n'est pas définie que

sur V^+, mais aussi sur V. Cela permet d'encoder les conditions d'applications des événements sur les mondes d'une structure de croyances ou de probabilités définis sur le vocabulaire V.

Dans le cas de la représentation symbolique de PDEL, nous avons vu que nous utilisons la précondition conditionnelle symbolique $\Theta^+_{\mathfrak{F}}$ (édf. 6.18).

Pour cela montrer l'équivalence entre les deux modèles, nous avons besoin de définir une équivalence entre deux structures de probabilités très similaires.

Définition 6.31. *Soient deux structures de probabilités* $\mathfrak{F}^1 = \langle V, \Omega, \Pi^1, \theta\rangle$ *et* $\mathfrak{F}^2 = \langle V, \Omega, \Pi^2, \theta\rangle$. Π^1_a *et* Π^2_a *sont dites équivalentes si et seulement si :*

$$\forall \phi \in \mathcal{L}_{\text{PDEL}},\ \forall s \in 2^V,\ \langle \mathfrak{F}^1, s\rangle \models \phi \iff \langle \mathfrak{F}^2, s\rangle \models \phi$$

Nous exprimons l'équivalence des deux structures grâce à la proposition suivante.

Proposition 6.32. *Soit deux structures de probabilités* $\mathfrak{F}^1 = \langle V, \Omega, \Pi, \theta\rangle$ *et* $\mathfrak{F}^2 = \langle V, \Omega, (\Pi_a \cdot \theta')_{a\in\mathfrak{A}}, \theta\rangle$. $\mathfrak{F}^1$ *et* $\mathfrak{F}^2$ *sont équivalentes.*

Démonstration. On veut montrer que les structures $\mathfrak{F}^1$ et $\mathfrak{F}^2$ sont équivalentes quand l'une des deux contient la loi des mondes primée dans la loi de probabilités.

Les deux structures $\mathfrak{F}^1$ et $\mathfrak{F}^2$ sont identiques en tout point sauf pour les lois de probabilités. Pour montrer que le *model checking* est équivalent, il suffit de montrer que celui-ci est identique dans le cas de l'utilisation de lois de probabilités, i.e. la traduction locale pour vérifier une formule probabiliste est identique. Il suffit alors de montrer que $||\phi||_{\mathfrak{F}^1} = ||\phi||_{\mathfrak{F}^2}$.

Rappelons que la traduction d'une formule probabiliste est la suivante : $||\alpha_1 \mathrm{Pr}_a(\phi_1) + \cdots + \alpha_k \mathrm{Pr}_a(\phi_k) \geq \beta||_{\mathfrak{F}} \coloneqq$

$$\mathrm{Cut}_{\geq}(\textstyle\sum_{i=1}^{k} \alpha_i \times \mathrm{Marg}^+_{V'}\left(\Pi_a \cdot \theta' \cdot (||\phi_i||_{\mathfrak{F}})'\right), \beta \times \mathrm{nonzero}(\mathrm{Marg}^+_{V'}\left(\Pi_a \cdot \theta'\right)))$$

Soit ϕ une formule probabiliste, nous avons alors :

$||\phi||_{\mathfrak{F}^2}$

$= \mathrm{Cut}_{\geq}(\sum_{i=1}^{k} \alpha_i \times \mathrm{Marg}^{+}_{V'}\left((\Pi_a \cdot \theta') \cdot \theta' \cdot (||\phi_i||_{\mathfrak{F}^2})'\right), \beta \times \mathrm{nonzero}(\mathrm{Marg}^{+}_{V'}\left((\Pi_a \cdot \theta') \cdot \theta'\right)))$

or on a $\theta' \wedge \theta' \equiv \theta'$ et par la même occasion $\Pi_a \cdot \theta' \equiv \Pi_a \cdot \theta' \cdot \theta'$

$= \mathrm{Cut}_{\geq}(\sum_{i=1}^{k} \alpha_i \times \mathrm{Marg}^{+}_{V'}\left(\Pi_a \cdot \theta' \cdot (||\phi_i||_{\mathfrak{F}^2})'\right), \beta \times \mathrm{nonzero}(\mathrm{Marg}^{+}_{V'}\left(\Pi_a \cdot \theta'\right)))$

par hypothèse d'induction sur les cas non probabilistes :

$= \mathrm{Cut}_{\geq}(\sum_{i=1}^{k} \alpha_i \times \mathrm{Marg}^{+}_{V'}\left(\Pi_a \cdot \theta' \cdot (||\phi_i||_{\mathfrak{F}^1})'\right), \beta \times \mathrm{nonzero}(\mathrm{Marg}^{+}_{V'}\left(\Pi_a \cdot \theta'\right)))$

$= ||\phi||_{\mathfrak{F}^1}$

□

Une façon plus simple de voir les choses est la suivante : par définition, Π_a est toujours couplé à la fonction θ'. C'est pourquoi pour $\mathfrak{F}^1$ et $\mathfrak{F}^2$ sont équivalents car avoir θ' dans Π_a^2 n'est qu'une simple redondance en terme de *model checking*.

Nous allons maintenant pouvoir montrer qu'il est possible de représenter un modèle observationnel par un modificateur conditionnel, de sorte à ce que le *product update* soit équivalent à partir de toute structure de probabilités.

Proposition 6.33. *Soit* $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mu^{\mathcal{E}}, \mathrm{pre}^o \rangle$ *un modèle observationnel. Pour toutes structure de probabilités* $\mathfrak{F}$, $\lambda_E^{V^+}$, *et* θ^+ *issu de* $\mathrm{symb}_{\lambda_E^{V^+}}(\mathcal{E})$ *et* $\Theta^+_{\mathfrak{F}}$ *issu de* $\mathrm{symb}_{\lambda_E^{V^+}}(\mathrm{cond}(\mathcal{E}))$ *(*cond *: déf. 4.53), on a :*

$$||\theta^+||_{\mathfrak{F}} = \Theta^+_{\mathfrak{F}}$$

Démonstration. cond ne modifie pas E, $R^{\mathcal{E}}$ et $\mu^{\mathcal{E}}$, ainsi les représentations symboliques de ces élements pour $\mathrm{symb}_{\lambda_E^{V^+}}(\mathcal{E})$ et $\mathrm{symb}_{\lambda_E^{V^+}}(\mathrm{cond}(\mathcal{E}))$ sont identiques.

Par la définition de $\mathrm{cond}(\mathcal{E})$, on obtient Φ et μ_{pre} :

- $\Phi = \{\mathrm{pre}^o(e) \mid e \in E\}$;
- $\forall \phi \in \Phi \forall e \in E,\ \mu_{\mathrm{pre}}(\phi)(e) = \begin{cases} 1 & \text{si } \mathrm{pre}^o(e) = \phi \\ 0 & \textit{sinon} \end{cases}$.

et on obtient :

$$\begin{aligned} \mathrm{symb}_{\lambda_E^{V^+}}(\mu_{\mathrm{pre}})(\phi) &= \{\lambda_E^{V^+}(e) \sqsubseteq V^+ \colon 1 \mid e \in E \text{ si } \mathrm{pre}^o(e) = \phi\} \\ &= \sum_{e \in E} \lambda_E^{V^+}(e) \sqsubseteq V^+ \text{ si } \mathrm{pre}^o(e) = \phi,\ 0 \text{ sinon} \\ &= \lambda_E^{V^+}(e) \sqsubseteq V^+ \text{ pour } e \in E \text{ tel que } \mathrm{pre}^o(e) = \phi \end{aligned}$$

Ce qui revient à avoir $\forall e \in E,\ \theta^{\text{pre}}(\phi)(\lambda_E^{V^+}(e)) = 1$.

Par la définition de la représentation symbolique d'un modèle observationnel 6.7, on a : $\theta^+ = \bigvee_{e \in E}(\lambda(e) \sqsubseteq V^+ \wedge \text{pre}^o(e))$.

Par définition de $\Theta^+_{\mathfrak{F}}$ (déf. 6.18), on a :

$$
\begin{aligned}
\Theta^+_{\mathfrak{F}} &= \sum_{\phi \in \Phi} \text{symb}_{\lambda_E^{V^+}}(\mu_{\text{pre}})(\phi) \cdot ||\phi||_{\mathfrak{F}} \\
&\text{Ayant } \Phi = \{\text{pre}^o(e) \mid e \in E\}, \text{ on a :} \\
&= \sum_{e \in E} \text{symb}_{\lambda_E^{V^+}}(\mu_{\text{pre}})(\text{pre}^o(e)) \cdot ||\,\text{pre}^o(e)||_{\mathfrak{F}} \\
&\text{or } \text{symb}_{\lambda_E^{V^+}}(\mu_{\text{pre}})(\text{pre}^o(e)) = 1 \text{ seulement pour l'argument } \lambda_E^{V^+}(e) \\
&= \sum_{e \in E} \lambda_E^{V^+}(e) \sqsubseteq V^+ \wedge ||\,\text{pre}^o(e)||_{\mathfrak{F}} \\
&= ||\theta^+||_{\mathfrak{F}}
\end{aligned}
$$

□

Notons qu'ici $\Theta^+_{\mathfrak{F}}$ est bien une PBF mais n'a des valeurs que dans $\mathbb{B}$, c'est pourquoi il est possible de la comparer à la fonction propositionnelle qu'est $||\theta^+||_{\mathfrak{F}}$.

Nous pouvons donc voir que dans le cas du passage d'un modèle observationnel à un modèle conditionnel, le θ^+ qu'on aurait pu avoir sans passer à un modèle conditionnel revient à Θ^+ dans le cas conditionnel.

Ainsi, un modèle conditionnel peut simuler un modificateur observationnel puisqu'il est possible de construire une précondition conditionnelle μ_{pre} qui permet d'obtenir la formule de θ^+. Notons que cette équivalence n'est possible que parce les formules de Φ peuvent ici avoir des modèles en commun.

Maintenant que nous pouvons faire un *product update*, il est possible de comparer les résultats après la transformation cond que nous avions défini plus tôt (sec. 6.3, déf. 4.53).

Théorème 6.34. *Soit $\mathcal{E} = \langle E, R^{\mathcal{E}}, \mu^{\mathcal{E}}, \text{pre}\rangle$ un modèle observationnel pour un vocabulaire V et $\lambda^E_{V^+}$ la fonction d'étiquetage utilisée pour les deux transformations utilisées.*

Pour toute structure de probabilités $\mathfrak{F}$, on a $\mathfrak{F} \otimes \text{symb}_{\lambda_E^{V^+}}(\mathcal{E})$ et $\mathfrak{F} \otimes \text{symb}_{\lambda_E^{V^+}}(\text{cond}(\mathcal{E}))$ sont équivalents.

Démonstration. Notons $\mathfrak{F} = \langle V, \theta, \Omega, \Pi\rangle$, $\mathfrak{F}_o = \mathfrak{F} \otimes \text{symb}(\mathcal{E}) = \langle V_o, \theta_o, \Omega_o, \Pi_o\rangle$ et $\mathfrak{F}_c = \mathfrak{F} \otimes \text{symb}(\text{cond}(\mathcal{E})) = \langle V_c, \theta_c, \Omega_c, \Pi_c\rangle$ avec $\text{symb}(\mathcal{E}) = \langle V_o^+, \theta^+, \Omega_o^+, \Pi_o^+\rangle$ et $\text{symb}(\text{cond}(\mathcal{E})) = \langle V_c^+, \theta^{\text{pre}}, \Phi, \Omega_c^+, \Pi_c^+\rangle$.

Les nouveaux vocabulaires et relations symboliques sont définies identiquement de sorte à ce que $V_o = V_c$ (V et $\lambda^E_{V^+}$ sont identiques dans les deux cas : $V^+_o = V^+_c$) et $\Omega_o = \Omega_c$ (déf. de symb : 6.7, 6.14 et déf. des *product update* : 6.2 et 6.23).

i) Il faut maintenant montrer que θ_o est identique à θ_c. Montrons que $\theta \wedge ||\theta^+||_{\mathfrak{F}}$ est équivalent à $\theta \wedge \text{supp}_{\text{prop}}(\Theta^+_{\mathfrak{F}})$.

Pour cela, nous devons montrer que $||\theta^+||_{\mathfrak{F}}$ est équivalent à $\text{supp}_{\text{prop}}(\Theta^+_{\mathfrak{F}})$.

Or, par la proposition 6.33 nous savons que : $||\theta^+||_{\mathfrak{F}} = \Theta^+$, et $\Theta^+_{\mathfrak{F}}$ étant booléens, on a donc $||\theta^+||_{\mathfrak{F}} = \text{supp}(\Theta^+)$.

ii) Montrons maintenant que pour tout $a \in \mathfrak{A}$, Π_{oa} est équivalent à Π_{ca} (déf. 6.31).

Montrons que $\Pi_a \cdot \Pi^+_{oa}$ est équivalent à $\Pi_a \cdot (\Theta^+)' \cdot \Pi^+_{ca}$.

Or on sait que $\theta_o = \theta_c = ||\theta^+||_{\mathfrak{F}} = \text{supp}(\Theta^+)$, donc $\Pi_a \cdot \Pi^+_{oa} = \Pi_a \cdot \theta'_c \cdot \Pi^+_{oa}$.

Comme les PBFs Π^+_{oa} et Π^+_{ca} sont identiques de par les transformations symb et cond, nous avons donc $\Pi^+_{oa} \cdot (\theta_o)' \equiv \Pi^+_{ca} \cdot (\theta_c)'$.

Par la proposition 6.32, $\mathfrak{F} \otimes \text{symb}(\mathcal{E})$ et $\mathfrak{F} \otimes \text{symb}(\text{cond}(\mathcal{E}))$ sont donc équivalents.

□

Nous avons donc vu que par la fonction cond, il est possible d'obtenir un modèle conditionnel à partir d'un modèle observationnel, et que les représentations symboliques de ces modèles sont équivalents en terme de *model checking* via une structure de probabilités donné. C'est-à-dire qu'il est possible de simuler un modèle observationnel à partir d'un modèle conditionnel particulier. Ce résultat n'est pas surprenant, pourtant il n'est pas trivial. Néanmoins, il permet d'illustrer un sous-cas du fonctionnement d'un modèle conditionnel et cela n'est possible seulement parce que le modèle conditionnel en question est dénormalisé (avec des Φ mutuellement incohérentes entre elles).

6.4 Discussions supplémentaires

Nous allons maintenant discuter quelque peu nos modélisations et nos choix.

6.4.1 Modificateurs conditionnels ou observationnels

Nous avons défini deux types d'actions modélisables avec des préconditions et utilisables avec le *framework* de PDEL : les modèles conditionnels de la littérature définis par PDEL (déf. 4.11) et les modèles observationnels tels que définis précédemment (déf. 4.39), mais qui ne sont pas utilisés dans la littérature.

Pour des actions probabilistes qui ne possèdent pas de probabilités conditionnelles intrinsèques, il n'y a pas besoin de modéliser un modèle conditionnel comme

PDEL le fait. Pour tout événement naturel dont on ne connait pas ce type de probabilités, mais sur lesquels nous aurions envie de raisonner avec des probabilités, il semble tout à fait naturel d'opter pour l'utilisation des modèles observationnels.

Typiquement, c'est le cas pour tout jeu de cartes, par exemple, où l'on voudrait représenter les probabilités d'avoir une carte, comme au Poker ou à Hanabi ; ce sont deux exemples où les structures de Kripke sont valuation-injectives et n'ont pas besoin de ces probabilités conditionnelles.

C'est ce que nous expliquions lors des exemples d'un lancé de pièce (ex. 4.45) et d'un lancé de pièce truquée (ex. 4.46).

Ainsi, comme nous le verrons dans la section 7.3, pour notre modélisation du jeu Hanabi, nous opterons pour une modélisation avec des modèles d'observations (définis de manière symbolique grâce à des modificateurs de probabilités).

6.4.2 Obtenir une loi de probabilités Π à partir d'une loi d'observations Ω

De par les définitions des structures de probabilités, extension des structures de croyances, et des modificateurs observationnels et des modificateurs conditionnels, extensions des modificateurs de croyances, nous pouvons nous demander s'il est possible d'obtenir une structure probabilisée à partir d'une struture non probabilisée, et de la même manière avec les modificateurs.

Deux conditions sont nécessaires pour que la transformation soit naturelle : premièrement, avoir des distributions de probabilités a-uniformes et a-consistantes (déf. 4.4). Mais surtout, deuxièmement, avoir des probabilités uniformes entre les mondes, i.e. chaque monde possède un poids identique. En effet, l'a-uniformité ne suffit pas : les lois de probabilité discrètes depuis tous les mondes sont identiques mais les probabilités attribuées à tous les mondes à partir d'un monde donné ne sont pas uniformes (l'uniformité des probabilités implique l'a-uniformité, mais l'inverse n'est pas vrai).

Rappelons l'a-uniformité : $w, w' : (w, w') \in R_a \implies \mu_a(w) = \mu_a(w')$.

Précisons maintenant l'uniformité des lois de probabilité discrètes que nous souhaitons, qui soit cohérente avec l'a-consistance : $\forall w \in W, \mu_a(w)$ est uniforme, i.e. $\forall w' \in W$ tel que wR_aw', $\mu_a(w)(w') = 1/n$, où n est le nombre de mondes successeurs de w via R_a.

Effectivement, on ne souhaite pas que $\mu_a(w)$ soit purement uniforme, car elle est définie sur tous les mondes. Nous souhaitons alors qu'elle soit uniforme sur tous les mondes successeurs accessibles via R_a, pour garantir l'a-consistance.

Grâce à la définition de la normalisation qui intègre la loi des mondes (voir déf. 5.11, rq. 5.12 et ex. 5.13), la loi d'observations des agents sera alors exactement la loi de probabilités.

De cette manière et grâce à la modélisation en loteries à la place de distributions de probabilités, il est possible de calculer les différentes lois d'observations $\Omega_a, a \in \mathfrak{A}$, et de s'en servir pour créer les différentes lois de probabilités $\Pi_a,\ a \in \mathfrak{A}$.

Comme les valeurs des loteries sont identiques grâce à l'uniformité des probabilités, l'idée est de conserver les fonctions booléennes calculées dans Ω_a. Ainsi, toutes les valeurs dans la PBF Π_a seront de 0 ou 1 et la normalisation de cette PBF permettra d'obtenir les valeurs normalisées voulues. Le fait que toutes les valeurs soient identiques correspond à la a-uniformité. Le fait que s'il n'y a pas de relation entre deux mondes, alors la probabilité reliant les deux mondes est de 0 correspond à la a-consistance $((w, w') \notin R_a \implies \mu_a(w)(w') = 0)$.

Ainsi, la création de l'ensemble de fonctions Π d'une structure de probabilités se fait naturellement grâce à l'ensemble de fonctions booléennes Ω dans les cas a-uniforme, a-consistant et avec des lois de probabilité discrètes uniformes pour chaque agent et chaque monde.

Dans les cas contraires, il est difficile de prévoir une procédure de création des PBFs Π_a à partir de Ω_a. Dans le cas non a-uniforme, il est possible d'imaginer multiplier la PBF voulue par une PBF sur V qui attribuerait à chaque état un poids, un peu comme une loi des mondes pondérée. Dans le cas non a-consistant, il faut construire la PBF directement à partir des données du problème.

6.4.3 La dénormalisation initiale

Discutons maintenant du fait d'utiliser des loteries à la place des distributions de probabilités. En soi, pourquoi voulons-nous utiliser des structures dénormalisées ?

Ce choix a deux grands interêts dans la représentation des structures symboliques probabilistes :

- premièrement, cela peut permettre d'éviter une étape de normalisation lors de la création des structures ;
- deuxièment, elle permet d'avoir des PBFs aux domaines moindres. En effet, si comme nous l'avons vu précédemment, nous pouvons modéliser Π_a comme étant l'équivalent de Ω_a dans le cas a-uniforme et a-consistant, alors le domaine de Π_a sera binaire : avec 0 et 1. Ainsi, le nombre de valeurs étant plus petit, la structure de données ADD est plus compacte et comme nous le verrons plus tard, la compilation de ces fonctions sera plus efficace.

Cette utilisation est simple dans le cas où nous utilisons des ADDs pour représenter les PBFs et des BDDs pour les fonctions booléennes, car un BDD est un ADD si on considère $\top$ comme 1 et $\bot$ comme étant 0. En effet, nous avions vu que le nombre de feuilles des ADDs est égal au nombre de valeurs du domaine des

fonctions. Cela revient en fait à utiliser des BDDs le plus possible au lieu d'utiliser des structures similaires mais plus complexes comme les ADDs.

6.4.4 La normalisation au product update ou au model checking

Maintenant, demandons-nous pourquoi il est utile de ne pas normaliser après un *product update*. En effet, il serait tout à fait possible de normaliser après une mise à jour des connaissances.

Rappelons les différentes définitions que nous avons vues pour le *product update* probabiliste, respectivement, avec modificateur observationnel ou modificateur conditionnel :

- $\Pi_a^\otimes \coloneqq \Pi_a \cdot \Pi_a^+$ dans le cas d'un modificateur observationnel ;
- $\Pi_a^\otimes \coloneqq \Pi_a \cdot (\Theta^+)' \cdot \Pi^+$ dans le cas d'un modificateur conditionnel.

De manière peu formelle, si nous avions à définir un *product update* avec une re-normalisation il serait calculé de la sorte :

- $\Pi_a^\otimes \coloneqq \frac{\Pi_a \cdot \Pi_a^+}{\text{nonzero}(\text{Marg}^+_{V^+}(\Pi_a \cdot \Pi_a^+ \cdot (\theta^\otimes)'))}$, ou ;
- $\Pi_a^\otimes \coloneqq \frac{\Pi_a \cdot (\Theta^+)' \cdot \Pi^+}{\text{nonzero}(\text{Marg}^+_{V^+}(\Pi_a \cdot (\Theta^+)' \cdot \Pi_a^+ \cdot (\theta^\otimes)'))}$.

Mais cela ne serait en fait pas suffisant. En effet, il faudrait effectuer un filtre sur les mondes $\theta^\otimes$ afin que les probabilités sont corrigées en conséquence. L'idée est exactement la même que présentée précédement à la section 6.4.2 : il faut ajouter une loi des mondes pour rectifier correctement les probabilités. Nous aurions alors :

- $\Pi_a^\otimes \coloneqq \frac{\Pi_a \cdot \Pi_a^+ \cdot (\theta^\otimes)'}{\text{Marg}^+_{V^+}(\Pi_a \cdot \Pi_a^+ \cdot (\theta^\otimes)')}$ ou,
- $\Pi_a^\otimes \coloneqq \frac{\Pi_a \cdot (\Theta^+)' \cdot \Pi_a^+ \cdot (\theta^\otimes)'}{\text{Marg}^+_{V^+}(\Pi_a \cdot (\Theta^+)' \cdot \Pi^+ \cdot (\theta^\otimes)')}$.

Et si nous utilisions des postconditions, rappelons que $\theta^\otimes$ est calculé de la sorte :

- $\theta^\otimes \coloneqq [V_- \mapsto V_-^\circ](\theta \wedge ||\theta^+||_\mathfrak{F}) \wedge \bigwedge_{p \in V_-}(p \leftrightarrow [V_- \mapsto V_-^\circ](\theta_-(p)))$, ou encore ;
- $\theta^\otimes \coloneqq [V_- \mapsto V_-^\circ](\theta \wedge \text{supp}_{\text{prop}}(\Theta^+)) \wedge \bigwedge_{p \in V_-}(p \leftrightarrow [V_- \mapsto V_-^\circ](\theta_-(p)))$.

Nous admettrons sans problème que ces formules de renormalisation sont plus complexes et calculatoires : il faut ajouter une multiplication avec $\theta^\otimes$ qui n'était pas nécessaire précédemment et il faut normaliser les PBFs. Bien évidemment, cette étape de normalisation devra être faite, notamment lors du *model checking*, néanmoins l'idée est de conserver une PBF la plus simple possible afin que la structure de données soit la plus petite possible, et que donc les calculs effectués dessus soient les plus rapides possibles.

L'idée derrière est de pouvoir enchaîner différents *product update* sans n'avoir jamais à re-normaliser les structures. De cette manière, il n'y aura à normaliser qu'à un seul et unique moment : lors d'un *model checking*. Celui-ci normalise alors la structure et met ses calculs en cache de sorte à ne pas avoir à les refaire, si d'autres *model checking* ont lieu par la suite.

Admettons que nous souhaitons normaliser lors du *product update* et ne pas avoir à le faire lors du *model checking*. Ainsi, nous aurions une définition de la traduction locale différente :

$||\alpha_1 \mathrm{Pr}_a(\phi_1) + \cdots + \alpha_k \mathrm{Pr}_a(\phi_k) \geq \beta||_{\mathfrak{F}} \coloneqq$
$\mathrm{Cut}_{\geq}(\Sigma_{i=1}^{k} \alpha_i \times \mathrm{Marg}^{+}_{V'}((\Pi_a \cdot \theta') \cdot (||\phi||_{\mathfrak{F}})'), \beta \times \mathrm{nonzero}(\mathrm{Marg}^{+}_{V'}(\Pi_a \cdot \theta')))$
serait alors :

$\mathrm{Cut}_{\geq}(\Sigma_{i=1}^{k} \alpha_i \times \mathrm{Marg}^{+}_{V'}((\Pi_a \cdot \theta') \cdot (||\phi||_{\mathfrak{F}})'), \beta)$

Cette méthode semble nettement plus simple à première vue, à juste titre, mathématiquement.

Il s'agit alors clairement d'un compromis et de ce que l'on souhaite garder en mémoire. L'utilisation du cache dans tous les cas permettra de ne faire la normalisation qu'une seule et unique fois.

