THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut Supérieur de l'Aéronautique et de l'Espace (ISAE)

Présentée et soutenue par :
Christophe CHARETON
le vendredi 20 juin 2014

Titre :

Modélisation formelle d'exigences et logiques temporelles multi-agents

---

École doctorale et discipline ou spécialité :
ED MITT : Sureté de logiciel et calcul de haute performance

Unité de recherche :
Équipe d'accueil ISAE-ONERA MOIS

Directeur(s) de Thèse :

Mme Laurence CHOLVY (directeur de thèse)
M. Julien BRUNEL (co-directeur de thèse)

Jury :

M. Jean-Paul BODEVEIX - Président du Jury
M. Julien BRUNEL - Co-directeur de thèse
M. David CHEMOUIL
Mme Régine LALEAU - Rapporteur
M. Nicolas MARKEY - Rapporteur
Mme Sophie PINCHINAT

Remerciements

Je remercie les personnes qui, directement ou indirectement, m'ont aidé aboutir à l'écriture de ce manuscrit et à la soutenance de ma thèse :

Mes encadrants d'abord, Julien et David : merci pour le temps pris pour discuter et mettre à l'épreuve chacun des aspects que j'ai abordés. Merci pour vos conseils, scientifiques mais aussi méthodologiques, pour m'avoir *introduit au métier* pendant ces quatre années de stage et doctorat. Merci Laurence, ma directrice, pour tes encouragements et m'avoir apporté ton retour sur ce travail, au long de son avancement.

Merci au département DTIM et à ses membres, de m'avoir accueilli et donné les moyens de ce travail.

Merci également aux membres de mon jury. Merci pour votre attention à mon travail et votre intérêt. Merci aussi à chacun de vous pour votre accueil dans vos équipes respectives au cours de cette thèse.

Merci aux copains : collègues doctorants avec qui j'ai pu partager les joies des avancées, ces moments où tout semble s'ouvrir... ainsi que les phases plus laborieuses ou plus denses, où votre simple présence à table le midi a parfois été la seule fenêtre de distraction de la journée. Ceux qui sont encore là, en particulier Alex, Costa et Tomek. Ceux de mon année, avec qui j'ai partagé l'émotion d'arriver au bout de la démarche, je pense à Antony, Matthias et Pierre. Et puis les historiques : aînés qui sont partis depuis bien longtemps mais qui ont marqué aussi ces années. Guillaume, le Russe et l'Espagnol c'est pour vous que je dis ça !

Merci aux copains plus anciens aussi, qui pour la plupart n'ont jamais rien compris à cette thèse ! Super souvenirs de sorties et week-ends à droite à gauche, en vélo et en rando notamment, bien nécessaires au cours de cette thèse. Lemeg, JB, Momal et Grégoite vous êtes nommés dans ma thèse !

Je remercie ma famille : mes frères et mes sœurs mais surtout mes parents. Merci de m'avoir encouragé et aidé, pas seulement dans les dernières années d'études qu'on été ce doctorat, mais aussi pour toutes les années avant, depuis mon départ du Loquidy.

L'écriture de ce manuscrit a été le point d'aboutissement d'une trajectoire bigarrée dans l'université. J'y ai appris beaucoup et rencontré beaucoup de personnalités marquantes. Parmi mes professeurs qui ont développé chez moi le goût d'apprendre et qui seraient sans doute pour certains bien surpris de se trouver nommés dans un mémoire de thèse informatique : André Vincent, Iqbal Goulhamoussen, Marguerite Léna, Lucien Bély, Michel Carmona, Gabriel Sandu et Arnaud Durand.

Bien des pensées pour vous tous, tout ce qui allongerait trop l'écriture de cette page où n'y aurait pas sa place. La liste est bien loin d'être exhaustive : les personnes qui comptent ou ont compté pendant ces années se connaissent et se reconnaissent pour ça, je l'espère.

Résumé

Ces travaux concernent la modélisation formelle d'exigences et les interactions entre agents. Nous y avons développé un langage de modélisation pour les exigences d'un système à élaborer, KHI. En s'inspirant notamment des méthodes KAOS et TROPOS-i^*, KHI synthétise les concepts essentiels relatifs aux buts et aux agents. Il permet en particulier d'exprimer la question de la capacité effective des agents à assurer la satisfaction des spécifications qui leurs sont assignées. Nous appelons cette question le *problème de l'assignation*. Dans KHI, nous exprimons ce problème comme la question de la satisfaction d'un certain nombre de critères de correction par un modèle.

Pour donner un formalisme aux concepts de KHI et un moyen de résolution du problème de l'assignation, nous introduisons également une logique temporelle multi-agents, USL. Elle s'inspire des travaux dans le domaine, en particulier ATL^*_{sc} et SL. Comme ces derniers formalismes, elle utilise des contextes de stratégies pour exprimer des capacités d'agents à assurer la satisfaction de propriétés temporelles. Elle se distingue des autres formalismes existants principalement par deux aspects : d'abord elle utilise des stratégies non-déterministes. Nous les appelons des *multi-stratégies*. Nous pouvons ainsi exprimer des propriétés de raffinement entre les multi-stratégies. Par ailleurs, nous utilisons pour USL des exécutions du système qui ne sont pas nécessairement infinies. Nous pouvons alors formaliser les notions d'engagement contradictoire pour un agent et de capacités d'actions conflictuelles pour un ensemble d'agents.

Nous réduisons ensuite la satisfaction des critères de correction qui expriment le problème de l'assignation dans KHI à des instances du problème de *model-checking* pour une version adéquate de USL, USL_{KHI}. Nous donnons un algorithme de résolution pour ce problème, il tourne en espace polynomial. L'ensemble des concepts et des outils présentés est par ailleurs illustré par un cas d'étude décrivant des missions d'observation spatiale.

Table des matières

Table des figures

Liste des tableaux

Introduction

Sommaire

Cycle de vie d'un système et ingénierie des exigences

En suivant Michael Jackson [Jac01], on peut présenter l'élaboration d'un système comme une *réponse* apportée à un *problème* posé dans un *environnement*. On a représenté le processus schématiquement dans la Figure 1, tirée de [Jac01]. Les phases de conception du système proprement dite, comme l'architecture et la réalisation détaillée, sont précédées par une étude des buts à remplir par le futur système. Cette étude mène de l'identification des besoins aux spécifications précises du système, tel qu'il doit être élaboré. Ces phases de travail, préliminaires à la réalisation, sont le domaine d'étude de l'ingénierie des exigences.

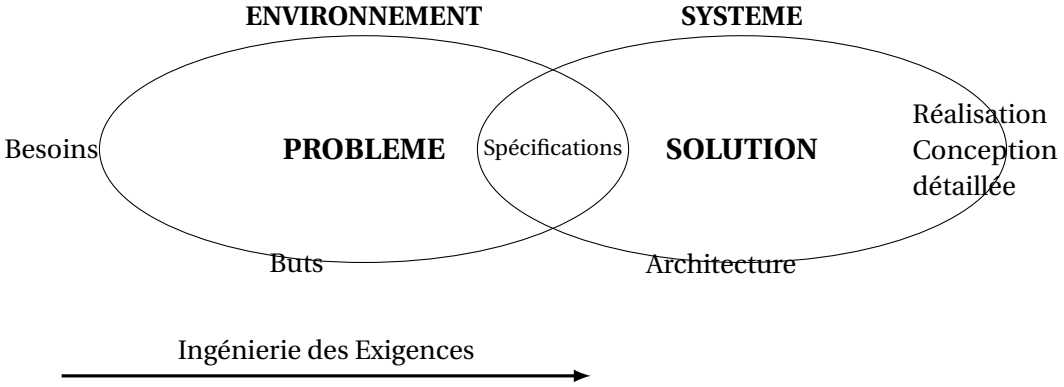

FIGURE 1 – L'ingénierie des exigences dans le développement d'un système

L'importance de cette étude est soulignée par un des outils majeurs de l'analyse du cycle de vie d'un système pour l'ingénierie : le cycle en V. C'est une description schématique du cycle de vie d'un système. On l'a représentée dans la Figure 2. Le développement du système suit les différentes étapes de sa spécification à sa validation, en passant par la conception architecturale, la conception détaillée, l'implémentation et les différentes phases de tests. La relation de vis-à-vis entre deux étapes du cycle, sur la figure, marque une dépendance mutuelle. Plus une erreur est commise en amont, plus sa détection risque d'être tardive et plus ses conséquences risquent d'être coûteuses.

Ce cycle a été conçu pour limiter les retours en arrière lors des constats d'erreurs : des erreurs constatées lors des tests n'obligent pas à reprendre l'intégralité du cycle. En revanche, il fait apparaître l'importance d'une spécification solide et fiable préalablement à toute autre avancée. Celle-ci est en effet la condition nécessaire pour pouvoir tirer profit de l'utilisation du cycle en V et éviter des retours à la première phase de développement du système lors des tests.

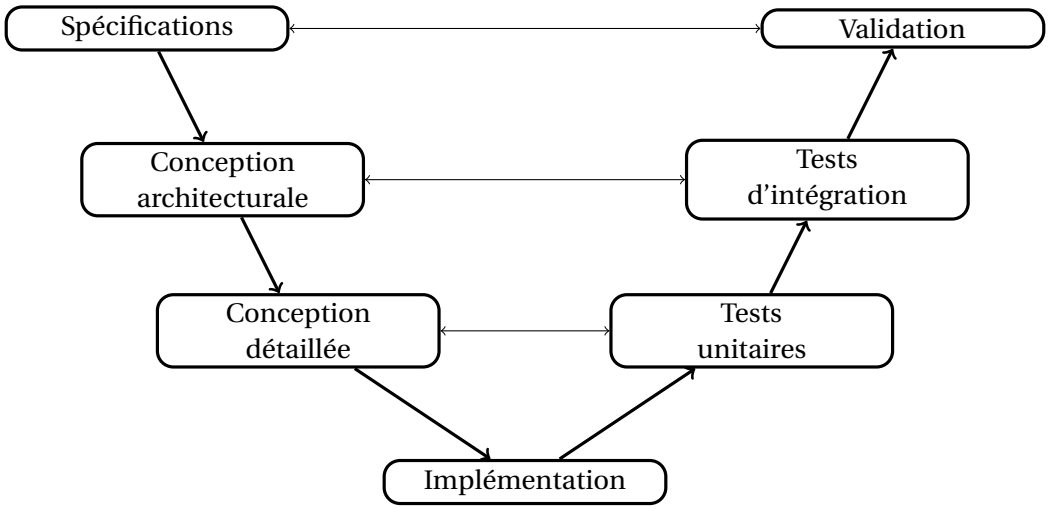

FIGURE 2 – Le développement d'un système : le cycle en V

C'est de ces observations qu'est née l'ingénierie des exigences et qu'elle s'est développée comme domaine d'étude autonome. Elle se donne notamment pour objectifs d'identifier les exigences à remplir par le futur système et d'en dériver les spécifications.

L'ingénierie des exigences développe, en particulier, un ensemble de langages de modélisation d'exigences. C'est à ces langages que nous nous intéressons ici, ainsi qu'aux méthodes formelles et semi-formelles qu'ils mettent en œuvre. Notre contribution s'enracine dans un constat : celui de l'absence, dans ces méthodes, d'une formalisation adéquate des capacités d'actions des agents.

Agents et formalisation des exigences : le problème de l'assignation

Les langages de modélisation en ingénierie des exigences permettent de dériver les spécifications à satisfaire par un système à élaborer, à partir de l'énoncé des buts de plus haut niveau qu'il devra remplir.

Le langage KAOS [vL09, LVL02, vL03, Let02] en particulier offre une méthode de raffinement progressif des buts à assurer par un système. Dans KAOS, ce processus aboutit à la description de spécifications d'opérations sur le système. La satisfaction de chacun des buts de plus bas niveau (*i.e.* de chacune des *exigences*) par des spécifications d'opérations est confiée à un agent. Un agent est donc responsable de la satisfaction des exigences qui lui sont *assignées*.

L'intérêt de cette méthode est notamment de fournir, à l'aide d'une formalisation en logique temporelle linéaire (LTL), un langage qui unifie dans un même formalisme :

– les énoncés de buts, qui portent sur les exécutions entières du système

– les spécifications d'opérations, qui sont des descriptions individuelles des transitions de ce système.

Ce traitement permet notamment de spécifier les relations de précédence entre les opérations : on parle de spécification *comportementale* du système. Le processus définit par ailleurs des critères formels de correction. Ces critères permettent de valider les spécifications comme assurant effectivement les buts de plus haut niveau. Cependant, le critère proposé pour l'assignation des buts de plus bas niveau aux agents nous semble insuffisant, pour des raisons développées notamment dans le Chapitre 1 de ce mémoire : la partie formelle de KAOS n'intégre pas la description des agents et de leurs actions sur le système.

Il nous semble qu'il s'agit là d'un manque important dans la formalisation de ce langage : délaisser le problème des capacités des agents à remplir leurs spécifications aboutit à une analyse des exigences qui ne prend pas en compte les moyens possibles à mettre en œuvre pour les remplir. Cette analyse peine alors à se connecter à la conception effective de l'architecture d'un système.

Nous appelons cette question, de la capacité effective des acteurs à jouer les rôles qui leur sont assignés, le *problème de l'assignation*.

Ce problème a été abordé dans des extensions de la méthode TROPOS [CDGM10a, CDGM10b, MS06, CS09]. Les auteurs y distinguent deux notions différentes d'agents :

– D'une part les agents fournis (les acteurs) : ils sont décrits avec un ensemble de capacités d'actions.

– D'autre part des agents requis (les rôles) : ce sont des descriptions du comportement requis des agents, c'est-à-dire des ensembles de spécifications issus de l'étude des buts du système.

Une relation d'*assignation* relie les rôles à des acteurs en charge de les réaliser dans le système, et le problème de l'assignation est réduit à une propriété de cette relation.

Cependant, le traitement qui est fait du problème de l'assignation dans ce cadre est mené dans le formalisme de la logique propositionnelle. Il ne permet donc pas de spécifier les exigences du système dans leur dimension comportementale. Par ailleurs, chaque agent y est évalué dans sa capacité à jouer les rôles qui lui sont assignés de manière absolument autonome. Le critère fourni par cette approche est donc à nouveau restrictif : il ne permet pas de tenir compte des interactions et des synergies entre les différents acteurs du système, et donc d'en tirer des bénéfices pour décider d'une assignation.

L'ingénierie des exigences offre donc des méthodes qui permettent à la fois de spécifier la dynamique des opérations d'un système et de poser la question de la capacité des acteurs à jouer les rôles qui leur sont assignés. Cependant, aujourd'hui et dans les limites de ce que nous avons pu observer, ces deux aspects sont traités dans des langages distincts. Par ailleurs, les moyens offerts pour traiter cette question ne tirent pas profit des interactions entre les agents.

Objectifs et structure du mémoire

Ces observations justifient notre étude. Elle s'est donnée le triple objectif suivant :

– Conceptualiser le problème de l'assignation dans un langage de modélisation qui permet une description comportementale des opérations.

– Réduire le problème de l'assignation à un problème formel, de satisfaction de formules logiques dans des modèles.

– Proposer des moyens algorithmiques pour la résolution du problème de l'assignation ainsi formalisé.

Ce mémoire commence par un aperçu de l'état de l'art en ingénierie des exigences (Chapitre 1). On y approfondit les langages de modélisation existants, en vue d'exprimer le problème de l'assignation.

Cette présentation est prolongée par un bilan critique des moyens formels existants pour la formalisation de système multi-agents (Chapitre 2).

Notre apport consiste d'abord à conceptualiser les notions d'acteurs et de rôles ainsi que la relation d'assignation qui les lie dans un langage d'ingénierie des exigences décrivant les opérations de manière comportementale (KHI). Cette modélisation aboutit à une expression semi-formelle de différentes versions du problème de l'assignation, déclinées comme autant de critères de correction. Ils permettent notamment de prendre en compte les interactions possibles entre les acteurs, la participation d'un même acteur à différentes coalitions pour différents rôles et l'interaction éventuelle avec des agents qui ne sont pas contrôlés par l'ingénieur qui utilise le langage. Le langage KHI a été décrit dans [CBC11] et [CBC12]. Il est donné dans le Chapitre 3.

Pour donner une sémantique à KHI et en particulier à la relation d'assignation, nous définissons une logique multi-agents, USL. Ce travail a été présenté dans [CBC13, CBC]. Sa description couvre ici deux chapitres : le Chapitre 4 en donne la syntaxe et discute plusieurs sémantiques. Le Chapitre 5 s'intéresse au pouvoir expressif et à la complexité du *model-checking* pour les différentes sémantiques de USL.

L'adaptation de USL à son usage particulier pour KHI achève la formalisation de KHI. Une première version de la sémantique de KHI en USL a été présentée dans [CBC12]. Elle permet d'utiliser les moyens expressifs et de calculs propres à USL pour la formalisation et la résolution du problème de l'assignation. Les différents critères de correction de la relation d'assignation peuvent ainsi être vérifiés algorithmiquement. (Chapitre 6).

Dans ce mémoire, on illustre également l'ensemble des concepts présentés à travers un cas d'étude de missions d'observation spatiale. On a fait le choix de présenter ce modèle de manière exhaustive dans l'Annexe C.

Conventions Techniques et Notations utilisées dans ce mémoire

On introduit ici certaines notations utilisées au long de ce mémoire :

On utilise la notation suivante pour une fonction partielle f d'un ensemble A dans un ensemble B : $f : A \nrightarrow B$. On écrit $dom(f)$ pour le *domaine de définition* d'une fonction partielle f.

Soient deux ensembles A et B, et une relation binaire $R \subseteq A \times B$. Soit aussi $a \in A$, on écrit $R(a)$ pour l'ensemble $\{b \in B \mid \langle a, b\rangle \in R\}$ et $dom(R)$ pour la *projection gauche* de R.

Soit un ensemble S, on écrit $|S|$ pour le *cardinal* de S, $\mathscr{P}(S)$ pour l'ensemble des parties de S, $\mathscr{P}_{>0}(S)$ pour l'ensemble des parties non-vides de S, et $\mathscr{P}_{>0}^{<\omega}(S)$ pour l'ensemble des parties finies non-vides de S.

Soit un ensemble S, on écrit S^* pour l'ensemble des séquences finies dans S, S^+ pour l'ensemble des séquences finies non-vides dans S, S^ω pour l'ensemble des séquences infinies et $S^{\omega+}$ pour l'ensemble des séquences possiblement vides, finies ou infinies, dans S.

Soient S et S' deux ensemble et soient $\tau \in S^*$ et $\lambda \in S^{\omega+}$. On écrit alors $\tau \cdot \lambda$ la concaténation de τ et de λ.

Soit une séquence λ, on écrit λ_0 pour son premier élément et λ_i pour son $(i+1)$-ième élément. La longueur d'une séquence finie λ (*i.e.* le nombre de ses éléments) est écrite $|\lambda|$ et son dernier élément est écrit $last(\lambda)$.

On écrit également $\lambda_{\leqslant i}$ pour la sous-séquence finie $(\lambda_0, \ldots, \lambda_i)$ de λ, et $\lambda_{\geqslant i}$ pour la sous-séquence de λ commençant à l'indice i.

Soit $\mathscr{L}$ un langage formel interprété par une classe de modèles $\mathscr{C}$, et soit $\models_{\mathscr{L}}$ la relation de satisfaction entre un modèle dans $\mathscr{C}$ et une formule de $\mathscr{L}$. On note également $\models_{\mathscr{L}}$ la relation de conséquence sémantique sur les formules de $\models_{\mathscr{L}}$: soient φ_1 et φ_2 deux formules de $\mathscr{L}$. Alors, $\varphi_1 \models_{\mathscr{L}} \varphi_2$ ssi tout modèle $\mathscr{M}$ de $\mathscr{C}$ tel que $\mathscr{M} \models_{\mathscr{L}} \varphi_1$ est aussi tel que $\mathscr{M} \models_{\mathscr{L}} \varphi_2$. Cette notation est étendue à des ensembles de formules : soient Γ un ensemble de formules de $\mathscr{L}$ et φ une formule de $\mathscr{L}$. Alors, $\Gamma \models_{\mathscr{L}} \varphi$ ssi tout modèle $\mathscr{M}$ de $\mathscr{C}$ tel que pour toute $\varphi' \in \Gamma$, $\mathscr{M} \models_{\mathscr{L}} \varphi'$ est aussi tel que $\mathscr{M} \models_{\mathscr{L}} \varphi$.

Avec les notations précédemment introduites, on note également $\mathscr{M} \not\models_{\mathscr{L}} \varphi$ pour : *ce n'est pas le cas que* $\mathscr{M} \models_{\mathscr{L}} \varphi$.

Soit $\mathscr{L}$ un langage formel qui utilise un ensemble de variables X, et soit φ une formule de $\mathscr{L}$. On note $\mathsf{Var}(\varphi)$ l'ensemble des variables de X qui ont une occurence dans φ. Cette notation s'étend à un ensemble de formules Γ : $\mathsf{Var}(\Gamma) = \bigcup_{\varphi \in \gamma} \mathsf{Var}(\varphi)$.

Abbréviations utilisées dans ce mémoire

t.q.	:	tel(le)(s) que
i.e.	:	c'est-à-dire (*id est*)
cf.	:	confert
v.a.v.	:	vis-à-vis
ssi	:	si et seulement si

Chapitre 1

Ingénierie des Exigences

Sommaire

On s'intéresse, dans ce chapitre, aux langages de modélisation développés en ingénierie des exigences, et aux méthodes formelles et semi-formelles qu'ils mettent en œuvre. On y précise en particulier le constat, mentionné dans notre introduction, de l'absence, dans ces méthodes, d'une formalisation adéquate des capacités d'actions des agents.

- D'abord, une méthode *orientée buts* : la méthode KAOS [vL09, LVL02, vL03, Let02]. On présentera en particulier les moyens formels qu'elle offre pour traduire les exigences à satisfaire en spécifications du système (Section 1.1). On dégage de cette étude une formulation du *problème de l'assignation*.
- On s'intéresse ensuite aux méthodes TROPOS et i^* (ces deux méthodes sont apparentées et on désigne leurs éléments communs sous le nom TROPOS-i^*). TROPOS-i^* prend en compte les agents participant au futur système pour déterminer les exigences du système. On s'attachera à dégager les enjeux de cette prise en compte, ainsi que ses limites telle qu'elle est menée dans la méthode TROPOS-i^* (Section 1.2).

La présentation de ces méthodes est guidée par notre objectif, d'exprimer, de formaliser et de résoudre le *problème de l'assignation*. Par ailleurs, nous ne parcourrons, dans l'Ingénierie des Exigences, que le développement de langages de modélisation.

1.1 Ingénierie des Exigences orientée buts : la méthode KAOS

La méthode KAOS [vL09, LVL02, vL03, Let02] fournit un langage de modélisation pour le processus d'identification des exigences et des spécifications. KAOS a en particulier l'intérêt de présenter une formalisation partielle de ses concepts en Logique Temporelle Linéaire (LTL). On donne avant tout les définitions relatives à ce formalisme et nécessaires pour l'introduction de KAOS (Section 1.1.1). Le Chapitre 2 est par ailleurs entièrement consacré aux pré-requis formels nécessaires pour le reste de ce mémoire. KAOS propose différents modèles de description liés au futur système. Nous présentons ici les modèles de buts (Section 1.1.2) et la manière dont KAOS permet de traduire les buts en spécifications du système (Section 1.1.3). Nous nous pencherons ensuite sur le traitement et la place des agents dans cette méthode (Section 1.1.4).

1.1.1 La Logique Temporelle Linéaire : LTL

LTL est une extension de la logique propositionnelle qui permet de décrire l'évolution d'un système dans le temps. Dans la sémantique, ce système est représenté par des séquences infinies d'états (on parle d'exécutions). LTL utilise les opérateurs temporels suivants :

- X (next), qui exprime la succession : pour toute formule φ, $\mathsf{X}\varphi$ est satisfaite dans un état d'une exécution si φ est satisfaite dans l'état suivant de la même exécution.
- U (until), exprime le fait qu'une certaine propriété reste satisfaite jusqu'à ce qu'une deuxième propriété le soit : $\varphi_1 \mathsf{U} \varphi_2$ est satisfaite dans un état d'une exécution si φ_1 est satisfaite dans cet état et le reste dans l'exécution jusqu'à un état où φ_2 est satisfaite.

Formellement, on a la définition suivante pour les formules de LTL :

Définition 1.1 (Syntaxe de LTL)**.** *Soit At un ensemble de propositions atomiques. Alors, le langage LTL sur At est généré par la grammaire suivante :*

$$\varphi := p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathsf{X}\varphi \mid \varphi_1 \mathsf{U} \varphi_2$$

où $p \in At$.

On utilise également, tout au long de ce mémoire, les abréviations suivantes définies sur les opérateurs introduits par LTL :

$$\begin{array}{lcl} \varphi_1 \vee \varphi_2 & \triangleq & \neg(\neg\varphi_1 \wedge \neg\varphi_2) \\ \top & \triangleq & p \vee \neg p \text{ (pour n'importe quel } p \in At) \\ \bot & \triangleq & \neg\top \\ \Diamond\varphi & \triangleq & \top \mathsf{U} \varphi \\ \Box\varphi & \triangleq & \neg\Diamond\neg\varphi \end{array}$$

On définit les modèles de LTL. Ce sont des exécutions valuées sur un ensemble de propositions atomiques At :

Définition 1.2. *Soit At un ensemble de propositions atomiques. Une exécution valuée sur At est un couple* $\langle \lambda, val \rangle$ *t.q. :*

- $\lambda = \lambda_0, \lambda_1, \lambda_2, \ldots$ *est une séquence infinie d'états indicés par les éléments de* $\mathbb{N}$.
- *val est une* valuation*, c'est-à-dire une fonction de* $\{\lambda_n\}_{n\in\mathbb{N}}$ *dans les parties de At* ($val : \{\lambda_n\}_{n\in\mathbb{N}} \rightarrow \mathscr{P}(At)$).

On s'autorise dans la suite de ce mémoire l'abus d'écriture qui consiste à noter λ pour une exécution $\langle \lambda, val \rangle$ et, pour tout $n \in \mathbb{N}$, λ_i pour le couple $\langle \lambda_i, val(\lambda_i) \rangle$.

On peut alors définir la satisfaction sémantique pour LTL :

Définition 1.3 (Satisfaction sémantique pour LTL)**.** *Soient At un ensemble de propositions atomiques. Alors, la* relation de satisfaction $\models_{LTL}$ *est définie par induction sur les formules, pour toute exécution λ valuée sur At, comme suit :*

- $\lambda \models_{LTL} p$ *ssi* $p \in val(\lambda_0)$*, pour tout* $p \in At$
- $\lambda \models_{LTL} \neg\varphi$ *ssi* $\lambda \not\models_{LTL} \varphi$
- $\lambda \models_{LTL} \varphi_1 \wedge \varphi_2$ *ssi* $\lambda \models_{LTL} \varphi_1$ *et* $\lambda \models_{LTL} \varphi_2$
- $\lambda \models_{LTL} \mathsf{X}\varphi$ *ssi* $\lambda_{\geqslant 1} \models_{LTL} \varphi$
- $\lambda \models_{LTL} \varphi_1 \mathsf{U} \varphi_2$ *ssi il existe* $i \in \mathbb{N}$ *t.q.* $\lambda_{\geqslant i} \models_{LTL} \varphi_2$ *et pour tout* $0 \leqslant j < i$, $\lambda_{\geqslant j} \models_{LTL} \varphi_1$

Dans ce mémoire on utilisera comme ensemble de propositions atomiques les formules d'un langage de comparaisons de valeurs entre des variables et des constantes. On en présente ici une première version, $\text{Cond}_{\mathbb{Z}}$, dont on se sert dans ce chapitre. Une version plus élaborée de cet ensemble de propositions (Cond) sera ensuite introduite dans le Chapitre 3.

Définition 1.4 (L'ensemble de propositions $\text{Cond}_{\mathbb{Z}}$)**.** *Soit X un ensemble de variables, l'ensemble de proposition $\text{Cond}_{\mathbb{Z}}$ sur X (on écrit $\text{Cond}_{\mathbb{Z}}(X)$) est donné par la grammaire suivante :*

$$\varphi := x \sim n \mid x - y \sim n \mid x + y \sim n \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi$$

où $x, y \in X$, $n \in \mathbb{Z}$ *et* $\sim \in \{<, >, =, \leqslant, \geqslant\}$.

Remarque 1.1. *On utilise ainsi des formalismes définis avec un ensemble de propositions atomiques qui est infini. Quand on envisagera l'utilisation de procédures de* model-checking *sur ces formalismes, dans le Chapitre 6, on opérera donc au préalable une sélection sur l'ensemble de propositions atomiques, afin de le réduire à un ensemble fini.*

On introduit également la remarque de terminologie suivante :

Remarque 1.2 (terminologie : modèles)**.** *Par la suite, on emploiera couramment le terme de* modèle *dans deux sens distincts :*

- *En Ingénierie des Exigences, on parlera de* modèles *comme objet de la modélisation. Il s'agit donc d'une description dans les langages de modélisation.*
- *On appelle par ailleurs* modèle*, en logique, une structure interprétant un langage formel donné.*

1.1.2 Modèle de buts

Un modèle de buts KAOS est une exposition structurée des *buts* à réaliser par le futur système, les *buts* étant définis comme des énoncés prescriptifs que le système doit satisfaire.

Les buts sont hiérarchisés par une relation de raffinement : la satisfaction d'un but de plus haut niveau (le but raffiné) doit être assurée par la satisfaction des buts de plus bas niveau (les buts raffinants). Cette satisfaction est exigée étant donnée une éventuelle description partielle du contexte : les *propriétés du domaine*. Les *exigences* du système sont alors identifiées comme les buts de plus bas niveau dans cette hiérarchie.

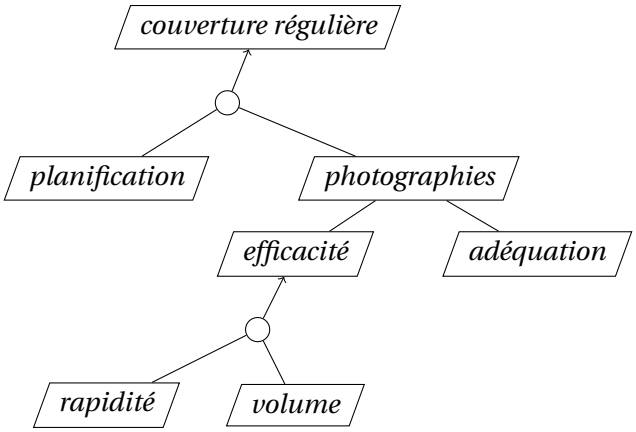

FIGURE 1.1 – Modèle de buts KAOS pour l'*agence cartographique*

On introduit ici un cas d'étude, par lequel on illustrera les concepts présentés et employés dans ce mémoire. Une agence de cartographie (*agence cartographique*) souhaite mettre en place un dispositif de couverture de la Terre par images satellites. Dans le Chapitre 3, on enrichira cet exemple par la prise en compte, également, des besoins pour une organisation militaire (l'*armée*). L'Annexe C offre par ailleurs une description exhaustive de l'exemple des deux missions d'observation, dans le langage de modélisation KHI.

Dans la Figure 1.1, on présente le modèle de buts pour l'*agence cartographique*. Les buts y sont représentés par des parallèlogrammes. Le graphique a la structure d'un arbre orienté par la relation de raffinement. Les buts raffinés sont représentés au-dessus des buts raffinants. À la racine se trouve le but de plus haut niveau. Il consiste en l'offre d'une couverture par image correspondant aux besoins de l'agence de cartographie (*couverture régulière*). Ce but est raffiné par les deux buts qui apparaissent comme *fils* dans le diagramme : l'identification des besoins de nouvelles images (*planification*), et la prise effective des photographies correspondant à ces besoins (*photographies*). Le but *photographies* lui-même doit répondre à une exigence d' *adéquation* (les images collectées doivent correspondre aux besoins identifiés par la *planification*) et les photos doivent être prises de manière efficace. L'efficacité est encore raffinée par les exigences de *volume* (les images doivent être collectées par paquets de deux) et de *rapidité* (après identification d'un besoin, l'image correspondante doit être obtenue dans un intervalle de temps court).

La méthode KAOS permet aussi de représenter, dans le modèle de buts, des alternatives entre plusieurs raffinements possibles pour un but (on parle alors de *raffinements OU*). Cet aspect n'est cependant pas développé dans ce mémoire.

Selon la terminologie déjà employée dans le paragraphe précédent, les buts de plus bas niveau, les buts feuilles dans la représentation graphique, sont appelés *exigences*. Ils sont suffisamment précis pour être assignables à des agents du système.

Dans la formalisation de KAOS [vL09], le système est décrit à travers une succession d'états valués sur l'ensemble des variables du langage. On utilise le langage LTL dont les atomes sont des formules de $\mathrm{Cond}_{\mathbb{Z}}$. Une exigence g fait l'objet d'une formalisation $Form(g)$ en logique LTL.

Ce traitement permet notamment d'intégrer, dans un même modèle, l'ensemble du processus de raffinement de but. Aux deux extrémités du modèle de buts on trouve :

– Le besoin premier, tel qu'originellement identifié dans l'environnement et formulé en langage naturel. Ce besoin est le but de plus haut niveau.

– Les exigences : les spécifications dans KAOS sont des moyens de les satisfaire. Identifier les exigences permettra ainsi d'entrer dans la phase formelle d'identification des spécifications du système.

Nous proposons ci-dessous une formalisation possible pour l'exigence de *volume*. Une formalisation complète des exigences est décrite et justifiée dans l'Annexe C. Dans la formule ci dessous, les variables $ordre^1$ et $ordre^2$ représentent les ordres pour de nouvelles images élaborées par la *planification*.

$$Form(volume) \triangleq \Box((ordre^1 = -1 \wedge \mathsf{X}(ordre^1 \geqslant 0)) \leftrightarrow (ordre^2 = -1 \wedge \mathsf{X}(ordre^2 \geqslant 0))$$

L'exigence de *volume* requiert que les ordres soient envoyés deux par deux. Dans notre formalisation, la valeur -1 pour ces ordres signifie que l'ordre correspondant n'est pas activé, l'envoi d'un ordre $ordre^i$ ($i \in \{1,2\}$) est donc décrit par la sous formule $ordre^i = -1 \wedge \mathsf{X}(ordre^i \geqslant 0)$. La formule **??** spécifie donc bien que les deux ordres sont toujours envoyés en même temps.

.

Les *propriétés du domaine* décrivent l'environnement dans lequel le système est introduit. On les formalise également par des formules $Form(propDomaine)$ de LTL.

Le cas d'étude de l'agence cartographique utilise par exemple des agents qui sont des satellites. Soit le satellite *S1*, son mouvement est décrit par la propriété $domPropMouv(S1)$. On la formalise comme suit :

$$Form(domPropMouv(S1)) \triangleq \Box(\mathsf{X}(positionSatellite^1 = positionSatellite^1.old + 1[mod 6]))$$

où $positionSatellite^1$ encode la position de *S1* sur une trajectoire modélisée par l'intervalle $[O,\ldots 5]$ sur $\mathbb{Z}$, et dans chaque état, $positionSatellite^1.old$ désigne la valeur de $positionSatellite^1$ dans l'état précédent.

On obtient alors la définition suivante pour la correction du raffinement d'un but :

Définition 1.5 (Correction d'un raffinement de buts [Let02])**.** *Un but g est correctement raffiné par les buts raffinant $g_1,\ldots,g_n$, avec les propriétés du domaine propDomaine, ssi la satisfaction de ces buts raffinants est nécessaire et suffisante pour assurer g, c'est-à-dire :*

$Form(g_1),\ldots,Form(g_n),Form(propDomaine) \models_{LTL} Form(g)$	*complétude*
$\bigwedge_{j\neq i} Form(g_j),Form(propDomaine) \not\models_{LTL} Form(g)$, *pour tout* $i \in [O,\ldots n]$	*minimalité*
$Form(g_1),\ldots,Form(g_n),Form(propDomaine) \not\models_{LTL} Form(g)\bot$	*consistence*

Les exigences étant identifiées, leur réalisation est assurée par des *opérations*. Dans la section suivante, on présente les moyens de KAOS pour décrire ces opérations.

1.1.3 Opérationnalisation et modèle d'opérations

Dans KAOS, les opérations sont les actions élémentaires sur le système. Ce sont elles qui en génèrent les transitions. Elles sont décrites à l'aide de deux types de conditions :

– D'une part les *conditions du domaine* identifient chaque occurence d'une opération. Il s'agit de la partie descriptive de la caractérisation. Chaque opération est ainsi définie par une *pré-condition du domaine* (*domPre*) et une *post-condition du domaine* (*domPost*). Dans une exécution λ, l'opération *op* se produit en l'état λ_i ssi sa pré-condition *op.domPre* est satisfaite en l'état λ_i et sa post-condition *op.domPost* est satisfaite en l'état λ_{i+1}. Formellement :

Définition 1.6 (Sémantique d'une opération)**.** *Pour toute opération op, on définit la formule Form*(*op*) *par :*

$$Form(op) \triangleq op.domPre \wedge \mathsf{X}\, op.domPost$$

– Par ailleurs, des *conditions requises* précisent la manière dont ces opérations doivent être déclenchées. Il s'agit cette fois d'énoncés prescriptifs. Les conditions requises d'une opération sont déterminées par les exigences du système. Chacune de ces conditions est ainsi relative à la fois à l'opération qu'elle spécifie et à une exigence pour laquelle elle est requise. Ces conditions se répartissent en *pré-condition requises* (*reqPre*), *post-conditions requises* (*reqPost*) et *conditions de déclenchement* (*reqTrig*). Dans le système à élaborer, une opération donnée ne devra jamais être déclenchée sans que ses *reqPre* ne soient satisfaites. Elles devra toujours être déclenchée d'une manière qui assure en même temps la satisfaction de ses *reqPost*, et elle devra toujours être déclenchée dès que ses *reqTrig* et sa *domPre* seront satisfaites. Formellement :

Définition 1.7 (Sémantique des conditions requises pour une opération)**.** *Soit op une opération, alors :*

$$\begin{array}{lcl} Form(op.reqPre) & \triangleq & \Box(Form(op) \rightarrow op.reqPre) \\ Form(op.reqPost) & \triangleq & \Box(Form(op) \rightarrow \mathsf{X}\, op.reqPost) \\ Form(op.reqTrig) & \triangleq & \Box((op.domPre \wedge op.reqTrig) \rightarrow Form(op)) \end{array}$$

On appelle *spécifications* pour une opération *op*, et on note *Spec*(*op*), l'ensemble de ses conditions requises. Les *spécifications du système* sont l'union des spécifications *Spec*(*op*) pour toutes les opérations identifiées. Le modèle d'opérations (voir Figure 1.2) représente la relation entre une exigence g et une opération *op* tels qu'il y a une spécification de *op* pour g. Les exigences y sont représentées dans des parallèlogrammes et les opérations dans des hexagones. Le paramètre i est utilisé comme une variable dans $\{1,2\}$. Cette utilisation est généralisée dans la description de ce cas d'étude.

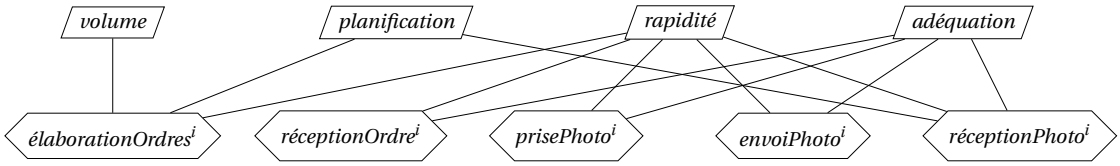

FIGURE 1.2 – Modèle d'opérations pour l'agence de cartographie , $i \in \{1,2\}$

Le circuit d'élaboration et de traitement des ordres se résume comme suit : quand un ordre $ordre_i$ est élaboré (opération $élaborationOrdres^i$), il est reçu par une unité embarquée (opération $réceptionOrdre^i$) qui prend la photographie correspondante (opération $prisePhoto^i$) et la renvoie au sol (opération $envoiPhoto^i$). Cette photo est alors reçue au sol (opération $réceptionPhoto^i$) et un nouvel ordre peut-être élaboré (nouvelle occurrence d'une opération $élaborationOrdres^i$.

Par exemple, l'exigence de *volume* est assurée par les seules opérations $élaborationOrdres^1$ pour l'élaboration de l'ordre $ordre^1$ et $élaborationOrdres^2$ pour l'élaboration de l'ordre $ordre^2$, . On donne dans le Tableau 1.1 ses conditions requises : les deux ordres $ordre^1$ et $ordre^2$ doivent être élaborés en même temps. Chacun de ces ordres $ordre^i$ ne peut donc être élaboré que si le deuxième ordre est désactivé (*reqPre* : $ordre^{3-i} = -1$). Et son activation doit activer également ce second ordre (*reqPost* : $ordre^{3-i} \geqslant 0$).

$élaborationOrdres^i$	$domPre$	$ordre^i = -1$
	$domPost$	$ordre^i \geqslant 0$
	$reqPre$ pour $volume$	$ordre^{3-i} = -1$
	$reqPost$ pour $volume$	$ordre^{3-i} \geqslant 0$

TABLE 1.1 – Opérationnalisation de $élaborationOrdres^i$ pour l'exigence *volume*

L'action d'identifier les opérations du système et de les spécifier selon les conditions requises est appelée *opérationnalisation* dans KAOS. L'opérationnalisation donne lieu à la définition d'un second critère de correction du modèle :

Définition 1.8 (Correction d'une opérationnalisation d'exigence [vL09, LVL02])**.** *Une exigence g est correctement opérationnalisée par les spécifications d'opérations $Spec_1, \ldots, Spec_n$ ssi elles sont nécessaires et suffisantes pour assurer g, c'est-à-dire :*

$$\begin{array}{ll} Form(Spec_1), \ldots, Form(Spec_n) \models_{LTL} Form(g) & \textit{complétude} \\ Form(g) \models_{LTL} Form(Spec_1), \ldots, Form(Spec_n) & \textit{minimalité} \\ Form(Spec_1), \ldots, Form(Spec_n) \not\models_{LTL} \bot & \textit{consistence} \end{array}$$

Ce critère appelle les commentaires suivants :

Commentaire 1.1.