Ici, l'idée à retenir est que l'on souhaite de nouveau conserver des structures de données (ADDs) les plus légères possibles en taille et donc en contraintes, notamment en terme de nombre de feuilles. De cette manière, les algorithmes polynomiaux auront moins de données à traiter et seront plus rapides.

Après avoir fait ces quelques discussions et remarques, nous allons pouvoir passer au chapitre suivant qui détaille la mise en œuvre informatique du *framework* SMCDEL avec des probabilités, en terme de code, d'implémentation, d'optimisation faites, ainsi que la modélisation de notre exemple qu'est Hanabi.

7 | Mise en œuvre

Maintenant que vous avons vu les définitions formelles de la modélisation mathématique des structures de probabilités, nous allons voir comment nous avons pu mettre cela en œuvre.

Avant de discuter de notre implémentation en terme de code (sec. 7.2), de notre modélisation du jeu Hanabi (sec. 7.3) et des expérimentations faites dessus (chap. 8), nous allons faire une brève section sur un travail ayant eu lieu lors de la première année de thèse (sec. 7.1).

7.1 Hintikka's World

Nous n'allons pas nous étendre sur cette partie, néanmoins nous nous devions de mentionner un travail qui a été fait en 2019 en collaboration avec François Schwarzentruber et Tristan Charrier, de l'IRISA à Rennes.

Hintikka's World est un site internet [Sch18] destiné à promouvoir la logique épistémique via une représentation graphique ludique. Y sont présents des puzzles logiques classiques de l'intelligence artificielle afin de visualiser les états mentaux des agents, typiquement liés à la logique épistémique modélisée par des structures de Kripke. La première version du site implémentait les modèles de Kripke de manière explicite [Sch18], ce qui permettait d'implémenter des exemples simples. En 2019, nous avons cherché à y proposer des problèmes plus complexes, notamment le démineur et un Hanabi simplifié, grâce à l'implémentation symbolique des structures de Kripke avec des BDDs, via un binding de CUDD (*Colorado University Decision Diagram*, implémentation en C des BDDs et ADDs [Som18]), en Wasm [1], utilisé en NodeJS [2].

Ce travail a fait l'objet d'une démonstration à la conférence internationale IJCAI sous le nom *Hintikka's World : Scalable Higher-order Knowledge* en 2019

1. WebAssembly, abrégé wasm, est un standard du World Wide Web pour le développement d'applications. Il est conçu pour compléter JavaScript avec des performances supérieures.
2. Node.js est une plateforme logicielle libre en JavaScript, orientée vers les applications réseau évènementielles.

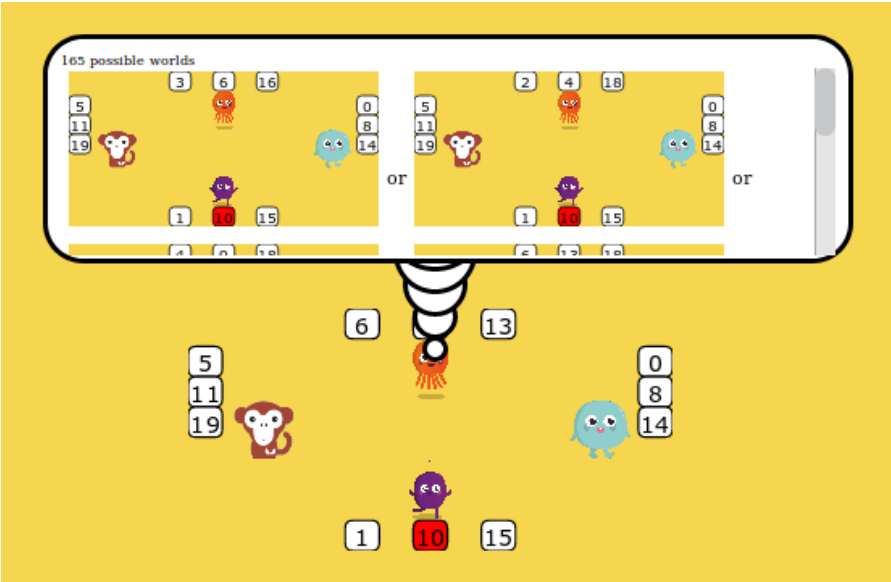


FIGURE 7.1 – Capture d'écran du site Hintikka's World

[CGNS19].

La figure 7.1 présente une capture d'écran de *Hintikka's World* montrant une situation d'Hanabi à quatre joueurs et trois cartes par main. Nous ne détaillerons pas les détails de l'implémentation. La conception du modèle a été faite en n'ayant pas connaissance des travaux de Gattinger sur le *model checking* symbolique de DEL. Néanmoins, la représentation des structures de Kripke symboliques est très similaire. Les différences notables entre les deux modèles étaient :

- la représentation des structures d'événements ; en effet il nous était impossible d'introduire des préconditions épistémiques ;
- une utilisation différentes des postconditions qui étaient pré-compilées dans une seule et unique formule propositionnelle.

Notons que l'implémentation que nous avions proposée fonctionne grâce à des variables observées, à l'instar de DELPAO [HM17], ce qui est très commode lors de la création des formules logiques de jeux tels que Hanabi. Nos noterons que cette formalisation avec des variables d'observations est aussi utilisée par Gattinger dans le cadre de la représentation de ses structures de connaissances. Par ailleurs, l'implémentation ne permet pas de différencier des mondes dont les valuations sont identiques (déf. 2.4). Mais c'est suffisant pour les jeux présentés sur le site web implémentés de manière symbolique.

A postériori, le modèle symbolique de ce site pourrait être grandement amélioré et optimisé. Ce travail a été très intéressant et nous a permis de maturer des idées de la représentation symbolique. Peu avant, Gattinger a proposé ses définitions.

Son modèle étant publié et formalisant des notions de préconditions épistémiques que nous n'avions pas, nous avons donc choisi de repartir de ses définitions pour proposer une version probabiliste de SMCDEL.

7.2 Implémentation de notre framework

Dans cette section, nous allons évoquer les implémentations des structures de données qui nous avons pu déjà présenter et comment nous les avons utilisées.

De la même manière que Gattinger a pu évoquer des *model checkers* symboliques dédiés à CTL et non pas à DEL, nous avons trouvé des logiciels permettant d'effectuer du *model checking* sur des structures probabilistes, comme PRISM (*Probabilistic Symbolic Model Checker* [KNP02]), MRMC (*Markov Reward Model checker* [KZH$^+$09]) ou encore STORM ([DJKV17]). Dans tous les cas, la structure de données utilisée est le MTBDD (*Multiterminal Binary Decision Diagrams*)

Néanmoins les problèmes modélisés ici sont typiquement PCTL (*Probabilistic Computation Tree Logic*), des chaînes de Markov discrètes ou continues (*DTMCs* ou *CTMCs*), ou des des processus décisionnels de Markov (*MDPs* ou *PO-MDPs*)

C'est pourquoi, nous avons cherché à implémenter nous-même notre *model checker* dédié à PDEL.

En première approche, nous souhaitions exploiter les SLDDs. Nous avons donc trouvé le compilateur SALAAD [Sch22] écrit par Nicolas Schmidt lors de sa thèse [Sch15]. Celui-ci offre notamment les fonctionnalités suivantes : compilation ascendante de réseau de contraintes vers SLDD+ ou de réseau bayésien vers SLDD$\times$, traduction d'un problème d'un langage VDD (Diagramme de Décision Valué) à un autre et réalisation de requêtes et transformations sur ces VDD.

Malheureusement, le cas d'utilisation est différent du nôtre. SALADD a besoin de VCSP [3] ou réseaux bayésiens (avec traduction possible vers ADD).

Cette implémentation ne répondant pas à nos besoins, et ne voyant pas d'autres solutions, nous avons écrit un programme en Python permettant de manipuler des SLDDs. Malheureusement, le programme n'était pas très efficace. Soupçonnant que les SLDDs sont intrinséquement inadaptés pour notre utilisation, mais ne pouvant pas écarter des problèmes d'implémentation, nous nous sommes tournés vers les

3. Un CSP (Problème de Satisfaction de Contraintes) est un problème modélisé sous la forme d'un ensemble de contraintes posées sur des variables, chacune de ces variables prenant ses valeurs dans un domaine. Lorsqu'un CSP n'a pas de solution, on dit qu'il est surcontraint : il y trop de contraintes et on ne peut pas toutes les satisfaire. Dans ce cas, on peut souhaiter trouver l'affectation totale qui viole le moins de contraintes possibles. Une autre possibilité est d'affecter un poids à chaque contrainte (une valeur proportionnelle à l'importance de cette contrainte, et de chercher l'affectation totale qui minimise la somme des poids des contraintes violées. Un tel CSP est appelé CSP valué (VCSP).

ADDs, plus simples à implémenter et avec un algorithme de marginalisation plus efficace.

Nous avons donc cherché d'autres types de structures de données.

Aussi, nous avons tout d'abord cherché ce qu'il existait comme implémentation possible pour les PBFs, en python. Malheureusement, nous n'en avons trouvé que très peu. La littérature utilise en général CUDD en C et JavaBDD en Java[Wha07]. Il existe néanmoins aussi un *binding* en Python [joh]. Nous avions utilisé ce *binding* Python de CUDD pour nos premières expérimentations, dans le cadre de représentation symbolique de DEL, mais en testant la bibliothèque dans le cas d'utilisation des ADDs, plusieurs fonctionnalités étaient manquantes. Nous noterons aussi l'existence de PyEDA [Dra13] en Python qui permet l'utilisation de BDDs mais pas d'ADDs.

Nous avons alors cherché d'autres implémentations d'ADDs, sans succès.

Par ailleurs, nous avions déjà un programme en python manipulant DEL de manière explicite, les formules booléennes, et des exemples de jeux tel qu'Hanabi. En outre, sous souhaitions être rapide dans l'écriture du programme qui avait simplement pour but initial que de faire des tests, en guise de prototype. C'est pourquoi nous avons conservé le langage Python.

Par ailleurs, il existe une petite bibliothèque : *pyddlib* [Bue17], qui possède une structure embryonnaire de BDDs et d'ADDs. Le programme fonctionne bien ; il est non optimisé, puisqu'il n'utilise pas de cache et il manque beaucoup de méthodes dont nous avions besoin. Nous avons donc faire le choix de modifier ce code et d'en faire un programme beaucoup plus évolué et complet. À titre comparatif, le code de pyddlib faisait 700 lignes et le notre en fait 3600. En effet, nous avons ajouté des classes abstraites, des classes encapsulant les diagrammes de décision pour une manipulation simplifiée, des tests unitaires, et de nombreuses méthodes.

Notre prototype étant écrit en Python, il n'est pas fait pour être efficace. Ce choix de langage a été fait basiquement pour gagner du temps en terme de conception et tests, vis à vis des connaissances du langage.

L'intégralité du programme est présent dans un dépot GITHUB accessible à tous [Gam22] et sera sujet à des modifications, notamment en terme de nettoyage du code et d'accessibilité avec des exemples d'utilisation.

7.2.1 Structure du code

Nous n'allons pas décrire tout le code qui crée et gère les structures épistémiques explicites, symboliques, et ADDs. Néanmoins, nous allons présenter rapidement le fonctionnement et l'imbrication des classes grâce à des diagrammes

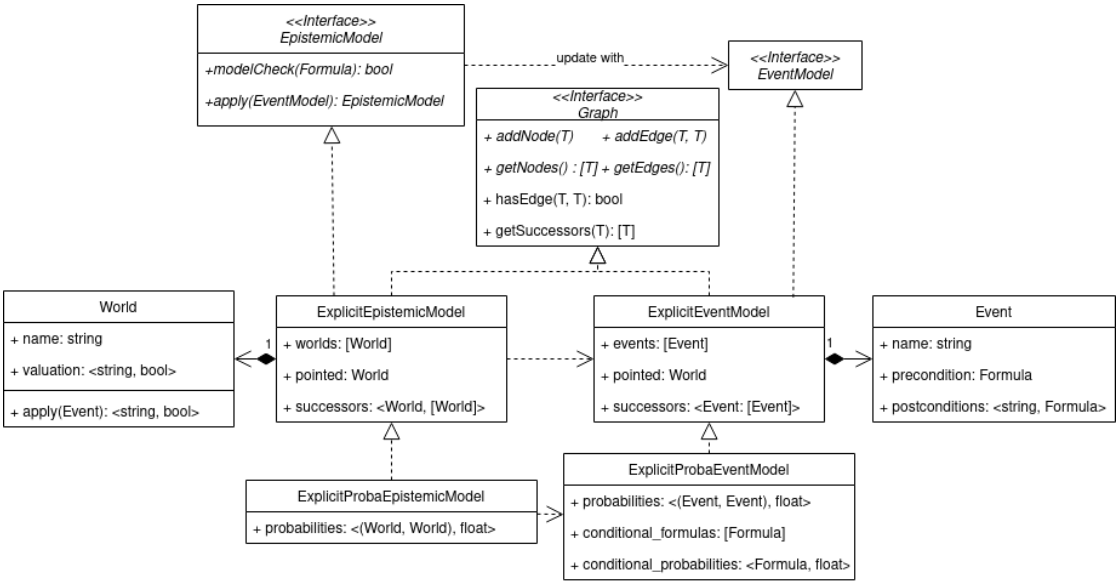


FIGURE 7.2 – UML de l'implémentation des structures épistémiques explicites

UML [4], qui illustrent le programme dans sa globalité, sans rentrer dans les détails. Sur la figure 7.2, nous pouvons voir les classes qui représentent les modèles de DEL (`ExplicitEpistemicModel` et `ExplicitEventModel`) et les modèles de PDEL (`ExplicitProbaEpistemicModel` et `ExplicitProbaEventModel`). Toutes ces structures explicites implémentent l'interface `Graph` de sorte à ajouter des nœuds qui sont soit des mondes pour la structure épistémique (`World`), soit des événements pour la structure d'événements (`Event`).

Sur la figure 7.3, nous voyons le pendant symbolique de l'UML de la figure 7.2. Nous pouvons constater que les `Structures` et les `Transformers` implémentent bien respectivement les interfaces `EpistemicModel` et `EventModel`, comme le faisant leurs équivalents explicites. Nous pouvons noter la présence de tous les éléments nécessaires à la modélisation des structures symboliques définies section 2.2 et chapitres 5 et 6 : la loi des mondes θ, les lois d'observations Ω, les lois de probabilités Π et de même pour les modificateurs : Ω^+, Π^+, θ_-. L'attribut `event_law` de `Transformer` est ici une abstraction de la loi des événements θ^+ d'un modificateur observationnel et de la loi de probabilités conditionnelles θ^{pre} d'un modificateur conditionnel.

Enfin, nous représentons l'UML des structures de données utilisées pour représenter les fonctions booléennes et pseudo-booléennes sur la figure 7.4. L'idée est la

4. Le Langage de Modélisation Unifié, de l'anglais Unified Modeling Language, est un langage de modélisation graphique à base de pictogrammes conçu comme une méthode normalisée de visualisation dans le domaine du développement logiciel et en conception orientée objet.

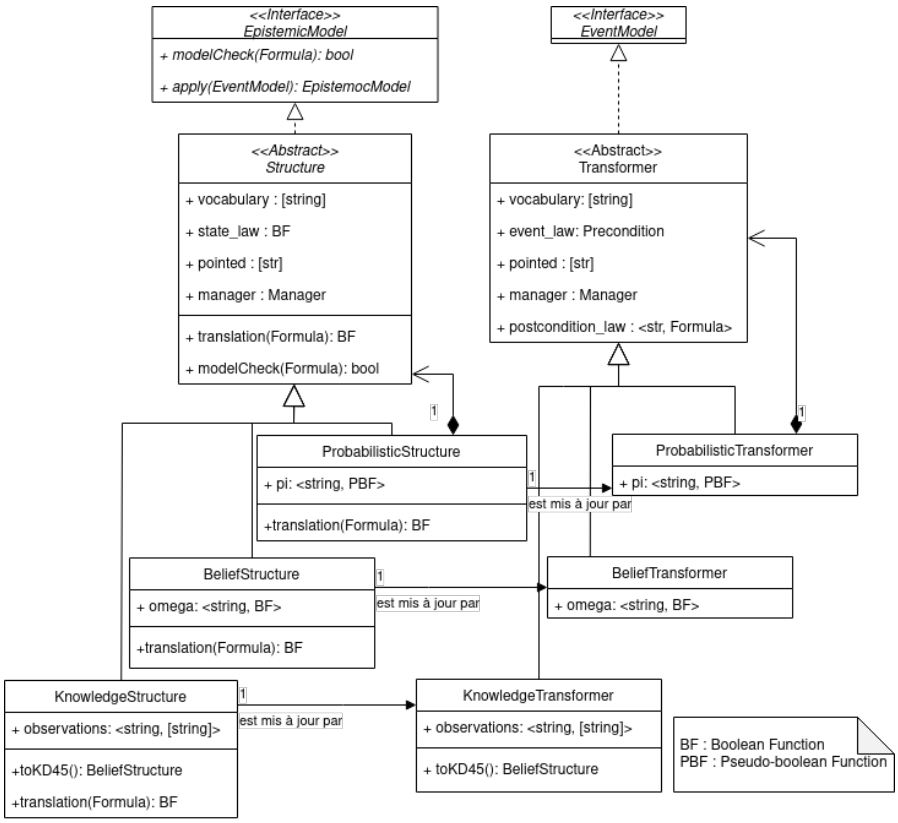


FIGURE 7.3 – UML des structures symboliques

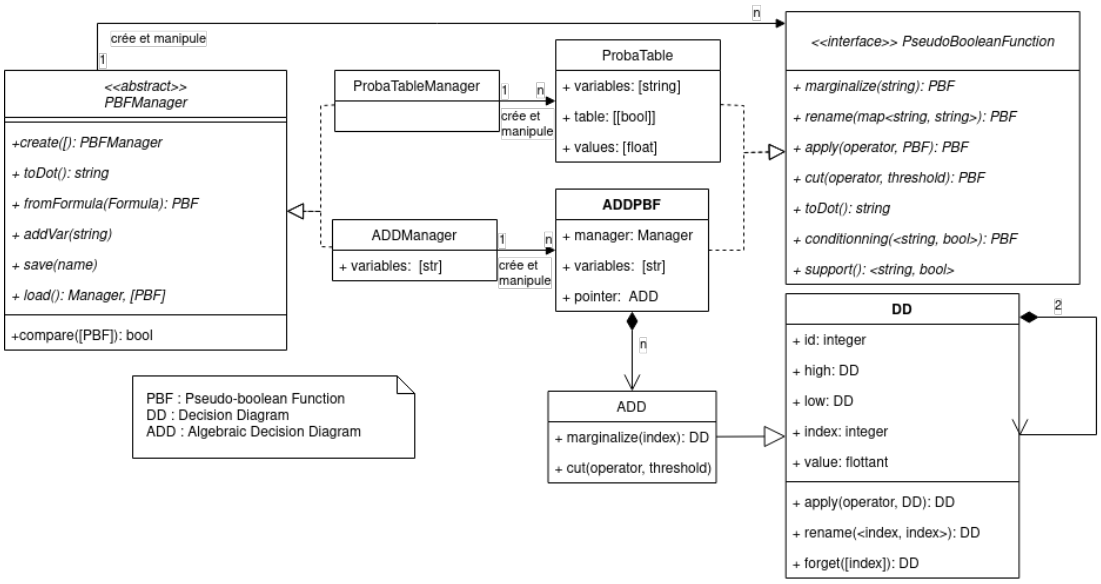


FIGURE 7.4 – UML de l'implémentation des ADDs

suivante :

- La classe `DD` représente un nœud d'un diagramme de décision binaire (Decision Diagram), ou de façon équivalente, le DD enraciné à ce nœud. `ADD` implémente évidemment cette classe. Chaque nœud a alors un identifiant (`id`), deux fils (`high` et `low`), un index (`index`) représentant son niveau dans l'arbre et une valeur (`value`) s'il s'agit d'un nœud terminal.
- La classe `ADDPseudoBooleanFunction` quant à elle est un *wrapper* de la classe `ADD`, elle contient donc un pointeur vers un `ADD`, mais surtout fait la liaison entre les variables « nommées » (noms des atomes propositionnels) par des chaînes de caractères et leurs index (entiers). Cette classe implémente toutes les méthodes fournies par l'interface `PseudoBooleanFunction` de sorte à manipuler les `ADDs` entre eux.
- Enfin, `ADDManager` est comme son nom l'indique un manageur qui gère tous les pointeurs utilisés dans les `ADDPseudoBooleanFunctions` et qui peut donc gérer des forêts d'`ADDs` au lieu d'arbres. Il permet aussi la sauvegarde et la chargement des objets Python (grâce à du JSON [5]).
- Notons la présence de `ProbaTable` et `ProbaTableManager` qui sont une représentation naïve des PBFs via des tables de vérité avec pour valeurs des nombres. Elles permettent de faire des tests unitaires : les algorithmes étant simples sur les `ProbaTable`, on peut comparer leurs résultats avec les résultats des algorithmes, plus complexes, des `ADDPseudoBooleanFunction`. Ces tests unitaires sont néanmoins raisonnablement petits, car les tables de vérité entières et explicites croîssent exponentiellement en taille avec le nombre de variables utilisées : ajouter une variable multiplie la taille de la table par 2.

7.2.2 Optimisations du code

Lors de la mise en pratique de ces programmes Python, nous souhaitions optimiser au mieux le code de sorte à ce que le moins possible de calcul soit faits et que ceux-ci soient plus rapides, bien que ce ne soit pas le but du programme.

Pour réaliser ces petites optimisations, nous avons utilisé différentes astuces.

Premièrement, en Python, l'utilisation du drapeau « -O » à l'exécution d'un programme en ligne de commande permet de désactiver les assertions du programme (typiquement en Python, `assert a == b`). Ainsi, si de nombreuses vérifications se font lors de l'exécution du programme, celles-ci seront simplement

5. JavaScript Object Notation (JSON) est un format de données textuelles dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée comme le permet XML par exemple.

Nombre de cartes	Sans `__slots__`		Avec `__slots__`		Gain total (%)
	Sans -O	Avec -O	Sans -O	Avec -O	
20	88.34	83.26	84.35	78.83	10.77
30	245.77	226.26	229.95	216.18	12.04
40	531.42	494.85	497.96	458.78	13.66
50	980.83	939.96	940.56	897.36	8.51

TABLE 7.1 – Temps de calculs (en secondes) en fonction de l'utilisation du drapeau -O et de `__slots__`, pour deux agents et deux cartes en main dans le jeu Hanabi (avec *model checking* et *product update*).

ignorées. En outre, la constante `__debug__` en python permet de savoir si le drapeau « -O » est désactivé. De cette manière, il est possible d'utiliser la condition « `if not __debug__:` » pour ignorer des pans entiers de programme.

Bien évidemment, il faut s'assurer que le programme est correct avant de désactiver les assertions. Une fois cela vérifié, il est possible de lancer les programmes avec ce drapeau pour augmenter leurs rapidités. Par exemple, dans notre cas, trois assertions ont lieu à chaque instanciation d'un `ADD`, vérifiant les valeurs des arguments et leurs cohérences entre elles. La désactivation de ces vérifications augmente donc légèrement la rapidité du programme.

Secondement, il existe aussi l'attribut de classe `__slots__`. Celle-ci permet de déclarer les attributs d'instance que contient un objet. Ainsi les attributs des instances d'une classe ne sont plus stockés dans un dictionnaire comme le fait nativement Python, mais dans une liste prédéterminée. Cela optimise l'utilisation de la mémoire pour accéder aux attributs, sans compter que cela réduit aussi la taille des objets manipulés.

Nous avons effectué quelques tests pour comparer l'efficacité de ces ajouts. Les résultats de ceux-ci sont présentés dans la table 7.1. Nous n'allons pas décrire les paramètres utilisées, car ils seront pleinement expliqués dans les sections 7.3 puis 8.

Les gains calculés dans la table 7.1 est simplement $(1 - (\mathrm{tps}_a \,/\, \mathrm{tps}_s))$ où tps_s est le temps de calcul sans `__slots__` ni la désactivation des assertions -O et tps_a est le temps de calcul avec `__slots__` et avec -O. Les gains ne sont ainsi pas extraordinaires mais restent non négligeables.

Notre code s'appuie énormément sur la technique de la *mémoïsation* (programmation dynamique.) ; c'est une technique d'optimisation utilisée principalement pour accélérer les programmes en stockant les résultats des appels de fonctions. Lorsqu'un prochain appel de ces fonctions ont lieu, alors il suffit de retourner la valeur anciennement stockée plutôt que refaire les calculs.

Dans le cas des diagrammes comme les BDD ou ADD, il est primordial d'utiliser la mémoïsation. En effet, les algorithmes manipulant les diagrammes sont récursifs et parcourent les nœuds un à un. Ces parcours sont amenés à rencontrer de nombreuses fois les mêmes nœuds car les diagrammes sont réduits et donc des chemins dans les diagrammes sont mutualisés (voir fig. 3.1, page 67 et 3.2a, page 69 sur les BDDs et RBDDs). Grâce à la mémoïsation chaque traitement d'un nœud sera mémorisé et ne sera fait qu'une seule et unique fois, évitant de nombreux traitements inutiles, qui croîtraient exponentiellement en nombre.

En outre, nous en avons profité pour faire un cache amélioré qui permet de mémoriser les résultats des opérations commutatives.

Exemple 7.1. *Soient f et g deux PBFs représentées par des ADDs, la résultat de l'opération $f \times g$ sera stocké en cache, mais aussi $g \times f$, ainsi que tous les résultats des sous-opérations récursives qui ont permis de générer les nœuds internes des ADDs.*

L'utilisation de la mémoïsation est primordiale dans ce type de programme. Grâce à cela, l'impact computationnel des redondances des calculs est grandement diminué : des calculs qui exploseraient exponentiellement deviennent polynomiaux.

Notre implémentation de la structure de données est complètement fonctionnelle, cependant un élément importante manque : le réordonnancement automatique des variables. En effet, il est possible de fixer l'ordre des variables en amont, ce qui est utile lorsque nous connaissons les données manipulées. Malheureusement, s'il n'y a pas une connaissance a priori de l'ordre voulu, un réordonnancement automatique pourrait être utile. Dans notre cadre, nous en avons pas nécesserement besoin. Nous tâcherons par la suite de l'ajouter au programme.

7.2.3 Outils de vérification

Nous avons aussi implémenté d'autres outils. Par exemple, des outils de vérification des modèles et de génération d'expériences sur la jeu Hanabi.

Ainsi, pour vérifier que nos implémentations fonctionnent correctement, nous avons écrit quelques programmes supplémentaires :

- Des tests unitaires sur toutes structures implémentant l'interface `PseudoBooleanFunction` via son *manager* `PseudoBooleanFunctionManager` dédié ;
- Des tests comparatifs entre les résultats de *model checking* faits sur une structure explicite aléatoire et sa transformation symbolique ;
- Des tests sur des exemples jouets, dont des célèbres, de la logique épistémique dynamique :
  - les enfants sales [FHMV95] ;

- l'anniversaire de Cheryl [vDHK+17] ;
- le dîner des cryptographes [vdMS04] ;
- Sally et Anne (adapté de [BCLF85]) ;
- l'exemple d'un lancé de pièce truqué pour PDEL (ex. 4.46).

Ensuite, nous avons évidemment un programme permettant de modéliser le problème d'Hanabi en version explicite et en version symbolique. Nous allons voir dans la prochaine section comment nous avons modélisé ce problème, ce qui nous servira par la suite pour effectuer nos expérimentations pour évaluer le passage à l'échelle des structures symboliques comparé aux structures explicites.

7.3 Modélisation de notre exemple : Hanabi

Rappelons pourquoi nous avons choisi un jeu comme Hanabi pour faire des expérimentations sur sa représentation symbolique :

- ce jeu contient un réel besoin de connaissances épistémiques pour que les agents puissent résoudre le jeu. En effet, pour qu'un joueur sache quelle annonce il doit faire, il se doit de compléter au mieux les connaissances de ses collègues. Pour cela, il doit savoir quelles connaissances possèdent les autres joueurs ;
- étant un jeu de cartes, la notion de probabilité est importante. En effet, avoir des connaissances épistémiques sur les cartes (savoir que l'on possède un 2 rouge) est important, mais savoir que la probabilité que notre première carte en main a 80% de chance d'être de couleur rouge peut être un facteur impactant pour faire des choix. Ainsi, PDEL a une réelle utilité lors de la modélisation de jeux stochastiques ;
- il est possible de fortement paramétrer le jeu : le nombre d'agents, le nombre total de cartes ou encore le nombre de cartes total en main peuvent être modifiés. Il est alors intéressant d'étudier la taille des structures de Kripke représentant le jeu, car tous ces paramètres augmentent de manière combinatoire le nombre de mondes possibles dans ces structures. Ainsi, en manipulant tous ces paramètres, nous allons pouvoir tester le comportement de nos structures de données et voir à quel point nos représentations symboliques (et leurs représentations en mémoire) pourront s'adapter à des problèmes réels aux tailles conséquentes.

Hanabi requiert donc de nombreuses caractéristiques positives. Malheureusement, cet exemple n'est pas parfait pour l'exploiter l'expressivité de PDEL. En effet, Hanabi n'est pas un jeu qui nécessite la modélisation de probabilités conditionnelles. Notre représentation de Hanabi symbolique est donc basée sur notre

définition de PDEL utilisant des modificateurs observationnels et des modèles observationnels dans leur version explicite. Ainsi, cet exemple ne nous permettra pas véritablement de faire des expérimentations sur des modificateurs conditionnels, chose qui aurait été attendue en représentant le *framework* de PDEL. Il y a trois explications à cela : premièrement, nous avions basiquement l'idée de faire nos expérimentations sur le jeu Hanabi, car il est actuellement étudié dans le domaine [BFC+20] et qu'il présente vraiment les propriétés de mise à l'échelle que nous souhaitions lors de nos expériences. Deuxièmement, nous avons montré que l'utilisation d'un modèle conditionnel peut être réduite à l'utilisation d'un modèle observationnel (sec. 4.5), il faudrait alors tout simplement ajouter une étape de précompilation les probabilités conditionnelles et probabilités d'observations dans des probabilités d'observations (voir déf. de cond 4.48). Troisièmement, nous n'avons pas trouvé de cas d'application du *framework* initial de PDEL dans un cas réaliste et naturel qui nous semblait intéressant, d'autant plus que nous souhaitions avoir un exemple où il est vraiment intéressant de raisonner sur des formules épistémiques probabilistes.

Notons aussi que notre exemple est spécifique dans PDEL : les probabilités sont a-consistantes, a-uniformes et uniformes (voir discussion sec. 6.4.2). Dans des cas réels d'application, à notre connaissance, il nous paraît peu intéressant de vouloir exploiter un exemple non a-consistant. Pour ce qui est de la a-uniformité, elle nous semble simplement refléter de nombreuses lois naturelles des probabilités et représente de nombreux cas d'utilisation. Typiquement, les probabilités peuvent ici être entièrement déduites des informations épistémiques de l'agent[vDHvdHK15]. Bien entendu, des cas applicatifs peuvent néanmoins être non a-uniforme et il serait aussi intéressant d'étudier le comportement de notre *framework*.

7.3.1 Les paramètres du jeu

Nous allons maintenant décrire comment nous avons implémenté le jeu Hanabi. Tout d'abord, voici quelques constantes du jeu essentielles à la construction des atomes propositionnels qui nous serviront à décrire les états physiques du jeu :

- Premièrement, il nous faut un ensemble d'agents (ou joueurs), que nous notons toujours $\mathfrak{A}$. Nous définissons $\mathfrak{A}$ comme étant l'ensemble $\{a_1, \ldots, a_n\}$ avec $n \in \{2, \ldots, 5\}$, car le nombre de joueurs est entre 2 et 5 et nous notons leur nombre $\mathsf{nbA} = |\mathfrak{A}|$.

Ensuite, trois valeurs seront importantes pour modéliser une carte dans Hanabi (et nous suivons les règles du jeu afin de les modéliser) : sa couleur, sa valeur et son identifiant définis comme suit :

- soient les couleurs des cartes $\{R, W, B, Y, G\}$ (pour rouge, blanc, bleu, jaune et vert). Nous noterons une couleur quelconque par la lettre $c \in$

$\{R, W, B, Y, G\}$;

- soient les valeurs des cartes, comprises entre 1 et 5 : $\{1, \ldots, 5\}$. Nous noterons une valeur quelconque par $v \in \{1, \ldots, 5\}$;
- soient les identifiants des cartes ids_v qui dépendent de la valeur des cartes v :

$$\mathsf{ids}_v = \begin{cases} \{1,2,3\} & \text{si } v = 1; \\ \{1,2\} & \text{si } v \in \{2,3,4\}; \\ \{1\} & \text{si } v = 5. \end{cases}$$

C'est à dire que pour chaque couleur, il y a 3 cartes de valeur 1, 2 cartes de valeur 2, 3 et 4, et 1 seule carte de valeur 5. Ainsi, il est nécessaire d'avoir trois identifiants pour les cartes de valeur 1, mais un seul pour les cartes de valeur 5.

Pour pouvoir manipuler simplement les ensembles $\{R, W, B, Y, G\}$, $\{1, \ldots, 5\}$ et ids_v, nous définissons l'ensemble de triplets noté Cards qui représente l'ensembles de toutes les cartes possibles :

- $\mathsf{Cards} = \{(c, v, i) \mid c \in \{R, W, B, Y, G\},\ v \in \{1, \ldots, 5\},\ i \in \mathsf{ids}_v\}$.

Par ailleurs, il nous faut définir les différents « possesseurs » des cartes.

- Soit $\{P, T, D\}$ l'ensemble des possesseurs potentiels des cartes qui ne sont pas des agents, c'est à dire respectivement la pioche, la table, ou la défausse.

Nous allons aussi utiliser deux autres variables qui vont nous permettre de paramétrer le jeu :

- nbC : le nombre de cartes total en jeu : $\mathsf{nbC} \in \{0, \ldots, 50\}$;
- nbH : le nombre total de cartes en main : $\mathsf{nbH} \in \{1, 2, 3, 4, 5\}$. Nous notons pos la liste des positions possibles des cartes dans les mains des joueurs : $\mathsf{pos} = \{1, \ldots, \mathsf{nbH} - 1\}$

Ces deux variables fixées en entrée du problème modifient l'ensemble des cartes possibles du jeu et les variables propositionnelles pour modéliser le jeu. En effet, pour moduler la difficulté du jeu, nous modifions nbC, et nous conservons donc les nbC premières cartes de la liste Cards. Par exemple, avec $\mathsf{nbC} = 10$, il n'y aura que 10 cartes en jeu, qui seront toutes rouges. Pour ce qui est d'avoir toutes les variables pour modéliser le jeu complet selon les règles d'Hanabi, il faut $\mathsf{nbC} = 50$ dans tous les cas (5 couleurs et 10 cartes par couleur), avec $\mathsf{nbH} = 5$ s'il y a 2 ou 3 joueurs et $\mathsf{nbH} = 4$ s'il y a 4 ou 5 joueurs.

Grâce à ces ensembles d'éléments, nous allons construire des variables propositionnelles qui vont décrire les valuations des mondes des structures de Kripke qui représenteront les connaissances épistémiques des agents :

Nous notons la variable propositionnelle $\mathsf{cardpos}^z_{(c,v,i)}$ avec $z \in (\mathfrak{A} \times \mathsf{pos}) \cup \{P, T, D\}$ et $(c, v, i) \in \mathsf{Cards}$, la variable représentant le fait que la carte identifiée

par (c, v, i) est dans la main de l'agent a en position p ($z = (a, p) \in (\mathfrak{A} \times \mathsf{pos})$) ou dans la pile de cartes z ($z \in \{P, T, D\}$).

Avec ces atomes, nous identifions deux ensembles, noté avec les symboles CP signifiant « cartes possibles » :

- $\mathsf{CP}_{(c,v,i)} = \{\mathsf{cardpos}^{(a,p)}_{(c,v,i)} \mid a \in \mathfrak{A},\ p \in \mathsf{pos}\}$, qui représente tous les atomes décrivant les cartes de couleur c, valeur v et id i ;
- $\mathsf{CP}^z = \{\mathsf{cardpos}^z_{(c,v,i)} \mid (c, v, i) \in \mathsf{Cards}\}$, qui représente tous les atomes décrivant les cartes pour un possesseur z. Ainsi $\mathsf{CP}^{(a,p)}$ représentent tous les atomes décrivant les cartes d'un agent a en position p et CP^o les atomes pour la pile $o \in \{P, T, D\}$.

Nous noterons simplement l'ensemble de tous ces atomes CP de sorte que $\mathsf{CP} = \{\mathsf{cardpos}^z_{(c,v,i)} \mid (c, v, i) \in \mathsf{Cards},\ z \in (\mathfrak{A} \times \mathsf{pos}) \cup \{P, T, D\}\}$.

Exemple 7.2. $\mathsf{CP}^{(a_0,0)}$ *identifiera donc tous les atomes propositionnels des cartes de l'agent* a_0 *en position 0 et* $\mathsf{CP}_{(R,1,0)}$ *tous les atomes identifiant la première carte rouge de valeur 1.*

Par ailleurs, nous ajoutons d'autres variables qui sont des compteurs pour la modélisation du nombre de jetons bleus et rouges dans le jeu, ainsi que pour représenter les tours des joueurs :

- Soit $\mathsf{B} = \{b_0,\ b_1,\ b_2,\ b_3\}$ l'ensemble des variables propositionnelles représentant les bits du compteur binaire pour les 8 jetons bleus (4 variables) ;
- soit $\mathsf{R} = \{r_0,\ r_1\}$ l'ensemble des variables propositionnelles représentant les bits du compteur binaire pour les 3 jetons rouges (2 variables) ;
- soit $\mathsf{Turns} = \{\mathsf{turn}_a \mid a \in \mathfrak{A}\}$ l'ensemble des variables propositionnelles représentant quel agent a doit jouer.

Nous pouvons voir d'ores et déjà que le paramètre nbC pourra être modulé pour apprécier le caractère combinatoire du jeu. Néanmoins, nbA et nbH auront aussi un fort impact sur le nombre total de variables pour modéliser le jeu.

7.3.2 Formules logiques pour la structure de Kripke

Pour représenter Hanabi de manière symbolique, nous allons donc définir la structure de probabilités $\mathfrak{F} = \langle V, \theta, \Omega, \Pi \rangle$. Comme nous l'avons vu, nous avons déjà notre ensemble de variables propositionnelles permettant de décrire Hanabi. Cet ensemble de variables forme notre vocabulaire V qui est alors simplement l'union de toutes nos variables propositionnelles : $V = \mathsf{CP} \cup \mathsf{Turns} \cup \mathsf{B} \cup \mathsf{R}$.

Afin d'écrire la loi des mondes θ, nous allons avoir besoin de la définition de quelques fonctions. Commençons par la construction récursive d'une formule

booléenne qui pour un ensemble de variables X donné est vraie quand exactement n des variables de X sont vraies.

Définition 7.3 (Parmi booléen)**.** *Soit X un ensemble de variables propositionnelles. La fonction booléenne* parmi $: N \times X \to \mathcal{L}_{\text{prop}}$*, avec $N \in \{0, \ldots, \mid X \mid\}$ est définie récursivement comme suit :*

- $\mathsf{parmi}(0, X) = \bigwedge_{x \in X} \neg x$
- *pour* $n \geq 1$, $\mathsf{parmi}(n+1, X) = \bigvee_{x \in X} (x \wedge \mathsf{parmi}(n, X \backslash \{x\}))$

Notons ici que la formule $\mathsf{parmi}(n, X)$ est de taille $|X|^n$.

En pratique, nous avons implémenté cette fonction de cette manière, en considérant X comme une liste triée :

- $\mathsf{parmi}(n, X) = (X[0] \wedge \mathsf{parmi}(n-1, X[1:])) \vee (X[0] \wedge \mathsf{parmi}(n, X[1:]))$

Cette fonction permet d'exploiter la programmation dynamique mais surtout de créer les niveaux du diagramme de décision un à un efficacement en considérant les variables dans l'ordre déclaré dans le diagramme. Grâce cette définition, nous pouvons construire de nouvelles formules qui seront des règles logiques permettant de modéliser les contraintes entres les variables propositionnelles, qui nous permettront de constituer la formule θ. En effet, l'idée est simple : nous ne voulons qu'il n'y ait qu'un seul atome vrai pour un couple (c, v, i), i.e. une carte n'est présente qu'une seule fois dans le jeu.

Définissons quelques formules qui constitueront θ :

- **Unicité des cartes :** il doit y avoir une seule carte pour chaque tuple (c, v, i) :
$$U = \bigwedge_{(c,v,i) \in \mathsf{Cards}} \mathsf{parmi}(1, \mathsf{CP}_{(c,v,i)})$$
- **Position des cartes :** il doit y avoir une seule carte à chaque position dans la main d'un joueur :
$$P = \bigwedge_{a \in \mathfrak{A}} \bigwedge_{p \in \mathsf{pos}} \mathsf{parmi}(1, \mathsf{CP}^{(a,p)})$$
- **Tours des joueurs :** un seul joueur peut jouer à la fois :
$$T = \mathsf{parmi}(1, \mathsf{Turns})$$

Nous pouvons ainsi définir la loi des mondes ainsi : $\theta = U \wedge P \wedge T$. Chaque modèle de cette formule représente les ensembles de variables propositionnelles qui représentent des états physiques légaux dans le jeu Hanabi. Nous pouvons maintenant construire les fonctions booléennes permettant de représenter les relations

symboliques entre les mondes : Ω. Comme Gattinger le précise dans sa thèse, il est possible de représenter cette fonction par des équivalences entre tous les atomes propositionnels que chaque agent observe. Ainsi, nous avons pour chaque agent $a \in \mathfrak{A}$ la relation d'équivalence Ω_a définie comme ci-dessous.

Soit Obs_a les variables qu'observe l'agent a. a observe donc toutes les variables propositionnelles représentant les cartes qu'il ne possède pas et qui ne sont pas dans la pioche (cf. les règles du jeu). Par ailleurs, il observe l'état des jetons et des tours des joueurs. Ainsi :

$$\mathsf{Obs}_a \coloneqq \mathsf{CP} \setminus \left(\left(\bigcup_{p \in \mathsf{pos}} \mathsf{CP}^{(a,p)} \right) \cup \mathsf{CP}^P \right) \cup \mathsf{Turns} \cup \mathsf{B} \cup \mathsf{R}$$

À partir de cet ensemble de variables Obs_a, nous pouvons construire pour chaque agent a la formule booléenne Ω_a qui est simplement définie ainsi :

$$\Omega_a \coloneqq \bigwedge_{v \in \mathsf{Obs}_a} v \leftrightarrow v'$$

Enfin, comme nous l'avons expliqué dans la section 6.4.2, nous pouvons directement obtenir Π_a à partir de Ω_a $\Pi_a = \Omega_a$.

Définition 7.4 (Structure de probabilités d'Hanabi)**.** *La structure de probabilités $\mathfrak{F} = \langle V, \theta, \Omega \rangle$ représentant les connaissances des agents au jeu Hanabi est définie comme suit :*

- $V = \mathsf{CP} \cup \mathsf{Turns} \cup \mathsf{B} \cup \mathsf{R}$ *;*
- $\theta = U \wedge P \wedge T$ *avec :*
  - $U = \bigwedge_{(c,v,i) \in \mathsf{Cards}} \mathsf{parmi}(1, \mathsf{CP}_{(c,v,i)})$ *;*
  - $P = \bigwedge_{a \in \mathfrak{A}} \bigwedge_{p \in \mathsf{pos}} \mathsf{parmi}(1, \mathsf{CP}^{(a,p)})$ *;*
  - $T = \mathsf{parmi}(1, \mathsf{Turns})$ *;*
- $\Omega_a = \bigwedge_{v \in \mathsf{Obs}_a} v \leftrightarrow v'$ *avec* Obs_a *défini comme ci-dessus ;*
- $\Pi_a = \Omega_a$.

Maintenant que nous avons une représentation symbolique des connaissances des agents dans le jeu Hanabi, nous allons définir les modificateurs observationnels permettant de modifier et mettre à jour les connaissances des agents.

7.3.3 Formules logiques pour les modèles observationnels

Les modèles d'événements utilisés pour modéliser Hanabi sont des modèles observationnels avec postconditions, comme présenté dans la section 4.3. Commençons par définir quelques opérations qui nous permettront de définir les actions du jeu Hanabi.

Définition 7.5. *La formule booléenne* $\mathsf{ite}(\phi,\ \psi_1,\ \psi_2)$ *(ou If-then-else) est définie comme suit :*

$$\mathsf{ite}(\phi,\ \psi_1,\ \psi_2) = (\phi \wedge \psi_1) \vee (\neg\phi \wedge \psi_2)$$

La formule booléenne $\Phi = \mathsf{ite}(\phi,\ \psi_1,\ \psi_2)$ est un simple *ou logique*. Si ϕ est vrai, alors ψ_1 doit être vrai, sinon Φ est faux. Sinon (si ϕ est faux), alors ψ_2 doit être vrai, sinon Φ est faux. Définissons maintenant l'opération d'incrémentation (incr) ou de décrémentation (decr) d'un compteur binaire représenté par un ensemble de n variables (ou bits, ici) $\{b_i \mid i \in \{0..n-1\}\}$. Le bit de poids faible est b_0 et le bit de poids fort b_{n-1}. Ces opérations vont nous permettre de pouvoir manipuler nos compteurs binaires représentant les nombres de jetons bleus (B) et rouges (R). Ces deux opérations vont retourner la loi de modifications θ_- – qui associe à chaque variable d'un ensemble donnée en paramètre une formule booléenne. Commençons par la décrémentation.

Définition 7.6. *Soit* $\mathsf{decr}(Bits)$ *la fonction de décrémentation d'un ensemble de variables propositionnelles* $Bits = \{b_0, \ldots, b_n\}$ *avec* $n \in \mathbb{N}$ *qui retourne la loi de modification* θ_- *:*

- $\theta_-(b_i) = \mathsf{ite}(\wedge_{j\in\{0,\ldots,i-1\}} b_j,\ \neg b_i,\ b_i)$.

Lors de la décrémentation d'un compteur binaire, l'idée est simple : le bit de poids faible est inversé (cela revient à $\theta_-(b_0) = \neg b_0$). Ensuite chaque bit b_i est inversé si et seulement si tous les bits précédents étaient faux, sinon il garde sa valeur. Par exemple, l'affectation $\{b_3 : 1,\ b_2 : 0,\ b_1 : 1,\ b_0 : 0\}$ 1010 (10 en décimal) deviendra $\{b_3 : 1,\ b_2 : 0,\ b_1 : 0,\ b_0 : 1\}$ (9) car b_0 est inversé (il passe de 0 à 1) et b_1 est inversé aussi puisque $\neg b_0$. Pour ce qui est de b_2 et b_3 ils ne changent pas car b_1 est vrai.

Qu'en est-il si l'on souhaite décrémenter un compteur égal à 0 ? Le compteur se vera assigné sa valeur maximal, soit $2^n - 1$. Ici, dans les exemples, nous avons 4 bits, ainsi le maximum sera de 15. À l'utilisation de cette fonction, il faudra donc s'assurer que le compteur ne soit pas nul.

Le fonctionnement de la fonction d'incrémentation est très similaire à celle de la décrémentation.

Définition 7.7. *Soit* $\mathsf{incr}(Bits)$ *la fonction de décrémentation d'un ensemble de variables propositionnelles* $Bits = \{b_0, \ldots, b_n\}$ *avec* $n \in \mathbb{N}$ *qui retourne la loi de modification* θ_- *:*

- $\theta_-(b_i) = \mathsf{ite}(\wedge_{j\in\{0,\dots,i-1\}}\neg b_j,\ \neg b_i,\ b_i)$.

Lors de l'incrémentation d'un compteur binaire, le bit de poids faible est aussi inversé. Par contre, chaque bit b_i est inversé si et seulement si tous les bits précédent sont vrais. Par exemple, l'affectation $\{b_3\colon 1,\ b_2\colon 0,\ b_1\colon 1,\ b_0\colon 0\}$ deviendra $\{b_3\colon 1,\ b_2\colon 0,\ b_1\colon 1,\ b_0\colon 1\}$ (11) car seul le bit de poids faible change ; et comme pour tous les autres bits, b_0 est faux, ils ne seront pas modifiés. Néanmoins, si on prend l'exemple de $\{b_3\colon 0,\ b_2\colon 1,\ b_1\colon 1,\ b_0\colon 1\}$ (7), alors tous les bits seront inversés pour arriver à $\{b_3\colon 1,\ b_2\colon 0,\ b_1\colon 0,\ b_0\colon 0\}$ (8).

Pour ce qui est du cas limite : incrémenter un compteur dont chaque bit est de valeur 1 affectera la valeur 0 au compteur. Il faudra donc s'assurer de ne jamais dépasser la valeur maximale du compteur.

Nous allons maintenant définir une opération de changement des tours des joueurs.

Définition 7.8. *Considérons la liste* Turns *comme étant ordonnée dans l'ordre des actions des joueurs.*

La fonction change_turn *retourne la loi de modification θ_- définie pour pour* turn_{a_i}, $a_i \in \mathfrak{A}$ *:*

- $\theta_-(\mathsf{turn}_{a_i}) = \mathsf{turn}_{a_{i-1}\ mod\ \mathsf{nbA}}$.

L'idée derrière cette opération est que la variable propositionnelle turn_{a_i} devient vraie si et seulement si c'était le tour du joueur précédent, et que donc, dans la liste ordonnée des tours Turns , la variable qui précède turn_{a_i} ($\mathsf{turn}_{a_{i-1} \bmod \mathsf{nbA}}$) est vraie. Le modulo est là simplement pour gérer le cas où $a_i = 0$, de sorte que $(a_i - 1) \bmod \mathsf{nbA} = \mathsf{nbA} - 1$, l'indice du dernier élément dans la liste, qui est la variable propositionnelle du dernier joueur.

Maintenant que nous avons ces trois opérateurs incr, decr et change_turn, nous allons pouvoir définir les modèles d'observations symboliques dans le jeu Hanabi.

Une annonce, comme nous l'avons vu dans la définition 2.46, se modélise symboliquement par une formule booléenne ϕ. Un annonce dans Hanabi consistant à annoncer une couleur ou une valeur sur une liste des positions dans la main d'un joueur, on définit l'opération annonceC qui prend en paramètre une couleur donnée c, un joueur donné a et une liste de positions $L \subseteq \mathsf{pos}$ et qui retournera la formule booléenne associée à cette annonce.

Définition 7.9. *Pour une couleur $c \in \{R, W, B, Y, G\}$, $a \in \mathfrak{A}$, et une liste de positions $L \subseteq \mathsf{pos}$, $\mathsf{annonceC}(c, a, L)$ est la précondition de l'annonce de la couleur c pour l'agent a de telle sorte que :*

$$\mathsf{annonceC}(c, a, L) = \bigwedge_{p\in L} \mathsf{parmi}(1,\ \mathsf{CP}_c^{(a,p)}) \wedge \bigwedge_{p\in \mathsf{pos}\setminus L} \mathsf{parmi}(0,\ \mathsf{CP}_c^{(a,p)})$$

où

$$\mathsf{CP}_c^{(a,p)} = \{\mathsf{cardpos}_{(c,v,i)}^{(a,p)} \mid v \in \{1,\ldots,5\},\ i \in \mathsf{ids}_v\}$$

De la même manière, on définit $\mathsf{annonceV}$ pour une valeur donnée v, un joueur donné a et une liste de positions $L \in \mathsf{pos}$.