- *La condition de* minimalité *impose que la réalisation des opérations n'apporte rien qui ne soit pas explicitement requis par le but. Couplée avec la condition de* complétude*, elle requiert en fait une équivalence complète entre les opérations et les exigences. Cette condition nous semble inutilement exigeante : en effet, un modèle qui serait tel que la satisfaction des exigences n'implique pas l'ensemble des spécifications serait lui aussi correct, dans le sens où les opérations spécifiées y garantiraient bien les exigences. Son seul défaut serait éventuellement de ne pas être optimal quant à l'économie de moyens mis en œuvre.*
- *Par ailleurs, les propriétés du domaine ne sont pas prises en compte dans l'évaluation de la correction de l'opérationnalisation. Or, les opérations sont bien destinées à être déclenchées dans un environement qui satisfait ces propriétés. Par ailleurs, elles sont bien prises en compte pour évaluer la correction du raffinement de but (cf. Définition 1.5).*

La méthode KAOS propose donc une décomposition progressive des buts du système, qui aboutit à une description des opérations requises pour la réalisation de ces buts. Ce processus est partiellement formalisé et permet la description de deux critères de correction. Ces deux critères permettent eux-mêmes d'assurer et de vérifier l'adéquation entre les buts identifiés et la description finale des opérations du système :

- Dans le raffinement des buts, un critère permet de vérifier que le raffinement garantit effectivement la satisfaction des buts de plus haut niveau par les buts de plus bas niveau.
- L'étape suivante est cruciale : elle fait passer de l'expression de prescriptions globales concernant l'exécution entière du système, à des prescriptions locales sur chacune des transitions. Le critère de correction de l'opérationnalisation permet également de vérifier la correction de cette étape.

On fait par ailleurs le commentaire suivant, sur l'ensemble du formalisme de KAOS :

Commentaire 1.2. *L'utilisation de LTL et des opérations propose une solution formellement correcte pour représenter la satisfaction des buts et sa décomposition en spécifications des transitions du système. Toutefois, l'utilisation de ce formalisme repose sur la capacité effective des ingénieurs utilisateurs du langage à opérer la formalisation des buts. Or, cette étape de formalisation des exigences constitue un enjeu fondamental. Un effort considérable est donc investi dans la recherche d'outils pour opérer la transition entre l'expression informelle des buts et leur formalisation. Elle développe en particulier des solutions pour adapter la méthode B à ce problème d'Ingénierie des Exigences [BMM+07, MGL09, MGL11]. Le problème de la formalisation des exigences ouvre donc un champ d'études qu'il nous permet important de signaler. Il ne sera cependant pas développé plus loin dans nos travaux. Ce commentaire s'applique donc à l'ensemble de ce mémoire.*

1.1.4 Agents et *problème de l'assignation*

Dans KAOS les buts opérationnalisés sont confiés à des *agents* en charge de les remplir. On parle d'*assignation* des buts à des agents. Les agents sont décrits par un ensemble de variables contrôlées et un ensemble de variables surveillées. Un agent peut déclencher une opération s'il surveille les variables sur lesquelles les pré-conditions de cette action (c'est-à-dire à la fois la condition *domPre* et les conditions *reqPre* et *reqTrig*) sont définies et s'il contrôle les variables sur lesquelles ses post-conditions (c'est-à-dire à la fois la condition *domPost* et les conditions *reqPost*) sont définies. Il y a donc encore là un critère de correction, pour l'assignation : l'assignation d'un but à un agent est correcte si l'agent en question peut s'assurer du respect de toutes les spécifications d'opérations pour ce but. On fait sur cette relation et ce critère les commentaires suivants :

Commentaire 1.3.

1. *Un agent qui contrôle une variable peut lui donner n'importe quelle valeur à n'importe quel moment. Dans le système modélisé, il peut cependant arriver qu'un agent puisse agir sur une variable, mais dans certaines limites. Dans notre exemple, le satellite a la capacité de prendre des photos. Ceci se traduit, comme nous le verrons plus loin, par une capacité à changer la valeur de la variable mémoireSatellite.pos ne peut cependant le faire que sous des contraintes correspondant à sa position au moment où il réalise cette opération. Donc à tout moment il peut agir sur la variable mémoireSatellite.pos, mais seulement dans une certaine fenêtre.* KAOS *ne permet pas de modéliser les capacités d'un tel agent.*
2. *Cette observation en amène directement une seconde : les capacités décrites dans* KAOS *ne sont pas pré-conditionnées. La capacité qu'a le satellite d'envoyer un signal vers la terre dépendra pourtant de sa position. L'absence de descriptions de pré-conditions des capacités des agents dans* KAOS *interdit d'utiliser toute interaction entre les agents : chacun garde ses capacités identiques quelles que soient les actions menées par les autres agents. La séquentialité des opérations menées par les différents agents est par exemple inexprimable. On préférera donc une description des capacités d'un agent qui, d'une part, précise quelle valeur cet agent peut donner à chacune des variables sur lesquelles il peut agir et qui, d'autre part, permette de définir des conditions sous lesquelles les capacités d'un agent sont effectives.*
3. *Dans* KAOS*, chaque variable est exclusivement contrôlée par, au plus, un agent. Or, dans la modélisation, il peut arriver que plusieurs agents aient la capacité d'agir sur la même*

variable. Modéliser les phénomènes qui en découlent (en particulier des capacités d'agents conflictuelles*) nécessitera également l'utilisation de moyens plus fins pour décrire les capacités des agents.*

Le *problème de l'assignation* est donc relevé dans KAOS. Nous considérons cependant que la réponse qui y est apportée n'est pas satisfaisante, en ce qu'elle ne permet pas de modéliser de manière fine les capacités des agents à agir sur le système.

Dans le Tableau 1.2, on présente un bilan des aspects formels vus dans la présentation de KAOS en termes d'atouts et de défauts relativement à notre objectif d'exprimer, de formaliser et de résoudre le *problème de l'assignation*. Ce bilan ne concerne que l'aperçu de KAOS, rapide et partiel, qui a été mené dans cette section. De nombreux aspects de cette méthode n'ont par ailleurs pas été abordés ici.

Atouts	– Description de l'exécution du système – Transition, grâce à LTL, entre l'identification de buts de haut niveau et la description individuelle des transitions du système – Existence de critères de correction pour la relation de raffinement et l'opérationnalisation – Première conceptualisation du *problème de l'assignation*

Défauts	– Moyens faibles pour exprimer et vérifier la correction d'une assignation de responsabilités aux acteurs :
	  – Description des capacités des agents difficile à utiliser
	  – Pas de prise en compte de l'interaction entre agents
	– Contraintes trop fortes pour la correction de l'opérationnalisation

TABLE 1.2 – Bilan sur la présentation des aspects formels de KAOS

Dans la section suivante, on poursuit cette étude des langages de modélisation en Ingénierie des Exigences avec les méthodes TROPOS et i^*. Elles permettent notamment d'enrichir les moyens proposés pour modéliser et résoudre le *problème de l'assignation*.

1.2 Ingénierie des Exigences orientée agents : la méthode TROPOS-i^*

Les méthodes TROPOS [BPG+04] et i^* [Yu09, Yu96] sont dites *orientées agents*. i^* est une méthode de modélisation des exigences proposée par Eric Yu. Elle a inspirée le développement de TROPOS. On désigne ici sous le même nom (TROPOS-i^*) les aspects communs à TROPOS et à i^*. TROPOS-i^* a notamment la particularité d'introduire le concept d'*agents intentionnels*. Ces travaux englobent tout le processus de conception d'un système. On y trouve donc aussi bien des phases de conception architecturale que des phases d'Ingénierie des Exigences. Cette intégration globale permet notamment de tenir compte, pendant l'identification des exigences du système, de la distribution de celles-ci à des agents qui seront en charge de les réaliser dans les phases de conception architecturale.

Dans cette section, on explicite le concept d'*agents intentionnels* dans i^* (Section 1.2.1). On étudie ensuite la manière dont TROPOS-i^* utilise cet apport conceptuel pour l'ingénierie des exigences (Section 1.2.2) . On présente finalement une extension de TROPOS par *engagements* et *protocoles*, qui permet un traitement du *problème de l'assignation* en logique propositionnelle [CDGM10a, CDGM10b, MS06, CS09] (Section 1.2.3).

1.2.1 Intentionalité distribuée

Le concept d'agents dans i^* rassemble non seulement des agents artificiels, qui sont crées pour le système, mais aussi des agents pré-éxistants. Ces derniers peuvent être des composants créés auparavant pour un autre système, et alors réutilisés. Mais ils peuvent également être des institutions, des agents humains... Les agents sont ainsi conçus non seulement comme des entités agissant sur le système, mais également comme des entités autonomes, décidant leurs aactions sur ce système. L'analyse des buts se fait alors par rapport aux agents : un but est considéré en tant qu'il est poursuivi par un agent. C'est en ce sens qu'on dit que les agents de TROPOS-i^* sont *intentionnels*, on les appelle également des *acteurs*. Un acteur agit donc sur le système en poursuivant ses propres buts.

Cette prise en compte des acteurs peut aider à l'identification des buts à remplir. Dans le Chapitre 6, grâce à la notion d'intentionalité, on pourra modéliser des acteurs qui ne se comportent pas tous de manière à satisfaire un unique ensemble de buts, mais ne sont pas pour autant nécessairement de simples adversaires les uns des autres. Les actions de ces acteurs peuvent avoir des effets de bords, qu'ils soient favorables ou défavorables au système à élaborer, mais qui sont à prendre en compte dans l'identification des spécifications.

1.2.2 Ingénierie des Exigences dans TROPOS

TROPOS est un langage de modélisation qui couvre à la fois les phases d'ingénierie des exigences pour un système à élaborer, et les phases de conception. Plus précisément, l'étude d'un système est réalisée à travers cinq phases correspondant à autant d'états d'avancement de la conception : l'analyse primitive des exigences (*Early requirements analysis*), l'analyse des exigences avancée (*Late requirements analysis*), la conception architecturale (*Architectural design*), la conception détaillée (*Detailed design*) et l'implémentation (*Implementation*).

Dans les paragraphes qui suivent, nous décrivons la première de ces phases. Nous donnons également un aperçu des objectifs des phases d'analyse des exigences avancée et de conception architecturale. Elles correspondent globalement aux phases de développement prises en charge par notre propre proposition dans le Chapitre 6. Nous en faisons une description rapide illustrée par notre cas d'étude. Nous tirons ensuite un bilan de ce que cette méthode apporte et de ce qui nous semble à améliorer pour la prise en compte des agents, de leurs capacités et de leurs interactions dans l'identification des exigences.

Analyse primitive des Exigences On pose ici une distinction entre deux notions de *système* utilisées dans TROPOS :

- Le *système tel qu'il est*, qui est une description d'un réseau d'acteurs interagissant et qui pré-existe à toute analyse.
- Le *système futur*, qui est l'objet de la conception. Il est donc constitué de l'ensemble des nouveaux acteurs qu'on se propose d'introduire dans le *système tel qu'il est*.

La phase d'Analyse primitive des Exigences est une description du *système tel qu'il est*. Cette description contient les buts identifiés pour les différents acteurs et les relations de dépendance entre ces acteurs, pour la satisfaction de ces buts. Une première analyse moyens-fins est par ailleurs introduite. Elle consiste en un énoncé des *plans* (ensembles d'opérations structurées) à mettre en œuvre pour la réalisation des buts.

On représente donc dans cette phase le réseau pré-existant d'acteurs et leurs interdépendance, ainsi que les buts à remplir qui émergent de ce réseau : les buts à remplir sont ceux qui sont présents dans les modèles de buts des acteurs mais ne sont pas satisfiables en l'état (*i.e.* ne sont pas satisfiables par le *système tel qu'il est*).

On illustre cette phase à l'aide de deux modèles : le *modèle d'acteurs du système* et le *modèle de buts*. On les représente, pour l'exemple de l'agence de cartographie, dans les Figures 1.3 et 1.4.

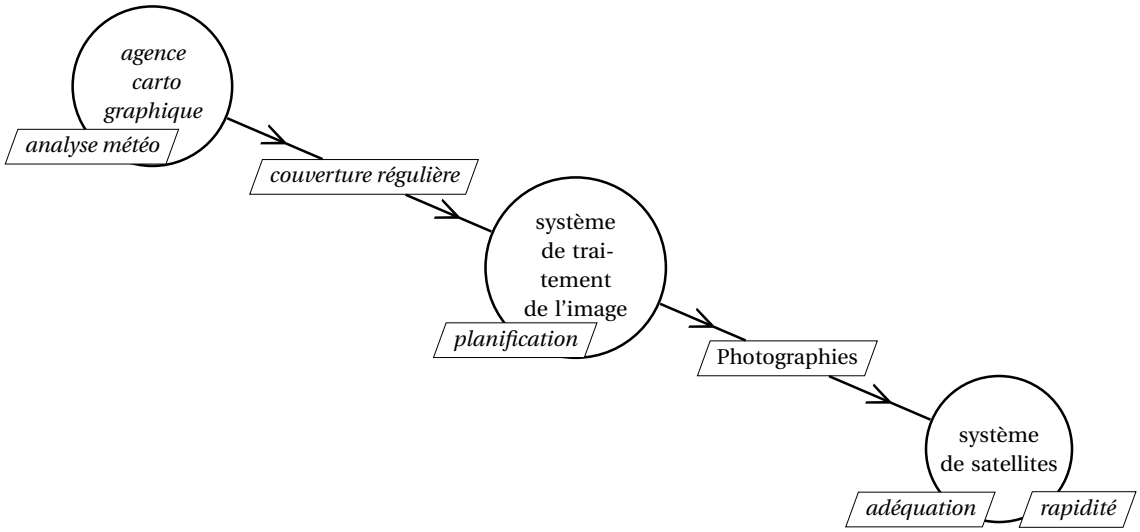

FIGURE 1.3 – Modèle d'acteurs pour l'agence météo

On s'intéresse d'abord au modèle d'acteurs : il schématise les buts de plus haut niveau ainsi que les besoins d'interaction entre les différents acteurs. Ici, l'*agence météo* poursuit le but de produire des analyses et dépend d'un système de traitement de l'image pour satisfaire le but *couverture régulière*. Ce système de traitement de l'image est lui-même dépendant d'un système de satellites pour se procurer les *photographies* à traiter.

On s'intéresse maintenant au modèle de buts pour l'*agence cartographique*. Il est représenté dans une ellipse en pointillés. Ce modèle permet d'identifier les buts, de les structurer par une relation de raffinement similaire à celle pour KAOS. On distingue aussi, dans ce modèle, ceux des buts qui sont déjà assurés par la présence des acteurs du *système tel qu'il est*, de ceux qui devront être assurés par le *système futur*. Dans cette figure, on a donc inclus ceux des buts pour l'*agence cartographique* qui sont déjà assurés par cette agence elle-même : ce sont le traitement des images et la communication des données. Par ailleurs, le graphique laisse apparaître les *plans* : un *plan* est un dispositif mis en œuvre pour la résolution de buts. Un plan joue donc un rôle similaire, dans TROPOS à celui joué par un ensemble d'opérations dans KAOS. La relation qui lie les exigences à ces plans, puis au *système futur* qu'est le *système de traitement des images*, est une relation de dépendance. Par exemple, le plan *réceptionPhoto* contribue à la satisfaction des buts *planification*, *adéquation* et *rapidité*. L'*agence cartographique* est dépendante du *système de traitement des images* pour la réalisation de ce plan.

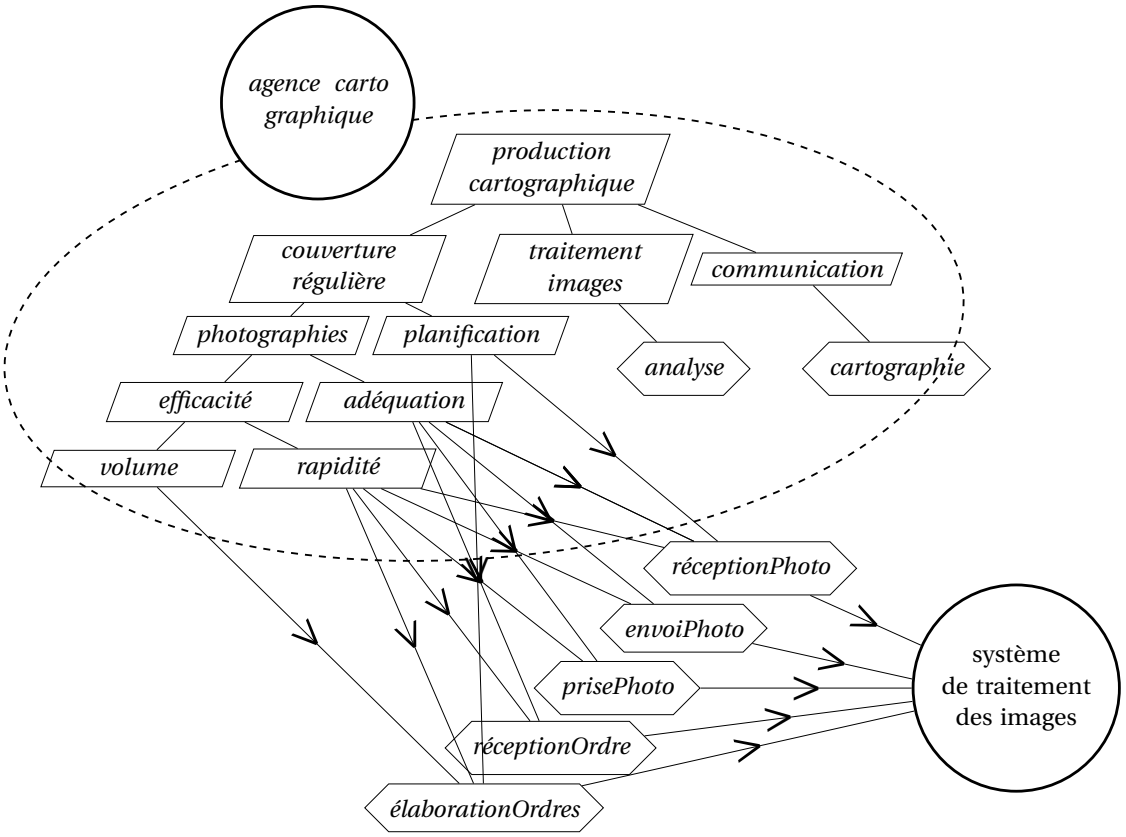

FIGURE 1.4 – Modèle de buts pour l'agence météo

Remarque 1.3. *Pour des raisons de convergence graphique sur l'ensemble de ce mémoire, on a adapté les notations graphiques à celles de* KAOS *dans ces deux figures : la représentation des buts en parallèlogramme, ainsi que celle des* plans *par des hexagones, ne sont pas les représentations habituelles pour* TROPOS-*i**.

On fait également les commentaires suivants sur cette modélisation :

Commentaire 1.4.

1. *Cette phase d'analyse primitive envisage le système d'une manière très large. En particulier, elle prend en compte des buts qui n'ont pas de rapport avec le système futur (comme* traitement images *et* communication *dans notre exemple).*
2. *Mentionner les* plans *(qui correspondent aux opérations dans* KAOS*) dès cette phase primitive de l'analyse nous semble discutable. On préfère, à ce titre, la démarche proposée par* KAOS*, dans laquelle l'identification des opérations est l'aboutissement du processus d'analyse des buts.*
3. *Enfin, la prise en compte des interdépendances entre acteurs, préalablement à l'analyse détaillée des buts, nous semble également difficile à mettre en œuvre : ces relations d'interdépendance émergent plutôt d'une double analyse :*
   - *une analyse des buts, qui aboutit aux spécifications d'opérations.*
   - *une analyse de la capacité des acteurs à assurer ces spécifications.*

L'identification des interdépendances nous paraît donc être la conclusion du traitement du problème de l'assignation*, plutôt que tenir ainsi sa place dans l'analyse primitive des exigences.*

1.2.3 Engagements et *problème de l'assignation*

Récemment, des travaux développés à partir du matériel fourni par TROPOS [CDGM10a, CDGM10b, MS06, CS09], ont proposé un traitement du *problème de l'assignation* en logique propositionnelle. Ceci est rendu possible par l'introduction d'un nouveau matériel dans le langage, dont les concepts essentiels sont les *engagements* et les *protocoles*. On présente ici ce traitement de l'assignation et on en fait un bilan critique.

Le matériel des *engagements* et des *protocoles* propose un moyen pour répondre à la question de la satisfaction du *support des buts* (en anglais : le *goal support*) : une fois les buts raffinés de manière satisfaisante, de quelle manière peut on s'assurer qu'ils sont remplis ?

Une première réponse a été apportée à cette question par KAOS : il s'agit de l'opérationnalisation des buts. Celle-ci permet en effet de décrire, de manière séquentielle, les transitions qui permettront la satisfaction des buts. Le matériel des engagements fournit des moyens similaires pour décrire les transitions requises du système (Section 1.2.3.1). Il introduit par ailleurs une dimension multi-agents, qui offre une première réponse au *problème de l'assignation* (Section 1.9).

On introduit d'abord une distinction entre deux concepts d'agents utilisés ici. Elle sera essentielle dans la suite de ce mémoire, en particulier pour l'expression du *problème de l'assignation*. On définit ainsi les *acteurs* et les *rôles* :

Définition 1.9 (Acteurs et rôles)**.**

- *Les* acteurs *sont les agents en tant qu'entités décrites. Il s'agit d'une donnée, soit présente dans le système tel qu'il est, soit dont l'ingénieur dispose pour élaborer son système. Dans le cadre de* TROPOS *étendu avec les engagements, ils sont décrits à la fois avec leur modèle de buts et par un certain nombre de* capacités *d'actions. Ces* capacités *sont des propositions atomiques du même langage que celui des buts et des engagements. Un acteur qui dispose d'une capacité est capable d'en assurer la satisfaction.*
- *Les* rôles *en sont la contre-partie prescriptive. Il s'agit d'une description du comportement attendu des acteurs. Ces rôles émergent de la spécification des protocoles.*

1.2.3.1 Engagements et protocoles

Un engagement est la contrepartie d'une opération dans KAOS : il s'agit également d'une description d'une transition correcte du système. Elle se fait par la donnée d'une pré-condition (l'*antécédent*) et d'une post-condition (le *conséquent*). Par ailleurs, un engagement est conçu comme une relation entre un rôle tenu de le respecter, et un rôle à l'égard duquel cet engagement est pris :

Définition 1.10 (Engagement, protocole)**.** *Un* engagement C *(*débiteur, créditeur, antécédent, conséquent*) est une relation entre deux rôles et deux conditions, telle que le* débiteur *est tenu v.a.v. du* créditeur *d'assurer l'*antécédent *depuis tout état dans lequel le* conséquent *est réalisé. Un* protocole *est un ensemble d'engagements.*

Formellement, l'*antécédent* et le *conséquent* sont décrits en Logique Propositionnelle (LP). Similairement à KAOS, la satisfaction des buts feuilles, dans un modèle de buts, est assurée par

un ensemble d'engagements (*i.e.* par un protocole). Le matériel des engagements présente donc, avec celui des opérations de KAOS, les éléments de comparaison suivants :

- Dans les deux cas, il s'agit de traduire la satisfaction d'un but par un ensemble de descriptions des transitions requises du système.
- La dimension *sociale* est introduite par les engagements et les protocoles, dans le sens où les rôles en charge de l'opération, tout comme les rôles au bénéfice desquels elle est réalisée, sont une donnée intégrante de l'identification des engagements.
- Les conditions qui permettent de décrire les transition requises sont de deux types seulement : *antécédent* et *conséquent*. Il n'y a en particulier pas de distinction entre les conditions du domaine et les conditions requises. Il n'y a donc pas non plus de distinction entre les différents types de conditions requises.

On peut cependant y simuler les conditions requises au sens de KAOS :

- une pré-condition requise pour une opération est un engagement à ne pas réaliser cette opération sans que cette pré-condition ne soit réalisée,
- une post-condition requise pour une opération est un engagement à ne pas réaliser cette opération sans que cette condition ne soit réalisée,
- une pré-condition suffisante de déclenchement pour une opération est un engagement à la déclencher automatiquement si cette condition et la pré-condition de l'opération sont remplies.

On a donc la traduction suivante, pour une condition requise *req* au sens de KAOS, selon son type, en engagement *Eng*(*req*). Dans cette traduction les variables de *rôles créditeur* et *débiteur* ne sont pas instanciées :

$$\begin{aligned} \mathit{Eng}(\mathit{op.reqPre}) &\triangleq \mathsf{C}(\mathit{débiteur}, \mathit{créditeur}, \mathit{domPre} \wedge \neg \mathit{reqPre}, \neg \mathit{domPost}) \\ \mathit{Eng}(\mathit{op.reqPost}) &\triangleq \mathsf{C}(\mathit{débiteur}, \mathit{créditeur}, \mathit{domPre}, \neg \mathit{domPost} \vee \mathit{reqPost}) \\ \mathit{Eng}(\mathit{op.reqTrig}) &\triangleq \mathsf{C}(\mathit{débiteur}, \mathit{créditeur}, \mathit{domPre} \wedge \mathit{reqTrig}, \mathit{reqPost}) \end{aligned}$$

On peut par exemple exprimer, en termes d'engagements, la spécification en KAOS du Tableau 1.1. Il nous faut pour cela identifier le créditeur et le débiteur. En se référant à la Figure 1.3, l'opération sera réalisée par le *Système de traitement de l'image* (*STI*) et destinée au *Système de satellites* (*SSat*). En introduisant dans le langage des protocoles les propositions construites dans $\mathrm{Cond}_{\mathbb{Z}}$, on obtient les engagements suivants :

$$\mathit{Eng}(\mathit{élaborationOrdres}^i.\mathit{reqPre}) \triangleq \mathsf{C}(\mathit{STI}, \mathit{SSat}, \mathit{ordre}^i = -1 \wedge \neg \mathit{ordre}^{3-i} = -1, \neg \mathit{ordre}^i \geqslant 0)$$

$$\mathit{Eng}(\mathit{élaborationOrdres}^i.\mathit{reqTrig}) \triangleq \mathsf{C}(\mathit{STI}, \mathit{SSat}, \mathit{ordre}^i = -1, \neg \mathit{ordre}^i \geqslant 0 \wedge \mathit{ordre}^{3-i} = -1)$$

1.2.3.2 Rôles et satisfaction des rôles

Ces termes étant posés, le *problème de l'assignation* se pose comme une question d'adéquation entre les acteurs et les rôles : la description des capacités d'actions des acteurs apporte un contenu formel à la question de leur capacité à jouer des rôles. Ce point est notamment développé dans [CDGM10a], on a la définition suivante :

Définition 1.11.

Capacité d'un acteur à jouer un rôle *Soit* Π *un protocole et soit* ρ *un rôle. Soit encore* $\{C\}_{d=\rho}$ *l'ensemble des engagements* C *(*débiteur, créditeur, antécédent, conséquent*) dans* Π *tels que* débiteur $= \rho$ *et soit* $\{\text{conséquent}\}_{d=\rho}$ *l'ensemble des conséquents pour les engagement dans* $\{C\}_{d=\rho}$. *Soit maintenant a un acteur qui a pour ensemble de capacités Cap(a). Alors, a peut jouer le rôle* ρ *ssi Cap(a)* $\subseteq \{\text{conséquent}\}_{d=\rho}$.

Correction d'une assignation *Soit* Π *un protocole et soit Rôle l'ensemble des rôles décrits dans* Π. *Soit maintenant A un ensemble d'acteurs et soit* f *une fonction d'assignation de Rôle dans A. Alors* f *est une assignation correcte ssi pour tout* $\rho \in$ *Rôle,* $f(\rho)$ *est un acteur qui peut jouer le rôle* ρ.

1.2.3.3 Commentaires sur le formalisme des engagements

Ce développement de TROPOS-i^* a le double avantage d'apporter le matériel conceptuel pour le traitement du *problème de l'assignation* et d'en proposer un mécanisme de vérification à l'aide de LP. On retiendra en particulier la distinction entre les deux concepts d'agents : un concept d'agents décrits (les acteurs) et un concept d'agents prescrits (les rôles), et l'expression du problème de l'assignation comme un problème d'adéquation entre les acteurs et les rôles. On fait cependant les commentaires suivants sur cette solution :

Commentaire 1.5.

1. *En premier lieu, comme le montre la Définition 1.11, cette solution envisage l'assignation des rôles exclusivement à des acteurs isolés, qui ne forment pas de coalitions. On perd donc la possibilité de mobiliser les capacités de différents acteurs pour la satisfaction d'un rôle.*
2. *Les engagements, qui sont l'unité de base pour l'action des acteurs, sont en fait une unité d'interaction entre acteurs. Toute action faite par un acteur est donc sociale. Ce cadre ne permet pas de décomposer les actions élémentaires qu'un acteur devrait effectuer pour réaliser une tâche complexe. Par ailleurs, on s'interroge sur la pertinence d'intégrer le paramètre* créditeur *dans tout engagement : si réaliser l'engagement est nécessaire pour la satisfaction d'un but, alors elle l'est quels que soient les acteurs qui en dépendent. Ce matériel risque de faire apparaître le paramètre* créditeur *comme une simple* case à remplir *et d'intégrer dans la description du modèle une surcharge d'information.*
3. *De même les remarques sur l'absence de moyens dans* KAOS *pour traiter le partage de variables entre plusieurs acteurs et les capacités conflictuelles s'appliquent ici.*
4. *L'étape de traduction des buts en termes d'engagements n'a pas été présentée ici. Comme relevé dans le Commentaire 1.2, le traitement de ce problème est en dehors de l'objet de nos travaux. Cependant, l'utilisation d'un formalisme temporel, tel que celui utilisé par* KAOS, *nous semble judicieux. Il permet à la fois :*
   - *Une expression des exigences en tant que caractérisations globales des exécutions acceptables du système.*

– *Une description individuelle des transitions requises du système, pour remplir ces exigences.*

5. *Si la question de la capacité d'acteurs à jouer des rôles est bien traitée, elle est faite de manière isolée pour chacun de ces acteurs. Or dans un modèle, la capacité d'un acteur ou d'une coalition d'acteurs à jouer un rôle sera souvent elle-même dépendante du comportement d'autres acteurs ou coalitions. Les commentaires faits sur la faiblesse des moyens de description des capacités des agents pour* KAOS *s'appliquent donc également à ce matériel.*
6. *La distinction entre un concept d'agents requis (les rôles) et un concept d'agents fournis (les acteurs) est judicieuse et permet de poser les termes du* problème de l'assignation. *Il est cependant parfois gênant que ces deux entités soient décrites avec le même vocabulaire. Dans la proposition du Chapitre 3, nous décrirons les capacités des acteurs comme des capacités à agir sur les valeurs de certaines variables. Dans ce cadre, utiliser les mêmes variables pour décrire une action possible de la part des acteurs et pour identifier des spécifications de rôles abstraits sera impossible. On utilisera alors un langage distinct pour la partie prescriptive (rôles, buts et spécifications) et pour la partie descriptive (les capacités des acteurs).*

On conclut à nouveau cette section par un bilan synthétique en termes d'atouts et de défauts de la méthode TROPOS-*i** et de l'extension au matériel des engagements et des protocoles :

Atouts	– Introduction du système dans un réseau d'agents préexistants – Possibilité de prendre en compte les effets de bord des actions des agents dans le *système tel qu'il est* – Description des capacités des acteurs – Distinction conceptuelle entre acteurs et rôle – Expression du *problème de l'assignation*
Défauts	– Modélisation d'informations peu pertinente – Faiblesse relative du langage de spécification – Formalisation uniquement dans LP – Pas de prise en compte des interactions dans l'utilisation du critère de correction de l'assignation – L'interdépendance entre les agents est prise en compte avant l'identification et l'assignation des spécifications

TABLE 1.3 – Bilan sur la présentation des aspects formels de TROPOS-*i**

1.3 Autres aspects de l'Ingénierie des Exigences

La présentation de l'Ingénierie des Exigences qui précède est très partielle et orientée par notre objectif de recherche de moyens d'expressions et de formalisation du problème de l'assignation.

De nombreux enjeux dans ce domaine n'ont pas été abordés. Les questions liées à la sécurisation du système et à la prévention des attaques par des agents malveillants sont par exemple investies largement, tant à partir du langage KAOS ([CvL13, vLL98, vL04]), que sur le matériel de TROPOS ([ABG07b, ABG+07a, AG06]).

1.4 Conclusion

Il ressort de ce premier chapitre que la formalisation et la résolution du *problème de l'assignation* nécessite l'utilisation d'une logique temporelle multi-agents. Nous poursuivons donc notre étude de l'état de l'art dans le domaine des logiques temporelles multi-agents. C'est l'objet du chapitre suivant.

Chapitre 2

Formalismes existants : logiques temporelles et logiques temporelles multi-agents

Sommaire

Dans ce chapitre, nous présentons différentes logiques temporelles et multi-agents. Che chapitre a donc une double fonction :

- Poursuivre dans le domaine logique la démarche du chapitre précédent, d'analyse de l'état de l'art et des solutions qu'il offre pour le *problème de l'assignation*.
- La formalisation de KHI passera également par une proposition dans le domaine logique, les logique USL et USL$_{\text{KHI}}$. Les Chapitre 4 et 5 seront consacrés à la présentation et à l'étude de ces formalismes. Le chapitre courant fournit alors les définitions des concepts communs aux logiques multi-agents et les termes de comparaisons qui seront faites entre ces langages, notamment dans la Section 5.2.

Les logiques temporelles multi-agents présentées dans ce chapitre sont des extensions des logiques du temps arborescent CTL et CTL*. On présente donc d'abord ces dernières (Section 2.1). On introduit ensuite les langages ATL et ATL*, dans la Section 2.2. On observe que pour exprimer le *problème de l'assignation*, ce formalisme manque d'une sémantique *contextualisée*. La Section 2.3 est consacrée à des formalismes qui utilisent une telle sémantique. On en tire un certain nombre d'observations qui concernent la *révocation des stratégies* dans un contexte et qui font l'objet de la Section 2.4. La Section 2.5 apporte par ailleurs des éléments pour interpréter les opérateurs temporels dans des séquences finies d'états.

2.1 Les logiques du temps arborescent : CTL et CTL*

On présente d'abord les langages CTL et CTL* [CE82]. Il s'agit de logiques du temps arborescent (l'acronyme CTL renvoie à *Computation Tree Logic*). Elles emploient les opérateurs de LTL, auxquels sont ajoutés deux quantificateurs sur les exécutions possibles du système : l'universel A et l'existentiel E.

2.1.1 Modèles de Kripke et exécutions

Les logiques du temps arborescent sont interprétées dans les *modèles de Kripke*. On les définit ici, ainsi que la notion d'exécutions dans ces modèles.

Définition 2.1 (Modèle de Kripke)**.** *Soit At un ensemble de propositions atomiques. Un modèle de Kripke sur At est un quadruplet $\mathcal{K} = \langle G, s_0, R, val\rangle$ où :*

- *G est un ensemble dénombrable non vide d'états,*
- *$s_0 \in G$ est l'état initial,*
- *R est une relation d'*accessibilité*,*
- *$val : G \to \mathcal{P}(At)$ est une fonction de valuation, qui associe à chacun des états s l'ensemble des propositions atomiques vraies dans s.*

Une exécution dans un modèle de Kripke est une séquence infinie d'états de ce modèle, qui respecte la relation d'accessibilité :

Définition 2.2 (Exécutions dans un modèle de Kripke)**.** *Soit $\mathcal{K} = \langle G, s_0 R, val\rangle$ un modèle de Kripke. L'ensemble des exécutions dans $\mathcal{K}$ est l'ensemble des séquences infinies d'états λ telles que :*

- *pour tout $n \in \mathbb{N}, \lambda_n \in G$*
- *pour tout $n \in \mathbb{N}, R(\lambda_n, \lambda_{n+1})$*

Pour tout état s de $\mathcal{K}$, on appelle également les issues *de s, et on note $out^{CTL}(s)$, l'ensemble des exécutions λ de $\mathcal{K}$ telles que $\lambda_0 = s$.*

On peut maintenant présenter les langages CTL et CTL*.

2.1.2 CTL

Les logiques CTL et CTL* diffèrent l'une de l'autre par une restriction syntaxique qui ne s'applique qu'à la première : alors que CTL* utilise librement les opérateurs de LTL et les quantificateurs, CTL, quant à elle, utilise une syntaxe où l'emploi d'un quantificateur est toujours lié à celui d'un opérateur temporel, et réciproquement :

Définition 2.3 (Syntaxe de CTL)**.** *Soit At un ensemble de propositions atomiques. Les formules de CTL sur At sont générées par la grammaire suivante :*

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{E}\mathsf{X}\varphi \mid \mathsf{A}\mathsf{X}\varphi \mid \mathsf{E}\varphi\,\mathsf{U}\,\varphi \mid \mathsf{A}\varphi\,\mathsf{U}\,\varphi$$

où $p \in At$.

Intuitivement, une formule $\mathsf{E}\mathsf{X}\varphi$ est vraie dans un état d'un modèle de Kripke ssi il existe une exécution de ce modèle à partir de cet état qui satisfait la formule φ. Le sens des formules $\mathsf{E}\varphi_1\,\mathsf{U}\,\varphi_2, \mathsf{A}\mathsf{X}\varphi$ et $\mathsf{A}\varphi_1\,\mathsf{U}\,\varphi_2$ est décrit de manière similaire : le quantificateur A agit comme un quantificateur universel sur l'ensemble des exécutions possibles depuis l'état courant, et U comme dans LTL. On définit formellement la satisfaction sémantique pour CTL :

Définition 2.4 (Satisfaction sémantique pour CTL)**.** *Soit $\mathcal{K} = \langle G, s_0, R, val\rangle$ un modèle de Kripke. Alors, la* relation de satisfaction $\models_{CTL}$ *est définie par induction sur les formules, pour tout état $s \in G$, comme suit :*

- $\mathcal{K}, s \models_{CTL} p$ *ssi* $p \in val(s)$*, pour toute proposition atomique p*
- $\mathcal{K}, s \models_{CTL} \neg\varphi$ *ssi* $\mathcal{K} \not\models_{CTL} \varphi$
- $\mathcal{K}, s \models_{CTL} \varphi_1 \wedge \varphi_2$ *ssi* $\mathcal{K}, s \models_{CTL} \varphi_1$ *et* $\mathcal{K}, s \models_{CTL} \varphi_2$
- $\mathcal{K}, s \models_{CTL} \mathsf{E}\mathsf{X}\varphi$ *ssi il y a une exécution* $\lambda \in out^{CTL}(s)$ *t.q.* $\mathcal{K}, \lambda_1 \models_{CTL} \varphi$
- $\mathcal{K}, s \models_{CTL} \mathsf{A}\mathsf{X}\varphi$ *ssi pour toute exécution* $\lambda \in out^{CTL}(s), \mathcal{K}, \lambda_1 \models_{CTL} \varphi$
- $\mathcal{K}, s \models_{CTL} \mathsf{E}\varphi_1\,\mathsf{U}\,\varphi_2$ *ssi il y a une exécution* $\lambda \in out^{CTL}(s)$ *t.q. il existe* $i \in \mathbb{N}$ *t.q.* $\mathcal{K}, \lambda_i \models_{CTL} \varphi_1$ *et pour tout* $0 \leqslant j < i, \mathcal{K}, \lambda_j \models_{CTL} \varphi_1$
- $\mathcal{K}, s \models_{CTL} \mathsf{A}\varphi_1\,\mathsf{U}\,\varphi_2$ *ssi pour toute exécution* $\lambda \in out^{CTL}(s)$*, il existe* $i \in \mathbb{N}$ *t.q.* $\mathcal{K}, \lambda_i \models_{CTL} \varphi_1$ *et pour tout* $0 \leqslant j < i, \mathcal{K}, \lambda_j \models_{CTL} \varphi_1$

Étant donnée une formule φ de CTL, on écrit $\mathcal{K} \models_{CTL} \varphi$ ssi $\mathcal{K}, s_0 \models_{CTL} \varphi$.

2.1.3 CTL*

On présente maintenant CTL*. Il s'agit d'une extension de CTL dans laquelle les opérateurs temporels et les quantificateurs ne sont pas liés. La syntaxe de CTL* distingue des *formules d'états* et des *formules de chemins* :

Définition 2.5 (Syntaxe de CTL*)**.** *Soit At un ensemble de propositions atomiques. Les formules de CTL* sur At sont générées par la grammaire suivante :*

- *formules d'état :*

  $$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{E}\psi \mid \mathsf{A}\psi$$

  où $p \in At$
- *formules de chemins :*

  $$\psi := \varphi \mid \neg\psi \mid \psi \wedge \psi \mid \mathsf{X}\psi \mid \psi\,\mathsf{U}\,\psi$$

On a alors la définition suivante, pour la relation $\models_{\mathrm{CTL}^*}$:

Définition 2.6 (Satisfaction sémantique pour CTL*)**.** *Soit $\mathcal{K} = \langle G, s_0, R, val\rangle$ un modèle de Kripke. Alors, la* relation de satisfaction $\models_{CTL^*}$ *est définie par induction sur les formules, pour tout état $s \in G$ et pour toute exécution λ de $\mathcal{K}$, comme suit :*

– *formules d'états :*
  – $\mathcal{K}, s \models_{CTL^*} p$ *ssi* $p \in val(s)$, *pour toute proposition atomique* p
  – $\mathcal{K}, s \models_{CTL^*} \neg\varphi$ *ssi* $\mathcal{K}, s \not\models_{CTL^*} \varphi$
  – $\mathcal{K}, s \models_{CTL^*} \varphi_1 \wedge \varphi_2$ *ssi* $\mathcal{K}, s \models_{CTL^*} \varphi_1$ *et* $\mathcal{K}, s \models_{CTL^*} \varphi_2$
  – $\mathcal{K}, s \models_{CTL^*} \mathsf{E}\psi$ *ssi il y a une exécution* $\lambda \in out^{CTL}(s)$ *t.q.* $\mathcal{K}, \lambda \models_{CTL^*} \psi$
  – $\mathcal{K}, s \models_{CTL^*} \mathsf{A}\psi$ *ssi pour toute exécution* $\lambda \in out^{CTL}(s), \mathcal{K}, \lambda \models_{CTL^*} \psi$
– *formules de chemins :*
  – $\mathcal{K}, \lambda \models_{CTL^*} \varphi$ *ssi* $\mathcal{K}, \lambda_0 \models_{CTL^*} \varphi$, *pour toute formule d'état* φ
  – $\mathcal{K}, \lambda \models_{CTL^*} \neg\psi$ *ssi* $\mathcal{K}, \lambda \not\models_{CTL^*} \psi$
  – $\mathcal{K}, \lambda \models_{CTL^*} \psi_1 \wedge \psi_2$ *ssi* $\mathcal{K}, \lambda \models_{CTL^*} \psi_1$ *et* $\mathcal{K}, \lambda \models_{CTL^*} \psi_2$
  – $\mathcal{K}, \lambda \models_{CTL^*} \mathsf{X}\psi$ *ssi* $\mathcal{K}, \lambda_{\geqslant 1} \models_{CTL^*} \psi$
  – $\mathcal{K}, \lambda \models_{CTL^*} \psi_1 \mathsf{U} \psi_1$ *ssi il existe* $i \in \mathbb{N}$ *t.q.* $\mathcal{K}, \lambda_{\geqslant i} \models_{CTL} \psi_2$ *et pour tout* $0 \leqslant j < i, \mathcal{K}, \lambda_{\geqslant j} \models_{CTL} \psi_1$

Étant donnée une formule φ *de CTL*, on écrit* $\mathcal{K} \models_{CTL^*} \varphi$ *ssi* $\mathcal{K}, s_0 \models_{CTL^*} \varphi$.

2.2 Les logiques du temps alternant : ATL et ATL*

Le travail pionnier dans le domaine des logiques temporelles multi-agents est le développement des formalismes ATL et ATL* [AHK02, AHK97] (l'acronyme ATL vaut pour *Alternating-time Temporal Logic*). Il s'agit d'extensions de, respectivement, CTL et CTL*, dans lesquelles on s'intéresse aux capacités d'ensembles d'agents (*i.e.* de *coalitions* d'agents) à assurer la satisfaction de certaines propriétés temporelles. Syntaxiquement, on obtient ces langages en remplaçant les quantificateurs E et A de CTL et CTL* par les quantificateurs $\langle\langle A \rangle\rangle$ et $[\![A]\!]$, où A est un agent ou une coalition d'agents.

2.2.1 Concepts sémantiques

On définit d'abord les concepts sémantiques liés à ATL et ATL*. Ces langages s'interprètent dans des extensions de modèles de Kripke qui modélisent les capacités des agents à influencer l'exécution du système. On les appelle des CGSs (de l'anglais *Concurrent Game Structures*). Elles ont été introduites dans [AHK97] puis utilisées dans de nombreux travaux [BDCLLM09, DLLM10, MMV10, MMPV11] derivés de ce premier article. Dans ces modèles, les transitions sont décidées par des actions effectuées par un ensemble d'agents. Dans chaque état de chaque exécution, chaque agent joue une action. La transition est alors déterminée par ces actions et l'état courant. La sémantique utilise une notion de *stratégie*, qui est une fonction de l'ensemble des successions finies d'états dans l'ensemble des actions. Intuitivement, le sens d'une formule $\langle\langle A \rangle\rangle\varphi$ est alors le suivant : pour chaque agent a dans la coaliton A, il existe une stratégie σ_a telle que si chaque agent a dans A joue toujours les actions indiquées par σ_a, alors la satisfaction de φ est assurée.

Définition 2.7. *Une CGS est un uplet* $\mathcal{G} = \langle Ag, G, s_0, At, val, Ac, tr \rangle$ *où :*

– *Ag est un ensemble fini non vide d'agents,*
– *G est un ensemble dénombrable non vide d'états,*
– $s_0 \in G$ *est l'état initial,*

– *At est un ensemble fini non vide de propositions atomiques*

– *val : $G \to \mathscr{P}(At)$ est une fonction de valuation, qui associe à chacun des états s l'ensemble des propositions atomiques vraies dans s,*

– *Ac est un ensemble dénombrable non vide d'actions,*

– *Soit $Dc = Ac^{Ag}$ l'ensemble des décisions,* i.e. *l'ensemble des fonctions totales des agents dans les actions. Alors $tr : G \times Dc \to G$ est la fonction de transition : elle détermine le successeur d'un état donné en fonction de cet état et de l'ensemble des actions jouées par les agents.*

On appelle aussi *préfixe d'exécution* une séquence finie d'états qui est possible dans une CGS :

Définition 2.8 (Préfixes d'exécution et exécutions)**.** *Soit $\mathscr{G}$ une CGS sur l'ensemble d'états G, et soit τ une séquence finie telle que $last(\tau) = \tau_k$ et pour tout $0 \leqslant i \leqslant k, \tau_i \in G$. Alors, τ est un préfixe d'exécution dans $\mathscr{G}$ ssi pour tout $0 \leqslant i < k$, il y a une décision $dc = \langle ac_{a_1}, \ldots, ac_{a_n} \rangle$ telle que $tr(\tau_i, dc) = \tau_{i+1}$. L'ensemble des préfixes d'exécution est noté Track. Similairement, une séquence infinie d'états possible dans $\mathscr{G}$,* i.e. *telle que tous ses préfixes sont dans Track, est appelée une exécution de $\mathscr{G}$.*

La notion de préfixe d'exécution permet notamment d'introduire la définition des stratégies :

Définition 2.9 (Stratégie)**.** *Soit $\mathscr{G} = \langle Ag, G, s_0, At, val, Ac, tr \rangle$ une CGS. Une stratégie σ dans $\mathscr{G}$ est une fonction partielle de Track dans Ac,* i.e. *$\sigma : Track \nrightarrow Ac$. On note $Strat_{\mathscr{G}}$ (ou tout simplement Strat s'il n'y a pas d'ambiguïté) l'ensemble des stratégies du modèle $\mathscr{G}$. Soient un état s et une stratégie σ, on dit de σ qu'elle est s-*totale *si tous les préfixes d'exécution τ de Track tels que $\tau_0 = s$ sont dans $dom(\sigma)$.*

Une coalition est simplement définie comme un ensemble d'agents :

Définition 2.10 (Coalition d'agents)**.** *Une coalition d'agents est un ensemble fini, éventuellement vide, d'agents. Soit a un agent, on écrit usuellement a pour la coalition $\{a\}$.*

À partir d'une coalition, d'un état dans une CGS et de la donnée d'une stratégie pour chaque agent de cette coalition, on peut définir la notion d'issues, il s'agit de l'ensemble des exécutions qui sont possibles à partir de cet état quand chaque agent dans cette coalition joue la stratégie considérée :

Définition 2.11 (Issues)**.** *Soit $\mathscr{G} = \langle Ag, G, s_0, At, val, Ac, tr \rangle$ une CGS. Soient $A \subseteq Ag, s \in G$ et f une fonction de A dans l'ensemble des stratégies s-totales. On appelle issues de (A, f, s) et on note $out^{ATL}(A, f, s)$ l'ensemble d'exécutions possibles quand chaque agent a dans A fait les choix indiqués par la stratégie $f(a)$ à partir de s : pour toute exécution λ de $\mathscr{G}$, $\lambda \in out^{ATL}(A, f, s)$ ssi :*

– *$\lambda_0 = s$*

– *Pour tout $n \in \mathbb{N}$, il y a une décision $dc \in Dc$ telle que pour tout $a \in A$, $dc(a) = f(a)(\lambda_{\leqslant n})$ et $tr(\lambda_n, dc) = \lambda_{n+1}$*

On peut maintenant définir les syntaxes et sémantiques respectives d'ATL et ATL*. Elles diffèrent l'une de l'autre par une contrainte syntaxique similaire à celle qui distingue CTL et CTL* : dans ATL, l'emploi des quantificateurs $\langle\langle A \rangle\rangle$ et $[\![A]\!]$ et celui des opérateurs temporels sont liés.

2.2.2 ATL

On définit d'abord la syntaxe d'ATL :

Définition 2.12 (Syntaxe d'ATL)**.** *Soient At un ensemble de propositions atomiques et Ag un ensemble d'agents. Les formules d'ATL sur At et Ag sont générées par la grammaire suivante :*

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A\rangle\rangle \mathsf{X}\varphi \mid [\![A]\!] \mathsf{X}\varphi \mid \langle\langle A\rangle\rangle \varphi \mathsf{U} \varphi \mid [\![A]\!] \varphi \mathsf{U} \varphi$$

où $p \in At$ et $A \subseteq Ag$

Intuitivement, une formule $\langle\langle A\rangle\rangle \mathsf{X}\varphi$ est vraie ssi les agents dans A disposent chacun d'une stratégie pour garantir ensemble que φ est satisfaite en l'état suivant. On a la définition suivante pour la satisfaction des formules de ATL :

Définition 2.13 (Satisfaction sémantique pour ATL)**.** *Soit $\mathcal{G} = \langle Ag, G, s_0, At, val, Ac, tr\rangle$ une CGS. Alors, la* relation de satisfaction $\models_{ATL}$ *est définie par induction sur les formules, pour tout état $s \in G$, comme suit :*

- $\mathcal{G}, s \models_{ATL} p$ *ssi* $p \in val(s)$*, pour toute proposition atomique p*
- $\mathcal{G}, s \models_{ATL} \neg\varphi$ *ssi* $\mathcal{G} \not\models_{ATL} \varphi$
- $\mathcal{G}, s \models_{ATL} \varphi_1 \wedge \varphi_2$ *ssi* $\mathcal{G} \models_{ATL} \varphi_1$ *et* $\mathcal{G} \models_{ATL} \varphi_2$
- $\mathcal{G}, s \models_{ATL} \langle\langle A\rangle\rangle \mathsf{X}\varphi$ *ssi il y a une fonction f de A dans l'ensemble des stratégies $s-$totales t.q. pour toute exécution $\lambda \in out^{ATL}(A, f, s)$, $\mathcal{G}, \lambda_1 \models_{ATL} \varphi$*
- $\mathcal{G}, s \models_{ATL} [\![A]\!] \mathsf{X}\varphi$ *ssi, pour toute fonction f de A dans l'ensemble des stratégies $s-$totales, pour toute exécution $\lambda \in out^{ATL}(A, f, s)$, $\mathcal{G}, \lambda_1 \models_{ATL} \varphi$*
- $\mathcal{G}, s \models_{ATL} \langle\langle A\rangle\rangle \varphi_1 \mathsf{U} \varphi_2$ *ssi il y a une fonction f de A dans l'ensemble des stratégies $s-$totales t.q. pour toute exécution $\lambda \in out^{ATL}(A, f, s)$ il existe $i \in \mathbb{N}$ t.q. $\mathcal{G}, \lambda_i \models_{ATL} \varphi_2$ et pour tout $0 \leqslant j < i, \mathcal{G}, \lambda_j \models_{ATL} \varphi_1$*
- $\mathcal{G}, s \models_{ATL} [\![A]\!] \varphi_1 \mathsf{U} \varphi_2$ *ssi pour toute fonction f de A dans l'ensemble des stratégies $s-$totales, pour toute exécution $\lambda \in out^{ATL}(A, f, s)$ il existe $i \in \mathbb{N}$ t.q. $\mathcal{G}, \lambda_i \models_{ATL} \varphi_2$ et pour tout $0 \leqslant j < i, \mathcal{G}, \lambda_j \models_{ATL} \varphi_1$*

Étant donnée une formule φ de ATL, on écrit $\mathcal{G} \models_{ATL} \varphi$ ssi $\mathcal{G}, s_0 \models_{ATL} \varphi$.

2.2.3 ATL*

La différence entre les langages ATL et ATL* est similaire à celle entre CTL et CTL* : dans ATL*, un opérateur temporel n'est pas nécessairement immédiatement précédé d'un quantificateur $\langle\langle A\rangle\rangle$ ou $[\![A]\!]$.

Définition 2.14 (Syntaxe d'ATL*)**.** *Soit At un ensemble de propositions atomiques. Les formules d'ATL* sur At sont générées par la grammaire suivante :*

- *formules d'état :*

  $$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A\rangle\rangle \psi \mid [\![A]\!] \psi$$

  où $p \in At$
- *formules de chemins :*

  $$\psi := \varphi\neg\psi \mid \psi \wedge \psi \mid \mathsf{X}\psi \mid \psi \mathsf{U} \psi$$

On définit la satisfaction pour ATL* :

Définition 2.15 (Satisfaction sémantique pour ATL*)**.** *Soit $\mathcal{G} = \langle Ag, G, s_0, At, val, Ac, tr\rangle$ une CGS. Alors, la* relation de satisfaction $\models_{ATL^*}$ *est définie par induction sur les formules, pour tout état $s \in G$ et pour toute exécution λ de $\mathcal{G}$, comme suit :*

- *formules d'états :*
  - $\mathcal{G}, s \models_{ATL^*} p$ *ssi* $s \in val(p)$*, pour toute proposition atomique* p
  - $\mathcal{G}, s \models_{ATL^*} \neg\varphi$ *ssi* $\mathcal{G}, s \not\models_{ATL^*} \varphi$
  - $\mathcal{G}, s \models_{ATL^*} \varphi_1 \wedge \varphi_2$ *ssi* $\mathcal{G}, s \models_{ATL^*} \varphi_1$ *et* $\mathcal{G}, s \models_{ATL^*} \varphi_2$
  - $\mathcal{G}, s \models_{ATL^*} \langle\langle A\rangle\rangle\psi$ *ssi il y a une fonction f de A dans l'ensemble des stratégies $s-$totales t.q. pour toute exécution $\lambda \in out^{ATL}(A, f, s)$, $\mathcal{G}, \lambda \models_{ATL^*} \psi$*
  - $\mathcal{G}, s \models_{ATL^*} [\![A]\!]\psi$ *ssi, pour toute fonction f de A dans l'ensemble des stratégies $s-$totales, pour toute exécution $\lambda \in out^{ATL}(A, f, s)$, $\mathcal{G}, \lambda \models_{ATL^*} \psi$*
- *formules de chemins :*
  - $\mathcal{G}, \lambda \models_{ATL^*} \varphi$ *ssi* $\mathcal{G}, \lambda_0 \models_{ATL^*} \varphi$*, pour toute formule d'état* φ
  - $\mathcal{G}, \lambda \models_{ATL^*} \neg\psi$ *ssi* $\mathcal{G}, \lambda \not\models_{ATL^*} \psi$
  - $\mathcal{G}, \lambda \models_{ATL^*} \psi_1 \wedge \psi_2$ *ssi* $\mathcal{G}\lambda \models_{ATL^*} \psi_1$ *et* $\mathcal{G}, \lambda \models_{ATL^*} \psi_2$
  - $\mathcal{G}, \lambda \models_{ATL^*} \mathsf{X}\psi$ *ssi* $\mathcal{G}, \lambda_{\geqslant 1} \models_{ATL^*} \psi$
  - $\mathcal{G}, \lambda \models_{ATL^*} \psi_1 \mathsf{U} \psi_2$ *ssi il existe* $i \in \mathbb{N}$ *t.q.* $\mathcal{G}, \lambda_{\geqslant i} \models_{ATL^*} \psi_2$ *et pour tout* $0 \leqslant j < i$, $\mathcal{G}, \lambda_{\geqslant j} \models_{ATL^*} \psi_1$

Étant donnée une formule d'état φ de ATL*, on écrit $\mathcal{G} \models_{ATL^*} \varphi$ ssi $\mathcal{G}, s_0 \models_{ATL^*} \varphi$.

On donne les commentaires suivant sur ce formalisme :

Commentaire 2.1.

1. *Les CGSs permettent de modéliser un système qui évolue dans le temps selon les actions d'un certain nombre d'agents. La fonction de transition est à la fois totale et déterministe : elle est définie de manière à ce que la donnée d'une action par agent depuis un état détermine de manière unique un successeur. À partir d'un état s, la donnée d'une stratégie $s-$totale par agent détermine donc une unique exécution. Ces deux contraintes vont être remises en cause dans ce mémoire :*
   - *Le fait que la fonction de transition soit complète impose une forme d'harmonie entre les actions des agents : toute combinaison d'actions par les différents agents est nécessairement cohérente et accepte des successeurs potentiels. C'est la contrepartie formelle de l'absence de* capacités conflictuelles *relevée pour les modèles de* KAOS *dans le Commentaire 1.3 de la Section 1.1. On proposera, dans la Section 4.5, une extension du concept de CGSs qui lève cette contrainte de complétude et permet la modélisation de* capacités conflictuelles.
   - *La contrainte qui concerne le déterminisme n'est pas remise en cause au niveau des CGSs dans ce mémoire, mais au niveau des stratégies. On proposera dans le Chapitre 4 des formalismes qui utilisent des stratégies non-déterministes, ou* multi-stratégies. *Ceci permettra en particulier d'exprimer et d'utiliser des* raffinements *de* multi-stratégies.

2. *Par ailleurs, observons à nouveau la sémantique des opérateurs de capacités. On prend l'exemple de la formule*

$$\langle\langle A\rangle\rangle\Box(\langle\langle B\rangle\rangle\varphi_1 \vee \langle\langle C\rangle\rangle\varphi_2) \tag{2.1}$$

Elle est vraie en l'état s d'une CGS $\mathcal{G}$ ssi il y a une fonction f_A de A dans l'ensemble des stratégies $s-$totales telle que pour toute exécution $\lambda \in out^{ATL}(A, f_A, s)$, pour tout $n \in \mathbb{N}$, il y a une fonction f_B de B dans l'ensemble des stratégies $s-$totales telle que pour toute exécution $\lambda' \in out^{ATL}(B, f_B, \lambda_n), \mathcal{G}, \lambda' \models_{ATL^*} \varphi_1$ et il y a une fonction f_C de C dans l'ensemble des stratégies $s-$totales telle que pour toute exécution $\lambda'' \in out^{ATL}(C, f_C, \lambda_n)$, $\mathcal{G}, \lambda'' \models_{ATL^*} \varphi_2$. L'évaluation des sous formules φ_1 et φ_2 s'effectue donc dans des exécutions λ' et λ", qui ne dépendent que des stratégies jouées par les agents dans B et C. Pour évaluer ces sous-formules, on oublie *la donnée des stratégies jouées par les agents de A (on dit que les agents de A* révoquent *leurs stratégies). Les liaisons d'agents à des stratégies ne sont pas cumulatives dans ATL et ATL*. Evaluer les effets combinés de plusieurs liaisons de stratégies à des agents requiert l'utilisation de sémantiques à* contexte. *Cela est nécessaire pour modéliser l'action concurrente de plusieurs coalitions sur un système.*

Dans la prochaine section, on présente des formalismes qui s'interprétent grâce à des sémantiques à contexte *et permettent donc cette modélisation.*

2.3 Utilisation des contextes de stratégies : ATL$^*_{sc}$ et SL

On s'intéresse ici à des formalismes multi-agents contextualisés. On commence par présenter la logique ATL$^*_{sc}$ [DLLM10, DCL11, BDCLLM09], qui est une sémantique alternative pour ATL (Section 2.3.1). On présente ensuite SL [MMV10, MMPV11]. Ce formalisme introduit en plus, dans la syntaxe, une référence explicite aux stratégies à l'aide de variables. On utilisera ce mécanisme dans les Chapitres 4 et 5 pour construire USL, un formalisme dans lequel les stratégies sont des objets raffinables.

2.3.1 La sémantique ATL$^*_{sc}$

2.3.1.1 Introduction : contextes et interactions entre coalitions

La sémantique ATL$^*_{sc}$ interprète les formules dans des *contextes de stratégies* (l'indice *sc* renvoie à l'anglais *strategy context*), qui sont des fonctions partielles de l'ensemble des agents dans l'ensemble des stratégies. Ce contexte enregistre les liaisons d'agents à des stratégies au cours de l'évaluation d'une formule.

La syntaxe d'ATL$^*_{sc}$ est similaire à celle d'ATL*. Comme la sémantique des opérateurs de capacité $\langle\langle A\rangle\rangle$ et $[\![A]\!]$ diffère, leur notation est cependant remplacée respectivement par $\langle\cdot A\cdot\rangle$ et $\cdot\rangle A\langle\cdot$.

Utiliser des contextes pour l'évaluation des formules permet en particuler de traiter leur composition, donc de spécifier l'interaction entre les stratégies jouées par différentes coalitions. Les auteurs dans [BDCLLM09] se servent de la formule suivante pour illustrer le mécanisme. Elle est obtenue de la formule 2.1 en adaptant à ATL$^*_{sc}$ la notation de l'opérateur de capacité :

$$\theta := \langle\cdot A\cdot\rangle\Box(\langle\cdot B\cdot\rangle\varphi_1 \wedge \langle\cdot C\cdot\rangle\varphi_2) \tag{2.2}$$

Intuitivement, cette formule a le sens suivant : pour chaque agent a de A il y a une stratégie $f_A(a)$ telle que, si chacun joue cette stratégie, alors tout au long de l'exécution, tandis que les agents de A continuent à jouer selon f_A, à la fois les agents dans B peuvent assurer φ_1 et les agents dans C peuvent assurer φ_2.

L'interprétation est donc différente de celle pour la formule 2.1. Dans ATL^*_{sc}, l'application de l'opérateur $\langle\cdot A\cdot\rangle$ ne révoque pas les stratégies des autres agents. La révocation peut par ailleurs être explicitement exprimée grâce à l'opérateur $\cdot\rangle A\langle\cdot$.

On peut maintenant donner les définitions relatives à ATL^*_{sc} : on définit d'abord les contextes de stratégies et leurs transformations, puis la syntaxe et la sémantique d'ATL^*_{sc}.

2.3.1.2 Concepts sémantiques utilisés pour ATL^*_{sc}

Définition 2.16 (Contextes de stratégie)**.** *Soit $\mathcal{G} = \langle Ag, G, s_0, At, val, Ac, tr\rangle$ une CGS. On appelle* contexte de stratégie pour ATL^*_{sc} *(on dit simplement* contexte *quand il n'y a pas d'ambiguïté) une fonction partielle χ de Ag dans l'ensemble des stratégies,* i.e. $\chi : Ag \nrightarrow Strat$.

On définit maintenant les *translations de stratégies* et les *translations de contextes* le long d'un préfixe d'exécution τ. On représente ainsi la manière dont une stratégie et un contexte sont "actualisés" pour prendre en compte le fait qu'un préfixe d'exécution τ est maintenant ajouté dans l'histoire du jeu.

Définition 2.17 (Translations de stratégies et de contextes)**.**

Stratégies *Soient $\mathcal{G} = \langle Ag, G, s_0, At, val, Ac, tr, s_o\rangle$ une CGS, σ une stratégie dans $\mathcal{G}$ et τ un préfixe d'exécution dans $\mathcal{G}$. On appelle* translation de σ le long de τ *la stratégie σ^τ t.q. $dom(\sigma^\tau) = \{\tau' \in Track \mid \tau \cdot \tau'_{\geqslant 1} \in dom(\sigma)\}$ et pour tout $\tau' \in dom(\sigma^\tau)$, $\sigma^\tau(\tau') = \sigma(\tau \cdot \tau'_{\geqslant 1})$.*

Contextes *Soit maintenant χ un contexte de stratégies pour ATL^*_{sc}. On appelle* translation de χ le long de τ *le contexte χ^τ t.q. $dom(\chi^\tau) = dom(\chi)$ et pour tout $a \in dom(\chi^\tau)$, $\chi^\tau(a) = (\chi(a))^\tau$*

La sémantique d'ATL^*_{sc} utilise les transformations de contextes que sont la liaison et la révocation de stratégies :

Définition 2.18 (Transformations de contextes)**.** *Soient χ un contexte de stratégies pour ATL^*_{sc}, A une coalition d'agents et f une fonction de A dans $Strat$. Alors :*

Liaison : *$\chi[f]$ est le contexte de domaine $dom(\chi[f]) = dom(\chi) \cup A$ t.q. pour tout $a \in dom(\chi[f])$,*

$$\chi[f](a) = \begin{cases} f(a) & \text{si } a \in A \\ \chi(a) & \text{sinon} \end{cases}$$

Révocation *$\chi_{\setminus A}$ est le contexte de domaine $dom(\chi_{\setminus A}) = \{a \in dom(\chi) \mid a \notin A\}$ t.q. pour tout $a \in dom(\chi_{\setminus A})$, $\chi_{\setminus A}(a) = \chi(a)$*

On définit enfin les issues d'un contexte et d'un état :

Définition 2.19 (Issues)**.** *Soit $\mathcal{G} = \langle Ag, G, s_0, At, val, Ac, tr\rangle$ une CGS. Soit χ un contexte de stratégies pour $\mathcal{G}$ qui ne renvoie que des stratégies $s-$totales (pour tout $a \in dom(\chi)$, $\chi(a)$) est une stratégie $s-$totale). On appelle issues de (χ, s) et on note $out^{sc}(\chi, s)$ l'ensemble d'exécutions possibles quand chaque agent a dans $dom(\chi)$ fait les choix indiqués par la stratégie $\chi(a)$ à partir de s : $out^{sc}(\chi, s) = out^{ATL}(dom(\chi), \chi, s)$*

2.3.1.3 Syntaxe et sémantique d'ATL$^*_{sc}$

La syntaxe d'ATL$^*_{sc}$ est similaire à celle d'ATL*, moyennant les remplacements de $\langle\langle A\rangle\rangle$ par $\langle\cdot A\cdot\rangle$ et de $[\![A]\!]$ par $\cdot\rangle A\langle\cdot$. La portée de l'opérateur $\cdot\rangle A\langle\cdot$ est par ailleurs limité aux formules d'états, pour des raisons que nous ne détaillons pas ici :

Définition 2.20 (Syntaxe d'ATL$^*_{sc}$)**.** *Soit At un ensemble de propositions atomiques. Les formules d'ATL$^*_{sc}$ sur At sont générées par la grammaire suivante :*

- *formules d'état :*

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\cdot A\cdot\rangle\psi \mid \cdot\rangle A\langle\cdot\varphi$$

  où $p \in At$

- *formules de chemins :*

$$\psi := \varphi \mid \neg\psi \mid \psi \wedge \psi \mid \mathsf{X}\psi \mid \psi \mathsf{U} \psi$$

Remarque 2.1. *Comme pour CTL* et ATL*, il existe aussi une version* sans étoile*, ATL$_{sc}$, d'ATL$^*_{sc}$. On ne la présente pas ici : il est prouvé dans [DCL11] que ces deux formalismes sont équivalents.*

La satisfaction $\models_{\text{ATL}^*_{sc}}$ est paramétrée par un contexte χ :

Définition 2.21 (Satisfaction sémantique pour ATL$^*_{sc}$)**.** *Soit $\mathcal{G} = \langle Ag, G, s_0, At, val, Ac, tr\rangle$ une CGS. Alors, la* relation de satisfaction $\models_{ATL^*_{sc,\chi}}$ *est définie par induction sur les formules, pour tout contexte χ, pour tout état $s \in G$, et pour toute exécution λ de $\mathcal{G}$, comme suit :*

- *formules d'états :*
  - $\mathcal{G}, s \models_{ATL^*_{sc,\chi}} p$ *ssi* $s \in val(p)$*, pour toute proposition atomique p*
  - $\mathcal{G}, s \models_{ATL^*_{sc,\chi}} \neg\varphi$ *ssi* $\mathcal{G}, s \not\models_{ATL^*_{sc,\chi}} \varphi$
  - $\mathcal{G}, s \models_{ATL^*_{sc,\chi}} \varphi_1 \wedge \varphi_2$ *ssi* $\mathcal{G}, s \models_{ATL^*_{sc,\chi}} \varphi_1$ *et* $\mathcal{G}, s \models_{ATL^*_{sc,\chi}} \varphi_2$
  - $\mathcal{G}, s \models_{ATL^*_{sc,\chi}} \langle\cdot A\cdot\rangle\psi$ *ssi il y a une fonction f de A dans l'ensemble des stratégies $s-$totales t.q. pour toute exécution $\lambda \in out^{sc}(\chi[f], s)$, $\mathcal{G}, \lambda \models_{ATL^*_{sc,\chi[f]}} \psi$*
  - $\mathcal{G}, s \models_{ATL^*_{sc,\chi}} \cdot\rangle A\langle\cdot\varphi$ *ssi* $\mathcal{G}, s \models_{ATL^*_{sc,\chi\setminus A}} \varphi$
- *formules de chemins :*
  - $\mathcal{G}, \lambda \models_{ATL^*_{sc,\chi}} \varphi$ *ssi* $\mathcal{G}, \lambda_0 \models_{ATL^*_{sc,\chi}} \varphi$*, pour toute formule d'état φ*
  - $\mathcal{G}, \lambda \models_{ATL^*_{sc,\chi}} \neg\psi$ *ssi* $\mathcal{G}, \lambda \not\models_{ATL^*_{sc,\chi}} \psi$
  - $\mathcal{G}, \lambda \models_{ATL^*_{sc,\chi}} \psi_1 \wedge \psi_2$ *ssi* $\mathcal{G}, \lambda \models_{ATL^*_{sc,\chi}} \psi_1$ *et* $\mathcal{G}, \lambda \models_{ATL^*_{sc,\chi}} \psi_2$
  - $\mathcal{G}, \lambda \models_{ATL^*_{sc,\chi}} \mathsf{X}\psi$ *ssi* $\mathcal{G}, \lambda_{\geqslant 1} \models_{ATL^*_{sc,\chi_{\leqslant 1}}} \psi$
  - $\mathcal{G}, \lambda \models_{ATL^*_{sc,\chi}} \psi_1 \mathsf{U} \psi_1$ *ssi il existe $i \in \mathbb{N}$ t.q. $\mathcal{G}, \lambda_{\geqslant i} \models_{ATL^*_{sc,\chi_{\leqslant i}}} \psi_2$ et pour tout $0 \leqslant j < i$, $\mathcal{G}, \lambda_{\geqslant j} \models_{ATL^*_{sc,\chi_{\leqslant j}}} \psi_1$*

Soit $\chi_\varnothing$ le contexte de domaine vide pour ATL$^*_{sc}$ et φ une formule d'état, on écrit $\mathcal{G}, s \models_{ATL^*_{sc}} \varphi$ ssi $\mathcal{G}, s \models_{ATL^*_{sc,\chi_\varnothing}} \varphi$, et $\mathcal{G} \models_{ATL^*_{sc}} \varphi$ ssi $\mathcal{G}, s_0 \models_{ATL^*_{sc}} \varphi$.

2.3.1.4 Stratégies à mémoire et stratégies positionnelles

Il y a par ailleurs une version d'ATL$^*_{sc}$ qui n'utilise que des stratégies *positionnelles*, c'est-à-dire des stratégies qui ne dépendent que de l'état initial. Cette sémantique est notée ATL$^{*0}_{sc}$. Nous ne la détaillons pas ici : on l'obtient, à partir d'ATL$^*_{sc}$, en adaptant seulement la définition d'une stratégie comme une fonction partielle de l'ensemble des états du modèle dans l'ensemble des actions, *i.e.* $\sigma : \mathrm{G} \nrightarrow Ac$.

On obtient ainsi une sémantique qui permet des gains substantiels en complexité, notamment pour le problème de *model-checking* : il est de difficulté NONELÉMENTAIRE pour ATL$^*_{sc}$, alors qu'il est PSPACE-complet pour ATL$^{*0}_{sc}$.

Un souci d'économie de moyens incitera donc à faire usage uniquement de stratégies positionnelles, quand c'est possible. On constatera par ailleurs, dans le Chapitre 6, que la formalisation du *problème de l'assignation* telle que nous l'aurons alors définie permet de se limiter à ces stratégies.

2.3.2 Un formalisme qui intégre la désignation des stratégies : SL

On présente ici SL. Précisons d'abord qu'il existe deux versions différentes de SL. Une première a été développée par K. Chatterjee, T. A. Henzinger et N. Piterman dans [CHP10] : il s'agit d'un formalisme d'interactions entre deux agents, dont les stratégies sont explicitement quantifiées et désignées par des variables. Il s'interpréte par des jeux à tour dans des CGSs. Un mécanisme similaire a par ailleurs été généralisé par F. Mogavero, A. Murano, G. Perelli et M. Y. Vardi, pour des jeux concurrents, dans des modèles comportant un nombre quelconque d'agents [MMV10, MMPV11] . C'est ce dernier formalisme que nous présentons ci-dessous et auquel nous faisons référence par la notation SL dans le reste de ce mémoire.

La principale différence entre ATL$^*_{sc}$ et SL est que, dans ce dernier langage, les stratégies sont explicitement désignées et quantifiées par des variables. Cette possibilité permet notamment d'exprimer le fait que des agents jouent selon la même stratégie. On se servira également de cette quantification explicite dans le formalisme (USL) que l'on utilisera pour le *problème de l'assignation* (cf. Chapitre 4). Dans USL, on pourra composer entre elles des stratégies pour un même agent. Les nommer explicitement permettra de comparer des stratégies et, éventuellement, de les raffiner ou d'exprimer leurs révocations par les agents.

L'idée originelle du développement de SL est la suivante : dans ATL*, le sens intuitif de l'opérateur $\langle\cdot a\cdot\rangle$ dans la formule $\langle\cdot a\cdot\rangle\varphi$ est qu'il existe une stratégie σ telle que, si a joue selon σ dans le contexte courant, alors φ est réalisée. Deux opérations sont ainsi réalisées successivement : une quantification sur les stratégies, qui identifie σ, puis une liaison de l'agent a à cette stratégie. Dans SL, on opére une distinction entre ces deux opérations. L'opérateur $\langle\cdot a\cdot\rangle$ d' ATL* est ainsi remplacé par deux opérateurs : un quantificateur sur les variables de stratégies $\langle\langle x\rangle\rangle$ et un opérateur de liaison (a, x).

Remarque 2.2. *Le paramètre de l'opérateur de liaison est un agent, et non une coalition comme dans l'opérateur $\langle\cdot A\cdot\rangle$ pour ATL ou ATL$^*_{sc}$: la syntaxe de SL n'utilise pas la notion de coalition.*

L'opérateur $[\![x]\!]$, quant à lui, est le quantificateur universel sur les stratégies. C'est le dual de $\langle\langle x\rangle\rangle$. On donne directement la syntaxe de SL avant d'apporter les définitions nécessaires pour sa sémantique.

Définition 2.22 (Syntaxe de SL)**.** *Soient Ag un ensemble d'agents, At un ensemble de propositions atomiques et X un ensemble de variables de stratégies. Alors la syntaxe de SL sur Ag, At et X est*

définie par la grammaire suivante :

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \varphi \,\mathsf{U}\, \varphi \mid \varphi \,\mathsf{R}\, \varphi \mid \langle\langle x \rangle\rangle \varphi \mid [\![x]\!] \varphi \mid (a, x)\varphi$$

où $p \in At$, $a \in Ag$ et $x \in X$.

SL utilise une notion d'*agents* et de *variables libres* dans une formule φ. On définit comme suit l'unique ensemble des agents et des variables libres dans une formule φ :

Définition 2.23. *Pour toute pseudo-formule φ de SL :*

- $Libre(p) = \varnothing$, *pour* $p \in At$
- $Libre(\neg\varphi) = Libre(\varphi)$
- $Libre(\varphi_1 \wedge \varphi_2) = Libre(\varphi_1 \vee \varphi_2) = Libre(\varphi_1) \cup Libre(\varphi_2)$
- $Libre(\mathsf{X}\varphi) = Ag \cup Libre(\varphi)$
- $Libre(\varphi_1 \,\mathsf{U}\, \varphi_2) = Libre(\varphi_1 \,\mathsf{R}\, \varphi_2) = Ag \cup Libre(\varphi_1) \cup Libre(\varphi_2)$
- $Libre(\langle\langle x \rangle\rangle \varphi) = Libre([\![x]\!] \varphi) = Libre(\varphi) \setminus \{x\}$
- $Libre((a, x)\varphi) = Libre(\varphi)$, *si* $a \notin Libre(\varphi)$
- $Libre((a, x)\varphi) = (Libre(\varphi) \setminus \{a\} \cup \{x\})$, *si* $a \in Libre(\varphi)$

Les énoncés de SL sont alors définis comme les formules φ telles que $Libre(\varphi) = \varnothing$. On relève en particulier le fait qu'un énoncé n'a pas d'agent libre, c'est-à-dire que tous les agents du langage y sont nécessairement liés à des stratégies. On reviendra sur cette particularité de SL dans le Chapitre 4.

Les stratégies utilisées pour SL sont les mêmes que celles pour ATL^*_{sc}. Les contextes pour SL, en revanche, prennent en charge non seulement les agents du langage, mais aussi ses variables :

Définition 2.24 (Contextes pour SL)**.** *Un* contexte pour SL *(on dit simplement un* contexte *quand il n'y a pas d'ambiguïté) est une fonction partielle χ de $Ag \cup X$ dans Strat,* i.e. $\chi : Ag \cup X \nrightarrow Strat$. *Pour tout état s, un contexte χ est dit s-*total *si pour tout $e \in dom(\chi)$, $\chi(e)$ est une stratégie $s-$totale. On appelle également* complet *un contexte χ t.q. $Ag \subseteq dom(\chi)$.*

Les transformations de contextes utilisées pour SL sont la liaison et l'instanciation :

Définition 2.25 (Transformations de contextes)**.** *Soient χ un contexte, a un agent, x une variable de stratégies, σ une stratégie et $e \in dom(\chi)$. La transformation notée $\chi[e \mapsto \sigma]$ est appelée* liaison *de e à σ si e est un agent, et* instanciation *de e par σ si e est une variable. Dans les deux cas elle est définie ainsi :*

$\chi[e \mapsto \sigma]$ est le contexte de domaine $dom(\chi) \cup \{a\}$ t.q. pour tout $e \in dom(\chi[e \mapsto \sigma])$,

$$\chi[e \mapsto \sigma](e) = \begin{cases} \chi(e) & \text{si } e \neq a \\ \chi(x) & \text{sinon} \end{cases}$$

Considérons maintenant un contexte χ complet et $s-$total. Ce contexte définit une stratégie $s-$totale pour tout agent. Il définit donc un choix pour chaque agent depuis l'état s, qui induit un successeur s', puis un choix pour tout agent après le préfixe d'exécution $s \cdot s'$, qui induit lui-même un successeur à s' et ainsi de suite. Finalement, la donnée d'un état s et d'un contexte complet et $s-$total détermine une unique exécution du système. On l'appelle *jeu associé à* (χ, s). Formellement :

Définition 2.26 (Jeu associé à (χ, s))**.** *Soient $\mathcal{G} = \langle Ag, G, s_0, At, val, Ac, tr\rangle$ une CGS, χ un contexte complet et $s-$total pour $\mathcal{G}$ et s un état de G. Le* jeu associé à (χ, s) *est l'exécution $jeu^{(\chi,s)}$ t.q. $jeu_0^{(\chi,s)} = s$ et pour tout $i \in \mathbb{N}$, $jeu_{i+1}^{(\chi,s)} = tr(jeu_i^{(\chi,s)}, dc)$ où, pour tout $a \in Ag$, $dc(a) = \chi(a)(s \cdot jeu_1^{(\chi,s)} \dots jeu_i^{(\chi,s)})$.*

On définit encore les translations générales. Il s'agit, étant donné un état s et un contexte complet et s-total χ, de la séquence des couples formés des successeurs successifs de s quand les agents jouent selon χ et des mises à jour successives de χ le long de cette séquence :

Définition 2.27 (Translation générale)**.** *Soit un état s de G et soit χ une assignation complète et $s-$totale. La i-*ème translation générale de (χ, s) *est la paire $(\chi, s)^i = (\chi_s^i, s_\chi^i)$ t.q.*

- $\chi_s^i = \chi^{jeu_0^{(\chi,s)} \dots jeu_i^{(\chi,s)}}$
- $s_\chi^i = jeu_i^{(\chi,s)}$

On peut alors définir la relation de satisfaction sémantique :

Définition 2.28 (Satisfaction sémantique pour SL)**.** *Soit $\mathcal{G} = \langle Ag, G, s_0, At, val, Ac, tr\rangle$ une CGS. Alors, la* relation de satisfaction $\models_{SL}$ *est définie par induction sur les formules, pour tout contexte χ, pour tout état $s \in G$, et pour toute exécution λ de $\mathcal{G}$, comme suit :*

- $\mathcal{G}, \chi, s \models_{SL} p$ *ssi* $p \in \lambda(s)$, *pour tout* $p \in At$
- $\mathcal{G}, \chi, s \models_{SL} \neg\varphi$ *ssi* $\mathcal{G}, \chi, s \not\models_{SL} \varphi$
- $\mathcal{G}, \chi, s \models_{SL} \varphi_1 \wedge \varphi_2$ *ssi* $\mathcal{G}, \chi, s \models_{SL} \varphi_1$ *et* $\mathcal{G}, \chi, s \models_{SL} \varphi_2$
- $\mathcal{G}, \chi, s \models_{SL} \varphi_1 \vee \varphi_2$ *ssi* $\mathcal{G}, \chi, s \models_{SL} \varphi_1$ *ou* $\mathcal{G}, \chi, s \models_{SL} \varphi_2$
- $\mathcal{G}, \chi, s \models_{SL} \langle\langle x\rangle\rangle\varphi$ *ssi il y a une stratégie $s-$totale σ t.q.* $\mathcal{G}, \chi[x \mapsto \sigma], s \models_{SL} \varphi$
- $\mathcal{G}, \chi, s \models_{SL} [\![x]\!]\varphi$ *ssi pour toute stratégie $s-$totale σ,* $\mathcal{G}, \chi[x \mapsto \sigma], s \models_{SL} \varphi$
- $\mathcal{G}, \chi, s \models_{SL} (a, x)\varphi$ *ssi* $\mathcal{G}, \chi[a \mapsto \chi(x)], s \models_{SL} \varphi$

Si de plus χ est un contexte de stratégie complet et s-total, alors :

- $\mathcal{G}, \chi, s \models_{SL} \mathsf{X}\varphi$ *ssi* $\mathcal{G}, (\chi, s)^1 \models_{SL} \varphi$
- $\mathcal{G}, \chi, s \models_{SL} \varphi_1 \mathbf{U} \varphi_2$ *ssi il y a* $i \in \mathbb{N}$ *t.q.* $\mathcal{G}, (\chi, s)^i \models_{SL} \varphi_2$ *et pour tout* $0 \leqslant j < i$, $\mathcal{G}, (\chi, s)^j \models_{SL} \varphi_1$
- $\mathcal{G}, \chi, s \models_{SL} \varphi_1 \mathsf{R} \varphi_2$ *ssi soit pour tout* $i \in \mathbb{N}$ $\mathcal{G}, (\chi, s)^i \models_{SL} \varphi_2$, *soit il y a* $i \in \mathbb{N}$ *t.q.* $\mathcal{G}, (\chi, s)^i \models_{SL} \varphi_1$ *et pour tout* $j < i$, $\mathcal{G}, (\chi, s)^j \models_{SL} \varphi_2$

Soient $\chi_\varnothing$ le contexte pour SL de domaine vide et φ un énoncé de SL, on écrit $\mathcal{G}, s \models_{SL} \varphi$ ssi $\mathcal{G}, \chi_\varnothing, s \models_{SL} \varphi$, et $\mathcal{G} \models_{SL} \varphi$ ssi $\mathcal{G}, s_0 \models_{SL} \varphi$

Le reste de ce mémoire fait largement référence à SL. C'est en particulier à partir de SL qu'est élaborée USL, dans le Chapitre 4.