Définition 7.10. *Pour une valeur $v \in \{1,\ldots,5\}$, $a \in \mathfrak{A}$, et une liste de positions $L \subseteq \mathsf{pos}$, $\mathsf{annonceV}(v,a,L)$ est la précondition de l'annonce d'une valeur v pour l'agent a de telle sorte que :*

$$\mathsf{annonceV}(v,a,L) = \bigwedge_{p \in L} \mathsf{parmi}(1,\ \mathsf{CP}_v^{(a,p)}) \wedge \bigwedge_{p \in \mathsf{pos}\setminus L} \mathsf{parmi}(0,\ \mathsf{CP}_v^{(a,p)})$$

où

$$\mathsf{CP}_v^{(a,p)} = \{\mathsf{cardpos}_{(c,v,i)}^{(a,p)} \mid c \in \{R,W,B,Y,G\},\ i \in \mathsf{ids}_v\}$$

Pour ces actions d'annonces où nous noterons simplement ϕ la formule annoncée, nous pouvons définir le modificateur observationnel correspondant.

Définition 7.11 (Le modificateur observationnel pour une annonce dans le jeu Hanabi)**.** *La structure de probabilités $\langle V^+, \theta^+, \Omega^+, \Pi^+, V_-, \theta_-\rangle$ pour l'action d'annonce est défini comme suit :*

- $V^+ = \{\}$;
- $\theta^+ = \phi$;
- $\Omega_a^+ = \top$;
- $\Pi_a^+ = \top$;
- $V_- = \mathsf{B} \cup \mathsf{Turns}$;
- $\theta_- = \mathsf{decr}(\mathsf{B}) \cup \mathsf{change\_turn}()$.

Une fois les actions d'annonce modélisée, il nous reste les actions qui permettent de jouer ou de défausser une carte.

Pour jouer une carte, il faut vérifier qu'une carte est jouable sur la table. Pour cela, nous définissons la fonction $\mathsf{possible}$qui utilise la couleur c et la valeur v de la carte que l'on souhaite jouer. Il est alors possible de vérifier s'il y a la carte de même couleur et de valeur inférieure de posée sur la table.

Définition 7.12. *Étant donné une couleur $c \in \{R,W,B,Y,G\}$ et une valeur $v \in \{1,\ldots,5\}$, $\mathsf{possible}(c,v)$ est une formule booléenne définie comme suit :*

- $\mathsf{possible}(c,1) = \neg \bigvee_{i \in \mathsf{ids}_v} \mathsf{cardpos}_{c,1,i}^T$;

- $\mathsf{possible}(c,v) = \bigvee_{i \in \mathsf{ids}_v} \mathsf{cardpos}^T_{(c,v-1,i)} \wedge \neg \bigvee_{i \in \mathsf{ids}_v} \mathsf{cardpos}^T_{c,v,i}$

C'est à dire que si l'on cherche à vérifier s'il est possible de jouer un 1, il suffit de vérifier qu'il n'y a pas de 1 de la même couleur sur la table. Si l'on cherche à vérifier qu'il est possible de poser une carte d'une autre valeur, il faut vérifier qu'il n'y a pas la valeur strictement inférieure sur la table (ici, au moins une, mais par construction du jeu, il ne peut y en avoir qu'une), et qu'il n'y a pas une carte de même valeur et même couleur sur la table.

Nous voulons désormais modéliser l'action « l'agent a joue sa carte en position p, de couleur c, de valeur v et d'ids_v » : $\mathsf{joue}(a,p,c,v,i)$, grâce à formule booléenne $\mathsf{possible}(c,v)$.

Définition 7.13 (Le modificateur observationnel pour l'action « jouer » à Hanabi)**.** *Le modificateur observationnel $\langle V^+, \theta^+, \Omega^+, \Pi^+, V_-, \theta_- \rangle$ pour l'action « jouer » est défini comme suit, pour des a, p, c, v, i donnés :*

- $V^+ = \{\}$;
- $\theta^+ = \mathsf{turn}_a \wedge \mathsf{cardpos}^{(a,p)}_{(c,v,i)}$;
- $\Omega^+_a = \top$;
- $\Pi^+_a = 1$;
- $V_- = \mathsf{CP}^T_{(c,v,i)} \cup \mathsf{CP}^D_{(c,v,i)} \cup \{\mathsf{cardpos}^{(a,q)}_{(d,w,j)} \mid q \in \{0,\ldots,p\},\ (d,w,j) \in \mathsf{Cards}\} \cup \mathsf{R}$;
- *et avec θ_- défini de sorte que :*
  - $\theta_-(\mathsf{cardpos}^D_{(c,v,i)}) = \neg\, \mathsf{possible}(c,v)$;
  - $\theta_-(\mathsf{cardpos}^T_{(c,v,i)}) = \mathsf{possible}(c,v)$;
  - $\forall r \in \mathsf{R},\ \theta_-(r) = \mathsf{ite}(\mathsf{possible}(c,v),\ r,\ \mathsf{decr}(\mathsf{R})(r))$;
  - $\theta_-(\mathsf{cardpos}^{(a,0)}_{(c,v,i)}) = \bot$;
  - $\theta_-(\mathsf{cardpos}^{(a,p)}_{(c,v,i)}) = \bot$;
  - $\forall (d,w,j) \in \mathsf{Cards}, q \in \{1,\ldots,p\},\ \theta_-(\mathsf{cardpos}^{(a,q)}_{(d,w,j)}) = \mathsf{cardpos}^{(a,q-1)}_{(d,w,j)}$.

L'ensemble $\{\mathsf{cardpos}^{(a,q)}_{(d,w,j)} \mid q \in \{0,\ldots,p\},\ (d,w,j) \in \mathsf{Cards}\}$ calculé dans V_- est l'ensemble des atomes propositionnels amenés à changer dans la main de l'agent a, i.e. tous les atomes représentant des cartes ayant une position inférieure ou égal à la position p.

Les postconditions, présentées dans θ_- sont ainsi formulées : s'il est possible de jouer la carte est posée sur la table : $\mathsf{cardpos}^T_{(c,v,i)}$ devient vrai ; sinon, c'est $\mathsf{cardpos}^D_{(c,v,i)}$ qui devient vrai.

La modification des variables propositionnelles $\mathsf{cardpos}^{(a,0)}_{(c,v,i)}$ et $\mathsf{cardpos}^{(a,q)}_{(d,w,j)}$ permettent de remettre en place les cartes dans la main de l'agent de sorte à ce que ce soit toujours l'emplacement en position 0 dans sa main qui soit vacant. Voici la fonctionnement : la carte en position 0 obtient la valeur faux. Les autres cartes, de position 1 à p, prendront obtiennent la valeur de leur atome homologue à la position précédente.

Ce tri dans la main du joueur permettra par la suite de pouvoir piocher une carte en position 0. Cette modélisation du replacement des cartes est totalement arbitraire et pourrait être faite autrement. Néanmoins, cette méthode a le mérite de ressembler à ce que les joueurs expériementés d'Hanabi font lors de parties réelles.

Pour ce qui est de l'action « l'agent a défausse la carte en position p de couleur c, de valeur v et d'id i »($\mathsf{defausse}(a,p,c,v,i)$), elle ressemble quant à elle un peu à l'action joue. Seulement cette fois-ci, il n'est pas nécéssaire de distinguer si l'action est possible ou non.

Définition 7.14 (Le modificateur observationnel pour l'action « défausser » à Hanabi)**.** $\langle V^\dagger, \theta^\dagger, \Omega^\dagger, \Pi^\dagger, V_-, \theta_- \rangle$ *pour la défausse est donc définie par :*

- $V^\dagger = \{\}$;
- $\theta^\dagger = \mathsf{turn}_a \wedge \mathsf{cardpos}^{(a,p)}_{(c,v,i)}$;
- $\Omega^\dagger_a = \top$;
- $\Pi^\dagger_a = 1$;
- $V_- = \mathsf{CP}^D_{(c,v,i)} \cup \{\mathsf{cardpos}^{(a,q)}_{(d,w,j)} \mid q \in \{0,\dots,p\},\ (d,w,j) \in \mathsf{Cards}\} \cup \mathsf{B}$;
- *et avec* θ_- *défini de sorte que :*
  - $\theta_-(\mathsf{cardpos}^D_{(c,v,i)}) = \top$;
  - $\forall b \in \mathsf{B},\ \theta_-(b) = \mathsf{ite}(\neg b_3,\ \mathsf{incr}(\mathsf{B})(b),\ b)$;
  - $\theta_-(\mathsf{cardpos}^{(a,0)}_{(c,v,i)}) = \bot$;
  - $\theta_-(\mathsf{cardpos}^{(a,p)}_{(c,v,i)}) = \bot$;
  - $\forall (d,w,j) \in \mathsf{Cards}, q \in \{1,\dots,p\},\ \theta_-(\mathsf{cardpos}^{(a,q)}_{(d,w,j)}) = \mathsf{cardpos}^{(a,q-1)}_{(d,w,j)}$.

Le fonctionnement de replacement des cartes est identique à celui de l'action « jouer ». L'incrémentation du compteur de jetons bleus s'effectue seulement s'il n'y a pas plus de 7 jetons ($\neg b_3$).

L'action la plus complexe est celle de la pioche. En effet, c'est dans cette structure d'événement que réside tout l'interêt de l'utilisation de la logique épistémique : l'agent qui pioche une carte ne doit ici pas savoir quelle carte il pioche. C'est pourquoi il faut pouvoir distinguer différents sous-événements (ce que nous appelions les événements atomiques des événements de DEL).

Considérons la précondition que nous voulions après avoir déplacé les cartes dans la main du joueur : il n'y a pas de carte en position 0. Vérifions ainsi que l'agent a ne possède pas cette carte, par la négation de cette formule :

$$\mathsf{peut\_piocher}(a) = \neg(\bigvee_{(c,v,i)\in\mathsf{Cards}} \mathsf{cardpos}^{(a,0)}_{(c,v,i)})$$

Soit l'action : « l'agent a pioche la carte de couleur c, de valeur v et d'id i » : $\mathsf{pioche}(a,c,v,i)$. Pour pouvoir modéliser cette action, nous aurons besoin de variables que l'on notera $\mathsf{pioche}^a_{(c,v,i)}$ et qui appartiendront au vocabulaire V^+. Nous pouvons donc construire θ^+, la loi des événements ainsi :

$\theta^+ = \neg\,\mathsf{peut\_piocher}(a) \wedge$

$$\bigvee_{(c,v,i)\in\mathsf{Cards}} (\mathsf{cardpos}^P_{(c,v,i)} \wedge \mathsf{pioche}^a_{(c,v,i)} \wedge \bigwedge_{(d,w,j)\in\mathsf{Cards}\setminus(c,v,i)} \neg\mathsf{pioche}^a_{(d,w,j)})$$

Par cette formule, nous nous assurons qu'un seul et unique événement atomique $\mathsf{pioche}^a_{(c,v,i)}$ soit utilisé dans la loi des événements, et que chaque événement atomique ait pour précondition affiliée le fait qu'il y ait bien l'atome propositionnel $\mathsf{cardpos}^P_{(c,v,i)}$ vrai.

Définition 7.15 (Le modificateur observationnel pour l'action « piocher »)**.** *Le modificateur observationnel $\langle V^+, \theta^+, \Omega^+, \Pi^+, V_-, \theta_-\rangle$ pour l'action de pioche étant donné une carte $(c,v,i) \in \mathsf{Cards}$ est :*

- $V^+ = \{\mathsf{pioche}^a_{(d,w,j)} \mid (d,w,j) \in \mathsf{Cards}\}$;
- θ^+ *comme défini ci-dessus ;*
- $\Omega^+_a = \top$;
- $\Omega^+_{b\in\mathfrak{A}\setminus a} = \bigwedge_{d,w,j\in\mathsf{Cards}} \mathsf{pioche}^a_{(d,w,j)} \leftrightarrow \mathsf{pioche}^a_{(d,w,j)}{}'$;
- $\Pi^+_a = \Omega^+_a$;
- $V_- = \mathsf{CP}^{(a,0)}_{(c,v,i)} \cup \mathsf{cardpos}^P_{(c,v,i)} \cup \mathsf{Turns}$;
- *et avec θ_- défini de sorte que :*
  - $\theta_-(\mathsf{cardpos}^P_{(c,v,i)}) = \mathsf{ite}(\mathsf{pioche}^a_{(c,v,i)},\ \bot,\ \mathsf{cardpos}^P_{(c,v,i)})$;
  - $\theta_-(\mathsf{cardpos}^{(a,0)}_{(c,v,i)}) = \mathsf{ite}(\mathsf{pioche}^a_{(c,v,i)},\ \top,\ \mathsf{cardpos}^{(a,0)}_{(c,v,i)})$;
  - $\forall b \in \mathfrak{A},\ \theta_-(\mathsf{turn}_b) = \mathsf{change\_turn}(\mathsf{turn}_b)$.

Ici, il est important de constater que $\Omega^+_a = \top$, c'est à dire qu'il n'y a aucune loi d'observations particulière pour l'agent a : il ne peut pas différencier les variables de V^+ et donc les actions atomiques. Par contre, les autres agents ont quant à eux

une loi d'observations qui spécifient qu'ils différencient clairement les variables de V^+.

Cet événement symbolique χ étant modélisé, il ne reste plus qu'à le combiner avec un événement pointé $x \in V^+$ de sorte à obtenir le modificateur $\langle \chi, x \rangle$.

Nous avons maintenant les différents ensembles d'actions possibles dans Hanabi :

- les annonces pour les couleurs :

$$\{\mathsf{annonceC}(c, a, L) \mid a \in \mathfrak{A}, c \in \{R, W, B, Y, G\}, L \subseteq \mathsf{pos}\};$$

- les annonces pour les valeurs :

$$\{\mathsf{annonceV}(v, a, L) \mid a \in \mathfrak{A}, v \in \{1, \ldots, 5\}, L \subseteq \mathsf{pos}\};$$

- les actions pour jouer une carte :

$$\{\mathsf{joue}(a, p, c, v, i) \mid a \in \mathfrak{A}, (c, v, i) \in \mathsf{Cards}, p \in \mathsf{pos}\};$$

- les actions pour défausser une carte :

$$\{\mathsf{defausse}(a, p, c, v, i) \mid a \in \mathfrak{A}, (c, v, i) \in \mathsf{Cards}, p \in \mathsf{pos}\};$$

- les actions de pioche possible :

$$\{\mathsf{pioche}(a, c, v, i) \mid a \in \mathfrak{A}, (c, v, i) \in \mathsf{Cards}\};$$

Nous précisons néanmoins que les actions joue et defausse doivent toujours être suivies par une action pioche. En effet, joue et defausse ne laissent par définition aucune carte en position 0 et ne changent pas le tour courant du joueur. C'est donc à l'action pioche de « remplir » la position 0 en main, puis de changer le tour du joueur.

Dans le programme des expérimentations, nous avons implémenté cela avec des pipelines d'actions, que ce soit en représentation explicite ou symbolique.

Nous allons maintenant pouvoir exploiter cette modélisation logique pour faire des expérimentations en pratique par des calculs sur ordinateur.

8 | Expérimentations

Toutes nos définitions de PDEL, modifiées de sorte à utiliser des loteries, sont formelles et ne permettent pas une mise en pratique directement. Dans la dernière section, nous avons utilisé ces définitions pour représenter le jeu Hanabi. Afin de mettre notre *framework* en pratique, nous avons implémenté la structure de données ADDs, qui représente en mémoire les PBFs, utilisées dans nos définitions. Nous allons maintenant illustrer comment se comportent nos définitions dans un cas pratique comme Hanabi en utilisant les ADDs.

Tout d'abord, nous allons présenter quelques élements de lecture afin de faciliter la lecture des graphiques qui vont suivre (sec. 8.1), puis nous montrerons des résultats expérimetaux sur les temps de création des structures (sec. 8.2), les tailles des structures (sec. 8.3), les temps de *model checking* (sec. 8.4), les temps de *product update* (sec. 8.5), et le comportement des structures lors de *product updates* successifs (sec. 8.6).

8.1 Éléments de lecture

Avant de montrer les résultats des expérimentations, nous allons décrire les codes couleurs utilisés afin de faciliter la lecture et l'interprétation des courbes que nous avons réalisées.

Des couleurs distinctes seront utilisées pour évoquer chaque type de structures :

- les structures de Kripke explicites probabilisées seront en rouge : (●) et étiquetées *Ex Pr.* dans les légendes (les résultats sur les structures non probabilisées sont très similaires et n'apportent rien) ;
- les structures de connaissances seront en vert clair (●) et étiquetées *SK* ;
- les structures de croyances seront en vert (●) et étiquetées *SC* ;
- les structures de probabilités dénormalisées seront en bleu (●) avec potentiellement les variantes ● et ● et étiquetées *SP* ;
- les structures de probabilités normalisées seront quant à elles en violet ● ou ● et étiquetées *SP-n*.

Pour faciliter la lecture des graphiques, quand un graphique ne mentionnera qu'une seule et unique structure, on utilisera la palette de couleur complète sans tenir compte de ces étiquetages.

Différents symboles seront utilisés pour chaque type d'élément (formules, nombre de cartes en main différents, etc.) ■ ◆ ▲ ▼ ●.

Par ailleurs, tous les calculs ont été effectués sur des serveurs dotés de 4 processeurs *AMD Opteron 6282SE 2.6GHz* avec 64 coeurs, pour un total de 512Go de RAM.

Cela représente beaucoup de mémoire vive et permet de faire tourner les expérimentations gourmandes qui ont lieu sur les structures de probabilités normalisées. En effet, il est possible d'atteindre des montants de RAM de 50Go pour ces processus.

Dans les cas non spécifiques, avec un nombre total de cartes de 50, un nombre d'agents de 2 avec 4 cartes en main ou avec un nombre d'agents de 3 avec 2 ou 3 cartes en main, le mémoire vive utilisée est en dessous de 16Go.

Cette quantité de mémoire vive a permis en réalité de pouvoir lancer de nombreux processus en parallèle sans saturer la mémoire.

Les calculs ont tous été réalisés plusieurs fois afin d'obtenir de moins de variance possible sur les courbes. Néanmoins, il apparaît des pics sur les courbes dont la provenance est difficilement explicable. L'allure des courbes est par contre toujours bel et bien lisible.

8.2 Temps pour la création

Les temps de créations généraux des structures sont présents sur la figure 8.1 ; ces temps représentent le temps total de création des instances représentant les connaissances/croyances des agents et les événements. Par exemple, dans le cas de PDEL explicite, il s'agit de la structure de Kripke probabilisée et de tous les modèles observationnels permettant de la modifier. Dans les cas symboliques, il s'agit du temps de création de la structure (de connaissance, de croyances, de probabilités ou de probabilités normalisées) avec les modificateurs adéquats.

Les courbes représentant la génération du jeu d'Hanabi de manière explicite, en rouge (●, ▲, ▼), ont visiblement une allure exponentielle. Notons que les courbes rouges ne sont présentes que pour 2 et 3 cartes ; pour 4 cartes, cela nécessitait d'avoir 8 cartes en jeu au total pour deux agents, ce qui est trop à modéliser de manière explicite. Ce résultat exponentiel n'est pas étonnant. Le nombre de mondes possibles dans la structure de Kripke augmente comme le cœfficient binomial $\binom{\mathsf{nbC}}{\mathsf{nbH}}$.

Les structures de croyances étant plus simples à modéliser grâce à des variables d'observation, nous retrouvons les temps de génération les plus bas pour les structures de connaissances (SK) en vert clair (●, ▲, ■, ▼).

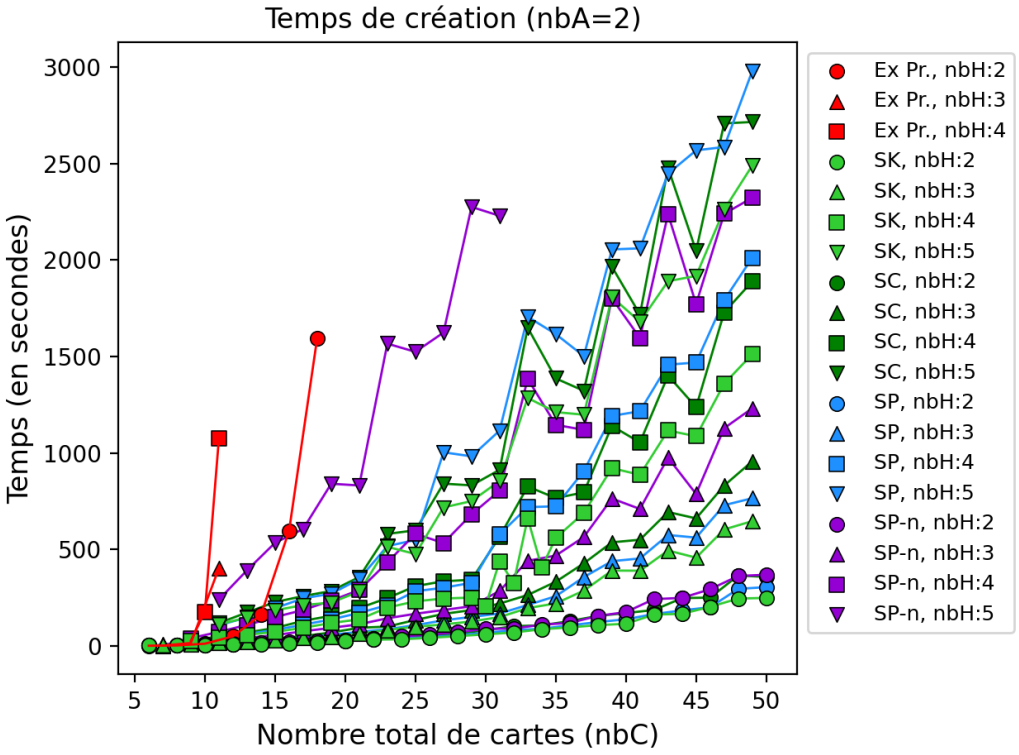


FIGURE 8.1 – Temps de création des différentes structures de Kripke (Ex Pr., ●), structures de connaissances (SK, ●), structures de croyances (SC, ●), structures de probabilités (SP, ●) et structures de probabilités normalisées (SP-n, ●), pour deux agents, en fonction du nombre de total cartes nbC, et en faisant varier le nombre de cartes en main nbH. Timeout=3000s

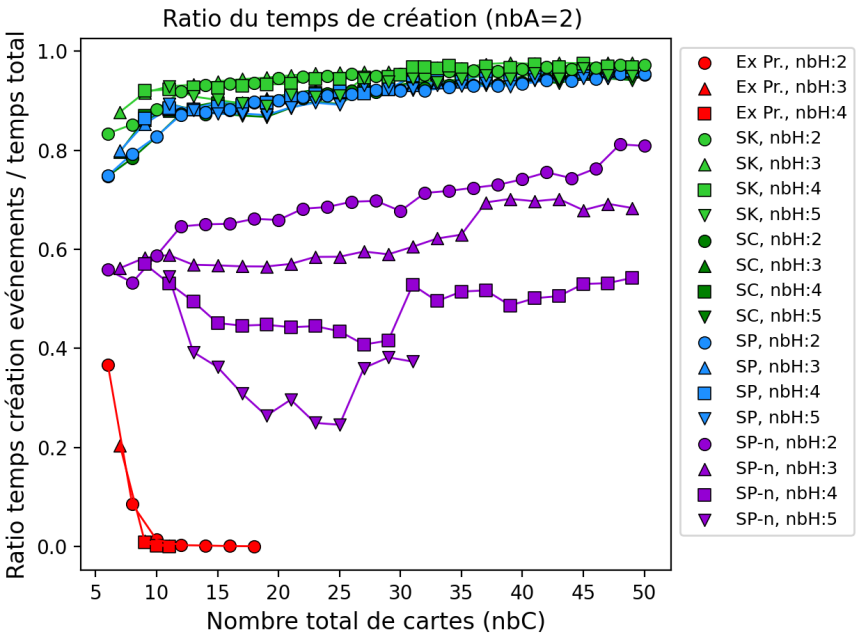
Ensuite, dans l'ordre croissant de temps de génération, nous trouvons les structures de croyances en vert (●, ▲, ■, ▼). Juste au dessus se trouvent les structures de probabilités en bleu (●, ▲, ■, ▼) : elles ne peuvent bien évidemment pas se trouver plus bas que les courbes des structures de croyances, car la création d'une structure de probabilités inclut le temps de création d'une structure de croyances. Néanmoins, leurs temps sont très similaires car Π_a est construit directement à partir de Ω_a.

De plus, la différence la plus notable est la différence entre le temps de création de la structure de probabilités (●, ▲, ■, ▼) et celui de la structure de probabilités normalisée (●, ▲, ■, ▼). La différence de temps de génération est ici simple à expliquer : comme dit précemment, pour modéliser Hanabi avec des probabilités, nous pouvons créer Π_a normalisé en utilisant la fonction boolénne qui est ici $\Omega_a \times \theta'$ (avec Ω la relation de croyances de la structure de croyances et θ la loi des mondes). Néanmoins, il faut ajouter un calcul supplémentaire pour obtenir Π normalisée, en utilisant la marginalisation (déf. 5.11). Enfin, le paramètre qui joue indéniablement sur la temps de création des structures est le nombre de cartes en main nbH (● pour deux cartes, ▲ pour trois, ■ pour 4 et ▼ pour 5). En effet, pour ce qui est des quatre courbes symboliques annotées par un rond (●, ●, ●, ●) on a :

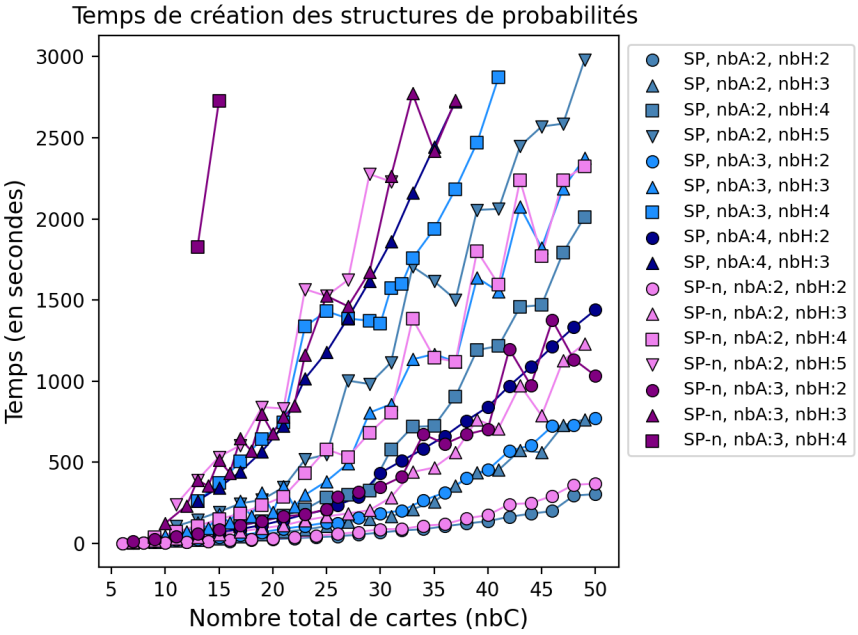
- pour un nombre de 50 cartes et $\mathsf{nbH} = 2$, les temps sont entre 250s (●) et 400s(●) ;
- pour $\mathsf{nbH} = 3$ les temps sont compris entre 600s (▲) et 1300s (▲) ;
- pour $\mathsf{nbH} = 4$, les temps s'écartent clairement, entre 1500s (■) et 2300s (■) ;
- enfin, pour $\mathsf{nbH} = 5$, tous les temps dépassent 2500s pour 50 cartes (▼, ▼ et ▼) mais la courbe normalisée (▼) n'atteint que 32 cartes avec plus de 2250 secondes (le timeout étant de 3000 secondes, le point suivant n'a pas été crée).

Ensuite, nous montrons des compléments à ces temps de calculs dans la figure 8.2, page 195. En haut (fig. 8.2a), nous présentons le pourcentage que prend le temps de création des structures d'événements comparé au temps total des structures, en fonction du nombre total de cartes en main. L'idée principale à voir est la suivante :

- pour les structures symboliques de connaissances (●, ▲, ■, ▼), de croyances (●, ▲, ■, ▼) et de probabilités (●, ▲, ■, ▼), le ratio est très élevé et globalement au dessus de 80%. Par ailleurs, plus le nombre total de cartes augmente, plus le ratio se rapproche de 1. Ainsi, le temps de calcul des structures de connaissances/croyances/probabilités est assez faible vis à vis du temps de calcul des modificateurs ;
- pour la structure explicite (●, ▲, ■), c'est exactement l'inverse qui se produit. La structure de Kripke prend rapidement tout le temps de création, car



(a) Représentation du temps de création des modèles d'événements sur le temps de création total des structures, pour chaque type de structures, en fonction du nombre total de cartes nbC, en faisant varier le nombre de cartes en main nbH.



(b) Temps de création total des structures symboliques dénormalisées et normalisées en fonction du nombre total de cartes nbC, en faisant varier le nombre de cartes en main nbH et le nombre d'agents nbA.

FIGURE 8.2 – Compléments sur les temps de création des structures symboliques probabilisées avec deux agents.

le nombre de mondes à créer est exponentiel.

- enfin, pour la structure de probabilités normalisée (●, ▲, ■, ▼), bien que le comportement semble plus chaotique, nous pouvons interpréter que la marginalisation pour créer la loi de probabilités Π prend beaucoup de temps et que ce temps rivalise avec le temps de créations de structures d'événements.

Nous pouvons aussi voir que la part du temps de calcul est globalement pris par la construction des modificateurs. La raison est simple : plus il y a de cartes en main, plus il y a de cartes jouables, plus il y a d'actions possibles à créer. Il en va de même avec le nombre total de cartes, augmentant le nombre d'actions « piocher » possible.

Pour ce qui est du second graphique présenté à la figure 8.2b, nous pouvons y voir les temps de création total des structures initiales symboliques probabilistes en fonction du nombre total de cartes en abscisse, en faisant varier le nombre d'agents (nbA, dénoté par les trois teintes de bleu (●, ● et ●) et deux de violet (●,●) et du nombre de cartes en main (nbH, dénoté par les différents symboles ●, ▲, ■, ▼). Ces valeurs sont déjà partiellement présentes dans la figure 8.1 que nous avons déjà commentée pour nbA = 2. Ici, l'interêt est de constater l'ajout d'autres agents dans la partie. Pour rappel, s'il y a 4 ou 5 agents, il faudrait pouvoir avoir 4 cartes en main, et s'il y a 2 ou 3 agents, il faudrait 5 cartes en main. Nous pouvons voir qu'avec deux agents, les temps de création sont importants mais raisonnables pour modéliser le jeu avec 50 cartes (▼ avec près de 3000 secondes). Nous pouvons voir qu'avec trois agents et trois cartes en main (▲), la génération est faite en 2250 secondes, et que pour quatre agents et trois cartes en main, la génération prend 2750 secondes pour 36 cartes (▲).

Nous pouvons par la même occasion comparer les temps de création entre les structures dénormalisées et normalisées : il est toujours plus long de créer les structures normalisées. Pour vérifier cela, nous pouvons par exemple comparer les couples de symboles (▲, ▲) ou encore (▼, ▼).

Ainsi, nous nous rapprochons fortement de pouvoir modéliser le jeu complet en un temps raisonnable pour deux agents ; pour trois ou quatre agents, cela semble plus complexe. Néanmoins, nous avons quelques perspectives d'amélioration : en effet, nous créons les formules booléennes (que ce soit pour la création des PS ou des PT) pour chaque joueur, mais par exemple, nous pouvons imaginer en créer une seule, et la dupliquer pour les autres agents en renommant des variables. Ceci n'est pour le moment pas possible, car le programme des ADDs ne permet pas le réordonnancement des variables.

8.3 Tailles des structures

Une fois toutes les structures créés, nous avons comparé leurs tailles. Notamment, nous avons comparé la taille des structures de données, calculée en termes de nombre de nœuds dans le diagramme de décision.

À la création de ces diagrammes, il est nécessaire de spécifier l'ordre d'apparition voulu des atomes propositionnels, si l'on veut que les structures n'explosent pas en taille. En effet, si celui-ci n'est pas spécifié, il est alors automatiquement établi en fonction de l'ordre l'apparition des atomes dans les formules propositionnelles qui servent à créer les lois telles que θ ou Ω. Nous avons donc fait le choix de fixer cet ordre.

8.3.1 Ordre des variables

Les atomes écrits $\mathsf{cardpos}^{(a,p)}_{(c,v,i)}$ décrivant le jeu Hanabi (voir sec. 7.3.1) sont décomposables en 5 éléments distincts : « agent - position - couleur - valeur - id ». Ces atomes ont ensuite des variantes : $\mathsf{cardpos}^{(a,p)}_{(c,v,i)}{}^{\circ}$, $\mathsf{cardpos}^{(a,p)}_{(c,v,i)}{}'$, et la combinaison des deux $\mathsf{cardpos}^{(a,p)}_{(c,v,i)}{}^{\circ\prime}$. Nous étudions dans la table 8.1 l'ordre des variantes en priorisant les variables dans l'ordre affiché dans la première colonne, en notant « cvi » pour la carte, « a » pour l'agent, « p » pour la position, « pr » pour la version primée de l'atome et « h » pour la version historicisée. Ainsi pour un ordre donné « c, ag, pos, pr, hist », toutes les variables avec même « *cvi* » seront côte à côte, puis même agent, puis même position, puis la version prime, et enfin, la version historicisée. En variant les priorités de ces 5 élements, on obtient 120 combinaisons possibles. Seulement 23 permettent de faire l'initialisation des structures en moins de 10 secondes pour 2 agents, 2 cartes en main et 20 cartes au total. L'ordre des variables impacte la taille des structures, que ce soit pour θ, Ω ou Π. Nous avons choisi de prendre l'ordre suivant : « cvi, a, p, pr, h » qui minimise la taille de Ω. Nous aurions pu choisir un des autres ordres présenté dans la table, car les résultats sont très similaires. Mais il est clair que des ordres mal choisis auraient été catastrophiques.

Notons que cet ordre choisi optimise le temps de construction des structures et des modificateurs. Nous sommes partis du principe qu'il vaut mieux créer la structure de manière efficace pour pouvoir y faire du *model checking* que de faire des mises à jour ; mais de manière globale, un autre ordre est peut-être meilleur pour effectuer de nombreux *product updates*.

Le choix de l'ordre impacte bien le temps de création des structures, mais aussi la taille des ADDs qui les représentent.

Ordonnancement des atomes	$\|\Omega\|$	$\|\theta\|$	création $\mathfrak{F}$	création χ	Temps total
cvi, a, p, pr, h	269	3221	0.321566	4.781376	5.107425
cvi, a, p, h, pr	269	3221	0.375349	5.307240	5.687694
cvi, a, h, p, pr	269	3221	0.323354	4.495437	4.823143
cvi, p, a, pr, h	269	3104	0.437863	5.092653	5.536484
cvi, p, a, h, pr	269	3104	0.344957	4.639441	4.989341
cvi, p, h, a, pr	269	3104	0.343941	4.833033	5.181351
cvi, h, a, p, pr	269	3221	0.363620	4.747005	5.115474
cvi, h, p, a, pr	269	3104	0.344755	4.848893	5.198476
cvi, a, pr, p, h	329	3221	0.369018	4.656556	5.030259
cvi, a, pr, h, p	329	3221	0.363393	4.780191	5.147888
cvi, a, h, pr, p	329	3221	0.478293	5.477156	5.960957
cvi, h, a, pr, p	329	3221	0.375428	5.448389	5.828443
cvi, p, pr, a, h	509	3104	0.390902	4.968729	5.364091
cvi, p, pr, h, a	509	3104	0.392995	4.994261	5.391883
cvi, p, h, pr, a	509	3104	0.401242	5.016821	5.422853
cvi, h, p, pr, a	509	3104	0.368489	4.589792	4.962522
cvi, pr, a, p, h	929	3221	0.551707	4.837305	5.394047
cvi, pr, a, h, p	929	3221	0.488440	4.713903	5.207376
cvi, pr, p, a, h	929	3104	0.472559	5.072484	5.550161
cvi, pr, p, h, a	929	3104	0.583755	5.198914	5.789208
cvi, pr, h, a, p	929	3221	0.628462	5.157758	5.793105
cvi, pr, h, p, a	929	3104	0.517745	5.384240	5.907098
cvi, h, pr, a, p	929	3221	0.490310	4.880220	5.375729
cvi, h, pr, p, a	929	3104	0.475768	5.135147	5.615755

TABLE 8.1 – Temps de génération et taille des ADD pour Hanabi avec 2 agents et 20 cartes en fonction de l'ordre des variables (avec un timeout de 10 secondes)

8.3.2 Nombre de variables et tailles des ADDs

Voyons comment peut varier la taille des ADDs, non pas par l'ordre des variables, mais tout simplement par leur nombre.

La modélisation du jeu d'Hanabi, comme nous l'avons vu, est paramétrable en fonction du nombre d'agents nbA, du nombre de cartes en main nbH et du nombre total de cartes en jeu nbC. Ces facteurs modifient le nombre total de variables propositionnelles pour représenter le problème. Nous allons faire varier ces paramètres et étudier l'évolution de la taille des ADDs.

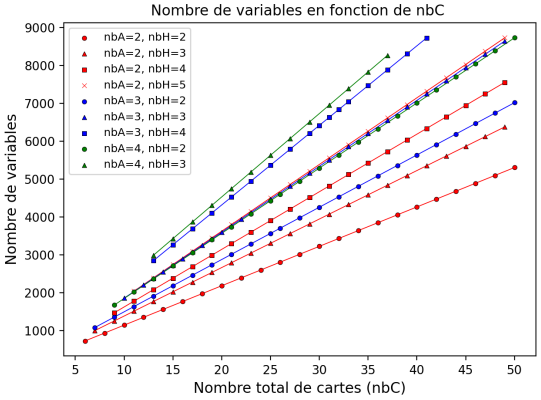
Nous précisons que les chiffres que nous allons montrer sont ceux après la création des structures, avec les variables nécessaires pour représenter le jeu (V) et les fonctions Ω et Π ($V \cup V'$), avec les variables d'historicisation (V°) présentes pour effectuer potentiellement quatre mises à jour. Par exemple, pour $\mathsf{nbA} = 2$, $\mathsf{nbH} = 2$ et $\mathsf{nbC} = 50$, il y aura 359 variables dans V mais 5 308 variables dans $V \cup V' \cup V^+ \cup V^{+'} \cup V^\circ \cup V'^\circ$. Pour commencer, nous avons besoin d'estimer l'évolution du nombre de variables présentes dans les formules booléennes et formules pseudo-booléennes représentant les structures symboliques. Ces informations sont présentes dans la figure 8.3a : clairement, le nombre de variables nécessaires pour représenter le jeu Hanabi est linéaire en fonction du nombre total de cartes en jeu : nbC, quelles que soient les tailles des mains des agents.

Nous présentons dans le graphique 8.3b la taille des lois de probabilités dans les cas dénormalisés et normalisés. Nous pouvons voir à quel point les tailles augmentent extrémement rapidement dans le cas normalisé. A contrario, comme, nous pouvons le voir dans le graphique 8.3c, qui est un zoom du graphique 8.3b, la taille des ADDs représentant les lois de probabilités dénormalisées sont linéaires en fonction du nombre total de cartes en jeu, peu importe le nombre de cartes en main ou le nombre d'agents. Néanmoins, nous voyons aussi clairement que la taille des ADDs pour représenter Π normalisé augmente bien plus rapidement que pour Π dénormalisé (qui, pour rappel, est équivalent à Ω). En outre, plus nbH augmente, plus la pente des droites affines est importante.

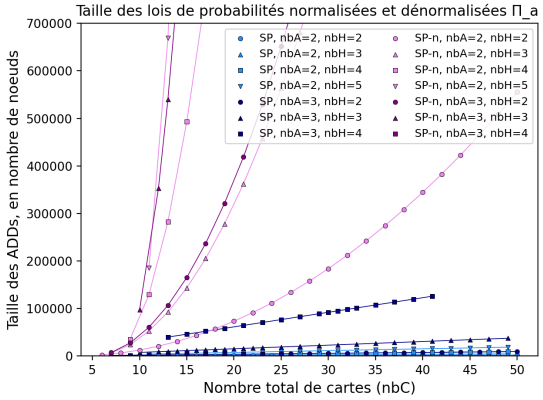
8.4 Temps pour le model checking

Après avoir vu comment se comportaient les courbes pour les temps de création des structures, nous allons voir quels sont les temps de calcul pour effectuer du *model checking*.

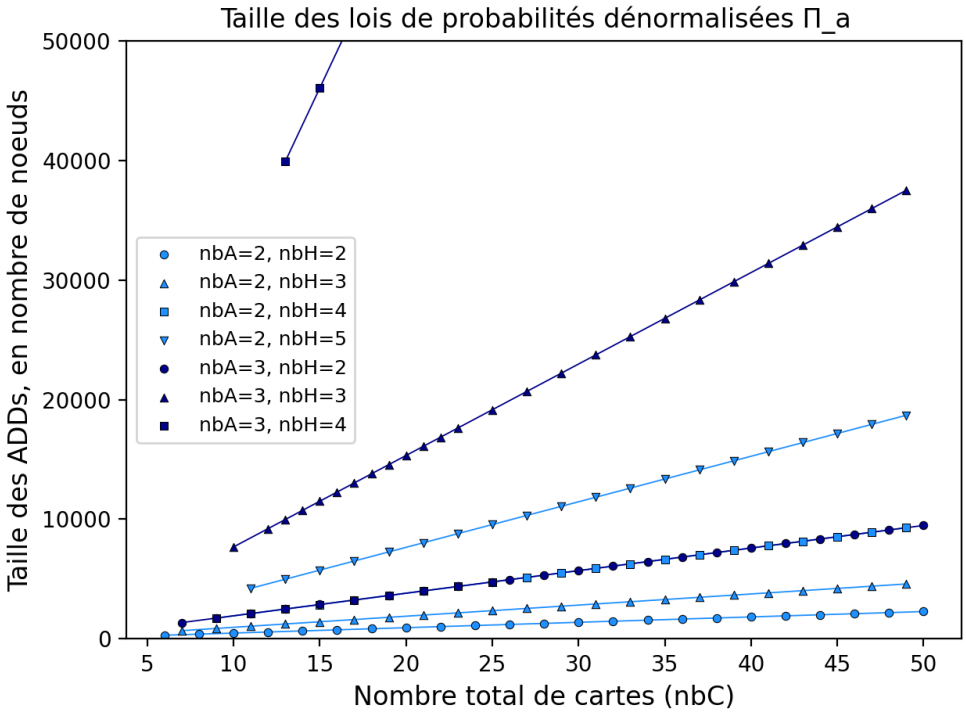
Nous allons commencer par la comparaison de temps de calculs pour le *model checking* de formules épistémiques non probabilistes pour les structures explicites et symboliques (fig. 8.4) : les structures de Kripke probabilistes, les structures de connaissances et les structures de croyances. Ici, nous ne présentons pas les



(a) Nombre de variables en fonction du nombre total de cartes.

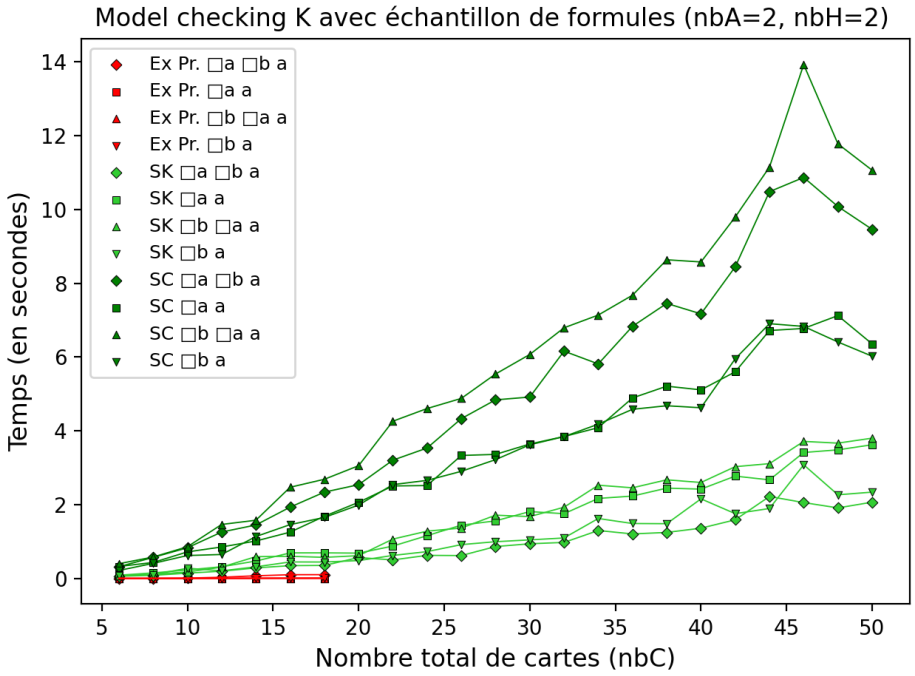


(b) Tailles des lois de probabilités dénormalisées et normalisées en fonction du nombre total de cartes.

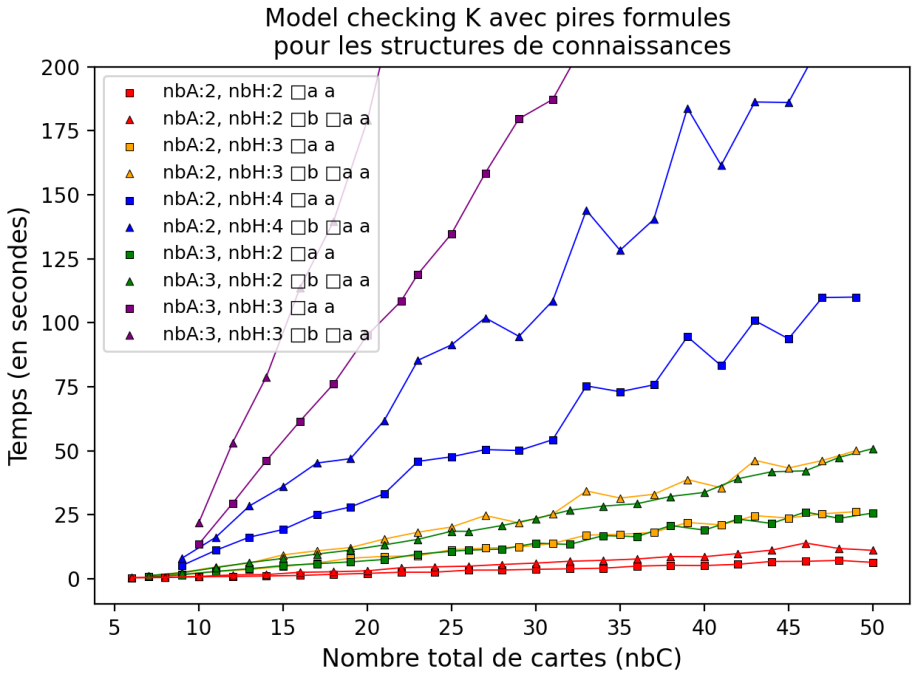


(c) Tailles des lois de probabilités dénormalisées en fonction du nombre total de cartes.

FIGURE 8.3 – Étude du nombre de variables et des tailles de Π en fonction du nombre de cartes en faisant varier nbA et nbH.



(a) Temps de calcul pour le *model checking* pour des formules avec l'opérateur □ en fonction de nbC.



(b) Temps de calcul pour le *model checking* pour des formules avec l'opérateur □ en fonction de nbC en variant nbAet nbH.

FIGURE 8.4 – Temps de *model checking* pour les formules épistémiques.

structures de probabilités, qui possèdent le même fonctionnement que les structures de croyances et qui présentent donc les mêmes résultats en terme de temps.

Nous avons fait des tests sur 8 formules différentes de profondeurs 1 et 2, à partir des deux sous-formules suivantes :

- $\phi_a = \bigvee_{c\in\{R,W,B,Y,G\}} \bigvee_{i\in\mathsf{ids}_1} \mathsf{cardpos}^{(a,0)}_{(c,1,i)}$, signifiant « l'agent a a une carte de valeur 1 en position 0 » ;
- $\phi_b = \bigvee_{c\in\{R,W,B,Y,G\}} \bigvee_{i\in\mathsf{ids}_1} \mathsf{cardpos}^{(b,0)}_{(c,1,i)}$, signifiant « l'agent b a une carte de valeur 1 en position 0 ».

Ces 8 formules sont les suivantes : $\Box_a\phi_a$, $\Box_a\phi_b$, $\Box_b\phi_b$, $\Box_b\phi_a$, $\Box_a\Box_b\phi_b$, $\Box_a\Box_b\phi_a$, $\Box_b\Box_a\phi_b$, $\Box_b\Box_a\phi_a$. Elles sont toutes les permutations possibles de profondeur 1 et 2 sans appliquer deux opérateurs identiques consécutifs (i.e. en modifiant les indices sur $\Box$). Dans nos graphiques, nous avons conservé seulement les formules dont le *model checking* prennent le plus de temps ; c'est à dire que dans la figure 8.4a les *model checking* de $\Box_a\Box_b a$ et $\Box_b\Box_a a$ prennent plus de temps que ceux de $\Box_a\Box_b b$ et $\Box_b\Box_a b$. Nous faisons de même pour la figure 8.4b. En outre, nous étiquetons, par exemple, « $\Box_a\Box_b a$ » pour la formule $\Box_a\Box_b\phi_a$.

Avec la figure 8.4a, nous comparons le *model checking* sur les 3 types de structures que nous avons à notre disposition, pour 4 formules épistémiques, en fonction du nombre total de cartes.

Nos travaux portent bel et bien sur les probabilités, néanmoins nous présentons les résultats sur les formules purement épistémiques pour pouvoir comparer les temps en fonction des formules et du nombre de cartes.

Deux informations importantes sont à retirer de ce graphique 8.4a. Premièrement, le *model checking* sur une structure de Kripke explicite est très rapide (●, ▲, ■, ▼). Néamoins, ne pouvant pas créer la structure au delà de 18 cartes, il est impossible de voir l'évolution de la courbe. Le nombre de mondes augmentant exponentiellement dans la structure, la courbe devrait suivre un aspect exponentiel. Deuxièmement et sans surprise, le *model checking* sur la structure de connaissances (●, ▲, ■, ▼) est le plus rapide d'entre toutes les structures.

Les temps de *model checking* sont plus longs pour la structure de croyances (●, ▲, ■, ▼) et nous pouvons voir apparaître un ordre dans les temps : $\Box_b\Box_a\phi_a > \Box_a\Box_b\phi_a > \Box_a\phi_a \geq \Box_b\phi_a$. Ayant omis d'afficher les formules les plus rapides, nous pouvons voir que ce sont les temps de *model checking* des sous-formules portant sur l'agent a qui sont les plus longs. En outre, plus la profondeur de la formule est grande, plus le temps est conséquent.