2.4 Le problème de la révocation systématique des stratégies

On évoque ici le lien implicite fait, dans les formalismes précédents, entre la liaison d'un agent (ou d'une coalition d'agents) à une (ou des) stratégie(s), et la révocation de sa (leur) stratégie(s) courante(s). On commence par faire le commentaire suivant. Il est mené sur SL mais s'adapte facilement à ATL^*_{sc} :

Commentaire 2.2. *Dans SL, la sémantique de l'opérateur de liaison* (a,x) *s'interprète par la transformation de contexte appelée* liaison *: la valeur pour a y est remplacée par la valeur pour x. Il y a donc à nouveau deux opérations successives qui sont menées :*

- *implicitement d'abord, une révocation par laquelle a est délié de sa stratégie dans le contexte courant,*
- *puis la liaison explicite à la nouvelle stratégie.*

La distinction entre l'opération de liaison et celle de révocation, qui est une des principales raisons pour la prise en compte des contextes dans la sémantique, opère donc toujours seulement entre deux agents différents : la liaison à une nouvelle stratégie entraîne encore une révocation implicite de sa stratégie courante pour l'agent auquel s'applique cette liaison.

Les enjeux liés à ce lien entre liaison et révocation seront développés dans la Section 4.1. On parle du *problème de la révocation automatique des stratégies*. On analyse dans cette section la manière dont il a été relevé et traité dans la littérature.

On fait d'abord un rapprochement entre la liaison avec révocation automatique des logiques multi-agents et les quantifications de CTL-CTL* et la logique du premier ordre FOL (Section 2.4.1). La sémantique de l'opérateur $\cdot\rangle A\langle\cdot$ d'ATL$^*_{sc}$ en fait un opérateur explicite de révocation. On revient dessus dans la Section 2.4.2. Enfin, on présente le formalisme IATL [ÅGJ07], qui soulève cette question et y répond en éliminant de sa sémantique toute possibilité de révocation de stratégie (Section 2.4.3).

2.4.1 Observation : révocation de stratégies et quantification

En premier lieu, la révocation automatique des stratégies stipule que la quantification est indépendante du contexte et des quantifications précédentes. La quantification sur les stratégies avec révocation automatique des stratégies opére donc sur l'ensemble des stratégies disponibles pour un agent, quel que soit le contexte courant.

A ce titre, elle peut être revendiquée comme une généralisation adéquate de la quantification à des modèles multi-agents. On observe en effet facilement l'analogie avec l'interprétation de la quantification pour CTL* :

$$\mathcal{K}, s \models_{\text{CTL}^*} \mathsf{E}\psi \text{ ssi il y a une exécution } \lambda \in out^{\text{CTL}}(s) \text{ t.q. } \mathcal{K}, \lambda \models_{\text{CTL}^*} \psi$$

où la quantification a pour domaine $out^{\text{CTL}}(s)$, indépendammment des précédentes étapes d'évaluation. Le fait que l'interprétation d'un quantificateur soit indépendante du contexte est également une caractéristique de l'évaluation des quantificateurs dans FOL.

Conformément à l'interprétation usuelle des quantificateurs donc, dans les logiques multi-agents, la liaison d'une stratégie à un agent est traditionnellement révocable : à tout moment, un agent peut choisir d'adopter une nouvelle stratégie. Il révoque alors la première. L'agent n'est donc pas *tenu* de jouer selon une certaine stratégie, celle-ci est un moyen révocable qu'il utilise pour satisfaire ses buts.

On estime cependant légitime de remettre en cause cet aspect. Une conception plus guidée par la théorie des jeux nous fait en effet voir les stratégies des agents comme des *plans d'actions* dans un réseau d'interactions. Cette vision est notamment défendue dans [ÅGJ07, ÅGJ08]. Là en effet, une stratégie décrit, de manière définitive, le comportement adopté par un agent. Un agent qui est lié à une stratégie est donc engagé v.a.v. de celle-ci à l'égard des autres agents avec lesquels il interagit dans le système modélisé.

Dans la suite de cette section, nous commentons deux suggestions issues de la litérature pour remettre en cause la révocation automatique des stratégies. Nous relevons également le caractère partiel de ces suggestions.

2.4.2 Une suggestion : l'opérateur de révocation explicite

La première de ces suggestions consiste à introduire dans le langage, en plus des opérateurs de quantification et/ou de liaison, un opérateur explicite de révocation de stratégie. C'est par exemple ce qui est mis en œuvre par ATL^*_{sc} avec l'opérateur $\cdot\rangle A\langle\cdot$. Sa sémantique correspond bien à une révocation, par les agents dans A, de leur stratégie courante. Une proposition contemporaine à ATL^*_{sc} et analogue est par ailleurs développée dans [ÅGJ08].

L'introduction d'un opérateur de révocation dans la sémantique permet de poser la distinction entre les opérations de liaison et de révocation. Il permet par ailleurs de simuler l'opérateur de capacité pour ATL dans une sémantique à contextes. Par exemple, la formule de ATL^*

$$\langle\langle A\rangle\rangle \mathsf{X}\langle\langle B\rangle\rangle \mathsf{X}\, p$$

est vraie ssi les agents de A disposent d'une stratégie telle que s'ils la suivent pendant une transition puis la révoquent, alors les agents de B disposent d'une stratégie qui leur permet d'assurer $\mathsf{X}\, p$. Elle est donc équivalente à la formule d'ATL^*_{sc} suivante :

$$\langle\cdot A\cdot\rangle \mathsf{X}\cdot\rangle A\langle\cdot\langle\cdot B\cdot\rangle \mathsf{X}\, p$$

Cependant, l'introduction d'un tel opérateur de révocation ne résout pas le problème de la révocation automatique des stratégies : l'interprétation de l'opérateur $\langle\cdot A\cdot\rangle$ ne tient pas compte des stratégies auxquelles les agents de A sont liés dans le contexte courant. Elle implique donc également une révocation de ces stratégies par ces agents. On a en effet le théorème suivant dans ATL^*_{sc} : pour tous CGS $\mathscr{G}$, état s, contexte χ, formule de chemin φ et coalition A :

$$\mathscr{G}, s \models_{\mathrm{ATL}^*_{sc,\chi}} \langle\cdot A\cdot\rangle\varphi \text{ ssi } \mathscr{G}, s \models_{\mathrm{ATL}^*_{sc,\chi}} \cdot\rangle A\langle\cdot\langle\cdot A\cdot\rangle\varphi$$

2.4.3 Une suggestion : la révocation interdite

Le problème de la révocation automatique des stratégies est également posé et analysé en détail dans [ÅGJ07]. Une sémantique alternative y est proposée pour ATL, appelée IATL (le I désigne l'Irrévocabilité des stratégies qui sont adoptées par les agents).

On ne détaille pas ici les définitions pour cette sémantique. Cependant : la liaison d'une coalition d'agents à un ensemble de stratégies est interprétée comme une restriction du modèle d'interprétation. Si l'on rencontre une sous-formule $\langle\langle A\rangle\rangle\psi$, on choisit une action depuis chaque état pour chaque agent de A (ce qui revient à choisir pour chacun d'eux une stratégie positionnelle), et on supprime du modèle toutes les autres actions pour ces agents. Cette restriction du modèle est définitive. Une stratégie adoptée par un agent ou une coalition d'agents dans IATL l'est donc une fois pour toutes. Une nouvelle occurence d'opérateur $\langle\langle A\rangle\rangle$ dans une sous-formule de ψ n'aura donc pas d'incidence sur l'évaluation. Ainsi, le langage IATL est équivalent à l'ensemble des formules de ATL défini par la condition suivante : pour toute formule $\langle\langle A\rangle\rangle\psi$ ou $[\![A]\!]\psi$, pour toute sous-formule $\langle\langle B\rangle\rangle\theta$ ou $[\![B]\!]\theta$ de ψ, $A\cap B=\varnothing$.

On fait le commentaire suivant sur cette proposition :

Commentaire 2.3.

1. *Elle a le mérite de poser le problème de la révocation explicite des stratégies et de constituer un langage dans lequel ce problème n'apparaît pas.*
2. *Elle fonctionne cependant, d'une certaine manière, en éliminant les cas problématiques. Le problème de la révocation automatique des stratégies peut-être exprimé comme ceci : si un agent est lié successivement à deux stratégies, il ne peut pas s'engager envers la deuxième sans révoquer la première. Dans le traitement proposé par ATL^*_{sc}, ATL^* et SL, cet agent s'engagera envers la deuxième stratégie en révoquant la première. Au contraire, dans le traitement proposé par IATL, cet agent ne révoquera pas la première stratégie, mais il ne s'engagera pas non plus envers la deuxième. Aucune de ces deux suggestions ne résoud donc le problème.*

2.4.4 Conclusion

Nous pensons qu'une interprétation légitime des interactions entre agents pour la résolution du *problème de l'assignation* en Ingénierie des Exigences devra fournir une solution au problème de la révocation automatique des stratégies.

À notre connaissance, il n'existe pas de formalisme multi-agents qui offre une solution satisfaisante à ce problème : *i.e.* telle qu'un agent puisse adopter une nouvelle stratégie sans révoquer l'ancienne.

Nous verrons dans le Chapitre 4 qu'une solution pour briser cette alternative est de rendre possible des *raffinements de stratégie*. Pour ce faire, on abandonnera le déterminisme des stratégies : une stratégie non déterministe peut en effet être raffinée par une autre sans pour autant être révoquée.

2.5 Opérateurs temporels dans les séquences finies d'états

Dans la suite de ce mémoire, on utilisera par ailleurs des formalismes multi-agents dans lesquels certaines combinaisons d'actions par les agents peuvent arrêter l'exécution du système. On parlera d'*actions conflictuelles* pour désigner ces ensembles d'actions qui, jouées ensemble, ne définissent pas de successeur à l'état courant. Nous n'avons pas rencontré de tel mécanisme dans l'état de l'art. Détecter de tels cas permet d'identifier des assignations incorrectes. Sur un plan plus technique, ce mécanisme apporte au formalisme un pouvoir expressif significatif. Il permet en particulier d'obtenir un prédicat d'égalité entre deux stratégies. Ces points sont explicités dans la Section 5.1.2. Les opérateurs temporels seront alors utilisés également pour spécifier des séquences finies d'états, on utilise alors l'expression d'exécutions possiblement finies.

On termine donc ce chapitre, consacré aux langages formels présents dans la littérature et utilisés dans ce mémoire, en discutant l'interprétation des opérateurs temporels dans de telles séquences. La question que nous abordons ici est donc transversale par rapport à ce qui a été évoqué dans le reste de ce chapitre.

L'adaptation de la sémantique pour LTL à des exécutions possiblement finies comporte une ambiguïté. On l'illustre grâce à l'opérateur X. Considérons d'abord une séquence à un état λ et l'évaluation de la formule $\mathsf{X}\varphi$ dans cette dernière. Comme dans le cas classique, on ampute λ de son premier état et on évalue φ dans la séquence vide.

Deux solutions s'offrent alors. La première est de considérer que φ est vraie dans la séquence vide (on parle de vue *faible* : $\mathsf{X}\varphi$ est vraie dans une exécution possiblement finie ssi elle n'y est pas explicitement falsifiée). Pour respecter l'équivalence $\varphi_1 \mathsf{U} \varphi_2 \leftrightarrow \varphi_2 \vee (\varphi_1 \wedge \mathsf{X}(\varphi_1 \mathsf{U} \varphi_2))$, $\varphi_1 \mathsf{U} \varphi_2$ sera alors vraie dans toute exécution possiblement finie telle qu'il n'y a pas d'état satisfaisant $\neg\varphi_1$

précédant tout état satisfaisant φ_2. La seconde solution, au contraire, est de considérer que φ est fausse dans la séquence vide (on parle de vue *forte* : φ est vraie dans une exécution possiblement finie ssi elle y est explicitement validée. Donc $\mathsf{X}\varphi$ est vraie ssi le successeur de l'état courant existe vraiment et satisfait φ. La formule $\varphi_1 \mathsf{U} \varphi_2$ est vraie, quant à elle, ssi il existe un état qui satisfait φ_2 et n'est précédé que d'états qui satisfont φ_1.

Dans [EFH+03], ces deux vues donnent lieu à deux contextes différents pour la sémantiques de LTL dans les exécutions possiblement finies, qui sont duaux l'un de l'autre ($\models_{\mathrm{LTL}^-}$ pour la vue faible et $\models_{\mathrm{LTL}^+}$ pour la vue forte).

Par ailleurs, également dans [EFH+03] ; une troisième vue, la vue *neutre*, (LTL^n) est présentée. Elle ne s'interpréte que dans les exécutions non vides et ne soulève donc pas le problème expliqué ci-dessus de l'interprétation des opérateurs dans la séquence vide. La vue *neutre* est équivalente à la vue forte sur les exécutions de longueur non nulle.

Dans l'usage que nous ferons des exécutions possiblement finies au cours de ce mémoire, la présence d'exécutions finies est due seulement à des combinaisons pathologiques de choix à partir d'un état. Toute exécution prise en compte a donc au moins un état et nous n'aurons pas besoin d'une sémantique qui puisse s'interpréter dans la séquence vide. On pourra donc limiter la sémantique des opérateurs temporels aux cas de séquences non vides en adoptant la vue *neutre* de [EFH+03]. On donne ci-dessous les définitions relatives à cette vue neutre. On commence par définir les exécutions possiblement finies :

Définition 2.29 (Exécutions possiblement finies)**.** *Soit At un ensemble de propositions atomiques. On appelle* exécution possiblement finie *sur At une séquence non vide $\lambda = \lambda_0 \dots \lambda_k$ (ou $\lambda = \lambda_0, \lambda_1 \dots$) d'états munis d'une valuation val : $Id_\lambda \to \mathscr{P}(At)$, ou Id_λ est l'ensemble d'indices i tels que λ_i est un état dans λ. Soit λ une exécution possiblement finie, on appelle* longeur de λ*, et on note $|\lambda|$ le plus petit $k \in \mathbb{N}$ t.q. λ_k n'existe pas, si un tel k existe, et ω sinon.*

On donne maintenant la définition pour la relation de satisfaction $\models_{\mathrm{LTL}^n}$. On inclut dans cette définition les opérateurs dérivés $\Diamond$ et $\Box$, en appliquant les équivalences suivantes : pour toute formule φ,

$$\begin{array}{lcl} \Diamond\varphi & \triangleq & \top \mathsf{U} \varphi \\ \Box\varphi & \triangleq & \neg\Diamond\neg\varphi \end{array}$$

Définition 2.30 (Satisfaction sémantique pour LTL^n)**.** *Soit λ une exécution possiblement finie sur At. Alors, la* relation de satisfaction $\models_{LTL^n}$ *est définie comme suit, par induction sur les formules :*

- $\lambda \models_{LTL^n} p$ *ssi* $p \in val(\lambda_0)$
- $\lambda \models_{LTL^n} \neg\varphi$ *ssi* $\lambda \not\models_{LTL^n} \varphi$
- $\lambda \models_{LTL^n} \varphi_1 \wedge \varphi_2$ *ssi* $\lambda \models_{LTL^n} \varphi_1$ *et* $\lambda \models_{LTL^n} \varphi_2$
- $\lambda \models_{LTL^n} \mathsf{X}\varphi$ *ssi* $|\lambda| > 1$ *et* $\lambda_{\geqslant 1} \models_{LTL^n} \varphi$
- $\lambda \models_{LTL^n} \varphi_1 \mathsf{U} \varphi_2$ *ssi il existe* i *t.q.* $|\lambda| > i$, $\lambda_{\geqslant i} \models_{LTL^n} \varphi_2$ *et pour tout* $0 \leqslant j < i$, $\lambda_{\geqslant j} \models_{LTL^n} \varphi_1$
- $\lambda \models_{LTL^n} \Diamond\varphi$ *ssi il existe* i *t.q.* $|\lambda| > i$ *et* $\lambda_{\geqslant i} \models_{LTL^n} \varphi$
- $\lambda \models_{LTL^n} \Box\varphi$ *ssi pour tout* i *t.q.* $|\lambda| > i$, $\lambda_{\geqslant 1} \models_{LTL^n} \varphi$

Remarque 2.3 (Caractérisation des exécutions infinie)**.** *On remarque que LTL^n permet notamment d'exprimer la propriété pour une exécution possiblement finie d'être infinie, grâce à la formule $\Box\mathsf{X}\top$. Soit en effet λ une trace finie, on a $\lambda_{\geqslant last(\lambda)} \not\models_{LTL^n} \mathsf{X}\top$ et donc $\lambda \not\models_{LTL^n} \Box\mathsf{X}\top$. Si λ est infinie par contre, pour tout $k \in \mathbb{N}$, $\lambda_{\geqslant k} \models_{LTL^n} \mathsf{X}\top$ et donc $\lambda \models_{LTL^n} \Box\mathsf{X}\top$.*

On donne également ici des définitions pour les satisfactions $\models_{\text{LTL}}$ et $\models_{\text{LTL}^n}$ dans les modèles de Kripke : une formule φ est vraie dans un modèle $\mathcal{K}$ d'état initial s_0 ssi φ est vraie dans toute exécution de $\mathcal{K}$ dont le premier état est s_0. Ainsi pour $\models_{\text{LTL}}$:

Définition 2.31 (Satisfaction $\models_{\text{LTL}}$ dans les modèles de Kripke)**.** *Soit $\mathcal{K} = \langle G, R, val, s_0, \rangle$ un modèle de Kripke, et soit φ une formule de LTL. Alors, pour tout état s de $\mathcal{K}$, $\mathcal{K}, s \models_{LTL} \varphi$ ssi pour toute exécution $\lambda \in out^{CTL}(s), \mathcal{K}, \lambda \models_{LTL} \varphi$. On écrit par ailleurs $\mathcal{K} \models_{LTL} \varphi$ ssi $\mathcal{K}, s_0 \models_{LTL} \varphi$*

Pour $\models_{\text{LTL}^n}$ on commence par définir l'ensemble des exécutions possiblement finies dans un modèle de Kripke. C'est l'ensemble des séquences non vides d'états de ce modèle qui respectent la relation d'accessibilité :

Définition 2.32 (Exécutions possiblement finies dans un modèle de Kripke)**.**
Soit $\mathcal{K} = \langle G, s_0, R, val, \rangle$ un modèle de Kripke. L'ensemble des exécutions possiblement finies λ dans $\mathcal{K}$ est tel que :

- *si λ est infinie alors pour tout $n \in \mathbb{N}, R(\lambda_n, \lambda_{n+1})$*
- *si λ est finie et $|\lambda| = n$ alors :*
  - *pour tout $i < n-1, R(\lambda_i, \lambda_{i+1})$*
  - *$\{s \mid R(\lambda_{n-1}, s)\} = \varnothing$*

On peut alors définir $\models_{\text{LTL}^n}$ dans les modèles de Kripke :

Définition 2.33 (Satisfaction $\models_{\text{LTL}^n}$ dans les modèles de Kripke)**.** *Soit $\mathcal{K} = \langle G, s_0, R, val, \rangle$ un modèle de Kripke, et soit φ une formule de LTL. Alors, pour tout état s de $\mathcal{K}$, $\mathcal{K}, s \models_{LTL^n} \varphi$ ssi pour toute exécution possiblement finie λ de $\mathcal{K}$ telle que $\lambda_0 = s, \mathcal{K}, \lambda \models_{LTL^n} \varphi$. On écrit par ailleurs $\mathcal{K} \models_{LTL^n} \varphi$ ssi $\mathcal{K}, s_0 \models_{LTL^n} \varphi$*

Pour terminer cette section, on donne un bilan général, en termes d'atouts et de défauts, des différents formalismes rencontrés :

Atouts	– Formalisation d'interactions entre agents (ATL, ATL*, ATL$^*_{sc}$, SL) – Contextes de stratégies (ATL$^*_{sc}$, SL, IATL) – Interprétation adéquate des opérateurs temporels dans les traces finies (LTL^n)
Défauts	– Pas de solution satisfaisante au problème de la révocation automatique des stratégies – Pas de modélisation satisfaisante des *capacités conflictuelles*

TABLE 2.1 – Bilan sur les logiques présentées.

Synthèse de l'état de l'art et introduction aux chapitres suivants

On tire ici une conclusion de ces deux premiers chapitres afin de synthétiser les besoins à satisfaire pour le langage KHI et son formalisme, qui feront l'objet des chapitres à venir. On présente ces besoins sous la forme d'un modèle de buts dans la Figure 2.1.

On fait avant tout une remarque sur la forme de ce graphique :

Remarque 2.4. *On y a appliqué certains des choix graphiques et des concepts présentés dans le Chapitre 1 ou qui seront présentés dans le Chapitre 3. La représentation avec les ellipses n'a pas été présentée jusqu'ici. Elle est issue de* TROPOS-*i* Il s'agit des rôles issus de l'analyse des exigences à remplir. Ils sont étiquetés par des cercles avec une corde horizontale. Les cercles sans corde sont les acteurs,* i.e. *les outils qui serviront à remplir les rôles. Il y a en effet une analogie entre notre démarche d'une recherche de solution pour exprimer et résoudre le* problème de l'assignation, *et certains aspects de la mise en œuvre de cette solution elle-même. Cependant, il s'agit seulement d'une analogie. Elle s'arrête à cette étude d'exigences à remplir, de laquelle émergent des rôles qui sont des descriptions des acteurs à fournir. La Figure 2.1 n'est pas le modèle de* KHI *en* KHI. *En effet :*

- *Les systèmes concernés par l'Ingénierie des Exigences sont prioritairement des systèmes d'informations et de calcul.* KHI *est un langage et ne répond donc pas lui-même à des exigences de calcul ou de production d'information, mais à des exigences d'expressivité. Il en résulte que les exigences ne donnent pas lieu à des spécifications d'opérations. Dans la Figure 2.1, les rôles sont constitués par des ensembles d'exigences. Dans* KHI, *comme nous le verrons, les rôles sont des ensembles d'opérations spécifiées.*
- *Les rôles, tels qu'on les a fait apparaître ici, sont caractérisés par des ensembles de buts. Dans* KHI, *ils seront caractérisés par des ensembles d'opérations.*
- *De manière plus anecdotique : on a fait figurer dans le graphique, dans des cercles en pointillés, les références principales que nous utiliserons, en relation avec les exigences pour lesquelles nous les utiliserons. Par exemple, la description comportementale que nous ferons des exigences dans* KHI *est inspirée de celle de* KAOS. *Cette notation graphique ne correspond à rien en* KHI.

Le *problème de l'assignation* sera d'abord exprimé dans un langage de modélisation (ce sera le langage KHI), puis cette modélisation sera instanciée dans un formalisme à même d'automatiser la résolution du *problème de l'assignation* (la logique USL_{KHI}).

Les exigences à remplir par KHI concernent notamment les concepts d'*agents* qu'il devra utiliser et le mécanisme de *spécifications des opérations* qu'il proposera. Concernant les agents, on devra disposer de *deux concepts distincts d'acteurs et de rôles*. On devra prendre en compte les effets de bord de l'action d'agents qui ne sont pas conçus et programmés pour le système

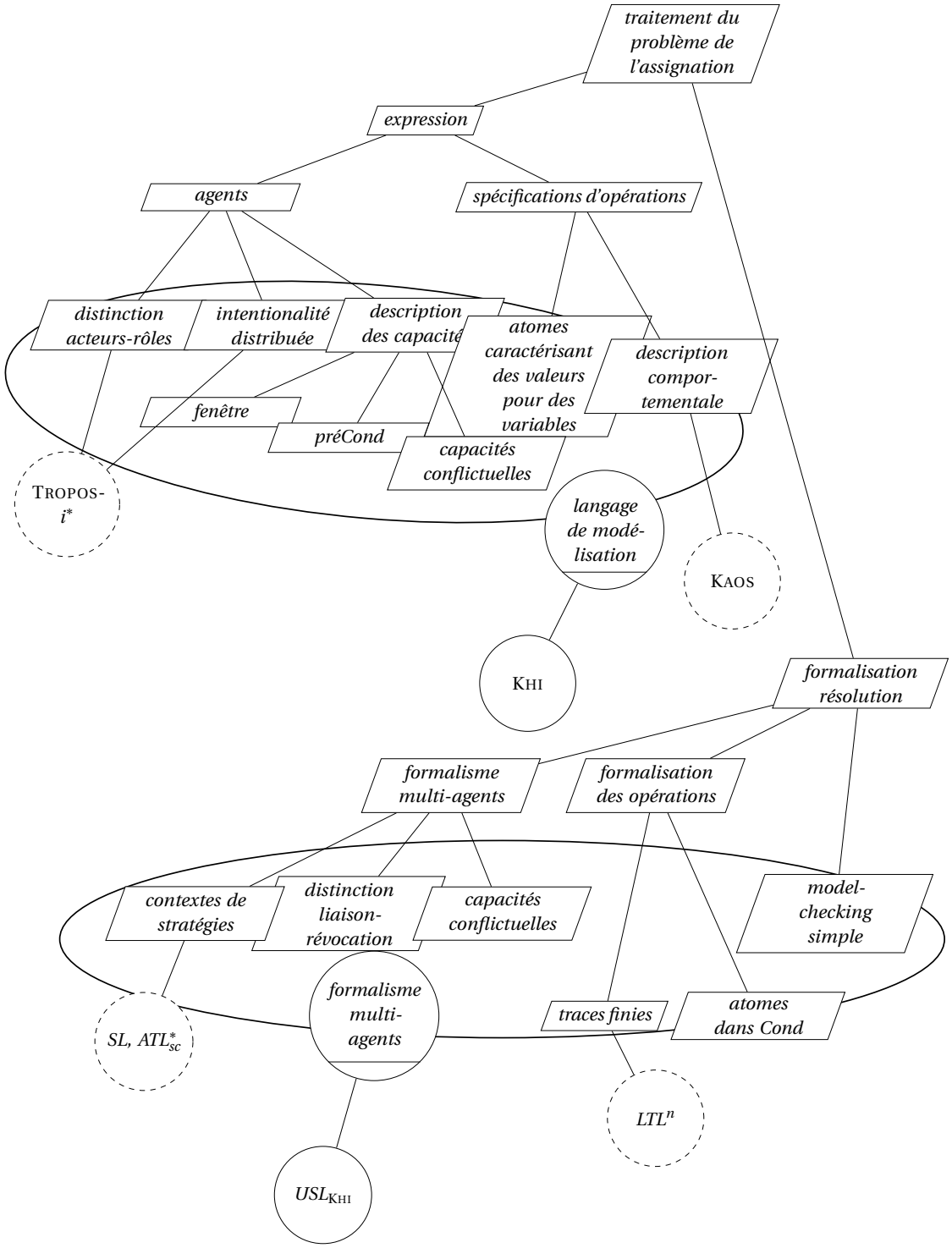

FIGURE 2.1 – Buts à remplir pour le traitement du *problème de l'assignation*

(exigence d'*intentionalité distribuée*). Ces deux aspects sont inspirés de TROPOS-i^*. On fournira également des mécanismes de *description des capacités* de ces agents qui soient bornées par des

fenêtres et pré-conditionnés. Ces capacités pourront être *conflictuelles*.

Les opérations seront spécifiées dans un formalisme qui, à la manière de KAOS, les décrit dans leur *comportementalité*. Ces spécifications utiliseront des propositions dans *Cond*. La description de KHI est l'objet du Chapitre 3. Elle aboutit à l'expression de plusieurs critères de correction pour une assignation de rôles à des coalitions d'acteurs.

Le langage nécessaire à la formalisation de ces critères et à la résolution du *problème de l'assignation* est une logique *multi-agents* avec une sémantique à *contextes*. Elle devra permettre de formaliser des situations où des agents adoptent cumulativement plusieurs stratégies sans les révoquer au fur et à mesure (exigence de *distinction liaison-révocation*) et des *capacités conflictuelles*. Il faudra par ailleurs permettre une *formalisation des opérations* telle que les opérateurs temporels décrivent des *exécutions possiblement finies* et qui utilise les *propositions atomiques de* Cond. Enfin, pour permettre la résolution du problème de l'assignation, quand il sera formalisé dans USL_{KHI}, ce langage doit permettre un traitement du *model-checking* qui soit le plus simple possible.

La proposition formelle que nous ferons, USL_{KHI}, est une instanciation d'un formalisme à portée plus général, USL^{cf}. USL^{cf} s'interpréte sur une classe plus large de modèles, de manière à pouvoir formaliser des *capacités conflictuelles* pour des agents. USL^{cf} est donc lui-même une extension d'USL, un formalisme multi-agent interprété sur des CGSs. La présentation d'USL et d'USL^{cf} fait l'objet des Chapitres 4 et 5. L'instanciation d'un modèle de KHI en une instance d'USL_{KHI} sera ensuite décrite dans le Chapitre 6.

La formalisation et la résolution du *problème de l'assignation*, seront également traités dans le Chapitre 6, selon les différents critères de correction exposés dans le Chapitre 3.

Chapitre 3

Un langage de modélisation : KHI

Sommaire

3.1 Introduction

Dans ce chapitre, on présente les différents éléments du langage KHI. On en donne aussi une formalisation partielle en LTL. Les principaux objectifs de KHI se résument ainsi :

- Il s'agit premièrement de reprendre et de prolonger la distinction entre acteurs et rôles telle qu'elle a été observée dans le matériel des engagements et des protocoles (Section 1.2.3), dans un langage qui intègre une spécification comportementale des exigences, telle qu'on l'a décrite pour KAOS.
- Il s'agit également de permettre l'expression, la formalisation et la résolution du *problème de l'assignation.*

Une première version de KHI a été décrite dans [CBC11, CBC12]. Dans KHI, la distinction entre acteurs et rôles s'étend aux langages formels utilisés pour leur description. On parle ainsi de deux langages distincts :

– Un langage descriptif. On l'appelle le langage des moyens. Les énoncés de ce langage sont des descriptions du domaine et des acteurs présents.

– Un langage prescriptif. On l'appelle le langage des exigences. C'est dans ce langage que les buts poursuivis et les opérations à réaliser sont formalisés. On y décrit aussi les différents rôles qui doivent être joués.

Chacun de ces langages a un concept d'agent (les acteurs pour les moyens et les rôles pour les exigences). Chaque langage est aussi identifié par un ensemble de variables :

– Var(M) désigne l'ensemble des variables des moyens

– Var(E) désigne l'ensemble des variables des exigences.

Ces deux langages sont liés d'une part par l'assignation, qui assigne les rôles à des (coalitions d') acteur(s), d'autre part par des *affectations* des moyens, qui sont des fonctions partielles des variables des moyens dans les ensembles de variables des exigences.

Le métamodèle de KHI est donné dans la Figure 3.1. Un modèle comprend un ensemble de *propriétés du domaine*, qui sont une description partielle de l'environnement dans lequel le modèle est élaboré.

On parcourt maintenant le métamodèle à partir du concept d'*acteur* : un *acteur* recherche la satisfaction de *buts*. L'ensemble des *buts* recherchés par chaque acteur est structuré, étant données des *propriétés du domaine*, par la relation *raffine*. Étant données, à nouveau, des *propriétés du domaine*, les *buts* de plus bas niveau (*i.e.* les exigences) sont *réalisés* par des *opérations*.

La prise en compte des propriétés du domaine dans la relation raffine constitue une différence par rapport à KAOS. Dans KHI en effet, on considère que la satisfaction des exigences par les opérations peut utiliser certaines propriétés du domaine. Dans le cas des observations par satellites par exemple, la propriété du domaine *domPropMouv* décrit le mouvement des satellites sur leurs trajectoires. Elle assure en particulier que toute zone susceptible d'être photographiée est en effet survolée par chaque satellite. Cette propriété est nécessaire pour assurer qu'une opération *prisePhoto* permette de prendre une photographie d'une zone donnée, quelle que soit cette zone.

La relation *réalise* est par ailleurs paramétrée par une *contrainte* qui contient les trois types de conditions requises de KAOS. Les ensembles d'opérations sont *pourvus* par des *rôles* et les rôles sont assignés, non pas directement à des acteurs mais à des coalitions, dans lesquelles un ensemble d'acteurs s'*allient*. Étant données des *propriétés du domaine*, cette coalition, si l'assignation est correcte, est en mesure de *jouer* ce rôle.

Les *acteurs* sont par ailleurs décrits par un certain nombre de *capacités*, chaque capacité étant décrite par une *préCondition d'exercice* et par une *fenêtre*.

Le circuit qui mène de l'identification des buts pour des acteurs à leur satisfaction par des coalitions décrit une boucle. Cette boucle passe par les opérations et les rôles. La correction générale du système dépend donc de la correction des trois relations suivantes :

– la relation raffine, qui structure l'ensemble des buts,

– la relation réalise, qui traduit la réalisation de ces buts en termes d'opérations,

– la relation peut, qui confie cette réalisation à des ensembles d'acteurs.

Parmi ces trois relations, les deux premières font déjà l'objet d'un critère de correction dans KAOS, comme nous l'avons vu dans le Chapitre 1. On adaptera ces critères à KHI. L'enjeu de la résolution du *problème de l'assignation* est de fournir un critère de correction pour la troisième relation, peut. On disposera ainsi d'une procédure formalisée et entièrement vérifiable, pour l'ensemble de la boucle qui mène de l'identification des buts à leur réalisation.

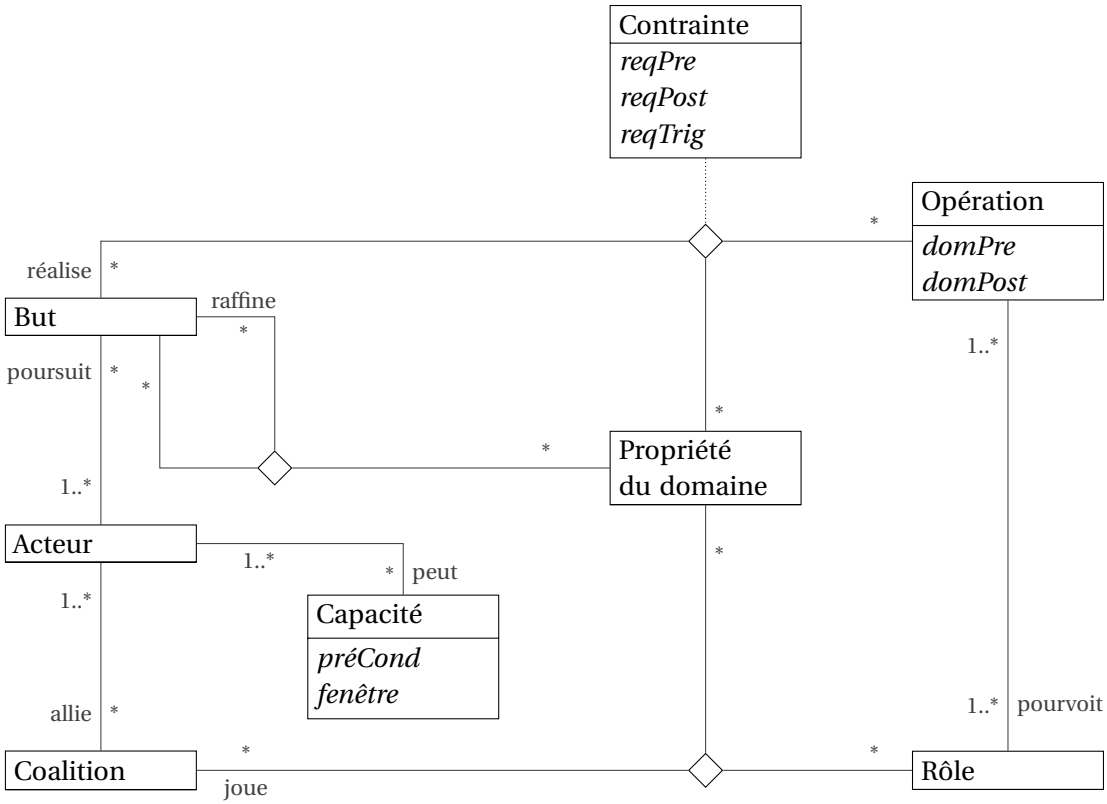

FIGURE 3.1 – Métamodèle du langage KHI

Remarque 3.1. *Le parcours dans le métamodèle de la Figure 3.1, qui mène des acteurs aux acteurs, en passant par le traitement complet des buts, contient en fait deux autres arêtes. Ce sont les relations pourvoit, entre un rôle et des opérations et allie, par laquelle des acteurs forment une coalition. Ces deux relations ne sont que des rassemblements d'éléments dans des ensembles (un rôle est un ensemble d'opérations contraintes et une coalition est un ensemble d'acteurs). Leur correction ne pose donc aucun problème. On relève également que les coalitions au sens de la modélisation sont des ensembles* non vides *d'acteurs, alors que les coalitions des logiques formelles peuvent éventuellement être vides.*

Dans la Section 3.2, on présente les buts ainsi que les variables qui sont utilisées et les propriétés du domaine. Les opérations et l'opérationnalisation sont définies dans la Section 3.3. On présente ensuite les différents concepts d'agents utilisés dans KHI et leurs caractéristiques (Section 3.4). Ces éléments permettent de modéliser et d'exprimer le problème de l'assignation. On élabore ainsi, dans la Section 3.5, différents critères de correction pour une assignation donnée. Ces critères sont autant de manières d'exprimer le problème de l'assignation. On conclut ce chapitre en observant la nécessité d'une logique adéquate pour vérifier la satisfaction de ces critères.

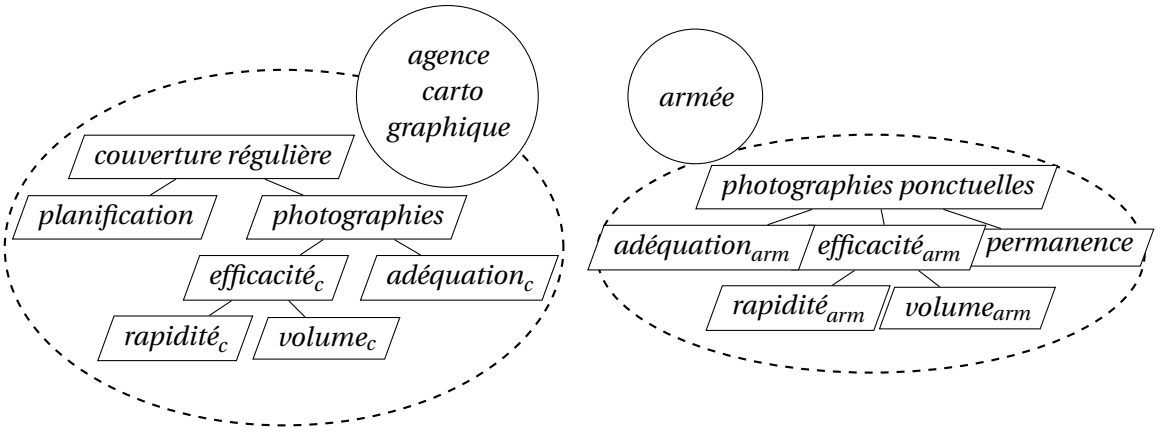

FIGURE 3.2 – Modèle de buts pour l'*agence cartographique* et l'*armée*

3.2 Buts, variables et propriétés du domaine

Le traitement des buts dans KHI emprunte à la fois à TROPOS-i^*et à KAOS. Dans une instance Kh de KHI, les buts sont identifiés par rapport aux acteurs qui ont besoin de les voir satisfaits, puis sont raffinés de manière progressive. Les buts feuilles dans l'arbre ainsi formé sont appelés les *exigences*. Elles sont réalisées par des opérations spécifiées selon les trois types de conditions requises de KAOS.

3.2.1 Diagrammes de buts

Dans la Figure 3.2, on a représenté les modèles de buts pour une nouvelle version de l'exemple de l'*agence cartographique* : on reprend le modèle de buts pour l'*agence cartographique*, tel qu'il qui a été présenté dans la Section 1.1.2. Mais on considère également un second modèle de but, pour une puissance militaire : l'*armée*. Tout comme l'*agence cartographique*, l'*armée* a besoin d'un dispositif de prise et de traitement d'images satellites. Contrairement à l'*agence cartographique*, il s'agit d'un dispositif exceptionnel, qui ne sera sollicité qu'en cas de crise nécessitant un diagnostic sur une région particulière. Ce but principal pour l'*armée* est appelé *photographies ponctuelles*. En cas de nécessité, il faudra que les images fournies par le dispositif soient adéquates à cette nécessité. Il faudra également que la réponse apportée soit efficace. C'est-à-dire, comme dans le modèle de buts pour l'*agence cartographique*, qu'elle réponde aux mêmes exigences d'*adéquation* et de *volume*. Enfin, entre chaque utilisation, il faudra que le système se maintienne dans un état où il est prêt à répondre à toute nécessité : ceci se modélise par l'introduction d'un but de *permanence*. Dans la suite de ce mémoire, on appellera *modèle des missions satellites*, cette extension dans KHI de l'exemple de l'*agence cartographique*. On le désigne par ailleurs par $\mathsf{Kh}_{C,A}$.

3.2.2 Variables

De même que dans KAOS, les buts de plus bas niveau sont formalisés dans une instance de LTL. On munit par ailleurs les variables utilisées dans le langage d'étiquettes. Elles permettent de les distinguer selon leur comportement. Une version plus rigoureuse et complète de ce travail devrait définir des structures de données. Cet aspect n'a cependant pas été suffisamment développé ici. Il serait à introduire dans le cadre de futur travaux sur KHI. Dans la présentation que

nous faisons ici de KHI, chaque étiquette est associée à un domaine de variation, c'est-à-dire à un ensemble de valeurs possibles pour les variables qui ont cette étiquette.