Nous avons ensuite fait les mêmes expériences qu'avec $\Box$, mais avec des formules probabilisées.

Lors du *model checking* de plusieurs formules différentes, des opérations sont répétées. C'est le cas des calculs de $\Pi_a \cdot \theta'$ et de $\mathrm{Marg}^+_{V'}(\Pi_a \cdot \theta')$ (voir def 5.15),

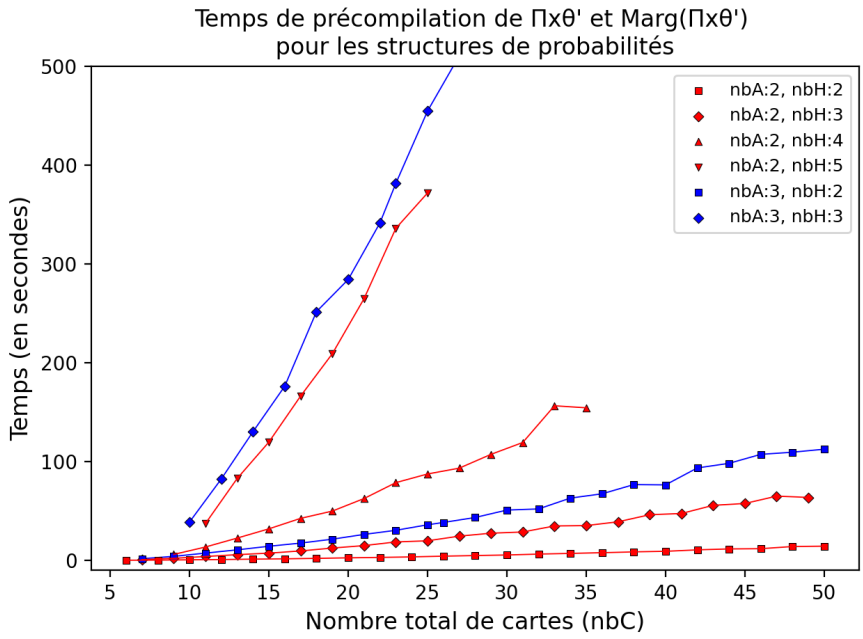


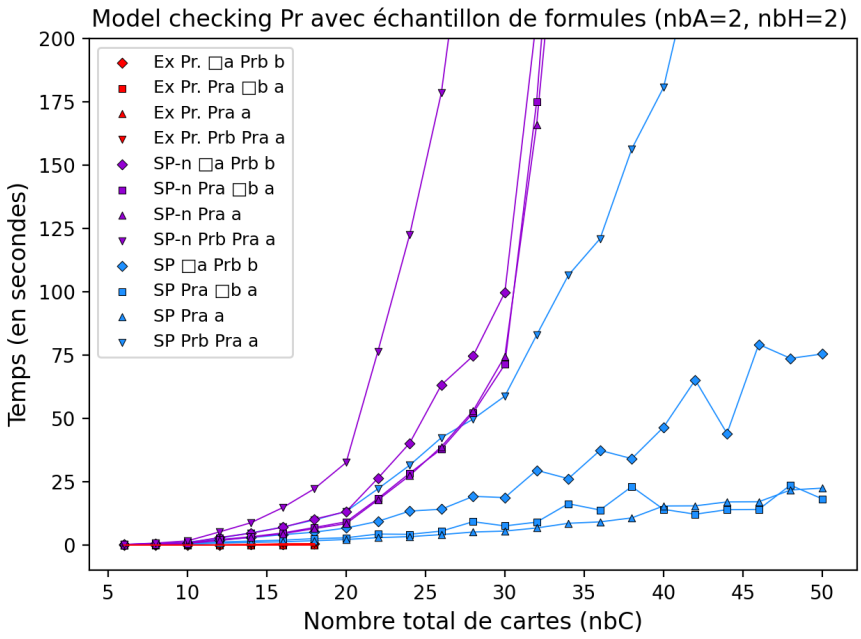
FIGURE 8.5 – Temps de calcul des précompilations de $\Pi_a \cdot \theta'$ et $\mathrm{Marg}^{+}_{\Pi_a \cdot \theta'}(V')$ servant au *model checking* pour des formules de $\mathcal{L}_{\mathrm{PEL}}$, dans le cas de structures de probabilités dénormalisées.

qui sont des constantes une fois une structure de probabilités créée. Dans notre programme, nous calculons ces informations une seule et unique fois, au premier *model checking* effectué. Ainsi, tous les autres *model checking* sur d'autres formules bénéficieront de ce pré-traitement. Nous présentons les temps de ces calculs sur la figure 8.5, en faisant varier nbA et nbH . Nous pouvons voir que ces temps sont linéaires en fonction du nombre total de cartes nbC, mais que les droites sont de plus en plus pentues en fonction de nbA et nbH.

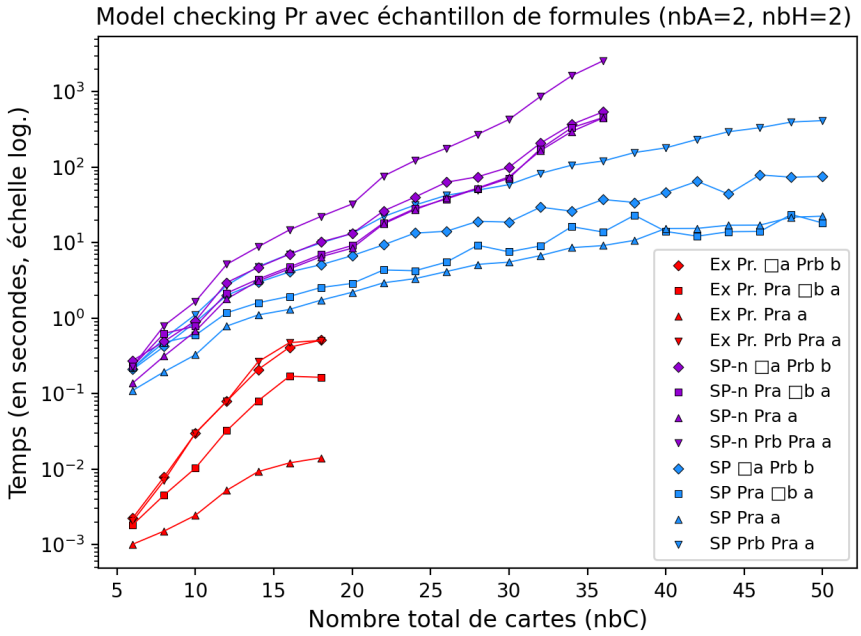
La figure 8.6 montre les résultats du *model checking*. Les formules utilisées sont les mêmes que précédemment avec $\Box$ (en remplaçant $\Box_i$ par Pr_j) mais en ajoutant des formules mêlant les opérateurs $\Box$ et Pr. Nous avons donc par exemple les formules :

- $\mathrm{Pr}_a(\phi_a) \geq 0.25$, notée « *Pra a* » dans le graphique ;
- $\mathrm{Pr}_a(\Box_b(\phi_a)) \geq 0.25$, notée « *Pra* $\Box b$ *a* » ;
- $\mathrm{Pr}_b(\mathrm{Pr}_a(\phi_a) \geq 0.25) \geq 0.25$, notée « *Prb Pra a* ».

Nous omettrons d'écrire les formules complètes et noterons par abus $\mathrm{Pr}_a(\phi_a)$ à la place de $\mathrm{Pr}_a(\phi_a) \geq 0.25$. Comme pour les formules épistémiques, ce sont toujours les formules dont le *model checking* est le plus long qui sont affichées et comme convenu, nous affichons chacun des types de structure avec une couleur distincte.



(a) Temps de calcul pour le *model checking* pour des formules de $\mathcal{L}_{\text{PEL}}$: explicite et symbolique.



(b) Temps de calcul pour le *model checking* pour des formules de $\mathcal{L}_{\text{PEL}}$: explicite et symbolique (échelle log).

FIGURE 8.6 – Comparaison des temps de *model checking* pour les formules de $\mathcal{L}_{\text{EL}}$ en fonction du nombre total de cartes.

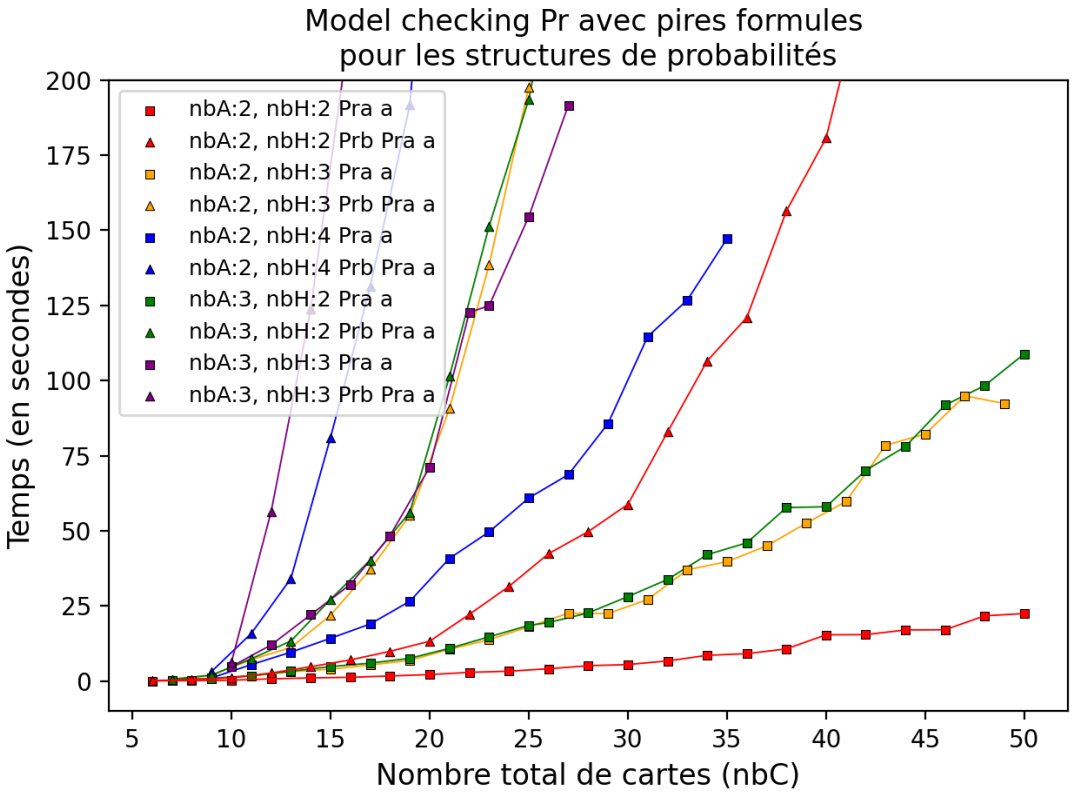


FIGURE 8.7 – Temps de calcul pour le *model checking* pour des formules de $\mathcal{L}_{\text{PEL}}$ sur les structures de probabilités dénormalisées, en faisant varier nbA et nbH.

Commençons par comparer les courbes présentes sur la figure 8.6a, qui présente le résultat du *model checking* pour les trois structures avec nbA et nbC fixés chacun à 2. D'abord, notons que le *model checking* est plus rapide pour la structure explicite (en rouge, ●, ▲, ■, ▼), ce que nous présentons avec la figure 8.6b, qui est la représentation en échelle logarithmique de 8.6a. Le comportement est identique au cas des formules épistémiques vu précédemment. Notons que le *model checking* semble instantané, mais il augmente rapidement pour les formules de prfondeur 2 : pour $\Box_a \mathrm{Pr}_b b$ et $\mathrm{Pr}_b \mathrm{Pr}_a a$ les temps sont de 0.1s pour $\mathsf{nbC} = 12$, 0.25s pour $\mathsf{nbC} = 14$ et 0.45s pour $\mathsf{nbC} = 16$. Le niveau de comparaison n'est simplement pas le même qu'avec les structures symboliques en terme de temps, mais le temps augmente bel et bien et de plus en plus.

Ensuite, nous pouvons clairement constater que le comportement du *model checking* pour les structures probabilistes normalisées est exponentiel (●, ▲, ■, ▼), alors qu'il est plutôt linéaire pour les structures probabilisées pour les formules $\Box_a \mathrm{Pr}_b b$, $\mathrm{Pr}_a \Box_b a$ et $\mathrm{Pr}_a a$. Pour ce qui est de la formule $\mathrm{Pr}_b \mathrm{Pr}_a a$, le temps est plus conséquent.

En fait, il s'avère qu'il est plus rapide de vérifier la formule $\mathrm{Pr}_b \phi$ que $\mathrm{Pr}_a \phi$ quand ϕ porte sur les variables que l'agent b observe. Nous avons condensé les temps de calculs dans la table 8.2 qui présente tous les temps de calculs des différents *model checking*. Dans la partie gauche du tableau, nous présentons les formules avec comme sous-formule ϕ_b et dans la partie droite ϕ_a. Chaque partie est divisée en deux : $a < b$ signifie que l'opérateur avec pour agent b est utilisé en priorité vis-à-vis de a, et $b < a$ signifie l'inverse.

Avec les formules épistémiques avec ϕ_a, nous avons :

$$\Box_b \Box_a \phi_a > \Box_a \Box_b \phi_a > \Box_a \phi_a > \Box_b \phi_a$$

et avec les formules purement probabilistes, nous avons aussi :

$$\mathrm{Pr}_b \mathrm{Pr}_a \phi_a > \mathrm{Pr}_a \mathrm{Pr}_b \phi_a > \mathrm{Pr}_a \phi_a > \mathrm{Pr}_b \phi_a$$

Ainsi, l'imbrication des opérateurs est important quant au temps de *model checking* : quand l'opérateur imbriqué (quel qu'il soit) porte sur le même agent que le sous-formule ϕ, le temps est plus long. En outre, le même comportement est constaté en version explicite.

L'opération Pr_a prend plus de temps que Pr_b et le temps se répercute sur les formules qui contiennent cette sous-formule. Pour comprendre pourquoi, nous avons analysé les tailles des ADDs (en nombre de nœuds) retournés par les traductions locales de ces formules : $|||\mathrm{Pr}_b \phi_a||_{\mathfrak{F}}| = 2890$ tandis que $|||\mathrm{Pr}_a \phi_a||_{\mathfrak{F}}| = 25528$. Ainsi, les opérations nécessaires pour effectuer le *model checking* à partir de cette structure nécessitent plus de calculs et sont donc plus longues. L'opération qui prend le plus de temps est la marginalisation somme (Marg_+).

Formules avec ϕ_b				Formules avec ϕ_a			
$a < b$		$b < a$		$a < b$		$b < a$	
$\Box_b\phi_b$	4.09	$\Box_a\phi_b$	3.65	$\Box_b\phi_a$	5.65	$\Box_a\phi_a$	5.79
$\Box_a\Box_b\phi_b$	8.47	$\Box_b\Box_a\phi_b$	7.09	$\Box_a\Box_b\phi_a$	9.17	$\Box_b\Box_a\phi_a$	10.24
$\mathrm{Pr}_a\Box_b\phi_b$	8.48	$\mathrm{Pr}_b\Box_a\phi_b$	45.17	$\mathrm{Pr}_a\Box_b\phi_a$	18.26	$\mathrm{Pr}_b\Box_a\phi_a$	9.13
$\mathrm{Pr}_b\phi_b$	19.51	$\mathrm{Pr}_a\phi_b$	8.35	$\mathrm{Pr}_b\phi_a$	7.64	$\mathrm{Pr}_a\phi_a$	22.52
$\Box_a\mathrm{Pr}_b\phi_b$	75.46	$\Box_b\mathrm{Pr}_a\phi_b$	9.05	$\Box_a\mathrm{Pr}_b\phi_a$	36.5	$\Box_b\mathrm{Pr}_a\phi_a$	66.72
$\mathrm{Pr}_a\mathrm{Pr}_b\phi_b$	394.15	$\mathrm{Pr}_b\mathrm{Pr}_a\phi_b$	27.02	$\mathrm{Pr}_a\mathrm{Pr}_b\phi_a$	22.73	$\mathrm{Pr}_b\mathrm{Pr}_a\phi_a$	414.8

TABLE 8.2 – Temps de calcul du *model checking* sur une structure de probabilités de toutes les formules pour $\mathsf{nbA} = 2, \mathsf{nbH} = 2$ et $\mathsf{nbC} = 50$.

Notons que les variables de l'agent a sont déclarées avant celles de l'agent b dans le diagramme de décision (i.e. les variables de l'agent b sont plus profondes dans le diagramme). Or, marginaliser ou oublier des variables présentes en bas du diagramme est plus rapide que de marginaliser ou oublier des variables qui sont qui sont proches de la racine. Ainsi, il est un peu plus rapide de calculer $||\Box_a\phi_b||_{\mathfrak{F}}$ ou $||\mathrm{Pr}_a\phi_b||_{\mathfrak{F}}$ que $||\Box_a\phi_a||_{\mathfrak{F}}$ ou $||\mathrm{Pr}_a\phi_a||_{\mathfrak{F}}$.

Un ordre différent des variables dans le diagramme de décision changerait sûrement les temps obtenus, mais peut-être aussi au détriment d'autres calculs, comme la création des structures. Néanmoins, il s'agit d'un compromis : déclarer les variables de l'agent b avant les variables de l'agent a inverserait le problème. Nous avons déjà discuté de l'ordre des variables utilisé (sec. 8.3.1) et nous ne pensons pas pouvoir trouver mieux.

Enfin, nous montrons sur la figure 8.7, page 205, les résultats du *model checking* des formules probabilistes, avec un ou deux opérateurs Pr, pour lesquelles le temps est le plus long ($\mathrm{Pr}_a\phi_a$ et $\mathrm{Pr}_b\mathrm{Pr}_a\phi_a$), en fonction du nombre total de cartes, en faisant varier nbA et nbH. Les courbes rouges sont identiques aux formules que l'on pouvait déjà voir dans le graphique pour $\mathsf{nbA} = 2$ et $\mathsf{nbH} = 2$ (■ et ▲). On peut voir que le *model checking* des formules de profondeur 1 (■) prennent un temps considérable dans les cas où $\mathsf{nbA} \neq 2$ et $\mathsf{nbH} \neq 2$. Cela s'explique par le nombre de variables pour représenter le problème. Comme nous l'avons vu, la taille des ADDs représentant les lois de probabilités Π_a est linéaire en fonction de nbC (fig. 8.3c), mais plus nbH ou nbA augmente, plus le cœfficient directeur des droites est important. Ainsi, les opérations de marginalisation présentes dans le *model checking*, qui sont polynomiales, prennent plus de temps.

Ainsi, nous pouvons dire que le *model checking* est possible sur Hanabi représenté de manière symbolique. Néanmoins, les résultats dépendent beaucoup des formules testées et le *model checking* des formules de profondeur modale 2 peut s'avérer long. Ainsi, sur des problèmes dédiés et en sachant à l'avance quelles for-

mules vont êtres vérifiées, il est possible de choisir l'ordre des variables de sorte à gagner en efficacité. Ici, ce n'est pas vraiment possible à cause de toutes les symétriques entre les relations d'accessibilité des agents : peu importe l'ordre, le *model checking* sur un agent sera moins efficace que sur un autre.

Un point d'amélioration potentiel pour gagner du temps lors du *model checking* est l'implémentation d'un algorithme spécifique qui remplacerait les opérations successives Marg et Cut. En effet, la marginalisation prend du temps. Or, de nombreux sous-ADDs seront par la suite « enlevés » par l'opération « coupe ». Il serait alors intéressant de profiter du parcours du diagramme et d'effectuer la coupe simultanément, de sorte à n'avoir à marginaliser que des ADDs réduits au strict minimum. En somme, il s'agirait de réaliser une opération équivalente aux deux opérations successives afin d'éviter des calculs fastidieux et inutiles.

8.5 Temps pour le product update

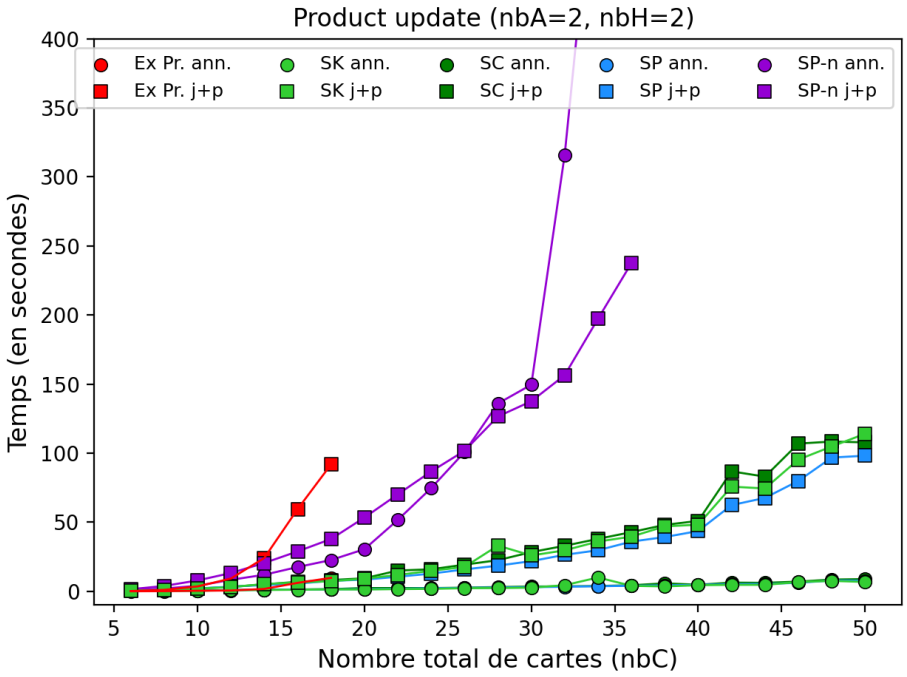
Nous avons ensuite fait le même type d'expériences avec la mise à jour des connaissances/croyances/probabilités. Pour cela, nous avons utilisé deux actions d'Hanabi :

- « l'agent a joue la première carte de sa main (i.e. en position 0) puis pioche », notée « j + p » ;
- « l'agent a annonce à l'agent b qu'il a une carte de valeur 1 en position 0 », notée « ann. ».

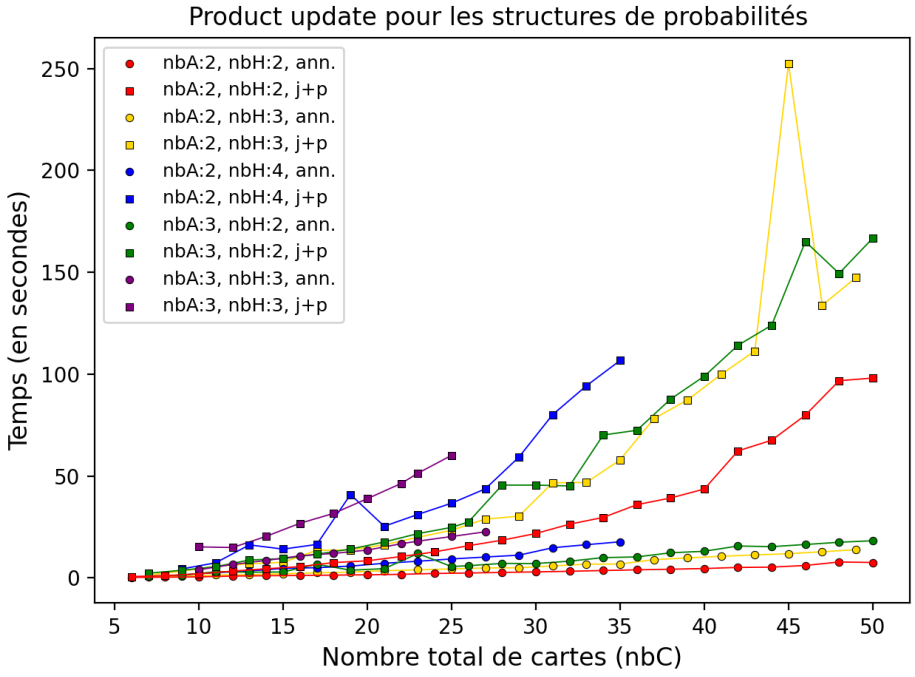
Comme précédemment, nous avons fait l'expérience sur tous les types de structures à notre disposition, et nous présentons les résultats sur la figure 8.8a. Premièrement, nous pouvons voir que les temps de *product update* sont très différents entre les deux types d'action. Les temps pour les actions d'annonces sont linéaires dans les cas symboliques (●, ● et ●) et ne dépassent pas les 8 secondes au bout de 50 cartes. Ce n'est pas le cas pour la structure explicite (●) qui effectue ce *product update* en 7.5s pour seulement 18 cartes.

Pour l'action « jouer », les résultats sur les structures symboliques sont très similaires (■, ■, ■) et sont très bons comparés à la structure explicite (■). Les résultats pour les structures de croyances et structures de probabilités sont censés être identiques puisque les définitions du *product update* sont identiques et que les lois de probabilités Π sont ici identiques aux lois d'observations Ω. Ce n'est pas le cas sur ces courbes et nous ne saurions l'expliquer.

Enfin, nous voyons que la structure normalisée (● et ■) présente des courbes beaucoup moins intéressantes (quoique meilleurs que l'explicite en rouge). Le temps de *product update* pour l'annonce a l'allure d'une exponentielle.



(a) Temps de calcul pour le *product update* avec nbA = 2 et nbH = 2 en fonction de nbC.



(b) Temps de calcul pour le *product update* pour la structure de probabilités en fonction de nbC, variant avec nbA et nbH.

FIGURE 8.8 – Comparaison des temps de calculs du *product update*.