Définition 3.1 (Etiquettes de variables)**.** *Une* étiquette de variables lab *caractérise un ensemble de valeurs bien ordonnées par la relation* $\leqslant$ *(le* domaine de l'étiquette, *noté* $dom(lab)$*). On utilise également la notation* $[x_i : lab]$ *pour :* x_i *est une variable étiquetée* lab*. Soit par ailleurs une variable* x*, on note* $lab(x)$ *l'étiquette* lab *telle que* $[x : l]$*. Cette notation s'étend à des ensembles de variables. Soit donc* X *un ensemble de variables étiquetées, on note* $lab(X)$ *l'ensemble des étiquettes* lab *tel qu'il y a une variables* x_i *dans* X *telle que* $[x_i : lab]$*.*

Pour un ensemble de variables étiquetées X, on définit l'ensemble de propositions $\text{Cond}(X)$.

Définition 3.2 (L'ensemble de propositions Cond (X))**.** *Soit* X *un ensemble de variables étiquetées, l'ensemble de propositions Cond sur* X *(on écrit Cond* (X)*) est donné par la grammaire suivante :*

$$\varphi := x \sim n \mid x - y \sim n \mid x + y \sim n \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi$$

où $x, y \in X$*,* $lab(x) = lab(y) = l$*,* n *est une constante dans* $dom(l)$*, et* $\sim \in \{<, >, =, \leqslant, \geqslant\}$*.*

On remarque que l'ensemble $\text{Cond}_{\mathbb{Z}}$ utilisé dans le Chapitre 1 est le cas particulier de $\text{Cond}(X)$ où toutes les variables dans X ont pour domaine $\mathbb{Z}$.

On peut dès lors définir le langage LTL_{KHI}, qui est utilisé pour la formalisation des buts et des opérations de KHI.

Définition 3.3 (LTL_{KHI})**.** *Soit* X *un ensemble de variables étiquetées, le langage* $LTL_{\text{KHI}}(X)$ *est l'instance de LTL dont les atomes sont des éléments de* $Cond(X)$*.*

Dans une instance Kh de KHI, l'ensemble des variables X est par ailleurs muni d'un ensemble d'axiomes $Ax(X)$. Ces axiomes décrivent les relations qui lient entre elles les différentes variables. Ils sont donnés dans le langage $\text{LTL}_{\text{KHI}}(X)$.

Dans l'exemple des missions d'observation, on utilise les deux étiquettes de variables **position** et **image** :

Une variable étiquetée position encode une coordonnée spatiale ou la position terrestre d'une zone à photographier. Les coordonnées sont schématisées par une position dans l'intervalle $[0, \ldots 5]$ dans $\mathbb{N}$. Le domaine de l'étiquette **position** contient aussi la valeur particulière -1, qui sert à encoder une absence de signal porté par la variable qui a cette valeur : $dom(\textbf{position}) = [-1, \ldots 5]$. Pour chaque variable x_i étiquetée **position**, on introduit deux autres variables $x_i.old$ et $x_i.succ$, ainsi que des axiomes tels que :

- $x_i.old$ encode la valeur de x_i dans l'état précédent.
- $x_i.succ$ a pour valeur la position suivant la valeur de x_i sur la trajectoire des satellites. Elle sert en particulier à la description, dans les moyens, du mouvement des satellites. L'axiomatisation est donnée dans le Tableau 3.1.

Une variable x_i étiquetée image encode une image transmise ou stockée. Le domaine de l'étiquette **image** est un ensemble fini de valeurs noté $\mathscr{I}$. Pour chaque variable x_i étiquetée **image**, on introduit deux autres variables $x_i.old$ d'étiquette **image** et $x_i.pos$ étiquetée **position**. L'axiomatisation de $x_i.old$ est similaire à celle pour les variables étiquetées **position** (voir Tableau 3.2). La variable $x_i.pos$ contient, pour une image x_i, la **position** de laquelle elle a été prise ou doit être prise. Elle est telle que deux images identiques correspondent à la même position.

$[x_i.old : \textbf{position}]$
$Form(ax(x_i.old)) \triangleq \{\Box(x_i = k \rightarrow \mathsf{X}(x_i.old = k))\}_{k\in[-1,...5]}$
$[x_i.succ : \textbf{position}]$
$Form(ax(x_i.succ)) \triangleq \{\Box(x_i = k \rightarrow x_i.succ = k+1[\text{mod}6])\}_{k\in[-1,...5]}$

TABLE 3.1 – Axiomes pour $x_i.old$ et $x_i.succ$, $[x_i : \textbf{position}]$

$[\textbf{image}.old : \textbf{position}]$
$Form(ax(x_i.old)) \triangleq \{\Box(x_i = k \rightarrow \mathsf{X}(x_i.old = k))\}_{k\in[-1,...5]}$
$[\textbf{image}.pos : \textbf{position}]$
$Form(ax(x_i.pos)) \triangleq \Box(x_i = x_j \rightarrow x_i.pos = x_j.pos)$

TABLE 3.2 – Axiomes pour $x_i.old$ et $x_i.pos$, $[x_i : \textbf{image}]$

On peut dès lors définir une affectation d'un ensemble de variables à un autre ensemble de variables.

Définition 3.4 (Affectations)**.** *Soient X_1 et X_2 deux ensembles de variables. Une affectation de X_1 dans X_2 est une fonction* Aff *de X_1 dans $\mathscr{P}(X_2)$ telle que pour toute $x_i \in X_1$, $lab(\mathsf{Aff}(x_i)) = lab(x_i)$. Par la suite, étant données une formule φ et une affectation* Aff *de* Var(φ) *dans un ensemble de variables quelconque, on note $\varphi[\mathsf{Aff}]$ l'ensemble des formules obtenues de φ en y remplaçant toute occurrence de variable x par une variable dans* Aff(x).

Cette notion permettra de traduire les capacités des acteurs, décrites dans le langage des moyens (cf. Section 3.4.2), en ressources pour réaliser les spécifications décrites dans le langage des exigences.

Dans notre étude, on utilisera des affectations de variables de Var(M) dans Var(E) (on parle alors d'*affectation des moyens*). On constate en particulier que, par cette définition, l'action d'un acteur sur une variable des moyens peut être affectée à plusieurs variables des exigences.

3.2.3 Modèles pour LTL_{KHI}

Pour tout ensemble de variables X, $\text{LTL}_{\text{KHI}}(X)$ est interprété dans des modèles de Kripke dont chaque état est donné par une valuation possible pour les variables dans X, selon leur étiquette.On introduit d'abord une notation pour désigner l'ensemble des valuations possibles pour un ensemble de variables étiquetées :

Notation 3.1. *Soit X un ensemble de variables étiquetées. On note $dom(X)^X$ l'ensemble des valuations possibles des variables de X dans leurs domaines respectifs.*

Soient maintenant X un ensemble de variables étiquetées et p une proposition dans $Cond(X)$. Pour tout $s \in dom(X)^X$, on note $s \models_{Cond} p$ si l'assignation des variables de X contenue dans s rend vraie la proposition p, avec les axiomes de l'arithmétique et des ensembles bien ordonnés. Un modèle pour $\text{LTL}_{\text{KHI}}(X)$ est un modèle de Kripke dont les états sont dans $dom(X)^X$.

Définition 3.5 (Modèles pour LTL_{KHI} (X))**.** *Soit X un ensemble de variables étiquetées. Un modèle de $LTL_{\text{KHI}}(X)$ est un quadruplet $\mathscr{K} = \langle G, s_0, R, val\rangle$ où :*

- $G \subseteq dom(X)^X$
- $s_0 \in G$ *est l'état initial*
- $R \subseteq G \times G$ *est une relation d'*accessibilité
- *pour tout* $s \in G$ *et pour toute* $p \in Cond(X)$, $p \in val(s)$ *ssi* $s \models_{Cond} p$.

Comme nous l'avons mentionné dans la Section 2.5, la formalisation complète de KHI s'effectuera dans un langage multi-agents dans lequel certaines combinaisons d'actions par les agents peuvent arrêter l'exécution du système. Les formules issues de la description des rôles dans KHI seront donc évaluées dans des modèles pouvant contenir des exécutions finies. En particulier, la correction des relations raffine et réalise est évaluée par rapport à la satisfaction dans des exécutions possiblement finies. C'est donc la relation LTL^n de la Définition 2.31 qui est utilisée pour évaluer les formules de LTL^n (on note également $\models_{\text{LTL}_{\text{KHI}}}$).

3.2.4 Propriétés du domaine et propriétés du contexte

Étant donnée une instance Kh de KHI, les *propriétés du domaine* pour Kh sont un ensemble d'énoncés descriptifs portant sur l'environnement du système à élaborer. On utilise couramment l'abréviation *propDomaine* pour les désigner. Les propriétés du domaine sont formalisées en un ensemble $\{Form(pdd)\}_{pdd \in propDomaine}$ de formules de $\text{LTL}_{\text{KHI}}(\text{Var(M)})$.

Soit Kh un modèle de KHI. On désigne l'union des propriétés du domaine et des axiomes des variables des moyens sous le nom de *propriétés du contexte pour Kh*. On utilise également l'abbréviation *propContexte* pour l'ensemble des propriétés du contexte.

Les propriétés du contexte peuvent décrire l'état initial du système, restreindre l'ensemble possible des états du système ou restreindre les exécutions possibles dans ce système. On a la définition suivante :

Définition 3.6 (Propriétés initiales, statiques, et dynamiques du contexte)**.** *Soit Kh un modèle de* KHI*, propContexte l'ensemble des propriétés du domaine de Kh et Var(E) l'ensemble des variables des exigences dans Kh.*

- *On appelle* ensemble des propriétés initiales du contexte *de* KHI*, et on note* $propContexte_i$*, l'ensemble des propriétés du contexte qui décrivent l'état initial du système. Une propriété initiale du contexte pdc est telle que* $Form(pdc) \in Cond$.
- *On appelle* ensemble des propriétés statiques du contexte *de* KHI*, et on note* $propContexte_s$*, l'ensemble des propriétés du contexte qui restreignent l'ensemble des états du système. Une propriété statique du contexte pdc est telle qu'il existe* $\psi \in Cond$ *telle que* $Form(pdc) = \Box(\psi)$.
- *On appelle* ensemble des propriétés dynamiques du contexte *de* KHI*, et on note* $propContexte_d$*, l'ensemble des propriétés du contexte qui restreignent les exécutions possibles du système.*

On a que $propContexte_i \cup propContexte_s \cup propContexte_d = propContexte$.

3.2.5 Relation raffine

Une exigence g de KHI est exprimée formellement par une formule $Form(g)$ de LTL_{KHI}. Comme dans KAOS, la relation de raffinement fait l'objet d'un premier critère de correction. Un raffinement est correct ssi la satisfaction des buts raffinants et des propriétés du domaine a pour conséquence sémantique la satisfaction du but raffiné.

On peut alors définir le critère de correction pour la propriété raffine. Elle est relative à une affectation Aff, qui traduit les propriétés du contexte dans le langage des exigences :

Définition 3.7 (Correction d'un raffinement)**.** *Soient g un but et $\{g_i\}_{i\in I}$ un ensemble de buts. Soit encore $\{pdc_j\}_{j\in J}$ un ensemble de propriétés du contexte tel que $\{g_i\}_{i\in I}$ raffine g étant donné $\{pdd_j\}_{j\in J}$ et soit* Aff *une affectation des moyens. Alors le raffinement est correct ssi :*

$$\{Form(pdc_j)[\mathsf{Aff}]\}_{j\in J}, \{Form(g_i)\}_{i\in I} \models_{LTL_{\mathrm{KH1}}} Form(g)$$

3.3 Opérations et opérationnalisation

Comme dans KAOS, la réalisation des exigences est assurée par des opérations qui décrivent les transitions que le système devra connaître pour assurer la satisfaction des buts. On distingue les mêmes deux types de conditions que dans KAOS pour spécifier les opérations.

- En premier lieu, pour toute opération, deux *conditions du domaine* décrivent les effets que cette opération a sur le système. Il s'agit d'une pré-condition (*domPre*) et d'une post-condition (*domPost*). Elles sont exprimées dans Cond. La définition pour la sémantique d'une opération est donc la même que celle pour KAOS (cf. Définition 1.6).

  Définition 3.8 (Sémantique d'une opération)**.** *Une opération op est définie par une occurence de op.domPre immédiatement suivie d'une occurence de op.domPost :*

  $$Form(op) \triangleq op.domPre \wedge \mathsf{X}\, op.domPost$$

- Par ailleurs, pour toute paire d'une opération et d'une exigence, on définit une *contrainte*. Cette contrainte contient les trois formes de conditions *requises* sur cette opération pour cette exigence : une pré-condition nécessaire, (*reqPre*), une post-condition nécessaire (*reqPost*) et une condition de déclenchement (*reqTrig*). Si elles ne sont pas explicitement spécifiées, *reqPre* et *reqPost* sont $\top$ et *reqTrig* est $\bot$. Les conditions des contraintes sont également exprimées dans Cond.

Remarque 3.2. *Le matériel des spécifications d'opérations diffère légèrement de celui de* KAOS *: on y ajoute en effet de manière explicite l'objet contrainte, qui rassemble les conditions requises sur une opération pour une exigence. On peut ainsi désigner l'ensemble des conditions requises d'une opération pour une exigence.*

L'identification des opérations et des contraintes pour les exigences doit être correcte dans le sens suivant : pour toute exigence g, si chaque opération réalisant g respecte sa contrainte pour g, alors g est réalisé. Pour formaliser ceci, on introduit d'abord un prédicat respecte entre une exigence et une opération, qui affirme que cette opération *respecte* sa contrainte pour cette exigence dans le sens suivant :

- À chaque fois que l'opération est déclenchée,
  - sa *reqPre* est satisfaite dans l'état courant,
  - sa *reqPost* est satisfaite dans le prochain état,
- et l'opération est toujours déclenchée dès que sa *reqTrig* et sa *domPre* sont satisfaites.

Formellement, on a la définition suivante :

Définition 3.9 (respecte)**.** *Soient g une exigence et op une opération qui réalise g, et soit c la contrainte pour g et op. Alors,*

$$\mathsf{respecte}(g, op) \triangleq \Box[(Form(op) \rightarrow c.reqPre) \wedge (Form(op) \rightarrow \mathsf{X}\ c.reqPost) \wedge ((c.reqTrig \wedge op.domPre) \rightarrow Form(op))]$$

On peut alors définir la correction de la réalisation. La réalisation est correcte pour une exigence g avec l'affectation Aff ssi g est réalisée dans tout modèle qui satisfait les propriétés du contexte interprétées par Aff et tel que chaque opération respecte sa contrainte pour g :

Définition 3.10 (Correction d'une réalisation)**.** *Soient g une exigence et $\{op_i\}_{i\in I}$ un ensemble d'opérations. Soit encore $\{pdc_j\}_{j\in J}$ un ensemble de propriétés du contexte tel que $\{op_i\}_{i\in I}$ réalise g étant donné $\{pdc_j\}_{j\in J}$ et soit* Aff *une affectation des moyens.. Alors la réalisation est correcte ssi :*

$$Form(pdc)_{i\in I}[\mathsf{Aff}], \{\mathsf{respecte}(g, op)\}_{j\in J} \models_{LTL^n} Form(g)$$

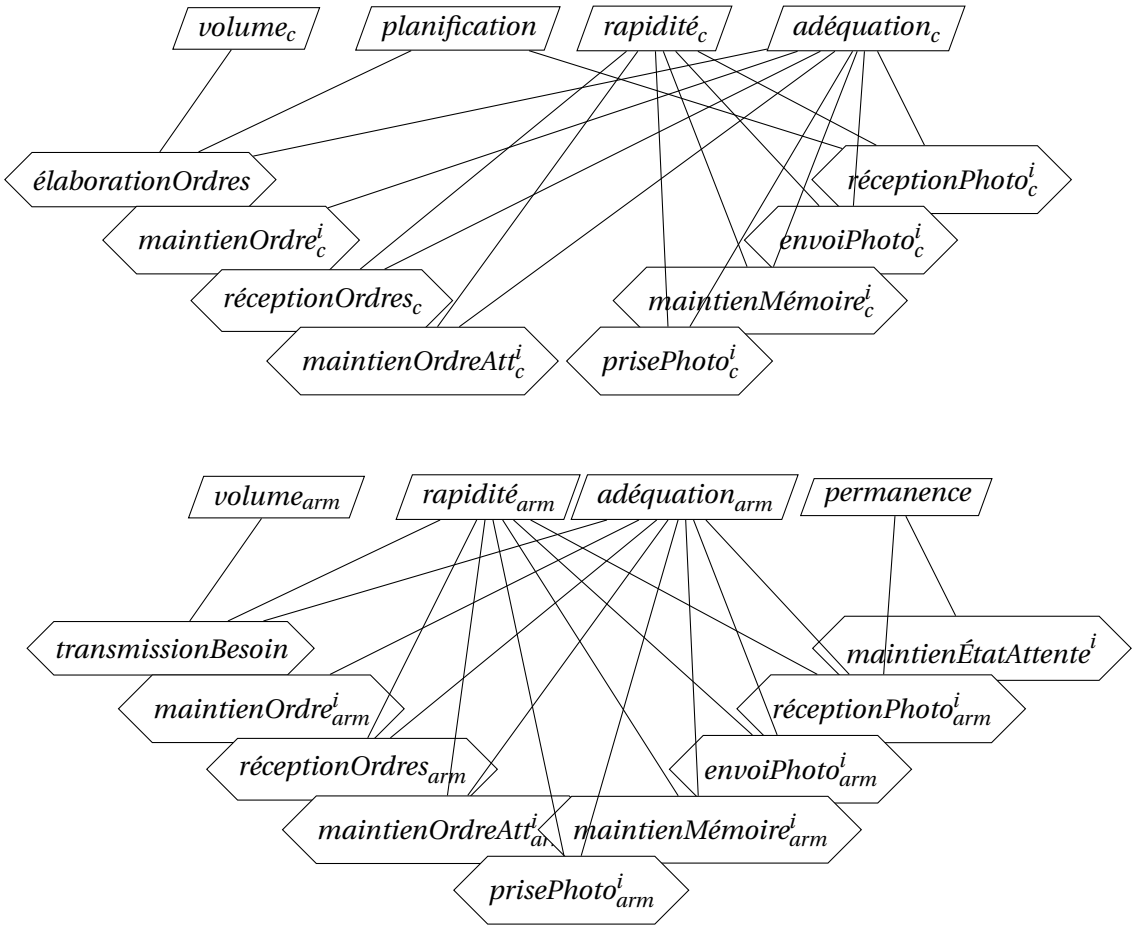

FIGURE 3.3 – Opérations pour les buts de l'*agence cartographique* et de l'*armée* ($i \in \{1,2\}$)

On donne dans la Figure 3.3 les opérations retenues pour satisfaire les exigences de l'*agence cartographique* et de l'*armée*. Leur formalisation et les contraintes qui s'y appliquent sont détaillées dans l'Annexe C.2. On résume ici l'enchaînement de ces opérations.

Pour l'*agence cartographique*, l'exécution du système suit le cours suivant : l'agence élabore des ordres (opération *élaborationOrdres*). Ces ordres sont reçus par des satellites (opération $réceptionOrdres_c$) et des photos qui leur correspondent sont prises (opérations $prisePhoto_c^1$ et $prisePhoto_c^2$). Les satellites envoient ces photos au sol (opérations $envoiPhoto_c^1$ et $envoiPhoto_c^2$). Finalement, l'*agence cartographique* récupère ces photos (opérations $réceptionPhoto_c^1$ et $réceptionPhoto_c^2$), elle élabore de nouveaux ordres, et un nouveau cycle démarre.

À ces opérations s'ajoutent trois mécanismes pour maintenir les valeurs pour des variables. Il s'agit, entre le moment où une variable reçoit une certaine valeur et le moment où cette valeur est utilisée, de la conserver. Pour chacune de ces variables, le mécanisme qui permet cette garantie se modèlise dans le formalisme des opérations, bien qu'il ne s'agisse pas, à strictement parler, d'une opération. Il s'agit d'abord, pour les satellites qui prennent les photos, de conserver la valeur des ordres reçus avant de prendre les photos correspondantes (opérations $maintienOrdreAtt_{act}^i$, pour $act \in \{c, arm\}$, $i \in \{1,2\}$) et de conserver ensuite en mémoire ces images (opérations $maintienMémoire_{act}^i$). Par ailleurs, quand un ordre est envoyé aux satellites, sa valeur est également conservée jusqu'à réception d'une réponse à cet ordre, afin de pouvoir exprimer et vérifier l'*adéquation* de cette réponse (opérations $maintienOrdre_{act}^i$).

La mission pour *armée* suit le même processus, à la différence que l'ordre n'est pas élaboré par le système. Il s'agit d'un besoin manifesté dans l'environnement et transmis aux satellites en tant qu'ordre auquel répondre. L'opération *élaborationOrdres* y est donc remplacée par une opération *transmissionBesoin*, qui consiste à transmettre aux satellites les besoins identifiés en tant qu'ordres. Par ailleurs, le but de *permanence* est assuré par des opérations $maintienÉtatAttente^1$ et $maintienÉtatAttente^2$, déclenchées dès qu'une réponse à un besoin a été apportée, et qui maintiennent le dispositif en attente de nouveaux besoins identifiés.

3.3.1 Cas du but de survie du système

L'interprétation des buts dans KHI est faite grâce à LTL_{KHI}, elle utilise donc des modèles dans lesquels les exécutions ne sont pas nécessairement infinies. Ceci servira notamment à formaliser les possibilités de conflits entre plusieurs acteurs.

Un cas particulier de but pour le système est donc que l'exécution ne s'arrête pas (on parle du *but de survie du système*). Ce but est satisfait dans une exécution ssi elle est infinie, *i.e.* ssi les agents n'y utilisent pas de *capacités conflictuelles*. Dans la pratique il arrive (et c'est notamment le cas pour l'exemple des missions d'observation satellite) que la survie du système soit une conséquence directe de la satisfaction des autres buts. Dans ce cas, il n'y a pas besoin de le mentionner explicitement.

Il peut cependant arriver que la survie à la fois soit requise et ne soit pas assurée par les autres buts. Elle doit alors être intégrée au modèle de buts. Ce but est formalisé par la formule $\Box \mathsf{X} \top$ (voir la Remarque 2.3, Section 2.5). Pour le satisfaire, il suffit que chaque état soit suivi d'un autre état. Plus généralement, la formule $\Box \mathsf{X} \top$ permet de raisonner sur l'absence d'exercice de capacités conflictuelles par les acteurs du système.

3.4 Rôles et acteurs

On s'intéresse maintenant aux deux concepts d'agents mis en jeu dans KHI. On présente donc successivement les rôles (Section 3.4.1) et les acteurs (Section 3.4.2).

3.4.1 Rôles

Les rôles sont les agents en tant qu'entités requises : les agents dans le langage des exigences. Ils sont des descriptions du comportement attendu des acteurs qui les rempliront et sont constitués par des ensembles d'opérations avec leurs contraintes. On dit alors que le rôle *pourvoit* ces opérations.

On a la formalisation suivante en $\mathrm{LTL}_{\mathrm{KHI}}$:

Définition 3.11 (Sémantique d'un rôle)**.** *La sémantique d'un rôle est la conjonction des sémantiques de respect pour chacune des opérations qu'il pourvoit, pour chacune des exigences pour lesquelles cette opération est spécifiée.*

$$Form(rl) \triangleq \bigwedge_{op \in rl.pourvoit} \bigwedge_{g \in op.réalise} \mathsf{respect}(g, op)$$

La répartition des opérations dans des rôles pour le modèle de l'*agence cartographique* et de l'*armée* est représentée dans la Figure 3.4. Pour chacune des missions, un segment embarqué prend en charge toutes les opérations qui relèveront des satellites mis en place. Et un segment sol gère les images reçues. Dans le cas de la mission pour l'*armée*, il s'agit de la transmission des besoins, du maintien en mémoire des ordres ainsi transmis, de la réception des photos et du maintien dans l'état d'attente de nouveau besoins. Le segment sol pour l'agence météo réalise l'élaboration des ordres, garde mémoire de ces ordres envoyés et reçoit les photos.

3.4.2 Acteurs

Les acteurs sont les agents dans le langage des moyens. D'une part ils définissent l'ensemble des buts à satisfaire. D'autre part, ils sont les unités agissantes, susceptibles de permettre ou d'empêcher la satisfaction de ces buts et la correction du modèle. On définit ici les *capacités des acteurs*. Cette définition utilise le fragment de *Cond* (Var(M)) qui définit des intervalles finis, on l'appelle $Cond^{fen}$(Var(M)) :

Définition 3.12 ($Cond^{fen}(X)$)**.** *Soit X un ensemble de variables étiquetées, le langage $Cond^{fen}(X)$ est engendré par la grammaire suivante :*

$$\varphi := a \leqslant x \wedge x \leqslant b \mid \varphi \wedge \varphi \mid \varphi \vee \varphi$$

où $x \in X, a \leqslant b$ et $a, b \in dom(x)$.

On a alors la définition suivante, pour une capacité d'un acteur :

Définition 3.13 (Capacité)**.** *Une capacité cap pour un acteur a est donnée par :*

- *une* pré-condition de capacité *cap.préCond. Elle est donnée dans Cond (Var(M)).*
- *une* fenêtre *cap.fenêtre, qui est un ensemble fini de valeurs pour certaines variables de Var(M). Elle est donnée dans $Cond^{fen}$(Var(M)).*

Le sens d'une capacité est le suivant : en tout état où la *préCond* est satisfaite, l'acteur correspondant peut contrôler les variables de la *fenêtre* dans les limites qui sont définies par cette *fenêtre*. Notre modélisation considère en effet qu'un agent qui peut agir sur des variables n'a pas forcément pour autant un contrôle tel qu'il peut leur donner n'importe quelles valeurs n'importe quand. Les *préCond* permettent donc de définir les conditions dans lesquelles il peut agir sur ces variables, et les *fenêtre* servent à exprimer les bornes dans lesquelles il peut agir dessus.

FIGURE 3.4 – Rôles pour le modèle de l'*agence cartographique* et de l'*armée* ($i \in \{1,2\}$)

L'exemple des missions d'observation met en jeu six acteurs : à l'*armée* et à l'*agence cartographique* elles-mêmes, s'ajoutent un *transmetteur* qui transmettra les besoins en photos identifiés pour l'*armée,* et trois satellites *S1*, *S2* et *S3*. L'*armée* n'a pas de capacité d'action directe sur le système. Les capacités pour les trois satellites Sj sont similaires. On les donne dans le Tableau 3.3 : chacun des satellites Sj a une capacité $enregistrementOrdre^j$ qui lui permet d'enregistrer un ordre de photo à prendre. L'ordre est stocké dans la variable $tâcheSatellite^j$, étiquetée **position**. Cette capacité peut être exercée à tout moment, sans condition :

$$enregistrementOrdre^j.préCond \triangleq \top$$

et en enregistrant comme ordre n'importe quelle valeur de $dom(\textbf{position}) \setminus \{-1\}$:

$$enregistrementOrdre^j.fenêtre \triangleq tâcheSatellite^j \geqslant 0 \wedge tâcheSatellite^j \leqslant 5$$

$enregistrementOrdre^j$	*préCond*	$\top$
	fenêtre	$tâcheSatellite^j \in [0,\ldots,5]$
$\{prisePhoto^j\}_{k\in[0,\ldots,5]}$	*préCond*	$positionSatellite^j = k$
	fenêtre	$mémoireSatellite^j.pos = k+1\,[mod 6]$
$libérationMémoireSat^j$	*préCond*	$mémoireSatellite^j.pos \geqslant 0$
	fenêtre	$mémoireSatellite^j.pos = -1$
$signal^j$	*préCond*	$\top$
	fenêtre	$signalSatellite^j \in \mathscr{I}$

TABLE 3.3 – Capacités de Sj, $j \in \{1,2,3\}$

Il a aussi la capacité de prendre une photo à tout moment. La photo enregistre alors une valeur correspondant à la position du satellite à l'instant suivant le déclenchement de cette opération. La capacité $libérationMémoireSat^j$ consiste à vider de son contenu la variable $mémoireSatellite^j$, donc à donner à $mémoireSatellite^j.pos$ la valeur -1. Le satellite Sj peut la déclencher à tout instant où cette variable est non vide. Enfin, par la capacité $signal^j$, le satellite Sj peut à tout moment envoyer des signaux par les canaux $signalSatellite^j$. Ils peuvent alors leur donner n'importe quelle valeur dans $\mathscr{I}$.

Remarque 3.3. *Les capacités des acteurs ne sont pas formalisées par des formules logiques, comme l'ont été les différents éléments du langage rencontrés jusqu'ici. Comme nous le verrons dans le Chapitre 6, leur prise en compte formelle s'opére sur le plan sémantique : l'ensemble des capacités des acteurs agissants d'un modèle servira à définir les actions possibles des agents dans les modèles utilisés pour la résolution du* problème de l'assignation.

La notion de *capacité conflictuelle* peut désormais recevoir une définition formelle. Il s'agit du cas où deux acteurs ont des capacités dont les *préCond* sont compatibles et qui leur permettent de modifier de manière différente la valeur d'une même variable. On apporte ainsi une réponse à l'exigence *capacités conflictuelles* relevée dans la Figure 2.1. On introduit d'abord les notations suivantes :

Notation 3.2.

- *Pour toute capacité cap, pour toute variable* $x_i \in \mathsf{Var}(cap.fenêtre)$, $cap.fenêtre_{\restriction x_i}$ *désigne l'ensemble des valeurs pour* x_i *rendues possibles par cap.*
- *Pour tout agent a, on désigne par* $\mathsf{Var}(a)$ *l'ensemble des variables utilisées dans la définition des capacités de a :*

$$\mathsf{Var}(a) = \bigcup_{cap\in a.peut} (\mathsf{Var}(cap.préCond \cup cap.fenêtre))$$

Cette notation s'étend à une coalition d'acteurs A :

$$\mathsf{Var}(A) \triangleq \bigcup_{a\in A} \mathsf{Var}(a)$$

On définit alors ainsi les capacités conflictuelles,

Définition 3.14 (Capacités conflictuelles)**.** *Soient deux agents* a_1 *et* a_2*, et soient* cap_1 *et* cap_2 *deux capacités telles que* $cap_1 \in a_1.peut$ *et* $cap_2 \in a_2.peut$*. Ces deux capacités sont conflictuelles ssi :*

– *Les pré-conditions de cap_1 et cap_2 sont compatibles entre elles :*

$$cap_1.préCond \wedge cap_2.préCond \not\models_{LTL^n} \bot$$

– *Il existe une variable $x_i \in Var(M)$ t.q. :*
  – $x_i \in \mathsf{Var}(cap_1.fenêtre) \cap \mathsf{Var}(cap_2.fenêtre)$
  – $|cap_1.fenêtre_{\restriction x_i} \cup cap_2.fenêtre_{\restriction x_i}| > 1$.

Cette définition se généralise à un ensemble de capacités de cardinal quelconque : soit C un ensemble de capacités. On dit de C que c'est un ensemble de capacités conflictuelle ssi C contient une paire de capacités conflictuelles.

On peut désormais étudier la relation entre les rôles et les acteurs : la relation d'assignation. C'est l'objet de la section suivante.

3.5 Assignation et problème de l'assignation

3.5.1 Introduction, définition informelle du problème

Le *problème de l'assignation* se pose pour l'ingénieur qui a identifié les exigences du système à élaborer et les a constituées en *rôles*. À ce stade, l'ingénieur dispose d'un ensemble d'acteurs et d'un ensemble de rôles à assigner à des coalitions. À la question "quelles coalitions vont jouer les différents rôles ?", il apporte une réponse possible dont il s'agit d'évaluer la pertinence. On appelle cette réponse possible une *assignation* :

Définition 3.15 (relation d'assignation)**.** *Une assignation est une fonction de l'ensemble des rôles dans l'ensemble des coalitions,* i.e. *joue* : *Rôle* $\rightarrow \mathscr{P}_{>0}$(*Acteurs*).

On présente, dans la Figure 3.5, l'assignation pour le modèle $\mathsf{Kh}_{C,A}$: l'*agence cartographique* prend en charge elle-même son rôle de *segment sol*, et celui pour l'*armée* est confiée au *transmetteur*. Les trois satellites disponibles *S1*, *S2* et *S3*, se répartissent les deux rôles de *segments embarqués* de la manière suivante : *segment embarqué carto* est assigné à la coalition $\{S1, S2\}$ et *segment embarqué armée* est assigné à la coalition $\{S2, S3\}$.

Pour évaluer une assignation, on introduit un ensemble de *critères de correction*. On ne donne pour l'instant qu'une définition informelle de ces critères. Leur formalisation sera menée dans le Chapitre 6. Elle nécessite en effet au préalable l'introduction des logiques USL et USL_{KHI}.

3.5.2 Correction locale

Le critère le plus naturel est celui selon lequel tout rôle est assigné à une coalition capable de le remplir. On l'appelle le critère de *correction locale* :

Définition 3.16 (CL, définition informelle)**.** *Un modèle vérifie le critère de correction locale ssi pour tout rôle $\rho_i \in$ Rôle il y a une affectation des moyens telle que les agents dans la coalition ρ_i.peut sont en mesure d'assurer la satisfaction de Form(ρ_i) quelles que soient les actions des autres acteurs. On note alors* CL(*Rôle*).

On fait le commentaire suivant sur ce critère :

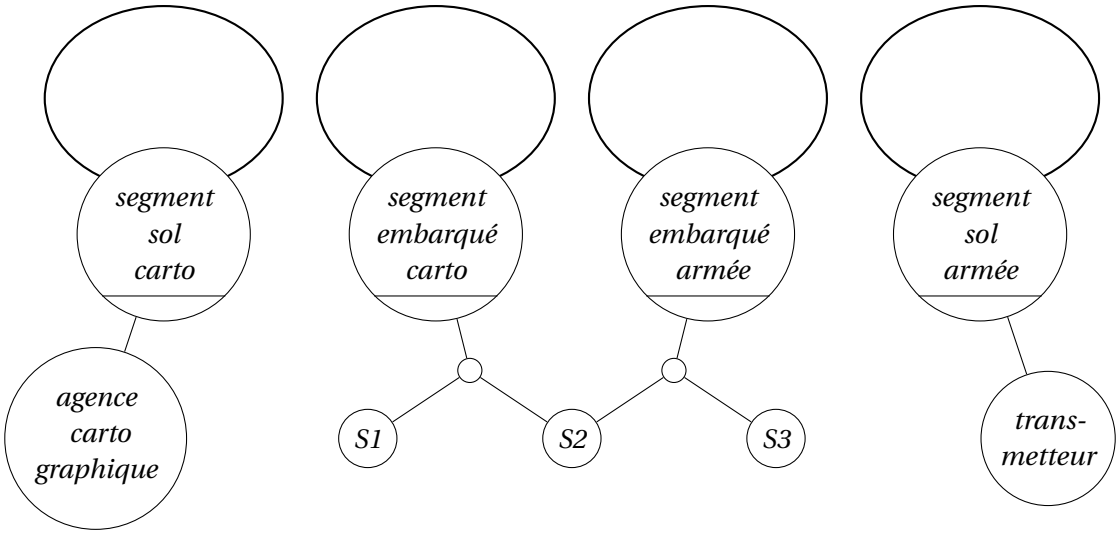

FIGURE 3.5 – Assignation pour le modèle $\mathsf{Kh}_{C,A}$

Commentaire 3.1. *La correction locale permet d'assurer que, isolément, chaque rôle est assigné à une coalition capable de le remplir. On parle de correction* locale *car la vérification se fait indépendamment pour chacun des rôles assignés. Ce critère constitue une condition suffisante pour la réalisabilité du système quand chaque agent est présent dans, au plus, une des coalitions pour un rôle. Il a cependant des inconvénients :*

1. *Si deux rôles distincts ρ_1 et ρ_2 sont assignés à deux coalitions qui ont un agent a en commun (c'est notamment le cas dans l'assignation présentée dans la Figure 3.5, où le satellite S2 participe aux coalitions pour chacun des deux rôles de segments embarqués), alors le critère de correction locale peut être satisfait si a est en mesure séparément de participer à la réalisation du rôle ρ_1 et du rôle ρ_2. Mais la satisfaction de ce critère n'assure pas que a puisse participer à la réalisation de ces deux rôles en même temps. Dans une telle situation, le critère est trop faible (il définit une condition non suffisante pour la correction de la solution d'assignation).*
2. *Par ailleurs, selon ce critère, chaque coalition doit être en mesure d'assurer son rôle de manière complétement autonome. En ce sens, le critère définit une condition qui peut également être inutilement forte. Elle ne tire en effet aucun profit des éventuelles interactions entre différentes coalitions. Il peut arriver par exemple que, étant donnés deux rôles ρ_1 et ρ_2 assignés respectivement aux coalitions A_1 et A_2, la coalition A_1 soit en mesure de satisfaire ρ_1 seulement moyennant une condition φ. Alors si A_2 assure cette condition φ, la solution d'assignation $(\rho_1 \mapsto A_1, \rho_2 \mapsto A_2)$ est correcte. Cependant, elle ne vérifie pas le critère de correction locale.*
3. *Enfin, l'utilisation de ce critère suppose que l'ingénieur qui utilise le langage peut disposer de toutes les capacités d'actions des acteurs. Il ignore donc les effets de bords des actions d'éventuels acteurs présents dans le système mais non contrôlés par cet ingénieur. On perd donc une partie du bénéfice de la prise en compte de l'intentionalité distribuée (cf. Section 1.2.1).*

On cherchera donc, pour les modèles qui le nécessitent, des critères de correction plus

adaptés. Le critère de *correction globale* répond au premier item de ce commentaire. La relation de *collaboration* permet de tirer parti des interactions entre coalition et répond au deuxième item. Enfin, la relation de contribution permet de modéliser les effets de bord des acteurs qui ne sont pas contrôlés par l'ingénieur qui utilise le langage.

3.5.3 Correction globale

Le critère de *correction globale* permet de prendre en compte des cas d'assignations où un agent est membre de plusieurs coalitions : il est respecté ssi l'ensemble des agents concernés par l'assignation peut agir de manière à ce que chaque coalition assure les rôles qui lui sont assignés.

Définition 3.17 (CG, définition informelle)**.** *Un modèle vérifie le critère de correction globale si et seulement s'il y a une affectation des moyens telle que pour tout rôle ρ de Rôle, les agents dans ρ.peut sont en mesure d'assurer la satisfaction de Form(ρ). On note alors* CG(*Rôle*).

3.5.4 Interactions entre des coalitions

Les critères CL et CG représentent des conditions pour une autonomie absolue de chaque coalition pour assurer son ou ses rôles. Dans le cas où une assignation ne satisfait pas CG, il est possible d'assouplir cette condition d'autonomie en acceptant des assignations qui utilisent des interactions entre des coalitions. On cherchera donc des critères de correction qui prennent en compte ces interactions. On définit ainsi, de manière informelle, deux types d'interactions entre les coalitions : la *collaboration* (Section 3.5.4.1) et la *contribution* (Section 3.5.4.2).

3.5.4.1 Collaboration

Tout d'abord, on s'intéresse aux cas où une coalition est en mesure d'assurer son propre rôle tout en rendant capable une autre coalition d'assurer le sien. Un critère de correction avec *collaboration* (Col) utilisera ainsi la relation suivante :

Définition 3.18 (Collaboration entre deux rôles, définition informelle)**.** *Soit Kh un modèle de* KHI*, soient deux rôles ρ_1 et ρ_2 dans Kh et soient $A_1 = \rho_1$.peut et $A_2 = \rho_2$.peut. ρ_1* collabore *avec ρ_2 ssi il y a une affectation* Aff *des moyens telle que A_1 peut agir dans le système de manière à assurer Form(ρ_1) tout en permettant à A_2 d'agir de manière à assurer Form(ρ_2). On note alors* Col(ρ_1, ρ_2).

Cette définition s'étend à des ensembles de rôles, de différentes manières. On donne ci-dessous les définitions pour la collaboration, locale et globale, d'un ensemble de rôles à un autre ensemble de rôles :

Définition 3.19 (Collaboration entre deux ensembles de rôles, définition informelle)**.** *Soit Kh une instance de* KHI*, et soient $Rôle_1$ et $Rôle_2$ deux ensembles de rôles dans Kh.*

- *On dit que les rôles dans $Rôle_1$* collaborent localement *aux rôles dans $Rôle_2$ ssi il y a une affectation des moyens telle que les agents dans $Role_1$.peut peuvent agir dans le système de manière à ce que*
  - *chaque rôle $\rho \in Rôle_1$ soit satisfait,*
  - *chaque coalition ρ'.peut pour $\rho' \in Rôle_2$ peut agir de manière à assurer Form(ρ'). On note alors* $\text{Col}_L(Rôle_1, Rôle_2)$.

– *On dit que les rôles dans* $R\hat{o}le_1$ collaborent globalement *aux rôles dans* $R\hat{o}le_2$ *ssi il y a une affectation des moyens telle que les agents dans* $Role_1.Act$ *peuvent agir dans le système de manière à ce que :*
  – *chaque rôle* $\rho \in R\hat{o}le_1$ *soit satisfait,*
  – *les agents dans* $Role_2.Act$ *peuvent agir dans le système de manière à ce que chaque coalition* $\rho'.peut$ *pour* $\rho' \in R\hat{o}le_2$ *assure la satisfaction de la formule* $Form(\rho')$

  On note alors $\mathsf{Col}_G(R\hat{o}le_1, R\hat{o}le_2)$.

3.5.4.2 Contribution

Prendre en compte la collaboration entre deux coalitions permet d'accepter comme correctes plus d'assignations qu'en employant le critère de correction globale. Le critère décrit dans la Définition 3.18 est utile si A_1 et A_2 sont deux coalitions d'acteurs dont le comportement est spécifié par un même ingénieur (ou par des ingénieurs qui s'accordent sur le comportement de ces acteurs). Alors cet ingénieur peut décider le comportement pour A_1 qui soit le plus favorable, à la fois directement à la satisfaction de ρ_1 et à ce que A_2 soit en mesure d'assurer ρ_2.

Supposons par contre que les comportements des acteurs dans A_1 et A_2 soient spécifiés par deux ingénieurs différents (et qui ne se concertent pas sur ces comportements).

Dans ce cas, l'ingénieur qui contrôle les acteurs dans A_2 doit anticiper le comportement des acteurs dans A_1. Toutefois, on ne doit pas forcément considérer A_1 comme une coalition d'agents malveillants : il s'agit d'acteurs en interactions avec le système et qui agissent pour la satisfaction de leurs propres buts, ici exprimés comme la satisfaction du rôle ρ_1.