Hormis pour l'action d'annonce de la version normalisée, nous constatons que l'action « j+p » prend plus de temps que l'annonce. En effet, l'action d'annonce symbolique n'est qu'une action simple qui n'a pas besoin de variables d'étiquetage pour différencier les événements du modèle d'événements. L'action « j+p », quant à elle, en a besoin car elle agglomère différentes actions atomiques qui correspondent à toutes les cartes différentes potentiellement piochées (voir sec. 7.3.3). Ainsi, il faut rajouter des variables aux structures Ω mises à jour. Par ailleurs, les postconditions de l'action « piocher » sont plus complexes et leur application ajoute de nombreuses contraintes entre les variables des ADDs.

Pour finir, nous comparons sur le graphique 8.8b les différences de temps de calcul en fonction du nombre de cartes en main nbH et nbA pour la structure symbolique dénormalisée.

Nous pouvons voir que pour les annonces (tous les ●), les courbes sont toujours linéaires et les temps de calculs ne dépassent pas 25 secondes, hormis pour $\mathsf{nbA} = 3$ et $\mathsf{nbH} = 3$, où les calculs se sont arrêtés car dépassant le timeout total de l'expérience (3000s pour le *model checking*). Dans tous les cas, nous voyons que les courbes ne sont pas exponentielles et que le *product update* est envisageable sur Hanabi représenté de manière symbolique.

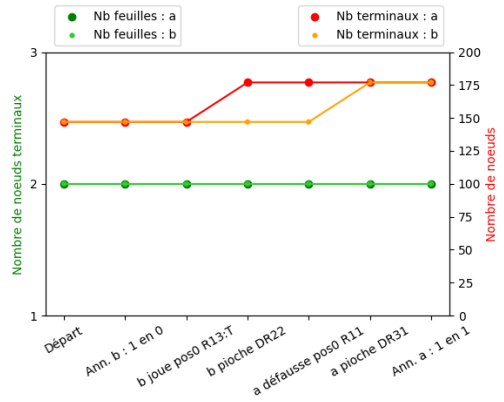
Comme précédemment avec le *model checking*, l'ajout de nombreuses variables dans la modélisation du jeu pour passer de 2 à 3 agents influe beaucoup sur la taille des structures et donc sur leurs manipulations, néanmoins nous pouvons dire que le *product update* passe bien à l'échelle en modifiant les paramètres du jeu.

8.6 Expérimentations avec plusieurs product updates

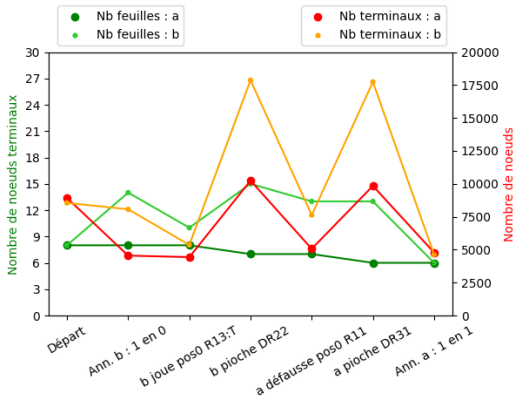
Nous avons effectué d'autres expérimentations, mêlant *product updates* et *model checking*. Le but ici est de voir l'évolution des temps de calculs des *model checking* et *product updates* dans une situation réelle, mais aussi l'évolution des tailles des ADDs représentant les PBFs. Nous souhaitons ainsi voir si maintenir l'état de connaissance est possible au fil du temps, ou si cet état explose en taille. En effet, la question se pose car que chaque *product update* pourrait multiplier les mondes possibles et augmenter en conséquence la taille des ADDs, ce qui conduirait, au fil du temps, à des calculs de plus en plus longs.

Nous utilisons toujours une structure épistémique et nous souhaitons lui y appliquer différentes actions à la suite. La suite d'actions est la suivante :

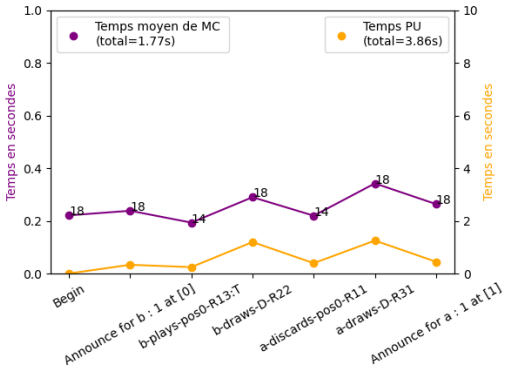
- Annoncer à l'agent b qu'il a une carte de valeur 1 en position 0 ;
- L'agent b joue sa carte en position 0 ;



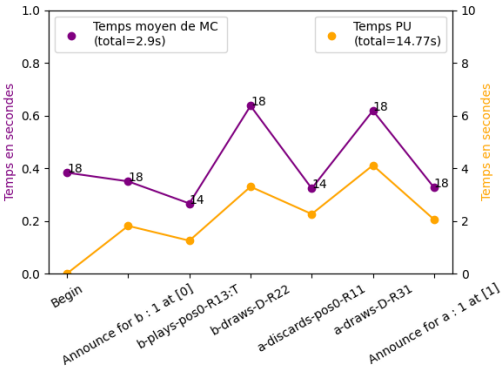
(a) Tailles des ADDs dénormalisés.



(b) Tailles des ADDs normalisés.



(c) Temps de calcul sur ADDs dénormalisés.



(d) Temps de calcul sur ADDs normalisés.

FIGURE 8.9 – Comparaison des tailles et temps de calculs sur ADDs normalisés/-dénormalisés pour 10 cartes, 2 agents et 2 cartes en main.

- L'agent b pioche ensuite une carte (l'événement pointé fait piocher la carte rouge de valeur 2 et d'id 2) ;
- L'agent a défausse sa carte en position 0 ;
- L'agent a pioche ensuite une carte (l'événement pointé fait piocher la carte rouge de valeur 3 et d'id 1) ;
- Annoncer à l'agent a qu'il a une carte de valeur 1 en position 1.

À chaque étape, nous appliquons un certain nombre d'opérations de *model checking* (diverses formules probabilistes) et nous étudions la taille des ADDs (i.e. le nombre de nœuds total dans l'arbre et le nombre de feuilles). Nous faisons cette expérience dans deux cas différents : pour 10 et 20 cartes dans le jeu, respectivement sur les figures 8.9 et 8.10, de manière normalisée (à droite) et de manière

dénormalisée (à gauche). Les graphiques a. et b. sont les graphiques représentant les tailles des ADDs et les graphiques c. et d. représentent les courbes des temps de calcul pour le *model checking* et le *product update*. Pour ce qui est du *model checking*, les nombres apparaissant à côté de chaque point représentent le nombre de formules qui ont été vérifiées. Attention, les échelles des axes des ordonnées varient quand il s'agit de la comparaison des tailles des structures (graphiques du haut) ; nous avons pris soin qu'elles aient un facteur en commun afin que les deux graphiques soient lisibles simultanément, sans que l'un des deux présente des courbes écrasées sur l'axe des abscisses.

- pour passer de 8.9a à 8.9b :
  - l'axe des ordonnées de gauche passe de 3 à 30 ($\times 10$) ;
  - l'axe des ordonnées de droite passe de 200 à 20000 ($\times 100$) ;
- pour passer de 8.10a à 8.10b :
  - l'axe des ordonnées de gauche passe de 3 à 150 ($\times 50$) ;
  - l'axe des ordonnées de droite passe de 700 à 700000 ($\times 1000$) ;

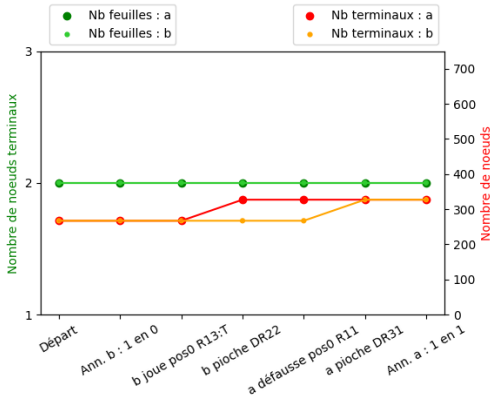
Le premier constat est simple : la taille des ADDs dénormalisés est nettement inférieure à celle des ADDs normalisés. Comme prévu, le nombre de feuilles de ces ADDs dénormalisés est de 2, contrairement au nombre de feuilles des ADDs dénormalisés qui varie de manière conséquente : entre 6 et 15 pour 10 cartes (fig. 8.9b) et 18 et 126 pour 20 cartes (fig. 8.10b).

Pour être plus précis, dans le cadre du nombre de cartes égal à 20, nous avons un ADD qui atteint au maximum 601170 nœuds de manière dénormalisée, alors que de manière normalisée, le maximum est 267 nœuds. Il y a un facteur supérieur à 2000 entre les deux valeurs.

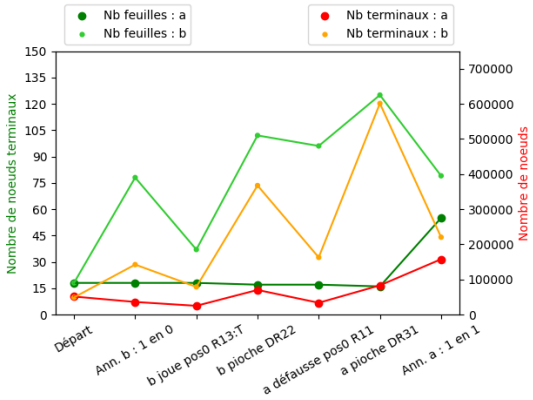
Quant aux temps de calcul, nous pouvons voir que plus la taille des ADDs est conséquente, plus ces temps sont importants. La corrélation semble directe. La différence de ces temps est visible bien que raisonnable (grossièrement deux fois plus de temps environ) pour 10 agents, mais n'est plus du tout discutable pour 20 agents. En effet, alors que le temps pour faire l'avant-dernier *product update* est de 4 secondes pour la version dénormalisée, la version normalisée prend 40 secondes (10 fois moins). Pour ce qui est du *model checking* : le *model checking* lors que l'avant-dernière étape prend en moyenne 33 secondes alors que pour la version normalisée il prend à peine 1 seconde (33 fois moins).

Nous pouvons par ailleurs commenter l'évolution des tailles des ADDs en fonction des actions réalisées dans le jeu. En effet, nous pouvons voir que la taille des ADDs est stable au fil du temps dans la version dénormalisée.

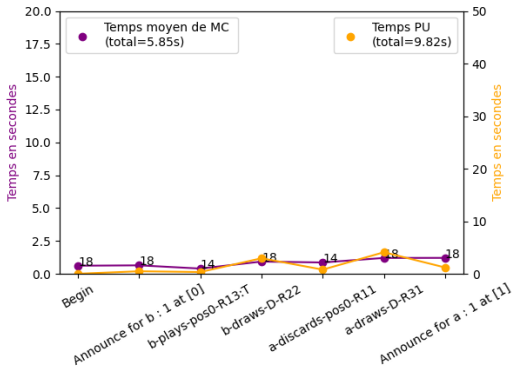
Dans la version normalisée, des types d'actions différentes mènent à des résultats différents. Une action d'annonce, qui peut se résumer brièvement à un filtre



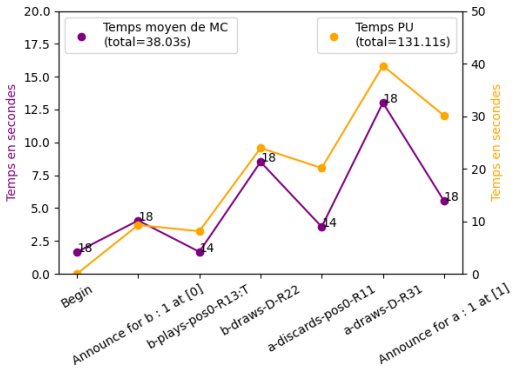
(a) Tailles des ADDs dénormalisés.



(b) Tailles des ADDs normalisés.



(c) Temps de calcul sur ADDs dénormalisés.



(d) Temps de calcul sur ADDs normalisés.

FIGURE 8.10 – Comparaison des tailles et temps de calculs sur ADDs normalisés/-dénormalisés pour 20 cartes, 2 agents et 2 cartes en main.

sur l'ensemble des mondes va diminuer la taille des ADDs. Au contraire, une action qui amène de l'incertitude dans les connaissances des agents va augmenter le nombre d'états possibles et de relations et donc aussi la taille des structures symboliques qui les représentent. Les actions de « jouer » ou de « défausser » une carte vont quant à elles aussi réduire la taille des ADDs pour l'agent qui fait l'action, car l'incertitude du joueur va diminuer ; pour l'autre joueur, c'est essentiellement l'application des postconditions qui va changer les structures, impactant peu la taille de celles-ci.

Ici, dans Hanabi, il n'y a qu'une seule action discriminante qui fait exploser l'espace d'états : « piocher ». Or, les règles imposent de faire des annonces, ou de jouer/défausser une carte régulièrement ; ainsi, les actions de pioche ne sont jamais enchaînées et l'espace d'états reste alors relativement stable.

Avec ces résultats, nous ne pouvons que constater qu'il est préférable de conserver des structures les plus petites possibles afin d'améliorer toutes les opérations sur les structures symboliques. C'est ce que permet une version dénormalisée des structures probabilisées.

Pour ce qui est du cas d'Hanabi, les structures sont a-consistantes, a-uniformes et les distributions de probabilités sont uniformes (voir sec. 6.4.2). Ainsi, la représentation dénormalisée permet aux PBFs d'avoir des valeurs booléennes et c'est pourquoi Π_a se comporte comme Ω_a. Les multiplications des lois de probabilités sont alors très efficaces, permettant des *product updates* rapides.

Bien qu'Hanabi soit un cas présentant les spécificités citeés ci-dessus, il n'en reste pas moins un exemple typique d'utilisation des connaissances épistémiques probabilistes.

D'autres cas d'applications s'avéreraient peut-être plus complexes à modéliser et maintenir symboliquement, ou au contraire, plus simples. Beaucoup de paramètres rentrent en ligne de compte : le nombre de variables augmentant la taille du problème, les dépendences des variables entre elles augmentant les contraintes dans le diagramme de décision, la caractère symétrique des lois d'observations/de probabilités des agents rendant le *model checking* potentiellement plus long, ou encore les caractéristiques probabilistes du problème en question, telles que l'uniformité des distributions de probabilités. Il est ainsi complexe de prévoir le comportement global du *framework* sur une situation donnée.

Néanmoins, par nos expériences, nous avons pu voir que dans un cas réel d'utilisation, même combinatoire, il est vraiment possible de maintenir et de modifier les structures de probabilités, notamment grâce aux définitions dénormalisées.

Conclusion

et perspectives

Nos travaux portent sur une représentation symbolique pour la logique épistémique dynamique probabiliste (PDEL). Cette logique modale est particulièrement bien adaptée pour modéliser des problèmes nécessitant de manipuler des informations épistémiques et probabilistes, et notamment les connaissances d'ordre supérieur des agents. Malheureusement, les structures de Kripke, qui représentent les états de connaissance dans PDEL, sont mathématiquement soumises à une explosion combinatoire du nombre d'états, ce qui rend PDEL inutilisable en pratique. Pour pallier ce problème, nous nous inspirons de la littérature de la représentation symbolique en général, en utilisant les formules propositionnelles pour encoder des ensembles de valuations, représentant des mondes, ou encore des relations entre ces mondes. Cette représentation modélise ces ensembles via des formules propositionnelles stockées en mémoire par une structure de données bien connue dans la littérature du *model checking* symbolique : le BDD (*Binary Decision Diagram*). Les algorithmes polynomiaux de manipulation des BDDs font de cette structure de données un classique en terme d'utilisation et d'efficacité.

Notre représentation symbolique de PDEL s'appuie sur SMCDEL, un *framework* permettant de faire du *model checking* symbolique sur DEL, la logique épistémique dynamique, dont PDEL est l'extension probabiliste. Avant d'introduire notre *framework*, nous avons généralisé PDEL en dénormalisant les probabilités, c'est-à-dire en remplaçant les lois de probabilité discrètes par ce que nous nommons des loteries qui n'ont pas la contrainte de sommer à 1. Nous avons également exhibé une variante du *modèle conditionnel*, que nous nommons *modèle observationnel* et que nous avons directement défini dans sa version dénormalisée. Celui-ci est similaire à un modèle d'événements de DEL auquel est ajouté des lois de probabilités.

Une des contributions de la thèse est d'avoir comparé les modèles conditionnels et les modèles observationnels. En effet, les modèles observationnels ne faisant pas partie de la littérature à proprement dit, aucune comparaison n'avait été faite.

Nous avons donc montré que le *model checking* était équivalent sur l'un ou sur l'autre, en passant d'un modèle à l'autre. Nous avons montré l'équivalence des définitions dénormalisées et des définitions initiales en définissant une nouvelle sémantique, qui consiste à appliquer les *product updates* de manière dénormalisée et à effectuer une normalisation des loteries seulement lors des *model checking* afin d'obtenir de nouveau des lois de probabilité discrètes.

Il y a deux motivations aux modifications des définitions. Tout d'abord, les définitions des représentations symboliques probabilistes ressemblent ainsi davantage aux définitions de SMCDEL, ce qui simplifie le parallèle entre les deux *frameworks*. Néanmoins, la motivation principale est l'aspect calculatoire. En effet, comme nous avons pu le montrer dans la partie expérimentale, le gain en temps est vraiment conséquent lorsque l'on en manipule des loteries et non des lois de probabilité discrètes.

Une contribution de nos travaux est d'avoir proposé une dénormalisation globale de toutes les définitions de PDEL (en comptant la modification des formules conditionnelles), tout en conservant l'équivalence des *model checking* et des *product update* avec les modèles initiaux.

Armés d'une structure de Kripke avec loteries et de ces deux types d'événements définis eux-mêmes sur des loteries, nous avons pu présenter notre version symbolique des structures de Kripke probabilistes grâce à des fonctions pseudo-booléennes (PBFs) qui encodent les loteries. Comme nous l'avons présenté, les PBFs se représentent très bien grâce à des structures de données adaptées : les ADDs (*Algebraic Decision Diagrams*). Nous avons par la même occasion présenté les fonctions spécifiques aux PBFs comme la marginalisation ou la coupe, qui nécessitent des algorithmes dédiés pour leur mise en pratique sur des ADDs.

La contribution théorique principale de nos travaux est d'avoir montré que cette représentation symbolique probabiliste était équivalente à la représentation explicite des structures de Kripke probabilistes grâce aux transformations symb et expl. Plus précisément, nous avons montré que le *framework* explicite normalisé est équivalent au *framework* explicite dénormalisé, qui est lui-même équivalent au *framework* symbolique dénormalisé.

Enfin, nous avons écrit un programme Python qui implémente les structures de données que nous utilisons (BDDs et ADDs) mais aussi toutes les définitions que nous avons présentées (SMCDEL et sa version probabiliste). Ce programme nous a permis une mise en pratique réelle en modélisant le jeu de cartes Hanabi, dont le nombre de variables propositionnelles est combinatoire en fonction du nombre de cartes en jeu et du nombre d'agents. Nous avons donc testé le passage à l'échelle de la représentation symbolique et de la représentation explicite en fonction des paramètres du jeu en comparant les temps de création des structures, les temps de *model checking*, et les temps de *product update*. Nous avons aussi illustré l'intérêt de

l'utilisation des loteries à la place de distributions de probabilités en comparant ces mêmes temps, mais aussi la taille des ADDs représentant les structures. Nous avons ainsi mis en exergue la possibilité de représenter les connaissances épistémiques et probabilistes avec un exemple concret et combinatoire, notamment grâce à la dénormalisation des définitions.

Maintenant que nous avons un moyen de représenter la logique épistémique dynamique probabiliste grâce à des structures symboliques, faisons un point sur les perspectives possibles.

Premièrement, on peut se demander si nos résultats expérimentaux sont représentatifs de l'efficacité qu'on peut attendre d'une implémentation symbolique de PDEL. D'une part, il serait intéressant d'expérimenter sur d'autres problèmes réalistes, avec des caractéristiques diverses –notamment utilisant des modèles conditionnels, et des *frames* ne vérifiant pas la a-consistance, la a-uniformité, ou l'uniformité des probabilités. D'autre part, il reste possible d'aller plus loin dans l'optimisation du programme.

En effet, l'ajout d'un réordonnancement automatique des variables dans les diagrammes de décisions pourrait être un atout, bien que cela nécessiterait de choisir une bonne politique de réordonnancement et donc des expérimentations supplémentaires.

Nous pourrions aussi envisager des optimisations dédiées au problèmes modélisés. En effet, les relations R des structures de Kripke peuvent présenter des symétries d'un agent à l'autre et il en va de même pour les probabilités. Ainsi, nous pourrions envisager des méthodes de calculs spécifiques afin, typiquement, de construire R_b à partir de R_a ou Π_b à partir de Π_a. Cela pourrait se faire grâce à des techniques de renommage des variables par exemple.

Par ailleurs, nous utilisons deux algorithmes de manière successive lors du *model checking* : la coupe et la marginalisation. Il semble possible d'écrire un algorithme spécifique appliquant les deux simultanément, ce qui est susceptible d'améliorer considérablement les temps de calcul du *model checking*.

Par ailleurs, nous avons vu qu'il est possible de représenter les structures de Kripke S5 symboliquement grâce à des variables d'observations. Nous nous interrogeons sur la possibilité de faire de même avec des probabilités. Il y a en effet beaucoup de symétries et il n'est peut-être pas nécessaire de représenter toutes les distributions de probabilités, mais seulement des ensembles de mondes liés à une valeur.

Deuxièmement, nous pourrions étendre notre représentation symbolique.

En effet, nous n'avons pas travaillé sur la représentation de la connaissance commune probabiliste des agents. Bien que cette notion ne soit pas présente dans

les définitions de PDEL, elle est présente dans DEL et est utilisée et implémentée par Gattinger. Cet opérateur serait bien utile, la littérature proposant plusieurs problèmes de planification qui nécessitent ce concept [CHM+16, BA11].

Ensuite, les concepts de représentation symbolique probabiliste pourraient être utilisés dans d'autres cas que PDEL. En effet, il existe d'autres types de structures de Kripke ; nous pensons notamment aux structures épistémiques décrites par Minghui Ma et Hans van Ditmarsch dans la logique nommée *Dynamic Graded Epistemic Logic* [MvD19]. Il s'agit de pondérer les arcs des structures de Kripke. Dans ce type de cadre, il semble envisageable de pouvoir définir une représenttation symbolique basée sur des PBFs, comme nous l'avons fait pour PDEL.

Dans un autre registre, DEL est actuellement étudié comme outil de résolution du problème de planification épistémique [BA11]. Ainsi, l'utilisation de SMCDEL pourrait permettre la mise en pratique de problèmes de planification épistémique avec DEL. Dans cette optique, le *framework* proposé dans cette thèse pourrait servir à la résolution en pratique de problèmes de *planification épistémique probabiliste*, dont l'incertitude serait quantifiée par des probabilités, ce qui permettrait de définir plus finement ce qu'on attend d'une stratégie pour qu'elle constitue une solution.

Gattinger dans la même lignée a envisagé des outils pour pouvoir utiliser la planification épistémique [Gat20] grâce à ses structures symboliques dans le cas S5 et grâce aux programmes mentaux définis par Charrier et Schwarzentruber [CS17]. Nous pourrions envisager le même type d'approches en utilisant des distributions de probabilités sur les mondes possibles, et donnant ainsi des pistes de résolution pour le problème de planification épistémique probabiliste.