La question pertinente, pour la faisabilité du système, est donc de savoir si tous les comportements de A_1 par lesquels elle assure la satisfaction de ρ_1 sont des comportements qui permettent à A_2 d'assurer ρ_2.

Un critère de correction avec contribution utilisera la relation suivante :

Définition 3.20 (Contribution entre deux rôles, définition informelle)**.** *Soient* ρ_1 *et* ρ_2 *deux rôles, et soient* $A_1 = \rho_1.peut$ *et* $A_2 = \rho_2.peut$. *On dit que* ρ_1 contribue *à* ρ_2 *ssi pour toute affectation* Aff_1 *de* $\mathsf{Var}(A_1)$ *, il y a une affectation* Aff_2 *de* $\mathsf{Var}(A_2)$ *telle que, quoi que fassent les acteurs dans* A_1*, s'ils garantissent* $Form(\rho_1)$*, alors les agents dans* A_2 *peuvent garantir* $Form(\rho_2)$. *On note alors* $\mathsf{Contr}(\rho_1, \rho_2)$.

Cette définition s'étend également à des ensembles de rôles, de différentes manières. On donne ci-dessous les définitions pour la contribution d'un ensemble de rôles, locale et globale, à un autre ensemble de rôles :

Définition 3.21 (Contribution entre deux ensembles de rôles, définition informelle)**.** *Soit Kh une instance de* KHI. *Soient* $R\hat{o}le_1, R\hat{o}le_2 \subseteq R\hat{o}le$, $A_1 = R\hat{o}le_1.joue$ *et* $A_2 = R\hat{o}le_2.joue$.

– *On dit que* $R\hat{o}le_1$ contribue localement *à* $R\hat{o}le_2$ *dans Kh ssi pour toute affectation* Aff_1 *de* $\mathsf{Var}(A_1)$ *, quoi que fassent les agents dans* A_1*, s'ils garantissent* $Form(\rho)$ *pour tout* $\rho \in R\hat{o}le_1$*, alors pour tout* $\rho' \in R\hat{o}le_2$ *il y a une affectation* $\mathsf{Aff}_{\rho'}$ *de* $\mathsf{Var}(\rho'.peut)$ *telle que* $\rho'.peut$ *peut garantir* $Form(\rho')$. *On note alors* $\mathsf{Contr}_L(R\hat{o}le_1, R\hat{o}le_2\})$.
– *On dit que* $R\hat{o}le_1$ contribue globalement *aux rôles dans* $R\hat{o}le_2$ *ssi pour toute affectation* Aff_1 *de* $\mathsf{Var}(A_1)$*, quoi que fassent les agents dans* A_1*, s'ils garantissent* $Form(\rho)$ *pour tout* $\rho \in R\hat{o}le_1$*, alors il y a une affectation* Aff_2 *de* $\mathsf{Var}(A_2)$ *telles que les acteurs dans* A_2 *peuvent jouer de manière à ce que pour tout* $\rho' \in R\hat{o}le_2$*, la coalition* $\rho'.peut$ *garantisse* $Form(\rho')$. *On note alors* $\mathsf{Contr}_G(R\hat{o}le_1, R\hat{o}le_2)$.

Dans les chapitres qui suivent, on décrit les formalismes qui permettent d'exprimer et de vérifier ces critères et ces relations : dans le Chapitre 4 on détaille une version générale de ce formalisme, USL, et on discute plusieurs sémantiques proposées pour USL. Le pouvoir expressif et le *model-checking* de ces différentes sémantiques sont étudiés dans le Chapitre 5. On utilise, pour la résolution du *problème de l'assignation* dans une instance de KHI, une instance particulière d'USL, appelée USL_{KHI}. Elle est décrite dans le Chapitre 6. Le Chapitre 6 décrit également l'utilisation d'USL_{KHI} pour formaliser les critères de correction de l'assignation.

Chapitre 4

Une logique multi-agent : USL

Sommaire

On s'intéresse ici au formalisme, USL, qui va être utilisé pour KHI dans le Chapitre 6.

Le langage USL [CBC13, CBC] va fournir un formalisme duquel on pourra dériver une description des critères de correction pour une assignation de rôles à des coalitions d'acteurs dans KHI, tels que décrit en Section 3.5. Ces critères de correction concernent les capacités croisées de coalitions d'acteurs à remplir des rôles, c'est-à-dire à assurer la satisfaction de formules exprimées en LTL. C'est donc d'une logique temporelle multi-agents que l'on a besoin.

La présentation de ce formalisme couvre les deux prochains chaitres. Dans le chapitre courant, on présente la syntaxe et différentes sémantique pour USL. Le Chapitre 5 décrit certaines des propriétés remarquables exprimables à l'aide d'USL, ainsi que les propriétés métathéoriques de ce formalisme.

4.1 Introduction

Tout d'abord, on décrit dans les paragraphes suivants des exemples minimaux, afin de préciser les exigences que le formalisme devra remplir.

Exemple 4.1 (Aide entre deux coalitions disjointes)**.** *Soient deux coalitions A_1 et A_2 sans acteur commun. Et soient deux rôle ρ_1 et ρ_2. Soit encore joue l'assignation telle que joue$(\rho_1) = A_1$ et joue$(\rho_2) = A_2$. Pour satisfaire ρ_1 (resp. ρ_2), chaque acteur dans A_1 (resp. A_2) suit une certaine stratégie. Comme défini de manière informelle dans la Section 3.5.4.1, pour que joue satisfasse* $\mathsf{Col}(\rho_1, \rho_2)$, *$A_1$ doit pouvoir aider A_2 à remplir son rôle ρ_2. Autrement dit, on ne s'intéresse à la capacité pour A_2 de remplir son rôle que dans des contextes où les agents dans A_1 jouent des stratégies pour ρ_1. On s'intéresse donc à la possibilité de composer la stratégie de A_2 avec celle de A_1.*

Comme illustré dans l'Exemple 4.1, l'expression de Col requiert une sémantique à contextes. Cependant, cet exemple spécifie que les coalitions prises en compte sont disjointes (*i.e.* qu'elles n'ont pas d'acteur en commun). Or l'assignation dans KHI est définie sans cette restriction. On veut pouvoir exprimer la relation de collaboration dans le cas général, *i.e.* tel qu'un acteur peut appartenir à différentes coalitions.

Exemple 4.2 (Aide entre deux coalitions)**.** *Soient maintenant deux coalitions A_1 et A_2 avec au moins l'agent a en commun. Soient toujours deux rôle ρ_1 et ρ_2 et joue l'assignation telle que joue $(\rho_1) = A_1$ et joue $(\rho_2) = A_2$. Comme précédemment, pour satisfaire* $\mathsf{Col}(\rho_1, \rho_2)$, *$A_1$ doit pouvoir aider A_2 à satisfaire son rôle ρ_2. Il doit être possible pour A_2, étant données les stratégies jouées par les agents dans A_1, de trouver des stratégies qui assurent ρ_2 dans ce contexte. Cependant, comme on l'a vu dans la Section 2.4.1, dans les formalismes connus a ne peut pas jouer une stratégie pour ρ_2 sans auparavant révoquer sa stratégie pour ρ_1. Donc, dès lors que deux coalitions ont un acteur en commun, la composition des contextes échoue entre ces deux coalitions dans les formalismes présentés jusqu'ici. Avec USL, on s'intéresse à la possibilité de composer la stratégie de a en tant qu'acteur de A_2 avec celle de a en tant qu'acteur de A_1.*

La situation de l'Exemple 4.2 est en fait simulable avec SL. On peut en effet y exprimer l'existence d'un unique vecteur de stratégies pour les deux coalitions, tel que :

- le seul fait que A_1 joue ces stratégies suffise à garantir la satisfaction de ρ_1,
- si les deux coalitions jouent en même temps ces stratégies alors les deux rôles sont satisfaits.

La formule de SL 4.1 exprime en effet une propriété équivalente à celle qui est décrite dans l'Exemple 4.2. Pour la lire, on introduit des notations pour des assignations collectives de stratégies :

Notation 4.1.

- *$\overrightarrow{x}$ désigne un vecteur de variables indicé par les agents du langage.*
- *Pour une coalition d'agents $A = \{a_1, \dots a_n\}$, $\overrightarrow{x}_A$ est la projection de $\overrightarrow{x}$ sur l'ensemble des variables indicées par un agent de la coalition A.*
- *Enfin $(A, \overrightarrow{x_A})$ abrège la succession de lieurs $(a_1, x_{a_1}), \dots (a_n, x_{a_n})$.*

Cette notation est abusive, comme elle utilise implicitement une fonction des agents de A dans les variables de $\overrightarrow{x_A}$, qui n'est pas définie. On ne s'en servira donc que dans les cas où l'on peut admettre implicitement la définition de cette fonction.

$$\langle\langle \overrightarrow{x} \rangle\rangle (A_1, \overrightarrow{x_{A_1}})(Form(\rho_1) \wedge (A_2, \overrightarrow{x_{A_2}})(Form(\rho_2))) \tag{4.1}$$

Ici, l'agent a, qui est commun aux deux coalitions, joue la même stratégie du début à la fin. La composition n'est donc pas nécessaire et l'évaluation de cette formule n'occasionne pas

de révocation de stratégie. Cependant, cette solution ne convient que dans la mesure où l'on n'exprime rien de particulier concernant la stratégie de a pour A_1. On ne la désigne que dans sa version composée avec la stratégie de a pour A_2. Des propriétés plus fines d'interaction entre les coalitions, mettant par exemple en jeu plus de deux coalitions, exigent une identification propre de cette stratégie :

Exemple 4.3 (Aide entre trois coalitions)**.** *Supposons encore qu'un troisième rôle ρ_3 soit pris en charge par l'assignation, et considérons l'assignation joue telle que joue $(\rho_1) = A_1$, joue $(\rho_2) = A_2$ et joue $(\rho_3) = A_3$. On cherche à formaliser la relation* $\mathsf{Col}_L(\{\rho_1\},\{\rho_2,\rho_3\})$, i.e. *le fait que les agents dans A_1 peuvent suivre des stratégies qui suffisent à satisfaire ρ_1 et qui peuvent être enrichies à la fois de manière à aider A_2 à satisfaire ρ_2 et de manière à aider A_3 à satisfaire ρ_3. Si l'agent a est toujours à la fois dans A_1 et A_2 et qu'il est encore dans A_3, ceci impose de considérer deux compositions potentielles pour sa stratégie pour A_1.*

Pour la situation décrite dans l'Exemple 4.3, la simulation de la composition de stratégies utilisée dans la formule 4.1 en assignant directement à a sa stratégie composée ne suffit pas. En effet, on veut ici considérer deux enrichissements potentiels de la même stratégie. Exprimer de telles propriétés dans le cas général nécessite donc un formalisme avec composition des stratégies pour un agent unique.

L'idée principale d'USL peut donc être exprimée comme suit : disposer d'un formalisme multi-agents par lequel un agent peut composer sa stratégie de plusieurs manières différentes. Si, dans un contexte où un agent est lié à une stratégie σ, ce même agent est à nouveau lié à une nouvelle stratégie σ', alors la liaison à σ' n'implique pas la révocation implicite de σ telle que nous l'avons relevée pour les langages ATL, ATL^*_{sc} et SL dans la Section 2.4. Au contraire, dans la suite de l'exécution, a jouera une stratégie composée à la fois de σ et de σ'. On résoudra ainsi le *problème de la révocation automatique des stratégies* soulevé dans la Section 2.4.

Ce langage utilise des stratégies *non déterministes* (on parle de *multi-stratégies*). Ce sont des stratégies qui indiquent des ensembles d'actions possibles après un préfixe d'exécution. Les multi-stratégies en USL peuvent être utilisées comme raffinement ou précision les unes des autres. La syntaxe d'USL est par ailleurs construite à partir de celle de SL, avec deux modifications :

- Le lieur de multi-stratégie est noté $(A \rhd x)$, où A est une coalition et x est une variable de multi-stratégies. En effet, dans USL, une coalition entière peut être liée à une multi-stratégie par l'application d'un seul opérateur. Par ailleurs, ce lieur n'implique pas la révocation par les agents de A des multi-stratégies auxquelles ils étaient éventuellement auparavant liés dans le contexte. La formule 4.2 par exemple formalise la propriété suivante : *Il y a un vecteur de multi-stratégies pour A qui lui permet d'assurer infiniment souvent P et qui peut à tout moment être amélioré de manière à assurer en plus infiniment souvent Q.*

$$\langle\langle \vec{x} \rangle\rangle (A \rhd \vec{x})(\Box\Diamond P \wedge \Box(\langle\langle \vec{y} \rangle\rangle (A \rhd \vec{y})\Box\Diamond Q)) \tag{4.2}$$

- Par ailleurs, afin d'unifier son formalisme avec les logiques préexistantes telles que SL ou ATL^*_{sc}, on introduit également dans USL la possibilité d'exprimer explicitement la révocation d'une multi-stratégie par un ou des agents. On utilise pour cela un opérateur explicite de déliaison (ou *révocation*) $(a \not\rhd x)$. Ainsi, la formule 4.3 exprime : *Il y a un vecteur de multi-stratégies pour A qui assure infiniment souvent P et qui peut à tout moment être abandonné par A de manière à ce que cette coalition adopte un nouveau vecteur de multi-stratégies qui assure infiniment souvent Q.*

$$\langle\langle \overrightarrow{x} \rangle\rangle (A \rhd \overrightarrow{x})(\Box\Diamond P \wedge \Box(\langle\langle \overrightarrow{y} \rangle\rangle (A \not\rhd \overrightarrow{x})(A \rhd \overrightarrow{y})\Box\Diamond Q)) \tag{4.3}$$

La formule 4.3 exprime la même propriété que la formule de SL 4.4, qui est elle-même obtenue de la formule 4.2 en y remplaçant le lieur de multi-stratégie d'USL par le lieur de multi-stratégie de SL :

$$\langle\langle \overrightarrow{x} \rangle\rangle (A, \overrightarrow{x})(\Box\Diamond P \wedge \Box(\langle\langle \overrightarrow{y} \rangle\rangle (A, \overrightarrow{y})\Box\Diamond Q)) \tag{4.4}$$

La Section 4.2 présente la syntaxe et les concepts sémantiques généraux pour USL. Le reste de ce chapitre est consacré à la présentation de différentes sémantiques pour USL.

Comme nous le verrons, dans la modélisation de la composition des multi-stratégies pour un agent ou une coalition, une question importante apparaît quand il s'agit d'interpréter la composition de deux multi-stratégies contradictoires, c'est-à-dire qui indiquent des ensembles d'actions en intersection vide après un préfixe d'exécution donné. La version standard de la sémantique modélise cette application de deux ensembles d'actions contradictoires par un arrêt de l'exécution. Elle interprète donc les opérateurs temporels dans des exécutions possiblement finies (Section 4.3). On définit egalement une version de la sémantique (USL^{lp}) qui est moins expressive mais utilise exclusivement des exécutions infinies (Section 4.4).

La sémantique USL^{cf} permet d'interpréter USL sur une classe de modèles qui englobe les CGSs : les CGS^{cf}*s*. Dans ces modèles, les combinaisons de un choix par agent depuis un état donné ne définissent pas nécessairement de successeur pour l'exécution : la fonction de transition est partielle. On peut donc modéliser des systèmes dans lesquels les différents agents ont des capacités conflictuelles (Section 4.5) en réponse à l'exigence *capacités conflictuelles* relevée dans la Figure 2.1.

Enfin, chacune de ces sémantique peut s'interpréter en ne faisant usage que de multi-stratégies positionnelles (Section 4.6).

4.2 Syntaxe et concepts sémantiques généraux

On définit d'abord le concept de multi-stratégie, ainsi que la syntaxe d'USL.

4.2.1 Multi-Stratégies

Une multi-stratégie indique un ensemble d'actions qui peuvent être jouées par un agent, en fonction de l'histoire des états déjà rencontrés.

La spécificité des multi-stratégies, par rapport aux stratégies usuellement utilisées dans la littérature, est que les multi-stratégies sont non déterministes. Cette approche permet notamment de décrire des situations dans lesquelles les stratégies sont sous-spécifiées. Une autre manière de voir les choses, mais qui n'est pas développée dans ce mémoire, est de considérer une multi-stratégie comme une version abstraite (sans probabilité) de stratégie mixte [SvdHed].

Les multi-stratégies qui sont utilisées pour USL sont des fonctions totales de l'ensemble des préfixes d'exécution vers les ensembles d'actions. En pratique, après un préfixe sur lequel elle n'est pas explicitement spécifiée, toute multi-stratégie sera considérée comme indiquant l'ensemble Ac de toutes les actions.

Définition 4.1 (*MStrat*)**.** *Soit $\mathcal{G} = \langle Ag, G, At, val, Ac, tr, s_o\rangle$ une CGS. Une multi-stratégie σ dans $\mathcal{G}$ est une fonction de l'ensemble des préfixes d'exécution dans l'ensemble des ensembles non vides*

d'actions, i.e. $\sigma : Track \nrightarrow \mathscr{P}_{>0}(Ac)$. *On écrit* $MStrat_{\mathscr{G}}$ *pour l'ensemble des multi-stratégies dans* $\mathscr{G}$ *(on omettra d'indiquer l'indice dans les cas dépourvus d'ambiguïté).*

On définit maintenant les *translations de multi-stratégies* le long de préfixes d'exécution. Cette définition est similaire à celle pour la translation des stratégies telle que définie pour ATL^*_{sc} (Définition 2.17), elle correspond à la mise à jour de la multi-stratégie pour prendre en compte le fait que τ est ajouté à l'histoire du jeu.

Définition 4.2 (Translation de multi-stratégie)**.** *Soient* $\mathscr{G} = \langle Ag, G, s_o, At, val, Ac, tr\rangle$ *une CGS,* σ *une multi-stratégie et* τ *un préfixe d'exécution dans* $\mathscr{G}$*. On appelle* translation de σ le long de τ *la multi-stratégie* σ^τ *t.q. pour tout préfixe d'exécution* τ' *dans* $\mathscr{G}$*,* $\sigma^\tau(\tau') = \sigma(\tau \cdot \tau'_{\geqslant 1})$.

4.2.2 Syntaxe d'USL

La syntaxe d'USL distingue entre formules d'état et formules de chemin. On décrit d'abord la notion de pseudo-formule. On définira ensuite les formules comme celles des pseudo-formules dans lesquelles tout quantificateur de variable de multi-stratégie opère sur une variable fraîche.

Définition 4.3 (Pseudo-formules d'USL)**.** *Soient Ag un ensemble d'agents, At un ensemble de propositions, et X un ensemble de variables de multi-stratégie. Alors, l'ensemble des* pseudo-formules d'USL *(Ag, At, X) est généré par la grammaire suivante :*

– Pseudo-formules d'état :

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle x\rangle\rangle\varphi \mid (A \rhd x)\psi \mid (A \not\rhd x)\psi$$

– Pseudo-formules de chemins :

$$\psi := \varphi \mid \neg\psi \mid \psi \wedge \psi \mid \mathsf{X}\psi \mid \psi \mathsf{U} \psi$$

où $p \in At$*,* $x \in X$ *et* $A \subseteq Ag$*.*

On remarque que, dans cette grammaire, les formules de chemins sont introduites exclusivement par $(A \rhd x)$ ou $(A \not\rhd x)$. En particulier, une formule de chemin ne peut pas suivre $\langle\langle A\rangle\rangle$. Cette restriction évite des effets contre-intuitifs dans les définitions sémantiques. Elle est appliquée sans perte de généralité.

Notation 4.2. *Finalement, le quantificateur universel sur les multi-stratégies est défini comme ceci : pour toute variable x et pour toute pseudo-formule d'état* φ*,* $[\![x]\!]\varphi \triangleq \neg\langle\langle x\rangle\rangle\neg\varphi$.

Définition 4.4 (Sous-pseudo-formules)**.** *L'ensemble* $Sf(\varphi)$ *des* sous-pseudo-formules *d'une pseudo-formule* φ *est défini par induction sur* φ *:*

– $Sf(p) = \{p\}$*, avec* $p \in At$
– $Sf(\neg\varphi) = \{\neg\varphi\} \cup Sf(\varphi)$
– $Sf(\varphi_1 \wedge \varphi_2) = \{\varphi_1 \wedge \varphi_2\} \cup Sf(\varphi_1) \cup Sf(\varphi_2)$
– $Sf(\langle\langle x\rangle\rangle\varphi) = \{\langle\langle x\rangle\rangle\varphi\} \cup Sf(\varphi)$
– $Sf((A \rhd x)\varphi) = \{(A \rhd x)\varphi\} \cup Sf(\varphi)$
– $Sf((A \not\rhd x)\varphi) = \{(A \not\rhd x)\varphi\} \cup Sf(\varphi)$
– $Sf(\mathsf{X}\varphi) = \{\mathsf{X}\varphi\} \cup Sf(\varphi)$

– $Sf(\varphi_1 \mathsf{U} \varphi_2) = \{\varphi_1 \mathsf{U} \varphi_2\} \cup Sf(\varphi_1) \cup Sf(\varphi_2)$

Comme les *noms* des variables de multi-stratégies sont pris en compte dans la sémantique des formules, ces noms de variables doivent être renouvelés quand des quantificateurs imbriqués sont utilisés. Ainsi, les formules (bien formées) sont les pseudo-formules telles que chaque quantificateur introduit une variable de multi-stratégie fraîche.

Définition 4.5 (Formules (bien formées))**.** *Une pseudo-formule φ est une* formule (bien formée) *ssi pour toute sous-pseudo-formule $\langle\langle x\rangle\rangle\varphi'$ de φ et pour toute sous-pseudo-formule $\langle\langle y\rangle\rangle\varphi''$ de φ', x et y sont des variables distinctes.*

On vérifie de manière immédiate que toute sous-pseudo-formule d'une formule est elle-même une formule.

Définition 4.6 (Sous-formules)**.** *Soit une φ formule d'USL, l'ensemble des* sous-formules *de φ est défini comme l'ensemble de ses sous-pseudo-formules. On le note également $Sf(\varphi)$. Pour toute formule φ, on appelle par ailleurs* sous-formule propre de φ *toute formule de $Sf(\varphi)$ autre que φ elle-même.*

Définition 4.7 (Variables libres)**.** *L'ensemble des* variables libres *d'une formule φ est écrit $VL(\varphi)$. Il est défini par induction sur φ :*

– $VL(p) = \varnothing$, *pour* $p \in At$

– $VL(\neg\varphi) = VL(\mathsf{X}\varphi) = VL(\varphi)$

– $VL(\varphi_1 \wedge \varphi_2) = VL(\varphi_1 \mathsf{U} \varphi_2) = VL(\varphi_1) \cup VL(\varphi_2)$

– $VL(\langle\langle x\rangle\rangle\varphi) = VL(\varphi) \setminus \{x\}$

– $VL((A \rhd x)\varphi) = VL((A \not\rhd x)\varphi) = VL(\varphi) \cup \{x\}$

Définition 4.8 (Enoncé)**.** *Une formule φ est appelée un* énoncé *si elle est* fermée, *c'est-à-dire si $VL(\varphi) = \varnothing$.*

Notation 4.3. *La* longueur *d'une formule φ est le cardinal de son ensemble de sous-formules. On l'écrit $|\varphi|$.*

Remarque 4.1 (Comparaison avec les énoncés de SL)**.** *Dans SL [MMV10, MMPV11], un énoncé est une formule qui n'a ni variable libre ni* agent libre. *Cette dernière condition impose que, pour qualifier un énoncé, tous les agents doivent être liés à une variable de multi-stratégie avant toute sous-formule temporelle. Par exemple, étant donné un ensemble d'agents $A = \{a_1, \ldots, a_n\}$ (avec $n \in \mathbb{N}$), la formule $\langle\langle x\rangle\rangle(a_n, x_n)\mathsf{X}\, p$ n'est pas un énoncé d'un langage de SL sur A. Pour qu'elle le devienne, une mention de chacun des agents autres que a_n doit être ajoutée, ce qui donne par exemple :*

$$[\![x_1]\!](a_1, x_1)\ldots[\![x_{n-1}]\!](a_{n-1}, x_{n-1})\langle\langle x\rangle\rangle(a_n, x_n)\mathsf{X}\, p$$

Conséquemment, l'écriture d'énoncés de SL peut devenir hardue. De manière plus importante, une unique spécification informelle peut se traduire par différents énoncés de SL, selon l'ensemble des agents du système qui ne sont pas mentionnés dans la spécification. À l'opposé, dans USL comme dans ATL ou ATL^*_{sc}, cette contrainte syntaxique n'est pas utilisée.

4.3 La sémantique standard, USL

Nous donnons maintenant un ensemble de définitions techniques qui sont requises pour définir la notion de satisfaction entre une CGS et une formule selon la sémantique standard. On définit d'abord les *contextes d'évaluation* qui sont utilisés pour USL, et différentes opérations sur ces contextes (Section 4.3.1). Dans la Section 4.3.2, on introduit l'ensemble des *issues* d'un *contexte d'évaluation* et d'un état. On donne ensuite la définition de la satisfaction $\models_{\text{USL}}$, (Section 4.3.3).

4.3.1 Contextes d'évaluation

Comme dans SL ou ATL$^*_{sc}$, USL repose sur une notion de contextes pour évaluer les sous-formules d'un énoncé. Cependant, contrairement à SL, on distingue deux éléments dans ces contextes : les *assignations*, qui gardent en mémoire les instanciations des variables de multi-stratégies, et les *engagements*, qui stockent les liaisons d'agents à des variables de multi-stratégies.

Définition 4.9 (Assignations, Engagements, Contextes)**.** *Une* assignation α *est une fonction partielle des variables de multi-stratégies vers les multi-stratégies,* $\alpha : X \nrightarrow \mathit{MStrat}$.

Un engagement γ *rassemble des liaisons entre un ensemble d'agents et les variables des multi-stratégies auxquelles ils sont liés,* i.e. *c'est une relation* $\gamma \subseteq \mathit{Ag} \times X$.

Un contexte χ *est une paire* $\langle \alpha, \gamma \rangle$ *d'une assignation* α *et d'un engagement* γ. *Il est "bien formé", dans le sens où toute variables de multi-stratégie associée à un ou des agent(s) dans l'engagement est instanciée,* i.e. $\gamma(\mathit{Ag}) \subseteq \mathit{dom}(\alpha)$.

Notation 4.4. *On écrit* $\alpha_\varnothing$ *pour l'assignation vide (t.q.* $\mathit{dom}(\alpha_\varnothing) = \varnothing$*),* $\gamma_\varnothing$ *pour l'engagement vide (t.q.* $\mathit{dom}(\gamma_\varnothing) = \varnothing$*), et* $\chi_\varnothing$ *pour le contexte vide (t.q.* $\chi_\varnothing = \langle \alpha_\varnothing, \gamma_\varnothing \rangle$*).*

La notion de translation de multi-stratégies est maintenant étendue aux assignations et aux contextes. On l'utilisera pour définir la notion d'issue dans la prochaine section. Une translation d'une assignation est une assignation dans laquelle chaque instance de multi-stratégie est elle-même translatée. La translation d'un contexte est définie par la translation de son assignation.

Définition 4.10 (Translation d'assignation, translation de contexte)**.** *Soient une CGS* $\mathcal{G} = \langle \mathit{Ag}, G, s_o, \mathit{At}, \mathit{val}, \mathit{Ac}, tr \rangle$*, un contexte* $\chi = \langle \alpha, \gamma \rangle$ *et un préfixe d'exécution* τ *dans* $\mathcal{G}$*. On appelle* translation de α le long τ *l'assignation* α^τ *t.q.* $\mathit{dom}(\alpha^\tau) = \mathit{dom}(\alpha)$ *et pour tous* $x \in \mathit{dom}(\alpha^\tau)$, $\alpha^\tau(x) = (\alpha(x))^\tau$*. La translation d'un contexte* χ *est définie comme* $\langle \alpha, \gamma \rangle^\tau = \langle \alpha^\tau, \gamma \rangle$.

Dans l'évaluation sémantique d'une formule, le contexte doit être transformé quand on rencontre un quantificateur de multi-stratégie ainsi qu'un opérateur de liaison ou de révocation. Pour un quantificateur $\langle\langle x \rangle\rangle$, on met simplement à jour l'assignation courante pour x. Pour un opérateur de liaison $(A \rhd x)$, l'engagement doit être mis à jour avec une paire $\langle a, x \rangle$ pour tout agent $a \in A$. Finalement, pour un opérateur de déliaison $(A \not\rhd x)$, toutes les paires $\langle a, x \rangle$ doivent être retirées de l'engagement, pour tout agent a dans A.

Définition 4.11 (Transformations de contexte)**.** *Soient un contexte* $\chi = \langle \alpha, \gamma \rangle$*, une coalition* $A \subseteq \mathit{Ag}$*, une variable de multi-stratégie* x *et une multi-stratégie* σ*. On définit les transformations* $[x \mapsto \sigma]$

sur les assignations et $[A \oplus x]$ *et* $[A \ominus x]$ *sur les engagements comme suit :*

$$\alpha[x \mapsto \sigma](y) = \begin{cases} \sigma & si\ x = y \\ \alpha(y) & sinon \end{cases}$$

$$\gamma[A \oplus x] = \gamma \cup \{\langle a, x\rangle \mid a \in A\}$$

$$\gamma[A \ominus x] = \gamma \setminus \{\langle a, x\rangle \mid a \in A\}$$

Cette notation est étendue aux contextes : $\chi[x \mapsto \sigma] = \langle \alpha[x \mapsto \sigma], \gamma\rangle$, $\chi[A \oplus x] = \langle \alpha, \gamma[A \oplus x]\rangle$ *et* $\chi[A \ominus x] = \langle \alpha, \gamma[A \ominus x]\rangle$.

Remarque 4.2. *On remarque que l'application de toutes ces transformations sur un contexte donne un contexte comme résultat.*

4.3.2 Issues

Dans SL [MMPV11], plusieurs définitions sont introduites pour caractériser un jeu (en particulier celles pour les stratégies et contextes "s-totaux", pour les contextes "complets"...). Ici, en raison du non-déterminisme des multi-stratégies, la situation est un peu différente. À la place d'un jeu unique, on prend en compte un ensemble *d'issues*. Plus précisément, un contexte définit une fonction d'issues qui indique, pour tout préfixe d'exécution λ, un ensemble d'exécutions possiblement finies qui peuvent être visitées si les agents, à partir de λ, jouent selon les multi-stratégies stockées dans χ.

Pour définir la fonction d'issues, il nous faut d'abord définir l'ensemble des *successeurs* immédiats potentiels d'un état donné dans un contexte $\chi = \langle \alpha, \gamma\rangle$:

Définition 4.12 (Fonction de successeurs)**.** *Soient une CGS* $\mathcal{G} = \langle Ag, G, s_o, At, val, Ac, tr\rangle$ *et un contexte* $\chi = \langle \alpha, \gamma\rangle$. *Alors la fonction de* successeurs $Msucc_\chi : Track \to \mathcal{P}(G)$ *indiquée par* χ *est définie, pour tout préfixe d'exécution* τ, *par :*

$$Msucc_\chi(\tau) = \{tr(last(\tau), dc) \mid \forall \langle a, x\rangle \in \gamma \cdot dc(a) \in \alpha(x)(\tau)\}$$

On remarque en particulier que, dans le cas où un agent a est lié dans χ à plusieurs multi-stratégies qui indiquent des ensembles d'actions contradictoires après un préfixe d'exécution τ, alors $Msucc_\chi(\tau) = \varnothing$ et l'exécution s'arrête. La sémantique USL intègre ainsi le traitement des engagements contradictoires et non contradictoires de manière unifiée.

On arrive à la définition des *issues* d'un contexte et d'un préfixe d'exécution : c'est l'ensemble des exécutions finies ou infinies qui sont rendues possibles par ce contexte après ce préfixe d'exécution :

Définition 4.13 (Issues pour USL)**.** *Soient une CGS* $\mathcal{G} = \langle Ag, G, s_o, At, val, Ac, tr\rangle$, *un contexte* $\chi = \langle \alpha, \gamma\rangle$ *et un préfixe d'exécution* τ. *On définit l'ensemble des* issues *de* χ *et* τ *dans* $\mathcal{G}$ *comme l'ensemble* $out(\chi, \tau)$ *des exécutions* λ, *dans* $\mathcal{G}$, *t.q.* $\lambda_0 = last(\tau)$ *et :*

- *si* λ *est infinie alors pour tout* $i \in \mathbb{N}$, $\lambda_{i+1} \in Msucc_{\chi^{\lambda_{\leqslant i}}}(\lambda_{\leqslant i})$
- *si* λ *est fini ,* $\lambda = \lambda_{\leqslant n}$ *alors :*
  - *pour tout* $i < n$, $\lambda_{i+1} \in Msucc_{\chi^{\lambda_{\leqslant i}}}(\lambda_{\leqslant i})$
  - *et* $Msucc_{\chi^\lambda}(\lambda) = \varnothing$.

Si τ *n'a qu'un seul élément* $\tau_0 = last(\tau)$, *on note également* $out(\chi, \tau_0)$ *pour* $out(\chi, \tau)$.

4.3.3 Satisfaction

Définition 4.14 (Satisfaction sémantique pour USL)**.**

Soit une CGS $\mathcal{G} = \langle Ag, G, s_o, At, val, Ac, tr\rangle$*. Alors, la* relation de satisfaction $\models_{USL}$ *est définie par induction sur les formules, pour tout contexte* χ*, pour tout état* $s \in G$ *et pour toute exécution* λ *dans* $\mathcal{G}$*, comme suit :*

- *Formules d'état*
  - $\mathcal{G}, \chi, s \models_{USL} p$ *ssi* $p \in val(s)$*, où* $p \in At$
  - $\mathcal{G}, \chi, s \models_{USL} \neg\varphi$ *ssi* $\mathcal{G}, \chi, s \not\models_{USL} \varphi$
  - $\mathcal{G}, \chi, s \models_{USL} \varphi_1 \wedge \varphi_2$ *ssi* $\mathcal{G}, \chi, s \models_{USL} \varphi_1$ *et* $\mathcal{G}, \chi, s \models_{USL} \varphi_2$
  - $\mathcal{G}, \chi, s \models_{USL} \langle\langle x\rangle\rangle\varphi$ *ssi il y a une multi-stratégie* $\sigma \in MStrat_{\mathcal{G}}$ *t.q.* $\mathcal{G}, \chi[x \mapsto \sigma], s \models_{USL} \varphi$
  - $\mathcal{G}, \chi, s \models_{USL} (A \rhd x)\psi$ *ssi pour toute* $\lambda \in out(\chi[A \oplus x], s)\mathcal{G}, \chi[A \oplus x], \lambda \models_{USL} \psi$
  - $\mathcal{G}, \chi, s \models_{USL} (A \not\rhd x)\psi$ *ssi pour toute* $\lambda \in out(\chi[A \ominus x], s), \mathcal{G}, \chi[A \ominus x], \lambda \models_{USL} \psi$
- *Formules de chemin*
  - $\mathcal{G}, \chi, \lambda \models_{USL} \varphi$ *ssi* $\mathcal{G}, \chi, \lambda_0 \models_{USL} \varphi$*, si* φ *est une formule d'état*
  - $\mathcal{G}, \chi, \lambda \models_{USL} \neg\psi$ *ssi* $\mathcal{G}, \chi, \lambda \not\models_{USL} \psi$
  - $\mathcal{G}, \chi, \lambda \models_{USL} \psi_1 \wedge \psi_2$ *ssi* $\mathcal{G}, \chi, \lambda \models_{USL} \psi_1$ *et* $\mathcal{G}, \chi, \lambda \models_{USL} \psi_2$
  - $\mathcal{G}, \chi, \lambda \models_{USL} \mathsf{X}\psi$ *ssi* $|\lambda| > 1$ *et* $\mathcal{G}, \chi^{\lambda_0\lambda_1}, \lambda_{\geqslant 1} \models_{USL} \psi$
  - $\mathcal{G}, \chi, \lambda \models_{USL} \psi_1 \mathsf{U} \psi_2$ *ssi il y a* $i \in \mathbb{N}$ *t.q.* $|\lambda| > i$*,* $\mathcal{G}, \chi^{\lambda_{\leqslant i}}, \lambda_{\geqslant i} \models_{USL} \psi_2$ *et pour tout* $0 \leqslant j < i$*,* $\mathcal{G}, \chi^{\lambda_{\leqslant j}}, \lambda_{\geqslant j} \models_{USL} \psi_1$

Étant donnée une formule d'état φ*, on écrit* $\mathcal{G}, s \models_{USL} \varphi$ *ssi* $\mathcal{G}, \chi_\varnothing, s \models_{USL} \varphi$*, et* $\mathcal{G} \models_{USL} \varphi$ *ssi* $\mathcal{G}, s_0 \models_{USL} \varphi$*.*

4.4 Une sémantique pour USL sans arrêt des exécutions : USL^{lp}

Comme on vient de le voir, la sémantique USL utilise des exécutions possiblement finies. Cette utilisation apporte un pouvoir expressif substantiel. Elle modélise par ailleurs les engagements potentiellement contradictoires de manière naturelle et unifiée. Cependant, elle a également des effets sémantiques indésirables. Dans cette section, on présente une sémantique pour USL qui n'utilise que des exécutions infinies. Elle est notée USL^{lp}.

Cette section s'organise comme suit : on discute l'utilisation des opérateurs temporels dans les traces finies (Section 4.4.1). On présente ensuite de manière intuitive la solution proposée par USL^{lp} pour éviter cette utilisation (Section 4.4.2). Cette solution consiste à donner, en cas d'engagement d'un agent à des multi-stratégies contradictoires, la priorité à la première multi-stratégie liée à cet agent. Comme c'est la multi-stratégie qui sera entrée dans le contexte pour interpréter le lieur le plus à gauche dans l'écriture de la formule, on parle de la *règle de priorité à gauche*. Ce nom explique l'indice *lp* dans l'écriture USL^{lp} (il fait référence à l'anglais *left priority*). La sémantique USL^{lp} nécessite des redéfinitions pour un certain nombre des concepts sémantiques. Elles sont données dans la Section 4.4.3).

4.4.1 Discussion sur la prise en compte d'exécutions finies dans USL

La sémantique USL permet de traiter la composition des multi-stratégies pour un agent. Elle permet d'exprimer de nombreuses propriétés nouvelles (voir Section 5.1.2). Cependant, en cas d'engagements contradictoires pour un agent, l'utilisation d'exécutions possiblement finies comporte certains inconvénients :

- D'une part, pour toute formule d'état φ, la formule $\mathsf{X}\varphi$ est toujours fausse dans le dernier état d'une exécution finie, de même que $\mathsf{X}\neg\varphi$. On perd ainsi l'équivalence de LTL entre $\neg\mathsf{X}\varphi$ et $\mathsf{X}\neg\varphi$. Équivalemment, on perd le théorème selon lequel $\mathsf{X}\top$ est vraie dans toute exécution.
- Par ailleurs, l'interprétation de l'opérateur dérivé $\Box$ pose certains problèmes : pour toute formule φ de LTL, la formule $\Box\varphi$ indique que φ reste vraie après n'importe quel nombre de transitions. Selon LTL^n cependant, $\Box\varphi$ est vraie ssi soit φ reste vraie après n'importe quel nombre de transitions, soit φ reste vraie jusqu'à la fin de l'exécution. $\Box\varphi$ est donc vraie dans des exécutions (finies) où φ ne reste vraie que durant un nombre fini de transitions.
- Enfin, l'interprétation faite par USL d'un contexte où un agent est lié à deux multi-stratégies contradictoires (*i.e.* comme un arrêt de l'exécution), induit implicitement la capacité, en tout état et pour chaque agent, d'arrêter l'exécution. Cette capacité implicite provient en fait de l'exercice simultané de plusieurs capacités. Elle ne peut être exercée par un agent que s'il se lie successivement à deux multi-stratégies.

  Or la satisfaction $\models_{\text{USL}}$ ignore cette capacité implicite dans le cas général. L'évaluation de la formule $\langle\langle x\rangle\rangle(a \rhd x)\mathsf{X}\,p$, par exemple, renvoie vrai si a dispose d'une multi-stratégie qui lui permet d'assurer $\mathsf{X}\,p$ quelles que soient les multi-stratégies jouées par les autres agents. Cependant, dans le cas général, chaque agent autre que a dispose au moins de deux multi-stratégies qui, combinées ensemble, s'interprétent comme un arrêt de l'exécution et empêchent la satisfaction de $\mathsf{X}\,p$. Depuis l'état s, s'il y a deux actions ac_1 et ac_2, il s'agit par exemple de tout couple de multi-stratégies (σ_1,σ_1) tel que $\sigma_1(s) = \{ac_1\}$ et $\sigma_2(s) = \{ac_2\}$. Les deux formules $\langle\langle x\rangle\rangle(a \rhd x)\mathsf{X}\,p$ et $[\![x]\!]\langle\langle y\rangle\rangle\langle\langle z\rangle\rangle(a \rhd x)(b \rhd y)(b \rhd z)\neg\mathsf{X}\,p$ peuvent donc être satisfaites en même temps. Cet aspect heurte une certaine intuition.

  Les limites de ces objections tiennent en deux éléments :

  - Premièrement, dans USL, l'arrêt de l'exécution n'est qu'un moyen sémantique pour représenter le fait que le contexte courant est pathologique, dans le sens où un agent ne peut pas respecter ses différents engagements. Il ne s'agit donc pas de modéliser un système dans lequel un agent commanderait effectivement un arrêt de l'exécution au moyen d'une contradiction, mais de relever ces cas de constructions de contextes impossibles. L'interprétation légitime de la formule $[\![x]\!]\langle\langle y\rangle\rangle\langle\langle z\rangle\rangle(a \rhd x)(b \rhd y)(b \rhd z)\neg\mathsf{X}\,p$ en langage courant n'est donc pas : *quoi que fasse a, b peut empêcher la satisfaction de* $\mathsf{X}\,p$ mais : *quoi que fasse a, il y a des couples d'ensembles d'actions que b ne peut pas jouer en même temps (a fortiori donc : que b ne peut pas jouer en même temps tout en assurant* $\mathsf{X}\neg p$*).*
  - Par ailleurs, il est possible de contourner ces effets sémantiques indésirables. En effet, pour toutes multi-stratégies x et y, la spécification $\Box((a \rhd x)(a \rhd y)\mathsf{X}\top)$ impose que ces multi-stratégies ne soient pas contradictoires pour a, après tout préfixe d'exécution τ. Plus radicalement, comme nous le verrons dans la Section 5.2, la règle de priorité à gauche est en fait entièrement simulable avec la sémantique d'USL,

de sorte que USL^{lp} se *plonge* (voir Définition 5.6) dans USL. Les inconvénients liés à l'utilisation des exécutions possiblement finies sont donc principalement des inconvénients de convivialité.

4.4.2 La règle de priorité à gauche

Pour introduire la règle de priorité à gauche, on se réfère à nouveau à la formule 4.2 de la Section 4.1. On la redonne ici pour une coalition unaire :

$$\langle\langle x\rangle\rangle(a \rhd x)(\Box\Diamond p \wedge \Box(\langle\langle y\rangle\rangle(a \rhd y)\Box\Diamond q)) \tag{4.5}$$

Par cette formule, on veut exprimer que a peut adopter une multi-stratégie qui assure infiniment souvent p et peut à tout moment être améliorée de manière à assurer également infiniment souvent q. On introduit ici une convention de notation :

Notation 4.5. *Étant donnée une variable x, on désigne par σ_x une multi-stratégie instanciant x.*

Supposons qu'au stade de l'évaluation pour la sous-formule $\langle\langle y\rangle\rangle(a \rhd y)\Box\Diamond q$, aucune multi-stratégie ne satisfasse les deux conditions suivantes, nécessaires pour instancier y : ne pas contredire la multi-stratégie σ_x et assurer infiniment souvent q.

De même que pour USL, l'évaluation devra conclure à l'insatisfaction de la formule. On s'en assurera ici en définissant comme résultat de la composition des ensembles d'actions indiqués par σ_x et σ_y le premier des deux. Ainsi, on considère que si la composition d'un nouvel ensemble d'actions dans le contexte courant ne peut avoir lieu, elle ne se fait pas : la tentative de composition échoue et le contexte courant est maintenu. C'est la règle de priorité à gauche.

Le principe suivi est donc de composer entre elles les multi-stratégies, en prenant l'intersection des ensembles d'actions qu'elles définissent *autant que c'est possible*, et de ne pas faire agir cette composition si elle n'est pas possible.

4.4.3 Définitions

L'application de la règle de priorité à gauche nécessite la redéfinition de certains concepts sémantiques. Deux modifications principales par rapport à l'ensemble des outils pour USL sont à prendre en compte : d'une part les engagements doivent enregistrer l'ordre d'introduction des différentes multi-stratégies. D'autre part, la nouvelle sémantique n'utilise que des exécutions infinies.

Les contextes pour USL^{lp} encodent, pour chaque agent, la succession des multi-stratégies auxquelles cet agent est lié. On représentera cette succession par une séquence finie de variables dans X.

Définition 4.15 (Engagements)**.** *Un* engagement γ *pour USL^{lp} est une fonction partielle de Ag dans X^*. Soient γ_1 et γ_2 deux engagements, on note $\gamma_1 \cdot \gamma_2$ l'engagement de domaine $dom(\gamma_1) \cup dom(\gamma_2)$ et tel que pour tout $a \in dom(\gamma_1 \cdot \gamma_2)$,*

$$\gamma_1 \cdot \gamma_2(a) = \begin{cases} \gamma_1(a) & \text{si } \gamma_2(a) \text{ est indéfini et } \gamma_1(a) \text{ est défini} \\ \gamma_2(a) & \text{si } \gamma_1(a) \text{ est indéfini et } \gamma_2(a) \text{ est défini} \\ \gamma_1(a) \cdot \gamma_2(a) & \text{sinon} \end{cases}$$

Une assignation α pour USL^{lp} est définie comme pour USL et un contexte est un couple $\chi = \langle \alpha, \gamma \rangle$ tel que pour tout $a \in dom(\gamma), \gamma(a) \in dom(\alpha)^*$. Les transformations d'engagements doivent être adaptées, de manière à prendre en compte l'ordre qui y a été introduit.

Définition 4.16 (Transformations d'engagements)**.** *Soient un engagement γ pour USL^{lp}, une coalition $A \subseteq Ag$ et une variable de multi-stratégie x. On définit les transformations $[A \oplus^{lp} x]$ et $[A \ominus^{lp} x]$ sur γ comme suit : pour tout agent a,*

$$\gamma[A \oplus^{lp} x](a) = \begin{cases} \gamma(a) \cdot x & \text{si } a \in A \\ \gamma(a) & \text{sinon} \end{cases}$$

$$\gamma[A \ominus^{lp} x](a) = \begin{cases} \gamma(a) \setminus x & \text{si } a \in A \\ \gamma(a) & \text{sinon} \end{cases}$$

où pour toute séquence $\overline{y}, \overline{y} \setminus x$ désigne la séquence $\overline{y}$ à laquelle on a retiré toutes les occurrences éventuelles de x : pour tous $y \in X$ et $\overline{y} \in X^*$,

$$y \setminus x = \begin{cases} \text{le mot vide de } X^* & \text{si } y_1 = x \\ y & \text{sinon} \end{cases}$$

$$(\overline{y} \cdot y) \setminus x = (\overline{y} \setminus x) \cdot y$$

La définition de la transformation $\alpha[x \mapsto \sigma]$ pour une assignation α et une multi-stratégie σ reste inchangée par rapport à celle pour USL.

Pour définir les issues d'un contexte, on introduit une notion de *multi-stratégie effective*, qui traduit la règle de priorité à gauche. Pour tout contexte χ et pour chaque agent a, $eS^{lp}(\chi, a)$ indique la multi-stratégie que a doit jouer, étant données les différentes multi-stratégies auxquelles il est lié dans χ. Tant que ces multi-stratégies lui indiquent des ensembles d'actions en intersection non vide, la *multi-stratégie effective* donne l'intersection de ces ensembles d'actions. Si l'ajout d'une nouvelle multi-stratégie pour un agent dans un contexte provoque une intersection vide après certains préfixes d'exécution, alors elle est ignorée après ces préfixes d'exécution dans la construction de la multi-stratégie effective :

Définition 4.17 (Multi-stratégie effective)**.** *Soit $\chi = \langle \alpha, \gamma \rangle$ un contexte bien formé. Alors, pour tout agent a, pour tout préfixe d'exécution τ,*

$$eS^{lp}(\chi, a)(\tau) = \begin{cases} Ac & \text{si } \gamma(a) \text{ n'est pas défini} \\ \alpha(x)(\tau) & \text{si } \gamma(a) \text{ est la séquence } \langle x \rangle \\ eS^{lp}(\langle \alpha, \gamma' \rangle, a)(\tau) \cap \alpha(x)(\tau) & \text{si } \gamma(a) = \gamma'(a) \cdot x \\ & \quad \text{et } eS^{lp}(\langle \alpha, \gamma' \rangle, a)(\tau) \cap \alpha(x)(\tau) \neq \varnothing \\ eS^{lp}(\langle \alpha, \gamma' \rangle, a)(\tau) & \text{si } \gamma(a) = \gamma'(a) \cdot x \\ & \quad \text{et } eS^{lp}(\langle \alpha, \gamma' \rangle, a)(\tau) \cap \alpha(x)(\tau) = \varnothing \end{cases}$$

On peut alors définir la fonction des ensembles de successeurs possibles, par simple application de la fonction de transition :

Définition 4.18 (Fonction de successeurs)**.** *Soient une CGS* $\mathscr{G} = \langle Ag, G, s_o, At, val, Ac, tr\rangle$ *et un contexte* $\chi = \langle \alpha, \gamma\rangle$*. Alors la fonction de* successeurs $Msucc^{lp}_{\chi} : Track \to \mathscr{P}(G)$ *indiquée par* χ *est définie, pour tout préfixe d'exécution* τ*, par :*

$$Msucc^{lp}_{\chi}(\tau) = \{tr(last(\tau), dc) \mid \forall a \in Ag, dc(a) \in eS^{lp}(\chi, a)(\tau)\}$$

Selon cette définition, pour tout contexte χ et pour tout préfixe d'exécution τ, $Msucc^{lp}_{\chi}(\tau)$ est non vide. Les issues possibles pour un contexte et un état sont les exécutions compatibles avec les ensembles de successeurs possibles. Du fait de cette dernière remarque, elles sont toutes de longueur infinie :

Définition 4.19 (Issues)**.** *Soit* χ *un contexte et soit* $\tau \in Track$*. Pour toute exécution* λ*,* $\lambda \in out^{lp}(\chi, \tau)$ *ssi* $\lambda_0 = last(\tau)$ *et pour tout* $n \in \mathbb{N}, \lambda_{n+1} \in Msucc^{lp}_{\chi^{\lambda_0 \cdots \lambda_n}}(\lambda_0 \cdots \lambda_n)$*.*

Finalement, la relation de satisfaction pour USL^{lp} est adaptée de celle pour USL, en interprétant les opérateurs temporels uniquement dans les exécutions infinies et en appelant les transformations $\oplus^{lp}$ et $\ominus^{lp}$ pour les engagements ainsi que la fonction d'issues définie ci-dessus :

Définition 4.20 (Satisfaction sémantique pour USL^{lp})**.**

Soit une CGS $\mathscr{G} = \langle Ag, G, s_o, At, val, Ac, tr\rangle$*. Alors, la* relation de satisfaction $\models_{USL^{lp}}$ *est définie par induction sur les formules, pour tout contexte* χ*, pour tout état* $s \in G$ *et pour toute exécution* λ *dans* $\mathscr{G}$*, comme suit :*

- *Formules d'état*
  - $\mathscr{G}, \chi, s \models_{USL^{lp}} p$ *ssi* $p \in val(s)$*, où* $p \in At$
  - $\mathscr{G}, \chi, s \models_{USL^{lp}} \neg\varphi$ *ssi* $\mathscr{G}, \chi, s \not\models_{USL^{lp}} \varphi$
  - $\mathscr{G}, \chi, s \models_{USL^{lp}} \varphi_1 \wedge \varphi_2$ *ssi* $\mathscr{G}, \chi, s \models_{USL^{lp}} \varphi_1$ *et* $\mathscr{G}, \chi, s \models_{USL^{lp}} \varphi_2$
  - $\mathscr{G}, \chi, s \models_{USL^{lp}} \langle\langle x\rangle\rangle\varphi$ *ssi il y a une multi-stratégie* $\sigma \in MStrat_{\mathscr{G}}$ *t.q.* $\mathscr{G}, \chi[x \mapsto \sigma], s \models_{USL^{lp}} \varphi$
  - $\mathscr{G}, \chi, s \models_{USL^{lp}} (A \rhd x)\psi$ *ssi pour toute* $\lambda \in out^{lp}(\chi[A \oplus^{lp} x], s)$*,* $\mathscr{G}, \chi[A \oplus^{lp} x], \lambda \models_{USL^{lp}} \psi$
  - $\mathscr{G}, \chi, s \models_{USL^{lp}} (A \not\rhd x)\psi$ *ssi pour toute* $\lambda \in out^{lp}(\chi[A \ominus^{lp} x], s)$*,* $\mathscr{G}, \chi[A \ominus^{lp} x], \lambda \models_{USL^{lp}} \psi$
- *Formules de chemin*
  - $\mathscr{G}, \chi, \lambda \models_{USL^{lp}} \varphi$ *ssi* $\mathscr{G}, \chi, \lambda_0 \models_{USL^{lp}} \varphi$*, si* φ *est une formule d'état*
  - $\mathscr{G}, \chi, \lambda \models_{USL^{lp}} \neg\psi$ *ssi* $\mathscr{G}, \chi, \lambda \not\models_{USL^{lp}} \psi$
  - $\mathscr{G}, \chi, \lambda \models_{USL^{lp}} \psi_1 \wedge \psi_2$ *ssi* $\mathscr{G}, \chi, \lambda \models_{USL^{lp}} \psi_1$ *et* $\mathscr{G}, \chi, \lambda \models_{USL^{lp}} \psi_2$
  - $\mathscr{G}, \chi, \lambda \models_{USL^{lp}} \mathsf{X}\psi$ *ssi* $\mathscr{G}, \chi^{\lambda_0\lambda_1}, \lambda_{\geqslant 1} \models_{USL^{lp}} \psi$
  - $\mathscr{G}, \chi, \lambda \models_{USL^{lp}} \psi_1 \mathsf{U} \psi_2$ *ssi il y a* $i \in \mathbb{N}$ *t.q.* $\mathscr{G}, \chi^{\lambda_{\leqslant i}}, \lambda_{\geqslant i} \models_{USL^{lp}} \psi_2$ *et pour tout* $0 \leqslant j < i$*,* $\mathscr{G}, \chi^{\lambda_{\leqslant j}}, \lambda_{\geqslant j} \models_{USL^{lp}} \psi_1$*.*

Étant donnée une formule d'état φ*, on écrit* $\mathscr{G}, s \models_{USL^{lp}} \varphi$ *ssi* $\mathscr{G}, \chi_\varnothing, s \models_{USL^{lp}} \varphi$*, et* $\mathscr{G} \models_{USL^{lp}} \varphi$ *ssi* $\mathscr{G}, s_0 \models_{USL^{lp}} \varphi$*.*

4.5 Une extension de la sémantique USL : USL^{cf}

On s'intéresse ici à une extension du concept de CGS qui permet de modéliser des systèmes où différents agents ont des capacités d'actions qui ne sont pas compatibles entre elles (les CGS^{cf}*s*). On parlera alors de *capacités conflictuelles* entre les différents agents du système (d'où l'indice *cf* dans les notations CGS^{cf}*s* et USL^{cf}). C'est notamment le cas dans KHI où, comme exposé dans la Section 3.4.2 et en réponse à l'exigence de *capacités conflictuelles* de la Figure 2.1, deux acteurs peuvent avoir la capacité d'agir sur une même variable dans un même état. La sémantique USL^{cf} est une extension de la sémantique standard pour USL sur cette classe de modèles. On y interprète un conflit comme un arrêt de l'exécution.

On donne une introduction informelle à cette sémantique (Section 4.5.1) avant d'en définir les aspects formels (Section 4.5.2).

4.5.1 Introduction : les fonctions de transition partielles

Dans les CGSs, la fonction de transition est totale. C'est-à-dire que le successeur d'un état est défini quels que soient cet état et les actions jouées par chacun des agents. Cette particularité traduit une forme de contrainte implicite d'indépendance entre les agent pour les systèmes modélisables avec des CGSs : quel que soit l'ensemble des actions jouées par chacun des agents, ces actions sont compatibles entre elles, dans le sens où, ensemble, elles déterminent toujours un successeur. Les possibilités d'actions de chacun des agents ne sont pas dépendantes des actions des autres agents.

Pour modéliser une instance de KHI où les capacités des acteurs représentent leur contrôle sur certaines valeurs de certaines variables, la contrainte correspondante serait que deux acteurs n'aient jamais de capacités conflictuelles.

Afin de pouvoir modéliser des instances de KHI dans lesquelles les agents ont éventuellement des capacités conflictuelles, on veut s'affranchir de cette contrainte. La solution retenue est de modéliser le conflit comme un cas d'arrêt de l'exécution quand les agents jouent des actions conflictuelles.

4.5.2 Définitions

On définit en premier lieu les *CGSs à conflits potentiels* (les CGS^{cf}*s*). Elles différent des CGSs uniquement en ce que leur fonction de transition est partielle sur l'ensemble des décisions. Elles sont donc une sur-classe des CGSs.

Définition 4.21 (CGS^{cf})**.** *Une CGS^{cf} est un tuple $\mathcal{G} = \langle Ag, G, s_0, At, val, Ac, tr\rangle$ où :*

- *Ag est un ensemble fini non vide d'agents*
- *G est un ensemble dénombrable non vide d'état*
- *$s_0 \in G$ est l'état initial*
- *At est un ensemble fini non vide de propositions atomiques*
- *$val : G \to 2^{At}$ est une fonction de valuation, qui indique en chaque état celles des propositions atomiques qui sont vraies en cet état*
- *Ac est un ensemble d'actions.*
- *Soit $Dc = Ac^{Ag}$ l'ensemble des décisions,* i.e. *l'ensemble des fonctions totales des agents dans les actions. Alors $tr : G \times Dc \nrightarrow G$ est une fonction* ***partielle*** *de transition.*

Sur les combinaisons (s, dc) d'un état et d'une décision qui ne sont pas dans son domaine, la fonction tr ne renvoie pas d'état successeur. Cette décision depuis cet état signifie donc l'arrêt de l'exécution. On peut alors donner un contenu formel à la notion d'action conflictuelle :

Définition 4.22 (Actions conflictuelles)**.** *Soient $\mathcal{G} = \langle Ag, G, s_0, At, val, Ac, tr\rangle$ une CGS^{cf}, $A \subseteq Ag$ un ensemble d'agents et $\{ac_a\}_{a\in A}$ un ensemble d'actions indicées par les agents dans A. Pour tout $s \in G$, $\{ac_a\}_{a\in A}$ est un ensemble d'actions conflictuelles depuis s pour les agents de A ssi aucune décision telle que chaque agent a dans A joue l'action ac_a n'est prise en charge par la fonction de transition,* i.e. *ssi*

$$\{dc \in Dc \mid \forall a \in A((dc(a) = ac_a)) \wedge (s, dc) \in dom(tr)\} = \varnothing$$

Dans l'ensemble de successeurs possibles associé à un contexte, on enregistre les terminaisons d'exécutions qui sont provoquées par des actions conflictuelles de la part des différents agents. Ceci permet d'identifier les exécutions terminées par des actions conflictuelles et donc de les inclure dans la fonction d'issues pour USL^{cf}. Pour cela, on étend le co-domaine de la fonction des successeurs potentiels ($Msucc^{cf}$) en y introduisant, en plus des états, un symbole $\overline{\varnothing}$ qui indique qu'une séquence finie s'est arrêtée dans l'état précédent à cause d'actions conflictuelles par différents agents :

Définition 4.23 (Fonction de successeurs)**.** *Soient une CGS^{cf} $\mathcal{G} = \langle Ag, G, At, val, Ac, tr, s_o\rangle$ et un contexte $\chi = \langle \alpha, \gamma\rangle$. Alors la fonction de* successeurs $Msucc^{cf}_{\chi} : Track \to \mathcal{P}(G \cup \{\overline{\varnothing}\})$ *induite par χ est définie, pour tout préfixe d'exécution τ, par :*

$$Msucc^{cf}_{\chi}(\tau) = \{tr(last(\tau), dc) \mid \forall \langle a, x\rangle \in \gamma, dc(a) \in \alpha(x)(\tau)\}$$
$$(\cup\, \{\overline{\varnothing}\} \text{ ssi } \exists dc.t.q.\langle last(\tau), dc\rangle \notin dom(tr) \text{ et } \forall \langle a, x\rangle \in \gamma(dc(a) \in \alpha(x)(\tau)))$$

Les issues d'un contexte χ et d'un préfixe d'exécution τ pour USL^{cf} ($out^{cf}(\chi, \tau)$) incluent les séquences terminées par des actions conflictuelles :

Définition 4.24 (Issues pour USL^{cf})**.** *Soient une CGS $\mathcal{G} = \langle Ag, G, s_o, At, val, Ac, tr\rangle$, un contexte $\chi = \langle \alpha, \gamma\rangle$ et un préfixe d'exécution τ. On définit l'ensemble des* issues *de χ et τ pour USL^{cf} dans $\mathcal{G}$ comme l'ensemble $out^{cf}(\chi, \tau)$ des exécutions λ, dans $\mathcal{G}$, t.q. $\lambda_0 = last(\tau)$ et :*

- *si λ est infinie : pour tout $i \in \mathbb{N}$, $\lambda_{i+1} \in Msucc^{cf}_{\chi^{\lambda_{\leqslant i}}}(\lambda_{\leqslant i})$*
- *si λ est fini , $\lambda = \lambda_{\leqslant n}$:*
  - *pour tout $i < n$, $\lambda_{i+1} \in Msucc^{cf}_{\chi^{\lambda_{\leqslant i}}}(\lambda_{\leqslant i})$*
  - *$Msucc^{cf}_{\chi^{\lambda}}(\lambda) = \varnothing$ ou $\overline{\varnothing} \in Msucc^{cf}_{\chi^{\lambda}}(\lambda)$*

Si τ n'a qu'un seul élément $\tau_0 = last(\tau)$, on note également $out(\chi, \tau_0)$ pour $out(\chi, \tau)$.

La définition pour la relation de satisfaction sémantique pour USL^{cf}, $\models_{\text{USL}^{cf}}$, est identique à celle pour USL.

4.6 Sémantiques à multi-stratégies positionnelles

Chacune des sémantiques d'USL définies jusqu'ici admet une version positionnelle. Pour chaque sémantique, on obtient sa version positionnelle en modifiant seulement la notion de

multi-stratégies prise en compte, pour ne garder que les multi-stratégies qui dépendent exclusivement du dernier état visité. De nombreux cas de programmations et de nombreuses situations d'interactions peuvent être modélisés en utilisant seulement des multi-stratégies positionnelles. En particulier, la modélisation des instances de KHI fera appel à la version positionnelle de la sémantique USL^{cf} (voir Chapitre 6).

Ici, on ne fait que donner la définition pour les multi-stratégies positionnelles :

Définition 4.25 (multi-stratégies positionnelles)**.**

Une multi-stratégie positionnelle est une fonction partielle σ de G dans les parties non vides de Ac, $\sigma : G \nrightarrow \mathscr{P}_{>0}(Ac)$. On note $MStrat^0$ l'ensemble des multi-stratégies positionnelles dans un modèle.

Toutes les définitions sémantiques données jusqu'ici pour USL (resp. USL^{lp} et USL^{cf}) s'adaptent pour USL^0 (resp. USL$^{lp,\,0}$ et USL$^{cf,\,0}$) en modifiant seulement la clause pour $\langle\langle x \rangle\rangle$ dans la définition de la satisfaction sémantique : dans les versions à multi-stratégies positionnelles, les variables de multi-stratégies parcourent le domaine $MStrat^0$ à la place de $MStrat$.

Chapitre 5

Pouvoir expressif et propriétés métathéoriques d'USL

Sommaire

Dans ce chapitre, on poursuit l'étude de USL et de ses différentes sémantique. La Section 5.1 présente certaines propriétés remarquables exprimables par ces sémantiques. On s'intéresse, dans la Section 5.2, aux comparaisons en pouvoir expressif, d'une part entre les logiques multi-agents existantes et USL, d'autre part entre nos différentes sémantiques pour USL. Finalement, la Section 5.3 traite le *model-checking* des différentes sémantiques pour USL. Il est de complexité non élémentaire pour les sémantiques avec des multi-stratégies à mémoire, et il est PSPACE-complet pour les différentes versions d'USL interprétées avec des multi-stratégies positionnelles.

5.1 Propriétés remarquables exprimables avec USL

On illustre dans cette section certaines propriétés remarquables exprimables par les différentes sémantiques pour USL. On présente ainsi :

– les propriétés de *contrôle pérenne*. Il s'agit de capacités pour des agents à toujours contrôler des propriétés d'état. Elles sont communes à toutes les sémantiques définies ici pour USL et s'appuient sur les mécanismes de composition entre les différentes multi-stratégies des agents (Section 5.1.1).

– des propriétés d'inclusion et d'égalité entre différentes multi-stratégies. Elles permettent en particulier de retrouver les quantificateurs classiques sur les stratégies déterministes et de caractériser des relations de dépendances entre multi-stratégies (Section 5.1.2). Elles sont exprimables avec USL et USL^{cf}.

– Par ailleurs, la sémantique USL^{cf} n'est pas à proprement parler une sémantique alternative pour USL, mais une extension d'USL aux CGS^{cf}*s*. On voit dans la Section 5.1.3 comment on peut y caractériser la classe des modèles sans conflit de capacité (autrement dit la classe des CGSs).

5.1.1 Le contrôle pérenne

On illustre d'abord le mécanisme des raffinements de multi-stratégies à l'aide des propriétés de *contrôle pérenne*. Les notions de contrôle pérenne sur une propriété ou un ensemble de propriétés sont introduites à l'aide d'un exemple simple et d'une définition intuitive (Section 5.1.1.1). On généralise dans un second temps ces notions, on les formalise et on présente des schémas de formules pour les caractériser (Section 5.1.1.2).

5.1.1.1 Illustration par l'exemple du serveur de chat

Exemple 5.1 (Le serveur de chat)**.** *On suppose qu'un client envoie régulièrement des requêtes de connexion à un serveur de chat. Le modèle $\mathcal{G}_{CHAT,1}$ de la Figure 5.1 représente les actions possibles du serveur : il y a trois états, son langage a un agent, le serveur, et un atome* accès. *La valuation est indiquée directement dans les états et les transitions sont étiquetées chacune par les actions que le serveur doit jouer pour les forcer. La requête de connexion est reçue dans l'état s_0. Depuis chacun des états s_0 et s_1, le serveur peut accorder la connexion en jouant l'action* pourvoit *ou la refuser, en jouant l'action* refuse. *Par chacune de ces deux actions, le serveur maintient l'exécution dans s_0 ou s_1, d'où il peut à nouveau faire ce choix. Depuis ces états il peut par ailleurs jouer l'action* bannit *et ainsi conduire l'exécution dans l'état s_2 qui n'a pas de sortie. Dans cet état, la connexion est définitivement fermée pour le client. Nous souhaitons exprimer par une formule logique que le* serveur *peut agir de manière à toujours pouvoir décider la valeur de vérité de* accès *dans l'état suivant. Cette disposition est ce qu'on appelle le* contrôle pérenne *du serveur sur la propriété* accès *(voir Définitions 5.1 et 5.2).*

À ce stade, on peut donner une définition informelle de la propriété de contrôle pérenne :

Définition 5.1 (Pérennité, définition informelle)**.** *Un agent a le contrôle pérenne sur une propriété ssi il a toujours les capacités à la fois de garantir la satisfaction de cette propriété et de garantir la satisfaction de sa négation, et il conserve ces deux capacités même s'il les exerce.*

On commence par illustrer l'inadéquation de SL pour exprimer le contrôle pérenne sur l'exemple du serveur de chat. Il s'agit en fait d'une conjecture, qui est présentée dans la Section 5.2.1.

Un cas de contrôle qui n'est pas pérenne Considérons la propriété : "*le serveur* (S) peut jouer de manière à toujours pouvoir assurer ou refuser *accès* dans l'état suivant". En SL, si S est le seul agent du système considéré, cette propriété est caractérisée par la formule 5.1 :

$$\langle\langle x_1\rangle\rangle(S,x_1)\Box((\langle\langle x_2\rangle\rangle(S,x_2)\,\mathsf{X}\,\mathit{accès})\wedge(\langle\langle x_3\rangle\rangle(S,x_3)\,\mathsf{X}\,\neg\mathit{accès})) \tag{5.1}$$

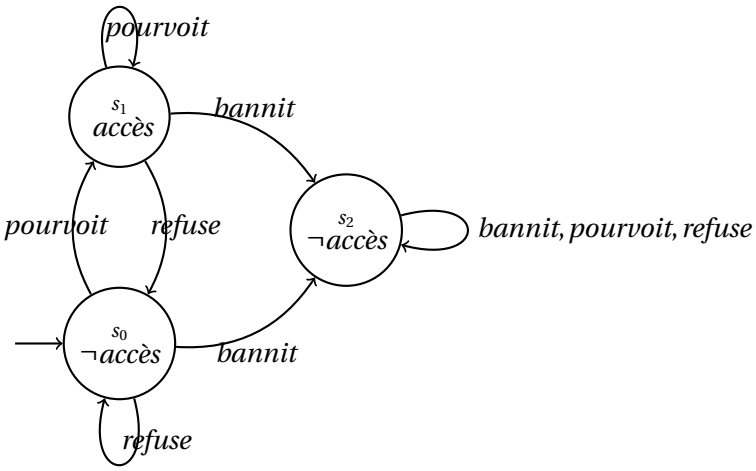

FIGURE 5.1 – Le modèle $\mathcal{G}_{CHAT,1}$ pour le serveur de chat

Dans la formule 5.1, la sous-formule $\langle\langle x_2\rangle\rangle(S, x_2)\,\mathsf{X}\,accès$ affirme que S peut adopter une stratégie σ_{x_2} qui assure $\mathsf{X}\,accès$, même si σ_{x_2} est en contradiction avec σ_{x_1} et lui fait perdre son contrôle sur la propriété *accès*. Et de même, la sous-formule $\langle\langle x_3\rangle\rangle(S, x_3)\,\mathsf{X}\,\neg accès$ affirme que S peut adopter une stratégie σ_{x_3} qui assure $\mathsf{X}\,\neg accès$ quitte à perdre son contrôle sur *accès*.

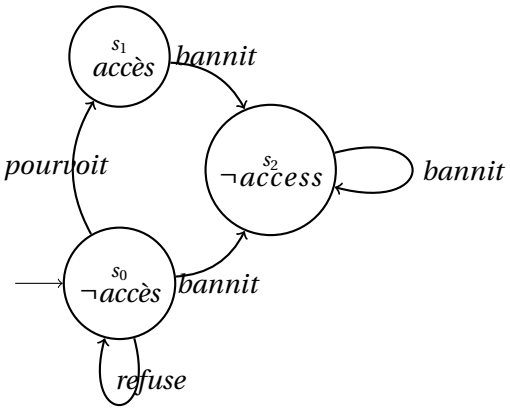

FIGURE 5.2 – Le modèle $\mathcal{G}_{CHAT,2}$ pour le serveur de chat

Considérons le modèle $\mathcal{G}_{CHAT,2}$ illustré dans la Figure 5.2. La formule 5.1 est vraie dans l'état s_0 de $\mathcal{G}_{CHAT,2}$. En effet, en jouant la stratégie *toujours-jouer*-refuse (*tjr*, définie par : pour tout $\tau \in Track$, $tjr(\tau) = refuse$, S maintient l'exécution dans l'état s_0, à partir duquel il peut toujours soit continuer à jouer cette stratégie pour assurer $\mathsf{X}\,\neg accès$, soit jouer *pourvoit* pour assurer $\mathsf{X}\,accès$. Cependant, pour assurer *accès* une fois, S doit révoquer *tjr* et adopter une nouvelle stratégie qui commence par le choix *pourvoit*. Ce faisant, S perd sa capacité à assurer $\mathsf{X}\,accès$.

Autrement dit, la formule 5.1 affirme que S peut choisir entre $\mathsf{X}\,accès$ et $\mathsf{X}\,\neg accès$ à tout moment, mais ne garantit pas que S puisse répéter ce choix plus d'une fois. Pour garantir cette possibilité, il faut pouvoir exprimer qu'en choisissant entre $\mathsf{X}\,accès$ et $\mathsf{X}\,\neg accès$, S continue en

même temps à agir selon la stratégie σ_{x_1}. On conjecture qu'un mécanisme de raffinement des stratégies est nécessaire pour exprimer cette propriété et qu'elle n'est donc pas exprimable dans SL. Cette conjecture est présentée dans la Section 5.2.1.

Un cas de contrôle pérenne Considérons maintenant la formule d'USL suivante. Elle est syntaxiquement similaire à la formule 5.1 (dans le sens où, syntaxiquement, l'une est obtenue de l'autre en changeant la notation pour le lieur) :

$$\langle\langle x_1\rangle\rangle(S \rhd x_1)\Box((\langle\langle x_2\rangle\rangle(S \rhd x_2)\mathsf{X}\,\mathit{accès}) \wedge (\langle\langle x_3\rangle\rangle(S \rhd x_3)\mathsf{X}\neg\mathit{accès})) \tag{5.2}$$

Pour que cette formule soit satisfaite, USL requiert que les multi-stratégies σ_{x_1} et σ_{x_2}, qui instancient x_1 et x_2, puissent être composées ensemble, de même que les multi-stratégies σ_{x_1} et σ_{x_3}. Ainsi, si S se lie à une des multi-stratégies σ_{x_2} ou σ_{x_3}, il ne révoque pas σ_{x_1}. La formule 5.2 exprime que S est capable de choisir entre $\mathsf{X}\,\mathit{accès}$ et $\mathsf{X}\neg\mathit{accès}$ autant qu'il le veut. Cette propriété n'est pas satisfaite dans le modèle $\mathcal{G}_{CHAT,2}$.

Revenons alors au modèle $\mathcal{G}_{CHAT,1}$ de la Figure 5.1, et considérons l'ensemble d'actions $\{\mathit{pourvoit}, \mathit{refuse}\}$. Cet ensemble d'actions correspond au choix de ne pas bannir le client du service, on le désigne également par *ne_bannit_pas*.

Comme énoncé formellement par l'Exemple 5.2 ci-dessous, la formule 5.2 est vraie dans l'état s_0 de $\mathcal{G}_{CHAT,1}$ avec la multi-stratégie *toujours-jouer*-ne_bannit_pas (*tjnb*, définie par : pour tout $\tau \in \mathit{Track}, \mathit{tjnb}(\tau) = \mathit{ne\_bannit\_pas}$) instanciant x_1. En effet, cette multi-stratégie peut toujours être raffinée en jouant l'action *pourvoit* (qui assure $\mathsf{X}\,\mathit{accès}$) ou en jouant l'action *refuse* (qui assure $\mathsf{X}\neg\mathit{accès}$) à n'importe quel moment. Avec la multi-stratégie *tjnb*, S reste toujours capable de décider entre $\mathsf{X}\,\mathit{accès}$ et $\mathsf{X}\neg\mathit{accès}$, autant de fois qu'il veut : son contrôle sur la propriété *accès* est pérenne. On en donne une preuve formelle dans ce qui suit :

Exemple 5.2 (Proposition)**.**

$$\mathcal{G}_{CHAT,1}, s_0 \models \langle\langle x_1\rangle\rangle(S \rhd x_1)\Box((\langle\langle x_2\rangle\rangle(S \rhd x_2)\mathsf{X}\,\text{accès}) \wedge (\langle\langle x_3\rangle\rangle(S \rhd x_3)\mathsf{X}\neg\text{accès}))$$

Démonstration. On présente d'abord les multi-stratégies utilisées dans la preuve pour instancier x_2 et x_3.

Pour x_2, on utilisera la multi-stratégie *jouer*-pourvoit-*une-fois* (*jpu*) définie par : $\mathit{jpu}(s_0) = \mathit{jpu}(s_1) = \mathit{jpu}(s_2) = \mathit{pourvoit}$.

Similairement, on utilisera pour x_3 la multi-stratégie *jouer*-refuse-*une-fois* (*jru*) définie par : $\mathit{jru}(s_0) = \mathit{jru}(s_1) = \mathit{jru}(s_2) = \mathit{refuse}$.

Avec ces notations, pour $i \in \{0,1\}$, on a :

$$\mathit{out}(\langle(x_1 \mapsto \mathit{tjnb}), \{\langle S, x_1\rangle\}\rangle, s_i) = s_i \cdot \{s_0, s_1\}^\omega$$

Ainsi, pour toute $\lambda \in \mathit{out}(\langle(x_1 \mapsto \mathit{tjnb}), \{\langle S, x_1\rangle\}\rangle, s_0)$, pour tout $n \in \mathbb{N}, \lambda_n \in \{s_0, s_1\}$ et donc :

$$\mathit{out}(\langle(x_1 \mapsto \mathit{tjnb}^{\lambda_0 \ldots \lambda_n}), \{\langle S, x_1\rangle\}\rangle, \lambda_n) = \lambda_n \cdot \{s_0, s_1\}^\omega$$

On a par ailleurs :

$$\mathit{out}(\langle(x_2 \mapsto \mathit{jpu}), \{\langle S, x_2\rangle\}\rangle, s_i) = s_i \cdot s_0 \cdot \{s_0, s_1, s_2\}^\omega$$

Et donc, pour le contexte $\langle(x_1 \mapsto \mathit{tjnb}^{\lambda_0 \ldots \lambda_n}, x_2 \mapsto \mathit{jpu}), \{\langle S, x_1\rangle\langle S, x_2\rangle\}\rangle$:

$$\mathit{out}(\langle(x_1 \mapsto \mathit{tjnb}^{\lambda_0 \ldots \lambda_n}, x_2 \mapsto \mathit{jpu}), \{\langle S, x_1\rangle\langle S, x_2\rangle\}\rangle, \lambda_n) = \lambda_n \cdot \{s_0, s_1\}^\omega \cap \lambda_n \cdot s_0 \cdot \{s_0, s_1, s_3\}^\omega = \lambda_n \cdot s_0 \cdot \{s_0, s_1\}^\omega$$

Il en résulte que pour tout

$$\lambda' \in out(\langle (x_1 \mapsto tjnb^{\lambda_0 \dots \lambda_n}, x_2 \mapsto jpu), \{\langle S, x_1\rangle\langle S, x_2\rangle\}\rangle, \lambda_n),$$

$acc\grave{e}s \in val(\lambda'_1)$. On peut maintenant dériver les étapes d'évaluation : pour tout

$$\lambda \in out(\langle (x_1 \mapsto tjnb), \{\langle S, x_1\rangle\}\rangle, s_0),$$

pour tout $n \in \mathbb{N}$, pour tout

$$\lambda' \in out(\langle (x_1 \mapsto tjnb^{\lambda_0 \dots \lambda_n}, x_2 \mapsto jpu), \{\langle S, x_1\rangle\langle S, x_2\rangle\}\rangle, \lambda_n)$$

on a :

$$\begin{array}{rl} & \mathcal{G}_{CHAT,1}, \langle (x_1 \mapsto tjnb^{\lambda_0 \dots \lambda_n}, x_2 \mapsto jpu^{\lambda_n \lambda'_1}), \{\langle S, x_1\rangle\langle S, x_2\rangle\}\rangle, \lambda'_1 \models_{\text{USL}} acc\grave{e}s \\ \text{donc} & \mathcal{G}_{CHAT,1}, \langle (x_1 \mapsto tjnb^{\lambda_0 \dots \lambda_n}, x_2 \mapsto jpu), \{\langle S, x_1\rangle\langle S, x_2\rangle\}\rangle, \lambda_n \models_{\text{USL}} \mathsf{X}\, acc\grave{e}s \\ \text{et} & \mathcal{G}_{CHAT,1}, \langle (x_1 \mapsto tjnb^{\lambda_0 \dots \lambda_n}, x_2 \mapsto jpu), \{\langle S, x_1\rangle\}\rangle, \lambda_n \models_{\text{USL}} (S \rhd x_2)\mathsf{X}\, acc\grave{e}s \end{array}$$

De manière similaire, on prouve que pour tout $\lambda \in out(\langle (x_1 \mapsto tjnb), \{\langle S, x_1\rangle\}\rangle, s_0)$, pour tout $n \in \mathbb{N}$:

$$\mathcal{G}_{CHAT,1}, \langle (x_1 \mapsto tjnb^{\lambda_0 \dots \lambda_n}, x_3 \mapsto jru), \{\langle S, x_1\rangle\}\rangle, \lambda_n \models_{\text{USL}} (S \rhd x_3)\mathsf{X}\, \neg acc\grave{e}s$$

On a donc successivement, pour tout $\lambda \in out(\langle (x_1 \mapsto tjnb), \{\langle S, x_1\rangle\}\rangle, s_0)$, pour tout $n \in \mathbb{N}$:

$$\begin{array}{l} \mathcal{G}_{CHAT,1}, \langle (x_1 \mapsto tjnb^{\lambda_0 \dots \lambda_n}), \{\langle S, x_1\rangle\}\rangle, \lambda_n \models_{\text{USL}} \\ \qquad (\langle\langle x_2\rangle\rangle(S \rhd x_2)\mathsf{X}\, acc\grave{e}s) \wedge (\langle\langle x_3\rangle\rangle(S \rhd x_3)\mathsf{X}\, \neg acc\grave{e}s) \\ \mathcal{G}_{CHAT,1}, \langle (x_1 \mapsto tjnb), \{\langle S, x_1\rangle\}\rangle, s_0 \models_{\text{USL}} \\ \qquad \Box((\langle\langle x_2\rangle\rangle(S \rhd x_2)\mathsf{X}\, acc\grave{e}s) \wedge (\langle\langle x_3\rangle\rangle(S \rhd x_3)\mathsf{X}\, \neg acc\grave{e}s)) \\ \mathcal{G}_{CHAT,1}, \langle (x_1 \mapsto tjnb), \gamma_\varnothing\rangle, s_0 \models_{\text{USL}} \\ \qquad (S \rhd x_1)\Box((\langle\langle x_2\rangle\rangle(S \rhd x_2)\mathsf{X}\, acc\grave{e}s) \wedge (\langle\langle x_3\rangle\rangle(S \rhd x_3)\mathsf{X}\, \neg acc\grave{e}s)) \\ \mathcal{G}_{CHAT,1}, \langle \alpha_\varnothing, \gamma_\varnothing\rangle, s_0 \models_{\text{USL}} \\ \qquad \langle\langle x_1\rangle\rangle(S \rhd x_1)\Box((\langle\langle x_2\rangle\rangle(S \rhd x_2)\mathsf{X}\, acc\grave{e}s) \wedge (\langle\langle x_3\rangle\rangle(S \rhd x_3)\mathsf{X}\, \neg acc\grave{e}s)) \\ \mathcal{G}_{CHAT,1} \models_{\text{USL}} \langle\langle x_1\rangle\rangle(S \rhd x_2)\Box(\langle\langle x_2\rangle\rangle(S \rhd x_2)\mathsf{X}\, acc\grave{e}s) \end{array}$$

□

Révocation de multi-stratégie On illustre également ici le mécanisme de la révocation des multi-stratégies. La multi-stratégie *tjnb* assure que l'exécution est maintenue entre les états s_0 et s_1. Or, à partir de chacun de ces états, le serveur peut également jouer l'action *bannit* et mener l'exécution en s_2 de manière à bannir le client. Comme l'action *bannit* est toujours contradictoire avec l'ensemble d'actions *ne_bannit_pas* indiquée par la multi-stratégie *tjnb*, le serveur doit d'abord révoquer la multi-stratégie *tjnb* pour jouer *bannit*. La description des capacités du serveur est donc achevée avec la formule 5.3 : il peut à tout moment, renoncer à son contrôle pérenne sur *accès* et refuser cet accès une fois pour toutes.

$$\begin{array}{l} \mathcal{G}_{CHAT,1}, s_0 \models \langle\langle x\rangle\rangle(S \rhd x)\Box(\langle\langle y\rangle\rangle(S \rhd y)\mathsf{X}\, acc\grave{e}s) \wedge (\langle\langle y\rangle\rangle(S \rhd y)\mathsf{X}\, acc\grave{e}s) \\ \qquad\qquad \wedge \langle\langle y\rangle\rangle(S \rhd y)\mathsf{X}\, \neg acc\grave{e}s \wedge \langle\langle z\rangle\rangle(S \not\rhd x)(S \rhd z)\Box \neg acc\grave{e}s \end{array} \quad (5.3)$$

5.1.1.2 Généralisation, définitions formelles et caractérisation en USL

On peut désormais généraliser les caractérisations de la pérennité et donner une version formelle de la Définition 5.1 :

Définition 5.2 (Pérennité, formalisation)**.** *Soient $\mathcal{G}$ une CGS sur l'ensemble d'états G, A une coalition d'agents, et soit une propriété des états représentée par un ensemble $P \subseteq G$. Pour tout état s de $\mathcal{G}$, on dit que A a le contrôle pérenne sur P en l'état s, et on note* $\mathsf{ContP}(s, A, P)$*, ssi :*

- *les agents de A disposent à la fois d'un vecteur d'actions qui garantit d'aller dans l'ensemble P et d'un vecteur d'actions qui garantit de ne pas y aller.*
- *chacun de ces vecteurs d'actions garantit par ailleurs une transition vers un état s' t.q.* $\mathsf{ContP}(s', A, P)$

Formellement, pour tout état s, $\mathsf{ContP}(s, A, P)$ *ssi pour tout agent $a \in A$, il y a des actions ac_1^a et ac_2^a telles que pour toute $dc \in Dc$:*

- *Si pour tout $a \in A, dc(a) = ac_1^a$ alors $tr(s, dc) \in P$ et* $\mathsf{ContP}(tr(s, dc), A, P)$
- *Si pour tout $a \in A, dc(a) = ac_2^a$ alors $tr(s, dc) \notin P$ et* $\mathsf{ContP}(tr(s, dc), A, P)$

Par ailleurs, on désigne par $\mathsf{ContP}(A, P)$ *la propriété, pour une CGS, d'être telle que* $\mathsf{ContP}(s_0, A, P)$*, où s_0 est l'état initial de $\mathcal{G}$. On écrit alors $\mathcal{G} \in \mathsf{ContP}(A, P)$.*

On a alors la proposition suivante :

Proposition 5.1. *Si P est une propriété des états des CGSs caractérisable par un énoncé φ d'USL (c'est-à-dire si pour toute CGS $\mathcal{G}$, pour tout état s de $\mathcal{G}$, $P(s)$ ssi $\mathcal{G}, s \models_{USL} \varphi$), alors pour toute CGS $\mathcal{G}$, en notant*

$$Form(\mathsf{ContP}(A, P)) \triangleq \langle\langle \vec{x} \rangle\rangle (A \rhd \vec{x}) \Box ((\langle\langle \vec{y_1} \rangle\rangle (A \rhd \vec{y_1}) \mathsf{X} \varphi) \wedge (\langle\langle \vec{y_2} \rangle\rangle (A \rhd \vec{y_2}) \mathsf{X} \neg \varphi))$$

on a :

$$\mathcal{G} \in \mathsf{ContP}(A, P) \text{ ssi } \mathcal{G} \models_{USL} Form(\mathsf{ContP}(A, P)) \text{ ssi } \mathcal{G} \models_{USL^{lp}} Form(\mathsf{ContP}(A, P))$$

Démonstration. (Résumé) La preuve vient directement, par application des définitions de la pérennité et des sémantiques USL et USL^{lp}. La formule $Form(\mathsf{ContP}(s, A, P))$ est vraie ssi A dispose d'une multi-stratégie telle qu'en la jouant, il peut toujours forcer la prochaine transition dans un état qui satisfait φ ou dans un état qui satisfait $\neg\varphi$, sans révoquer sa première multi-stratégie. □

La Proposition 5.1 assure par ailleurs que cette propriété a effectivement des instances.