Glossaire

$[\mathcal{E}, e]$ Opérateur d'application d'événement $\mathcal{E}, e$. 27

$\mathfrak{A}$ une liste d'agents. 7, 20

$\mathfrak{F}$ Symbole d'une structure de Kripke symbolique. 38

χ Symbole d'un modèle d'événement symbolique. 48

B Opérateur modal de croyance pour un agent. 20

$\Theta^{+}_{\mathfrak{F}}$ Symbole de représentation d'une précondition conditionnelle symbolique. 152

$\mathcal{E}$ modèle d'événement. 25

$\mathcal{F}$ Frame, une structure relationnelle. 14

$\hat{\mathrm{K}}$ Opérateur modal dual de la connaissance pour un agent. 21

K Opérateur modal de la connaissance pour un agent. 20

Kw Opérateur modal de la connaissance de la valeur d'une formule pour un agent. 21

λ Symbole d'une fonction d'étiquetage. 36

$\mathcal{L}$ un langage. 7

θ Symbole de représentation d'une loi d'ensemble symbolique. 35, 38

θ_{-} Loi de modification des atomes propositionnels modifiés par un modificateur de croyances et de probabilités. 54

θ^{+} Symbole de représentation d'une loi d'ensemble symbolique pour un modèle d'événements. 48

θ^{pre} Symbole de représentation des probabilités conditionnelles d'un modèle d'événement. 149

$\mathcal{L}_{\mathrm{DEL}}$ langage de la logique épistémique dynamique. 26

$\mathcal{L}_{\mathrm{DELCK}}$ langage de la logique épistémique dynamique avec connaissance commune. 26

$\mathcal{L}_{\mathrm{EL}}$ langage pour la logique épistémique. 13

$\mathcal{L}_{\mathrm{S5CK}}$ langage pour la logique épistémique avec la connaissance commune. 22

$\mathcal{L}_{\text{PAL}}$ langage pour la Public Announcement Logic. 23

$\mathcal{L}_{\text{PDEL}}$ langage pour la logique épistémique probabiliste dynamique . 102

$\mathcal{L}_{\text{PEL}}$ langage pour la logique épistémique probabiliste . 95

$\mathcal{L}_{\text{prop}}$ langage pour la logique propositionnelle. 11

$\mathcal{L}_{\text{S5}}$ langage pour la logique modale épistémique S5. 21

$\text{N}()$ Fonction de normalisation d'une fonction. 104

$\text{N}_\theta()$ Fonction de normalisation d'une loi de probababilités via une loi des mondes θ. 131

Ω Symbole de représentation d'une loi de relations symboliques. 38

Ω^+ Symbole de représentation d'une loi de relations symboliques pour un modèle d'événement. 48

Π Symbole de représentation d'une loi de probabilité symboliques. 128

Π^+ Symbole de représentation d'une loi de probabilité symboliques pour un modèle d'événement. 141

PS une liste d'atomes propositionnels $\{p,\ q,\ x,\ y\}$, qui peuvent être évalués à vrai ou faux. 7

μ symbole de distribution de probabilité dans une Frame. 94

$\overline{\mathbb{Q}}$ Ensemble de nombres rationnels incluant les infinis. 76

R une relation d'accessibilité. 14

V vocabulaire. 7

$V_{\_}$ Vocabulaire modifié d'un modificateur de croyances et de probabilités. 54

V^+ vocabulaire d'un modèle d'événements. 48

post postcondition. 25

pre precondition. 25

$\vdash$ symbole de démonstrabilité. 13

$\models$ symbole de validité. 13

Acronymes

AADD Affine Algebraic Decision Diagram. 78
ADD Algebraic Decision Diagram. 77
AP atomes propositionnels. 7, 220

BDD Binary Decision Diagram. 65
BNF Backus-Naur Form. 12, 13

DAG Directed Acyclic Graph. 66
DD Decision Diagram. 66
DEL Dynamic Epistemic Logic. 24

MC problème de la vérification de modèle (Model Checking). 22

PAL Public Announcement Logic. 23
PBF Pseudo-boolean function. 76
PDEL Probabilistic Dynamic Epistemic Logic. 91, 93

RBDD Reduced Binary Decision Diagram. 67
ROBDD Reduced Ordered Binary Decision Diagram. 68

SLDD Semiring Labeled Decision Diagram. 78
SMCDEL Symbolic Model Checking for Dynamic Epistemic Logic. 32
ssi Si et seulement si. 12

Bibliographie

[AA07] Michael Anderson and Susan Leigh Anderson. Machine ethics : Creating an ethical intelligent agent. *AI Magazine*, 28(4) :15, Dec. 2007.

[BA11] Thomas Bolander and Mikkel Birkegaard Andersen. Epistemic planning for single and multi-agent systems. *Journal of Applied Non-Classical Logics*, 21(1) :9–34, 2011.

[BCLF85] Simon Baron-Cohen, Alan M. Leslie, and Uta Frith. Does the autistic child have a "theory of mind" ? *Cognition*, 21(1) :37–46, 1985.

[BFC+20] Nolan Bard, Jakob N. Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H. Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, Iain Dunning, Shibl Mourad, Hugo Larochelle, Marc G. Bellemare, and Michael Bowling. The hanabi challenge : A new frontier for AI research. *Artif. Intell.*, 280 :103216, 2020.

[BFG+97] R. Iris Bahar, Erica A. Frohm, Charles M. Gaona, Gary D. Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Algebraic decision diagrams and their applications. *Formal Methods in System Design*, 10(2/3) :171–206, 1997.

[BFM10] Meghyn Bienvenu, Hélène Fargier, and Pierre Marquis. Knowledge compilation in the modal logic S5. In Maria Fox and David Poole, editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. AAAI Press, 2010.

[BHT13] Philippe Balbiani, Andreas Herzig, and Nicolas Troquard. Dynamic logic of propositional assignments : a well-behaved variant of PDL. In *28th Annual IEEE/ACM Symposium on Logic in Computer Science (LICS 2013)*, 28th Annual IEEE/ACM Symposium on Logic in Computer Science (LICS 2013), pages 143–152, New Orleans, LA, United States, June 2013.

[BMS98] Alexandru Baltag, Lawrence S. Moss, and Slawomir Solecki. *The Logic of Public Announcements and Common Knowledge and Private Suspicions*. 1998.

[BMV02] Patrick Blackburn, de Rijke Maarten, and Yde Venema. *Modal Logic*. Cambridge University Press, 2002.

[BRB90] Karl S. Brace, Richard L. Rudell, and Randal E. Bryant. Efficient implementation of a BDD package. pages 40–45, 1990.

[Bry86] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Computers*, 35(8) :677–691, 1986.

[Bue17] Thiago Pereira Bueno. pyddlib, a python3 library for manipulating decision diagrams : `https://github.com/thiagopbueno/pyddlib/`, 2017.

[CDSD+15] Christopher Cox, Jessican De Silva, Philip Deorsey, Franklin H. J. Kenter, Troy Retter, and Josh Tobin. How to make the perfect fireworks display : Two strategies for hanabi. *Mathematics Magazine*, 88(5) :323–336, 2015.

[CGNS19] Tristan Charrier, Sébastien Gamblin, Alexandre Niveau, and François Schwarzentruber. Hintikka's world : Scalable higher-order knowledge. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 6494–6496. ijcai.org, 2019.

[Cha18] Tristan Charrier. *Complexité thèorique du raisonnement en logique épistémique dynamique et étude d'une approche symbolique*. PhD thesis, Université de Rennes 1, 2018.

[CHhH02] Murray Campbell, A.Joseph Hoane, and Feng hsiung Hsu. Deep blue. *Artificial Intelligence*, 134(1) :57–83, 2002.

[CHL+16] Tristan Charrier, Andreas Herzig, Emiliano Lorini, Faustine Maffre, and François Schwarzentruber. Building epistemic logic from observations and public announcements. In Chitta Baral, James P. Delgrande, and Frank Wolter, editors, *Principles of Knowledge Representation and Reasoning : Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016.*, pages 268–277. AAAI Press, 2016.

[CHM+16] Martin Cooper, Andreas Herzig, Faustine Maffre, Frédéric Maris, and Pierre Régnier. Simple epistemic planning : generalised

gossiping. In *22nd European Conference on Artificial Intelligence (ECAI 2016)*, pages pp. 1563–1564, The Hague, Netherlands, August 2016.

[CS15] Tristan Charrier and François Schwarzentruber. Arbitrary public announcement logic with mental programs. In Gerhard Weiss, Pinar Yolum, Rafael H. Bordini, and Edith Elkind, editors, *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, pages 1471–1479. ACM, 2015.

[CS17] Tristan Charrier and François Schwarzentruber. A succinct language for dynamic epistemic logic. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pages 123–131, 2017.

[DJKV17] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is coming : A modern probabilistic model checker. In Rupak Majumdar and Viktor Kuncak, editors, *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II*, volume 10427 of *Lecture Notes in Computer Science*, pages 592–600. Springer, 2017.

[DK14] Lorenz Demey and Barteld Kooi. *Logic and probabilistic update*, volume 5 of *Oustanding Contributions to Logic*, pages 381–404. Springer, 2014.

[DM02] Adrian Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, pages 229–264, 2002.

[Dra13] Chris Drake. PyEDA. `https://pyeda.readthedocs.io/en/latest/bdd.html`, 2013. [Online ; accessed 30-september-2022].

[FH94] Ronald Fagin and Joseph Y. Halpern. Reasoning about knowledge and probability. *J. ACM*, 41(2) :340–367, 1994.

[FHM88] R. Fagin, J.Y. Halpern, and N. Megiddo. A logic for reasoning about probabilities. In *[1988] Proceedings. Third Annual Symposium on Logic in Computer Science*, pages 410–421, 1988.

[FHMV95] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. The MIT PRess, Cambdrige, Massachusetts, 1995.

[FMNS14] Hélène Fargier, Pierre Marquis, Alexandre Niveau, and Nicolas Schmidt. A knowledge compilation map for ordered real-valued decision diagrams. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 1049–1055. AAAI Press, 2014.

[FMS13a] Hélène Fargier, Pierre Marquis, and Nicolas Schmidt. Semiring labelled decision diagrams, revisited : Canonicity and spatial efficiency issues. In Francesca Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 884–890. IJCAI/AAAI, 2013.

[FMS13b] Hélène Fargier, Pierre Marquis, and Nicolas Schmidt. Semiring labelled decision diagrams, revisited : Canonicity and spatial efficiency issues. In Francesca Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 884–890. IJCAI/AAAI, 2013.

[FMS14] Hélène Fargier, Pierre Marquis, and Nicolas Schmidt. Compacité pratique des diagrammes de décision valués. Normalisation, heuristiques et expérimentations. *Revue des Sciences et Technologies de l'Information - Série RIA : Revue d'Intelligence Artificielle*, 28(5) :571–592, 2014.

[FvD08] Tim French and Hans P. van Ditmarsch. Undecidability for arbitrary public announcement logic. In Carlos Areces and Robert Goldblatt, editors, *Advances in Modal Logic 7, papers from the seventh conference on "Advances in Modal Logic," held in Nancy, France, 9-12 September 2008*, pages 23–42. College Publications, 2008.

[Gam22] Sébastien Gamblin. Github pour 'symbolic model checking for probabilistic dynamic epistemic logic' : `https://git.unicaen.fr/sebastien.gamblin/symbolic-representation-for-pdel`, 2022.

[Gat18a] Malvin Gattinger. Hascacbdd : https://github.com/m4lvin/HasCacBDD, 2018.

[Gat18b] Malvin Gattinger. *New directions in Model Checking Dynamic Epistemic Logic*. PhD thesis, Universiteit van Amsterdam, 2018.

[Gat18c] Malvin Gattinger. Smcdel code : https://github.com/jrclogic/SMCDEL, 2018.

[Gat18d] Malvin Gattinger. Smcdel web : https://w4eg.de/malvin/illc/smcdelweb/index.html, 2018.

[Gat19] Malvin Gattinger. Towards symbolic factual change in DEL. *CoRR*, abs/1912.10717, 2019.

[Gat20] Malvin Gattinger. Towards symbolic and succinct perspective shifts, 2020.

[GNB22] Sébastien Gamblin, Alexandre Niveau, and Maroua Bouzid. A symbolic representation for probabilistic dynamic epistemic logic. In Piotr Faliszewski, Viviana Mascardi, Catherine Pelachaud, and Matthew E. Taylor, editors, *21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022, Auckland, New Zealand, May 9-13, 2022*, pages 445–453. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2022.

[Hin64] Jaakko Hintikka. Knowledge and belief. an introduction to the logic of the two notions. *Journal of Symbolic Logic*, 29(3), 1964.

[HM92] Joseph Y. Halpern and Yoram Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artif. Intell.*, 54(2) :319–379, 1992.

[HM17] Andreas Herzig and Faustine Maffre. How to share knowledge by gossiping. *AI Communications*, 30(1) :1–17, March 2017. https ://content.iospress.com/articles/ai-communications/aic723.

[joh] johnyf. Binding python de CUDD : https://github.com/tulip-control/dd. https://github.com/tulip-control/dd. [Online ; accessed 30-september-2022].

[KdBCD14] Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic sentential decision diagrams. In Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter, editors, *Principles of Knowledge Representation and Reasoning : Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*. AAAI Press, 2014.

[KNP02] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM : probabilistic symbolic model checker. In Tony Field, Peter G. Harrison, Jeremy T. Bradley, and Uli Harder, editors, *Computer Performance Evaluation, Modelling Techniques and Tools 12th International Conference, TOOLS 2002, London, UK, April 14-17, 2002, Proceedings*, volume 2324 of *Lecture Notes in Computer Science*, pages 200–204. Springer, 2002.

[Koo03] Barteld P. Kooi. Probabilistic dynamic epistemic logic. *Journal of Logic, Language and Information*, 12(4) :381–408, 2003.

[KvB04] Barteld Kooi and Johan van Benthem. Reduction axioms for epistemic actions. 2004.

[KZH+09] Joost-Pieter Katoen, Ivan S. Zapreev, Ernst Moritz Hahn, Holger Hermanns, and David N. Jansen. The ins and outs of the probabilistic model checker MRMC. In *QEST 2009, Sixth International Conference on the Quantitative Evaluation of Systems, Budapest, Hungary, 13-16 September 2009*, pages 167–176. IEEE Computer Society, 2009.

[LR06] Alessio Lomuscio and Franco Raimondi. MCMAS : A model checker for multi-agent systems. In Holger Hermanns and Jens Palsberg, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 12th International Conference, TACAS 2006 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006, Vienna, Austria, March 25 - April 2, 2006, Proceedings*, volume 3920 of *Lecture Notes in Computer Science*, pages 450–454. Springer, 2006.

[LSSC08] Xiangyu Luo, Kaile Su, Abdul Sattar, and Yan Chen. Solving sum and product riddle via bdd-based model checking. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and International Conference on Intelligent Agent Technology - Workshops, 9-12 December 2008, Sydney, NSW, Australia*, pages 630–633. IEEE Computer Society, 2008.

[Lut06] Carsten Lutz. Complexity and succinctness of public announcement logic. In Hideyuki Nakashima, Michael P. Wellman, Gerhard Weiss, and Peter Stone, editors, *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*, pages 137–143. ACM, 2006.

[Maf16] Faustine Maffre. *Le bonheur est dans l'ignorance : logiques épistémiques dynamiques basées sur l'observabilité et leurs applications.* PhD thesis, Université de Toulouse, 2016.

[MPP14] Pierre Marquis, Odile Papini, and Henri Prade, editors. *Représentation des connaissances et formalisation des raisonnements*, volume 1 of *Panorama de l'Intelligence Artificielle.* Cépaduès, http ://www.cepadues.com, 2014.

[MvD19] Minghui Ma and Hans van Ditmarsch. Dynamic graded epistemic logic. *Rev. Symb. Log.*, 12(4) :663–684, 2019.

[MvdH95] J.-J. Ch. Meyer and W van der Hoek. *Epistemic Logic for Computer Science and Artificial Intelligence*, volume 41. 1995.

[NZ16] Alexandre Niveau and Bruno Zanuttini. Efficient representations for the modal logic S5. pages 1223–1229, 2016.

[Sch02] Thomas Schneider. *Complexity of Modal Logics.* 2002.

[Sch15] Nicolas Schmidt. *Compilation de préférences, application à la configuration de produit.* PhD thesis, Université d'Artois, 2015.

[Sch18] François Schwarzentruber. Hintikka's world : Agents with higher-order knowledge. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 5859–5861. ijcai.org, 2018.

[Sch22] Nicolas Schmidt. Semiring or affin labelled décision diagramms' : `https://www.irit.fr/~Helene.Fargier/BR4CP/CompilateurSALADD.html`, 2022.

[SHM+16] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot,

Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nat.*, 529(7587) :484–489, 2016.

[SM05] Scott Sanner and David A. McAllester. Affine algebraic decision diagrams (aadds) and their application to structured probabilistic inference. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 1384–1390. Professional Book Center, 2005.

[Som18] Fabio Somenzi. Colorado University Decision Diagram package . `https://github.com/johnyf/cudd`, 2018. [Online ; accessed 23-march-2022].

[SSL07a] Kaile Su, Abdul Sattar, and Xiangyu Luo. Model checking temporal logics of knowledge via obdds. *Comput. J.*, 50(4) :403–420, 2007.

[SSL07b] Kaile Su, Abdul Sattar, and Xiangyu Luo. Model checking temporal logics of knowledge via obdds. *Comput. J.*, 50(4) :403–420, 2007.

[TP97] Paul Tafertshofer and Massoud Pedram. Factored edge-valued binary decision diagrams. *Formal Methods in System Design*, 10(2/3) :243–270, 1997.

[Ult21] UltraBoardGames. Hanabi rules. `https://www.ultraboardgames.com/hanabi/game-rules.php`, 2021. accessed 2021-01-16.

[vBGK06] Johan van Benthem, Jelle Gerbrandy, and Barteld P. Kooi. Dynamic update with probabilities. Prepublication/extended version PP-2006-21 (March 15th, 2006), ILLC (Institute for Logic, Language and Computation), University of Amsterdam, 2006.

[vBGK09] Johan van Benthem, Jelle Gerbrandy, and Barteld P. Kooi. Dynamic update with probabilities. *Studia Logica*, 93(1) :67–96, 2009.

[vBvEGS15] Johan van Benthem, Jan Van van Eijck, Malvin Gattinger, and Kaile Su. Symbolic Model Checking for Dynamic Epistemic Logic. In Wiebe van der Hoek, H. Holliday Wesley, and Wang Wen-fang, editors, *Logic, Rationality, and Interaction ; 5th International Workshop, LORI 2015*, number 9394 in LNCS, pages 366–378, Taipei, Taiwan, 2015. Springer.

[VBvEGS18] Johan Van Benthem, Jan van Eijck, Malvin Gattinger, and Kaile Su. Symbolic model checking for Dynamic Epistemic Logic — S5 and beyond. *Journal of Logic and Computation*, 28(2) :367–402, 11 2018.

[vDHK+17] Hans van Ditmarsch, Michael Ian Hartley, Barteld Kooi, Jonathan Welton, and Joseph B.W. Yeo. Cheryl's birthday. *Electronic Proceedings in Theoretical Computer Science*, 251 :1–9, jul 2017.

[vDHvdHK15] H. van Ditmarsch, J.Y. Halpern, W. van der Hoek, and B.P. Kooi. *Handbook of Epistemic Logic*. College Publications, 2015.

[vdMS04] Ron van der Meyden and Kaile Su. Symbolic model checking the knowledge of the dining cryptographers. In *17th IEEE Computer Security Foundations Workshop, (CSFW-17 2004), 28-30 June 2004, Pacific Grove, CA, USA*, page 280. IEEE Computer Society, 2004.

[vDvdHK07] Hans van Ditmarsch, Wiebe van der Hoek, and Barteld Kooi. *Dynamic Epistemic Logic*. Springer Publishing Company, Incorporated, 1st edition, 2007.

[vDvdHR13] Hans van Ditmarsch, Wiebe van der Hoek, and Ji Ruan. Connecting dynamic epistemic and temporal epistemic logics. *Log. J. IGPL*, 21(3) :380–403, 2013.

[vDvdHvdMR06] Hans P. van Ditmarsch, Wiebe van der Hoek, Ron van der Meyden, and Ji Ruan. Model checking russian cards. *Electron. Notes Theor. Comput. Sci.*, 149(2) :105–123, 2006.

[vE08] Jan van Eijck. Demo — a demo of epistemic modelling. 01 2008.

[vE14] Jan van Eijck. Demo - s5 technical report. 2014.

[vES14] Jan van Eijck and François Schwarzentruber. Epistemic probability logic simplified. In Rajeev Goré, Barteld P. Kooi, and Agi Kurucz, editors, *Advances in Modal Logic 10, invited and contributed papers from the tenth conference on "Advances in Modal Logic," held in Groningen, The Netherlands, August 5-8, 2014*, pages 158–177. College Publications, 2014.

[Wha07] John Whaley. Java BDD : `https://javabdd.sourceforge.net/index.html`. `https://javabdd.sourceforge.net/index.html`, 2007. [Online ; accessed 30-september-2022].

Annexe : règles du jeu d'Hanabi

In this appendix we introduce the official rules of Hanabi [Ult21]

Game Material

50 fireworks cards in five colors (red, yellow, green, blue, white) :

- 10 cards per color with the values 1, 1, 1, 2, 2, 3, 3, 4, 4, 5,
- 10 colorful fireworks cards with same values as above
- 8 Clock (Note) tokens (+ 1 spare),
- 3 Storm (Fuse) tokens.

Aim of the Game

Hanabi is a cooperative game, meaning all players play together as a team. The players have to play the fireworks cards sorted by colors and numbers. However, they cannot see their own hand cards, and so everyone needs the advice of their fellow players. The more cards the players play correctly, the more points they receive when the game ends. The Game The oldest player is appointed first player and sets the tokens in the play area. The eight Clock tokens are placed white-side-up. The three Storm tokens are placed lightning-side-down. Now the fireworks cards are shuffled. Depending on the number of players involved, each player receives the following hand :

- With 2 or 3 players : 5 cards in hand,
- With 4 or 5 players : 4 cards in hand.

Important : For the basic game, the colorful fireworks cards and the spare Clock token(s) are not needed. They only come in to use for the advanced game. Important : Unlike other card games, players may not see their own hand ! The players take their hand cards so that the back is facing the player. The fronts can only be seen by the other players. The remaining cards are placed face down in the draw pile in the middle of the table. The first player starts.

Game Play

Play proceeds clockwise. On a player's turn, they must perform exactly one of the following :

A. Give a hint or

B. Discard a card or

C. Play a card.

The player has to choose an action. A player may not pass ! Important : Players are not allowed to give hints or suggestions on other players' turns !

A. Give a hint

To give a hint one Clock token must be flipped from its white side to its black side. If there are no Clock tokens white-side-up then a player may not choose the Give a hint action. Now the player gives a teammate a hint. They have one of two options :

1. **Color Hint.** The player chooses a color and indicates to their teammate which of their hand cards match the chosen color by pointing at the cards. Important : The player must indicate all cards of that color in their teammate's hand ! Example : "You have two yellow cards, here and here." Indicating that a player has no cards of a particular color is allowed ! Example : "You have no blue cards."
2. **Value Hint.** The player chooses a number value and gives a teammate a hint in the exact same fashion as a Color Hint. Example : "You have a 5, here." Example : "You have no Twos."

B. Discard a card

To discard a card one Clock token must be flipped from its black side to its white side. If there are no Clock tokens black-side-up then a player may not choose the Discard a card action. Now the player discards one card from their hand (without looking at the fronts of their hand cards) and discards it face-up in the discard pile near the draw deck. The player then draws another card into their hand in the same fashion as their original hand cards, never looking at the front.

C. Play a card

By playing out cards the fireworks are created in the middle of the table. The player takes one card from their hand and places it face up in the middle of the table. Two things can happen :

1. **The card can be played correctly.** The player places the card face up so that it extends a current firework or starts a new firework.
2. **The card cannot be played correctly.** The gods are angry with this error and send a flash from the sky. The player turns a Storm tile lightning-side-up. The incorrect card is discarded to the discard pile near the draw deck.

In either case, the player then draws another card into their hand in the same fashion as their original hand cards, never looking at the front.

The Fireworks

The fireworks will be in the middle of the table and are designed in five different colors. For each color an ascending series with numerical values from 1 to 5 is formed. A firework must start with the number 1 and each card played to a firework must increment the previously played card by one. A firework may not contain more than one card of each value.

Bonus

When a player completes a firework by correctly playing a 5 card then the players receive a bonus. One Clock token is turned from black side to white side up. If all tokens are already white-side-up then no bonus is received. Play then passes to the next player (clockwise).

Ending the Game

The game can end in three ways :

1. The third Storm token is turned lightning-side-up. The gods deliver their wrath in the form of a storm that puts an end to the fireworks. The game ends immediately, and the players earn zero points.

2. The players complete all five fireworks correctly. The game ends immediately, and the players celebrate their spectacular victory with the maximum score of 25 points.

3. If a player draws the last card from the draw deck, the game is almost over. Each player—Including the player who drew the last card—gets one last turn.

Finally, the fireworks will be counted. For this, each firework earns the players a score equal to the highest value card in its color series. The quality of the fireworks display according to the rating scale of the "International Association of Pyrotechnics" is :

- 0–5 : Oh dear ! The crowd booed.
- 6–10 : Poor ! Hardly applaused.
- 11–15 : OK ! The viewers have seen better.
- 16–20 : Good ! The audience is pleased.
- 21–24 : Very good ! The audience is enthusiastic !
- 25 : Legendary ! The audience will never forget this show !

Important Notes and Tips

- Players may rearrange their hand cards and change their orientation to help themselves remember the information they received. Players may not ever look at the front of their own cards until they play them.
- The discard pile may always be searched for information.
- Hanabi is based on communication—and non-communication—between the players. If one inter- prets the rules strictly then players may not, except for the announcements of the current player, talk to each other. Ultimately, each group should decide by its own measure what communication is permitted. Play so that you have fun !

Représentation symbolique pour la logique épistémique dynamique probabiliste

Cette thèse porte sur l'étude de la mise en pratique de la logique épistémique dynamique probabiliste, permettant de représenter les connaissances des agents à propos des connaissances des autres agents en utilisant des probabilités. Pour cela, en nous appuyant sur l'état de l'art du model checking symbolique, nous proposons un encodage des structures de Kripke et structures d'événements de notre logique via des fonctions booléennes et fonction pseudo-booléennes. Nous avons implémenté nos définitions en utilisant des structures de données adaptées : les Algebraic Decision Diagrams, qui généralisent les classiques Binary Decision Diagrams en permettant de manipuler des probabilités. Grâce à cette implémentation, nous avons pu mener des expérimentations qui nous ont permis de montrer que notre représentation symbolique passe mieux à l'échelle que la version explicite sur l'exemple du jeu de cartes Hanabi.

Mots-clefs : Intelligence artificielle ; Représentation des connaissances ; Logique épistémique dynamique probabiliste ; Vérification de modèles symbolique ; Diagrammes de Décision Valués ; Hanabi.

Symbolic representation for probabilistic dynamic epistemic logic

This thesis deals with the study of the practical application of the probabilistic dynamic epistemic logic, allowing to represent the knowledge of agents about the knowledge of other agents by using probabilities. For this, based on the state of the art of symbolic model checking, we propose an encoding of Kripke structures and event structures of our logic via boolean functions and pseudo-Boolean functions. We have implemented our definitions by using adapted data structures : Algebraic Decision Diagrams, which generalize the classical Binary Decision Diagrams by allowing the manipulation of probabilities. Thanks to this implementation, we have been able to conduct experiments that have allowed us to show that our symbolic representation scales better than the explicit version on the Hanabi card game example.

Key-words : Artificial intelligence ; Knowledge Representation ; Probabilistic Dynamic Epistemic Logic ; Symbolic Model Checking ; Valued Decision Diagrams ; Hanabi ;