5.1.2 Propriétés des multi-stratégies : inclusion, égalité, déterminisme

La formalisation des propriétés que nous allons exhiber ici est redevable de la modélisation de la composition des engagements contradictoires telle qu'elle est faite dans la sémantique standard pour USL. En modélisant des engagements contradictoires par un arrêt de l'exécution, cette sémantique permet de caractériser ces cas et donc d'effectuer des tests d'intersection vide entre les ensembles d'actions indiqués par différentes multi-stratégies. Les formules présentées ici utilisent donc la sous-formule $(a \rhd x)(a \rhd y)\mathsf{X}\top$. Elle est vraie après un préfixe d'exécution τ ssi x et y indiquent des ensembles d'actions en intersection non vide après τ.

Les propriétés présentées ci-dessous ne sont pas exprimables à l'aide d'USL^{lp} ou $USL^{lp,\,0}$. On présente d'abord des propriétés générales de comparaisons de multi-stratégies (Section 5.1.2.1).

On introduit ensuite des quantificateurs sur les stratégies (déterministes), dans la Section 5.1.2.2, on étudie les relations de *dépendance* entre les multi-stratégies quantifiées dans une formule (Section 5.1.2.3) et on caractérise une relation de *dépendance forte* entre multi-stratégies (Section 5.1.2.4).

5.1.2.1 Inclusion, égalité et unicité

On considère tout d'abord l'inclusion pour une transition : dans l'état courant, l'ensemble d'actions indiqué par la multi-stratégie x est inclus dans celui indiqué par y. Ceci est caractérisé par la formule 5.4 :

$$x \subseteq_{\mathsf{X}} y \triangleq [\![z]\!]((a \rhd x)(a \rhd z)\mathsf{X}\top \rightarrow (a \rhd y)(a \rhd z)\mathsf{X}\top) \tag{5.4}$$

En effet, cette formule affirme que tout ensemble d'actions en intersection non vide avec celui indiqué par x est aussi en intersection non vide avec celui indiqué par y. À partir de cette définition, on définit aisément l'égalité $=_{\mathsf{X}}$ pour une transition. On définit aussi l'inclusion globale $\subseteq$ (et l'égalité globale $=$) entre deux multi-stratégies.

$$x =_{\mathsf{X}} y \triangleq x \subseteq_{\mathsf{X}} y \wedge y \subseteq_{\mathsf{X}} x \quad x \subseteq y \triangleq \Box x \subseteq_{\mathsf{X}} y \quad x = y \triangleq x \subseteq y \wedge y \subseteq x$$

La formule $x \subseteq y$ affirme que l'ensemble d'actions indiqué par x est inclus dans celui indiqué par y, après tout préfixe d'exécution τ qui commence par l'état courant. Similairement, l'égalité $x = y$ caractérise des multi-stratégies qui indiquent le même ensemble d'actions après tout préfixe d'exécution qui commence par l'état courant. L'existence d'un prédicat d'égalité permet de *compter* le nombre de multi-stratégies possibles pour un but donné. En particulier, on peut exprimer l'existence d'une unique multi-stratégie pour satisfaire un but.

Définition 5.3 (Unicité)**.** *Le quantificateur d'unicité ($\langle\!\langle !x \rangle\!\rangle$) est défini comme suit : pour toute formule φ telle que x est libre dans φ :*

$$\langle\!\langle !x \rangle\!\rangle \varphi \triangleq \langle\!\langle x \rangle\!\rangle (\varphi \wedge [\![y]\!](\varphi[y \setminus x] \rightarrow x = y))$$

où $\varphi[y \setminus x]$ est obtenue de φ en y substituant toute ocurrence libre de x par y.

S'il y a une unique multi-stratégie pour assurer un but donné, alors la satisfaction de ce but détermine complètement la multi-stratégie de l'agent qui poursuit ce but.

5.1.2.2 Stratégies (déterministes)

L'inclusion et l'égalité permettent aussi d'exprimer qu'une multi-stratégie est déterministe, *i.e.* qu'elle indique une unique action après tout préfixe d'exécution.Pour exprimer en USL le fait qu'une variable x représente une multi-stratégie qui est déterministe *depuis l'état courant*, on spécifie que la seule multi-stratégie incluse dans x est x elle-même. Ceci est exprimé par la formule 5.5, dans laquelle x est une variable libre :

$$Det(x) \triangleq [\![y]\!](y \subseteq x \rightarrow y = x) \tag{5.5}$$

Une manière naturelle de raisonner sur les stratégies explicitement déterministes consiste à utiliser des quantificateurs spéciaux.

Définition 5.4 (Quantificateurs sur les stratégies)**.** *Les quantificateurs existentiel ($\langle\!\langle x \rangle\!\rangle_d$) et universel ($[\![x]\!]_d$) sur les stratégies sont définis comme suit :*

Existential *Pour toute formule φ, $\langle\!\langle x \rangle\!\rangle_d \varphi \triangleq \langle\!\langle x \rangle\!\rangle (Det(x) \wedge \varphi)$*

Universal *Pour toute formule φ, $[\![x]\!]_d \varphi \triangleq [\![x]\!](Det(x) \rightarrow \varphi)$*

5.1.2.3 Alternance de quantificateurs et dépendance

Ces quantificateurs sur les stratégies sont utiles pour comparer USL et les logiques qui utilisent des stratégies, comme SL (on les utilise notamment dans la Section 5.2.2.1 pour définir un *plongement* - cf. Définition 5.6 - de SL dans USL), mais ils sont également utilisés pour exprimer des propriétés qui impliquent des alternances de quantificateurs. En effet, dans le cas d'une alternance de quantificateurs de la forme $[\![x]\!]\langle\langle y\rangle\rangle\varphi$, on doit utiliser le quantificateur $[\![x]\!]_d$ à la place de $[\![x]\!]$ pour permettre à la multi-stratégie y de dépendre de x. On détaille cet aspect grâce à l'exemple suivant :

Exemple 5.3 (Relation de dépendance)**.** *Considérons la CGS $\mathcal{G}_{[\![]\!]\langle\langle\rangle\rangle}$ illustrée dans la Figure 5.3. Il y a deux agents a et b et deux actions ac_1 et ac_2. Il y a aussi trois états, s_0 (l'état initial), s_1 et s_2. L'état s_1 est atteint depuis s_0 ssi les deux joueurs jouent la même action et l'état s_2 est rejoint depuis s_0 ssi ils jouent différemment. Quelle que soit la décision jouée en s_1, l'exécution reste en s_1, et similairement pour l'état s_2. Le langage contient une unique proposition atomique p, qui est vraie seulement dans l'état s_1. On veut exprimer le fait que quelle que soit l'action jouée par a depuis s_0, b peut jouer de manière à assurer p. C'est trivialement vrai dans notre exemple, comme b peut toujours jouer la même action que a pour assurer p.*

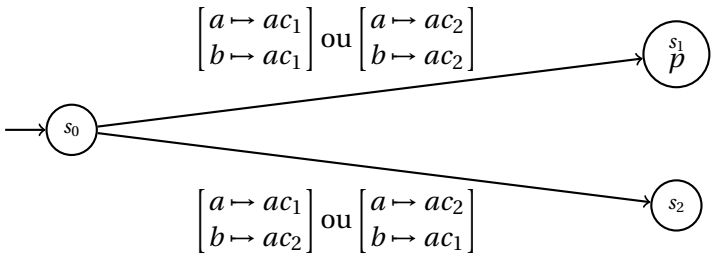

FIGURE 5.3 – Le Modèle $\mathcal{G}_{[\![]\!]\langle\langle\rangle\rangle}$

On explique d'abord pourquoi la formule

$$[\![x]\!]\langle\langle y\rangle\rangle(a \rhd x)(b \rhd y)\mathsf{X}\,p \tag{5.6}$$

ne convient pas pour exprimer cette propriété. En effet, elle spécifie que pour toute multi-stratégie σ_x, il y a une multi-stratégie σ_y t.q., en s_0, si a joue σ_x et b joue σ_y alors l'exécution va en s_1. Cependant, si σ_x est une multi-stratégie qui indique l'ensemble de toutes les actions possibles après s_0, alors il n'y a pas de multi-stratégie σ_y qui permette à b d'assurer $\mathsf{X}\,p$ si a joue σ_x. La formule 5.6 n'est donc pas satisfaite en s_0. Elle est en fait équivalente à la formule 5.7, obtenue de la formule 5.6 en commutant les quantificateurs (la preuve est laissée au lecteur) :

$$\langle\langle y\rangle\rangle[\![y]\!](a \rhd x)(b \rhd y)\mathsf{X}\,p \tag{5.7}$$

Ainsi, la quantification universelle sur les multi-stratégies ne semble pas adéquate pour exprimer les relations de dépendances usuelles liées aux alternances de quantificateurs. Il semble plutôt qu'il soit nécessaire pour cela d'utiliser le quantificateur universel sur les stratégies, comme dans la formule 5.8, vraie dans l'état s_0 de notre exemple :

$$[\![x]\!]_d\langle\langle y\rangle\rangle(a \rhd x)(b \rhd y)\mathsf{X}\,p \tag{5.8}$$

Cette dernière formule est vraie dans tout modèle où y dépend de x. Mais elle est aussi vraie dans tout modèle où y est complètement indépendant de x, *i.e.* tel qu'il y a une multi-stratégie y telle que pour toute stratégie x, $(a \rhd x)(b \rhd y)\mathsf{X}\, p$ est vraie. Autrement dit, la formule

$$[\![x]\!]_d \langle\langle y \rangle\rangle (a \rhd x)(b \rhd y)\mathsf{X}\, p$$

est une conséquence sémantique de la formule

$$\langle\langle y \rangle\rangle [\![x]\!]_d (a \rhd x)(b \rhd y)\mathsf{X}\, p.$$

La formule 5.8 affirme donc qu'il y a une relation de dépendance *potentielle* entre x et y. La formule spécifiant que y dépend de x est donc la suivante :

$$\big([\![x]\!]_d \langle\langle y \rangle\rangle (a \rhd x)(b \rhd y)\mathsf{X}\, p\big) \wedge \neg(\langle\langle y \rangle\rangle [\![x]\!]_d (a \rhd x)(b \rhd y)\mathsf{X}\, p) \tag{5.9}$$

En effet, elle affirme que la multi-stratégie y ne peut pas être la même quel que soit x, pour satisfaire $(a \rhd x)(b \rhd y)\mathsf{X}\, p$.

5.1.2.4 Dépendance forte

Grâce à l'égalité, on peut exprimer une relation plus fine de dépendance entre les multi-stratégies. Intuitivement, une façon d'étudier la force de la relation de dépendance entre deux multi-stratégies x et y est de regarder les conséquences sur y d'un changement de la valeur pour x. En particulier, on dit que y *dépend fortement* de x pour la satisfaction de φ si tout changement de x induit un changement de y pour la satisfaction de φ.

On illustre cet aspect à l'aide d'un exemple :

Exemple 5.4 (Tour de magie)**.** *On considère un jeu à deux joueurs, le magicien et le spectateur. Le spectateur choisit une carte dans un jeu de* 52 *cartes distinctes deux à deux, et la replace dans le paquet. Le magicien choisit à son tour une carte dans ce même paquet. Le jeu est itéré un nombre infini de fois. On distingue deux niveaux :*

Niveau débutant *Le magicien gagne ssi il choisit toujours une carte différente de celle choisie par le spectateur.*

Niveau expert *Le magicien gagne ssi il choisit toujours la même carte que le spectateur.*

On veut caractériser le fait que le niveau expert est plus difficile que le niveau débutant pour le magicien.

Le jeu est représenté par le modèle $\mathcal{G}_M$ de la Figure 5.4. Il y a trois états s_0 (l'état initial), s_1 et s_2 et autant d'actions que de cartes dans le jeu. La fonction de transition encode le fait que, depuis chaque état, l'exécution va dans :

- s_1 si les deux joueurs choisissent la même carte,
- s_2 s'ils choisissent des cartes différentes.

La proposition $p_=$ est vraie dans l'état s_1 et seulement dans cet état. Elle authentifie que les joueurs ont choisi la même carte dans le dernier tour du jeu.

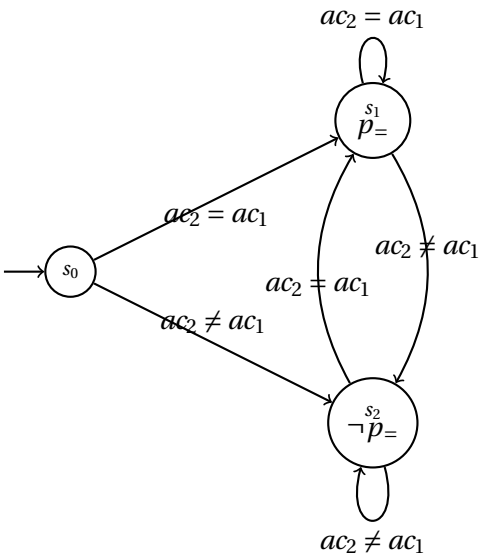

FIGURE 5.4 – Illustration du tour de magie : $\mathscr{G}_M$

Dans les deux niveaux du jeu, le magicien a une multi-stratégie gagnante, qui dépend de x. Ainsi, les formules suivantes, qui sont similaires à la formule 5.9, sont vraies dans notre modèle :

$$\mathscr{G}_M \models_{\text{USL}} \big([\![x]\!]_d \langle\langle y\rangle\rangle (\mathit{spectateur} \rhd x)(\mathit{magicien} \rhd y) \mathsf{X} \Box p_= \big) \wedge \neg(\langle\langle y\rangle\rangle [\![x]\!]_d (\mathit{spectateur} \rhd x)(\mathit{magicien} \rhd y) \mathsf{X} \Box p_=)$$

et

$$\mathscr{G}_M \models_{\text{USL}} \big([\![x]\!]_d \langle\langle y\rangle\rangle (\mathit{spectateur} \rhd x)(\mathit{magicien} \rhd y) \mathsf{X} \Box \neg p_= \big) \wedge \neg(\langle\langle y\rangle\rangle [\![x]\!]_d (\mathit{spectateur} \rhd x)(\mathit{magicien} \rhd y) \mathsf{X} \Box \neg p_=)$$

Le fait que le niveau expert est plus difficile que le niveau débutant vient de ce que pour le niveau expert, y *dépend fortement* de x : tout changement dans la stratégie jouée par le spectateur rend nécessaire un changement dans la multi-stratégie du magicien. Ce n'est pas le cas au niveau débutant, comme pour toute action du spectateur, le magicien dispose de 51 actions favorables possibles. Intuitivement, pour gagner dans le niveau expert, le magicien a besoin d'une information complète sur le choix du spectateur. Dans le niveau débutant, une information partielle serait suffisante (comme par exemple la couleur de la carte tirée par le spectateur). La formule suivante exprime le fait que que la multi-stratégie du magicien dépend fortement de la stratégie du spectateur pour gagner le jeu dans le niveau expert.

$$\mathscr{G}_M \models_{\text{USL}} [\![x]\!]_d \langle\langle y\rangle\rangle (\mathit{spectateur} \rhd x)(\mathit{magicien} \rhd y) \mathsf{X} \Box p_= \wedge \Big([\![x]\!]_d [\![y]\!] [\![x']\!]_d [\![y']\!] \Big(((\mathit{spectateur} \rhd x)(\mathit{magicien} \rhd y) \mathsf{X} \Box p_=) \wedge ((\mathit{spectateur} \rhd x')(\mathit{magicien} \rhd y') \mathsf{X} \Box p_=) \wedge x \neq x' \Big) \rightarrow y \neq y' \Big)$$

La première partie de cette formule établit que pour toute stratégie jouée par le spectateur, le magicien à une multi-stratégie gagnante, et le reste de la formule affirme que si deux paires $\langle x, y\rangle$ et $\langle x', y'\rangle$ d'une stratégie et d'une multi-stratégie (x et x' pour le spectateur et y et y' pour le

magicien) sont telles que $x \neq x'$ et permettent chacune d'assurer que le magicien gagne dans le niveau expert, alors elles sont également telles que $y \neq y'$. Donc, pour tout changement dans la stratégie du spectateur, le magicien est obligé de changer sa multi-stratégie pour gagner le jeu de niveau expert.

La formule similaire pour le niveau débutant (où $p_=$ est remplacé par $\neg p_=$) est évidemment fausse dans $\mathscr{G}_M$.

5.1.3 Propriétés des CGS^{cf}s, USL^{cf}

USL^{cf} permet d'exprimer des propriétés sur les CGS^{cf}s, qui sont une classe de modèle strictement plus importante que les CGSs. En effet, toute CGS est une CGS^{cf}. On peut définir la classe des CGSs à partir de la classe des CGS^{cf}s par la contrainte suivante : la fonction tr est totale. Or, cette contrainte est caractérisable par une formule d'USL^{cf} sous certaines conditions (on peut dire que la fonction de transition est totale dans tout état accessible depuis l'état d'évaluation). On présente ici cette caractérisation, ainsi que les résultats de comparaison de pouvoir expressif entre USL et USL^{cf} qui en découlent.

On définit d'abord la notion de *CGS atteignable* :

Définition 5.5 (CGSs atteignable, CGS^{cf}s atteignable)**.**

Soit une CGS $\mathscr{G} = \langle Ag, G, s_o, At, val, Ac, tr\rangle$. On dit de $\mathscr{G}$ qu'elle est atteignable *ssi pour tout état s' de $\mathscr{G}$, s' est accessible depuis l'état initial s_0 par une suite finie de décisions des agents. Formellement, ssi pour tout état s' il existe une séquence finie $s_0, \dots s_n$ telle que pour tout $k < n$, il existe $ac_1, \dots ac_{|Ag|}$ t.q. $tr(s_k, \langle (a_1 \mapsto ac_1) \dots (a_{|Ag|} \mapsto ac_{|Ag|})\rangle) = s_{k+1}$. On appelle CGS^{cf} atteignable une CGS^{cf} qui satisfait la même condition.*

Soit encore $\mathscr{G}$ une CGS ou une CGS^{cf}. On appelle la CGS (ou CGS^{cf}) atteignable engendrée par *$\mathscr{G}$ la CGS atteignable (ou la CGS^{cf} atteignable) obtenue de $\mathscr{G}$ en supprimant tous les états qui ne sont pas accessibles depuis l'état initial.*

Remarque 5.1. *Les résultats présentés ici sont valables sur les modèles atteignables. Ceci constitue, en toute rigueur, une restriction de leur portée. On relativise cependant la portée de cette restriction en observant qu'utiliser des CGSs non atteignables (*i.e. *qui contiennent des états non accessibles) n'a pas de pertinence dans la modélisation d'un système. Par ailleurs, deux CGSs qui engendrent le même modèle atteignable sont indistingables par l'ensemble des formalismes multi-agents considérés dans ce mémoire.*

On dresse les résultats suivants. Ils établissent d'une part que tout ce qui peut être décrit par USL sur des CGSs peut aussi l'être par USL^{cf}, d'autre part que pour toute formule φ, la classe des CGSs atteignables qui sont modèles de φ est définissable parmi les CGS^{cf}s atteignables.

Proposition 5.2.

1. *La classe des CGS^{cf}s $\mathscr{G}$ telles que la CGS^{cf} atteignable engendrée par $\mathscr{G}$ est une CGS est caractérisable par une formule d'USL^{cf}. Autrement dit, il existe une formule φ d'USL^{cf} telle que pour toute CGS^{cf} $\mathscr{G}$, $\mathscr{G} \models_{USL^{cf}} \varphi$ ssi la CGS^{cf} atteignable engendrée par $\mathscr{G}$ est une CGS.*
2. *Les deux sémantiques USL et USL^{cf} sont équivalentes sur la classe des CGSs : pour toute CGS $\mathscr{G}$ et pour toute formule φ d'USL, $\mathscr{G}, s \models_{USL} \varphi$ ssi $\mathscr{G}, s \models_{USL^{cf}} \varphi$*
3. *Pour toute formule φ, la classe des CGSs atteignables qui satisfont φ est caractérisable avec la sémantique USL^{cf} parmi les CGS^{cf}s atteignables : il existe une formule $\mathsf{usl}^{cf}(\varphi)$ telle que pour toute CGS^{cf} atteignable $\mathscr{G}$, $\mathscr{G} \models_{USL^{cf}} \mathsf{usl}^{cf}(\varphi)$ ssi $\mathscr{G}$ est une CGS et $\mathscr{G} \models_{USL} \varphi$.*

Démonstration. (Résumé) Ces trois assertions viennent de manière immédiate :

1. Une CGS^{cf} atteignable $\mathcal{G}$ est une CGS ssi sa fonction de transition est totale, c'est-à-dire ssi toute décision depuis chaque état accessible depuis l'état initial s_0 définit un successeur, donc ssi
$$\mathcal{G} \models_{\text{USL}^{cf}} [\![\vec{x}]\!](Ag, \vec{x})\Box \mathsf{X} \top$$
2. L'équivalence sémantique dans le cas où la fonction de transition est totale découle directement des définitions données pour USL^{cf}.
3. La troisième assertion vient immédiatement à partir des deux premières : pour toute formule φ d'USL, $\mathsf{usl}^{cf}(\varphi) \triangleq [\![\vec{x}]\!](Ag, \vec{x})\Box \mathsf{X} \top \wedge \varphi$.

□

5.2 Expressivité des différentes sémantiques pour USL

On s'intéresse dans cette section à la comparaison entre les pouvoirs expressifs des différentes sémantiques pour USL et des autres formalismes multi-agents connus et présentés dans le Chapitre 2. Ces comparaisons sont menées sur l'ordre partiel $\preceq$ défini comme suit pour des langages interprétant les CGSs :

Définition 5.6 (Equivalence, expressivité)**.**

Formules équivalentes *Soient $\mathscr{L}$ et $\mathscr{L}'$ deux logiques interprétées par des CGSs et soient deux énoncés φ et φ' respectivement de $\mathscr{L}$ et $\mathscr{L}'$. On dit que φ et φ' sont équivalentes ssi pour toute CGS $\mathcal{G}$:*
$$\mathcal{G} \models_{\mathscr{L}} \varphi \text{ ssi } \mathcal{G} \models_{\mathscr{L}'} \varphi'$$

Expressivité

- *On dit que $\mathscr{L}$ est au moins aussi expressive que $\mathscr{L}'$ et on note $\mathscr{L}' \preceq \mathscr{L}$ ssi pour toute formule de $\mathscr{L}'$ il y a une formule de $\mathscr{L}$ équivalente. On dit alors qu'il y a un* plongement *de $\mathscr{L}'$ dans $\mathscr{L}$.*
- *On dit que $\mathscr{L}$ est strictement plus expressive que $\mathscr{L}'$ et on note $\mathscr{L} \prec \mathscr{L}'$ ssi $\mathscr{L} \preceq \mathscr{L}'$ est vrai et $\mathscr{L}' \preceq \mathscr{L}$ n'est pas vrai.*
- *On dit que $\mathscr{L}$ et $\mathscr{L}'$ sont incomparables ssi à la fois $\mathscr{L} \not\preceq \mathscr{L}'$ et $\mathscr{L}' \not\preceq \mathscr{L}$.*

Les résultats principaux de cette section sont présentés dans la Figure 5.5. Soient deux langages $\mathscr{L}$ et $\mathscr{L}'$, une flèche en traits pleins de $\mathscr{L}$ vers $\mathscr{L}'$ indique la relation $\mathscr{L} \prec \mathscr{L}'$. Une flèche en pointillés indique la relation $\mathscr{L} \preceq \mathscr{L}'$. Dans la Figure 5.5, cette relation est utilisée entre SL et USL^{lp}. Elle y est barrée, dans les deux sens, pour indiquer que SL $\not\preceq$ USL^{lp} et USL^{lp} $\not\preceq$ SL. Comme indiqué dans la figure, ils s'agit de deux conjectures. Finalement, on a représenté par $\subsetneq$ la relation entre USL^{cf} et USL : USL est incluse dans USL^{cf} dans le sens où toute classe de modèles caractérisable par USL est également caractérisable par USL^{cf}. L'inclusion est stricte.

Cette dernière relation a été prouvée dans la Section 5.1.3. Les comparaisons entre SL, ATL$^*_{sc}$, IATL, ATL* et ATL sont traitées dans [DCL11]. Dans la présente section, on expose les deux conjectures sur l'absence réciproque de plongement entre SL et USL^{lp} (Section 5.2.1). On montre ensuite qu'USL^{lp} $\prec$ USL et que SL $\prec$ USL (Section 5.2.2).

5.2.1 Expressivité d'USL^{lp}

On expose ici les deux conjectures, selon lesquelles USL^{lp} et SL sont incomparables.

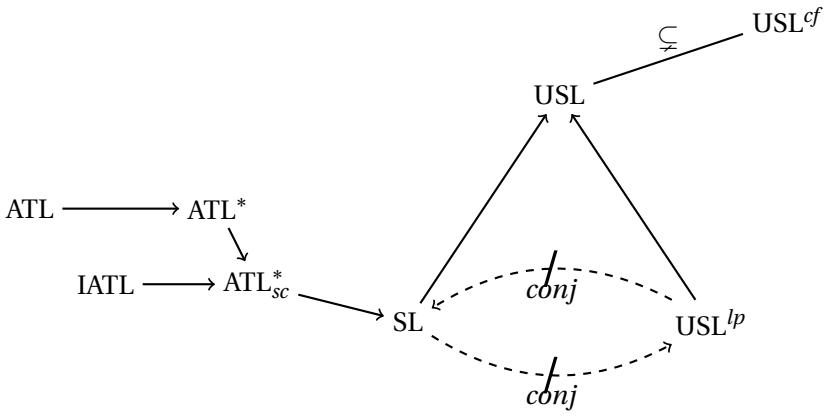

FIGURE 5.5 – Expressivité des formalismes multi-agents évoqués dans ce mémoire

5.2.1.1 Il n'y a pas de plongement d'USL^{lp} dans SL

On utilise le contrôle pérenne pour réfuter l'existence d'un plongement d'USL^{lp} dans SL. Le raisonnement est mené sur la formule d'USL^{lp} $Form(\mathsf{ContP}(a,p))$, dans la classe des CGSs qui interprétent le langage SL sur $\{a\}, At$ et X, où a est un agent, At contient la proposition p et X contient la variable x (on note SL$(\{a\}, At, X)$). La contrainte sur l'absence d'agent libre dans les énoncés de SL nous oblige en effet à travailler sur cette classe de CGSs réduite.

Conjecture 5.3.
Il n'y a pas de formule de SL équivalente à la formule d'USL^{lp} $Form(\mathsf{ContP}(a,p))$.

On présente ici une synthèse du raisonnement qui étaie cette conjecture. Une version détaillée en est par ailleurs donnée dans l'Annexe A.

1. On définit d'abord le fragment de SL$(\{a\}, At, X)$ dans lequel :
   - Les opérateurs dérivés $\vee$ et R sont introduits
   - la négation est réduite aux atomes
   - le quantificateur universel de stratégie $[\![x]\!]$ n'est pas utilisé.

   On désigne ce fragment par SL$^1_1$.
2. On prouve que s'il existe une formule de SL $\mathsf{sl}(Form(\mathsf{ContP}(a,p)))$ équivalente à $Form(\mathsf{ContP}(a,p))$, alors elle est dans SL$^1_1$. Cette preuve suit elle-même les étapes suivantes :
   (a) On définit une sous-classe ($\mathscr{C}$) des CGSs pour SL$(\{a\}, At, X)$. On montre que cette sous-classe s'interprète aussi comme classe de modèles pour la logique classique de premier ordre (FOL) et qu'elle y est axiomatisable.
   (b) On définit également un ensemble de formules $\{\mathsf{sl}(\mathsf{Cont}(a,p,n))\}_{n\in\mathbb{N}}$ de SL$(\{a\}, At, X)$. Dans $\mathscr{C}$, la conjonction infinie des $\mathsf{sl}(\mathsf{Cont}(a,p,n))$ est équivalente à $Form(\mathsf{ContP}(a,p))$. Par ailleurs on définit, pour tout $n \in \mathbb{N}$, un modèle $\mathscr{C}_n$ tel que $\mathsf{sl}(\mathsf{Cont}(a,p,n))$ est vraie dans $\mathscr{C}_n$ et $\mathsf{sl}(\mathsf{Cont}(a,p,n+1))$ y est fausse.
   (c) On montre que, dans $\mathscr{C}$, SL$^1_1 \preceq \Sigma^1_1$ où Σ^1_1 est le fragment de la logique de second ordre qui n'utilise pas le quantificateur universel d'ordre 2.

(d) Ce dernier plongement permet de formuler un argument de compacité : on montre ainsi que si $\mathsf{sl}(\mathit{Form}(\mathsf{ContP}(a,p))) \in \mathrm{SL}_1^1$, alors $\{\mathsf{sl}(\mathsf{Cont}(a,p,n))\}_{n\in\mathbb{N}} \cup \{\neg\,\mathsf{sl}(\mathit{Form}(\mathsf{ContP}(a,p)))\}$ est satisfiable dans $\mathscr{C}$. Ce qui contredit le contenu de l'item (b).

3. On conjecture que si SL $(\mathit{Form}(\mathsf{ContP}(a,p)))$ existe, alors elle admet une écriture dans SL_1^1. Pour cela :
   (a) On définit la notion de propriété *expansive*, qui caractérise une propriété stable par *simulation*.
   (b) On conjecture que toute propriété expansive caractérisable par une formule de $\mathrm{SL}(\{\Sigma\}, At, X)$ est caractérisable par une formule de SL_1^1.

5.2.1.2 Il n'y a pas de plongement de SL dans USL^{lp}

On expose ici les arguments pour l'absence de plongement de SL dans USL^{lp}.

Conjecture 5.4. *Il n'y a pas de plongement de SL dans USL^{lp}.*

Ces arguments rejoignent l'analyse faite de la dépendance dans la Section 5.1.2 : on y a fait l'observation que l'expression de relations de dépendances rendait apparemment nécessaire l'utilisation de quantificateurs sur les stratégies (déterministes). La Conjecture 5.4 repose donc à la fois sur cette hypothèse, et sur celle selon laquelle les quantificateurs sur les stratégies (déterministes) ne sont pas définissables dans USL^{lp}. Afin d'étayer cette seconde hypothèse, on approfondit ici les arguments donnés dans la Section 5.1.2.

On introduit d'abord la définition suivante :

Définition 5.7 (Stratégies contenues dans une multi-stratégie)**.** *Soit σ une multi-stratégie. On appelle* ensemble des stratégies contenues dans σ *l'ensemble des stratégies (déterministes) σ' telles que pour $\tau \in \mathit{Track}, \sigma'(\tau) \in \sigma(\tau)$. Soit maintenant un contexte $\chi = \langle \alpha, \gamma \rangle$. On appelle* ensemble des contextes (déterministes) contenus dans χ *l'ensemble des contextes $\chi' = \langle \alpha', \gamma \rangle$ t.q. $dom(\alpha) = dom(\alpha')$ et pour tout $x_i \in dom(\alpha), \alpha'(x_i)$ est une stratégie contenue dans $\alpha(x_i)$.*

Dans USL^{lp}, la sémantique du lieur $(a \rhd x)$ quantifie universellement sur l'ensemble des exécutions rendues possibles par le contexte courant enrichit par la transformation $[a \oplus^{lp} x]$. En préfixant cet opérateur par la négation, il est donc possible d'utiliser une quantification existentielle sur l'ensemble des exécutions rendues possibles par un contexte. On remplace alors toute occurrence d'un lieur $(a \rhd x)$ par $\neg(a \rhd x)\neg$. Ceci revient à une quantification existentielle sur l'ensemble des *déterminisations possibles* du contexte courant enrichi de $[a \oplus^{lp} x]$. La formule de SL

$$[\![x]\!]\langle\langle y\rangle\rangle(a,x)(b,y)\mathsf{X}\,p$$

est ainsi équivalente à la formule d'USL^{lp} suivante (la preuve en est laissée au lecteur) :

$$[\![x]\!]\langle\langle y\rangle\rangle\neg(a \rhd x)\neg\neg(b \rhd y)\neg\mathsf{X}\,p \tag{5.10}$$

selon laquelle, quelle que soit la multi-stratégie jouée par a, il existe une multi-stratégie par laquelle b empêche que $\neg\mathsf{X}\,p$ soit assurée.

Une objection à la Conjecture 5.4 semble donc formulable, selon laquelle une $\neg\neg$*-traduction*, par laquelle le lieur (a,x) de SL serait transformé en $\neg(a \rhd x)\neg$, suffirait pour définir un plongement de SL dans USL^{lp}. Cette solution est cependant partielle. Simuler ainsi le déterminisme par

la quantification existentielle sur l'ensemble des actions possibles n'est possible que globalement, sur un engagement dans un ensemble. En particulier, on ne peut pas rendre compte par ce moyen de la réidentification des stratégies permises par SL. Par exemple, la formule de SL

$$[\![x]\!]\langle\langle y\rangle\rangle(a,x)(b,y)(c,y)\mathsf{X}\,p \tag{5.11}$$

n'est pas équivalente à son image par la $\neg\neg$-*traduction* :

$$[\![x]\!]\langle\langle y\rangle\rangle\neg(a \rhd x)\neg\neg(b \rhd y)\neg\neg(c \rhd y)\neg\mathsf{X}\,p \tag{5.12}$$

On le vérifie en considérant le modèle $\mathscr{G}_{\{a,b,c\}}$ de la Figure 5.6 : il y a trois états, s_0 (l'état initial), s_1 et s_2. L'état s_1 est atteint depuis s_0 ssi b et c jouent des actions différentes et l'état s_2 est rejoint depuis s_0 ssi ils jouent la même action. Quelle que soit la décision jouée en s_1, l'exécution reste en s_1, et similairement pour l'état s_2. La proposition p est vraie exactement en s_1.

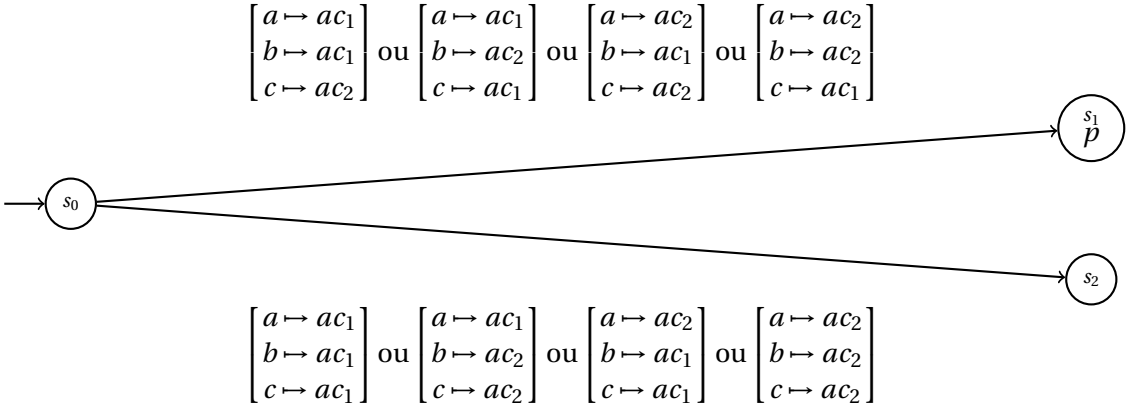

FIGURE 5.6 – Le modèle $\mathscr{G}_{\{a,b,c\}}$: illustration pour la conjecture SL $\not\preceq$ USL^{lp}

Le modèle $\mathscr{G}_{\{a,b,c\}}$ est tel que :

$$\mathscr{G}_{\{a,b,c\}} \not\models_{\mathrm{SL}} [\![x]\!]\langle\langle y\rangle\rangle(a,x)(b,y)(c,y)\mathsf{X}\,p$$

car, quelle que soit la stratégie σ_x, si b et c jouent la même action depuis s_0 alors l'exécution se poursuit dans l'état s_2 qui ne satisfait pas p. Mais on a pourtant

$$\mathscr{G}_{\{a,b,c\}} \models_{\mathrm{USL}^{lp}} [\![x]\!]\langle\langle y\rangle\rangle\neg(a,x)\neg\neg(b,y)\neg\neg(c,y)\neg\mathsf{X}\,p$$

car, quelle que soit la multi-stratégie σ_x, toute multi-stratégie σ_y telle que $\sigma_y(s_0) = \{ac_1, ac_2\}$ est telle que :

$$\mathscr{G}_{\{a,b,c\}}, ((x \mapsto \sigma_x, y \mapsto \sigma_y), \gamma_\varnothing), s_0 \models_{\mathrm{USL}^{lp}} \neg(a,x)\neg\neg(b,y)\neg\neg(c,z)\neg\mathsf{X}$$

5.2.2 Expressivité d'USL

On montre ici qu'USL est strictement plus expressive que SL (Section 5.2.2.1) et que USL^{lp} (Section 5.2.2.2).

5.2.2.1 USL et SL

On commence par comparer USL avec SL. On fait d'abord quelques considérations sur les formules de SL. On définit ensuite un plongement de SL dans USL. On montre enfin qu'USL ne se plonge pas dans SL.

Considérations techniques sur les formules de SL. On fait d'abord une distinction entre les formules d'état et les formules de chemin pour SL. Cette distinction sera utilisée dans notre plongement. Il s'agit seulement de définir deux classes de formules parmi les formules de SL. Ceci n'affecte pas la syntaxe de SL.

Définition 5.8 (Formules d'état et formules de chemin dans SL)**.** *Soient un ensemble d'agents Ag, un ensemble de propositions atomiques At et un ensemble de variables X. L'ensemble des formules de SL peut être divisé en deux sous-ensembles selon la grammaire suivante :*

- *formules d'état :* $\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle\langle x\rangle\rangle\psi \mid [\![x]\!]\psi \mid (a,x)\psi$
- *formules de chemin :* $\psi := \varphi \mid \neg\psi \mid \psi \wedge \psi \mid \psi \vee \psi \mid \mathsf{X}\psi \mid \psi \,\mathsf{U}\, \psi \mid \psi \,\mathsf{R}\, \psi$

où $p \in At$*, a est un agent et* $x \in X$*.*

Par ailleurs, afin d'assurer que notre plongement donne des formules d'USL bien formées, on adapte notre critère de bonne construction des formules d'USL (voir Définition 4.5) aux formules de SL.

Définition 5.9 (Formule de SL USL-compatible)**.** *Une formule* φ *de SL est* USL-compatible *ssi pour toute sous-formule* $\langle\langle x\rangle\rangle\varphi'$ *ou* $[\![x]\!]\varphi'$ *de* φ *et pour toute sous-formule* $\langle\langle y\rangle\rangle\varphi''$ *ou* $[\![y]\!]\varphi''$ *de* φ'*, x et y sont des variables distinctes.*

Dans ce qui suit, on ne considère que des formules de SL qui soient USL-compatibles. Ceci s'applique sans perte de généralité, comme un renommage adéquat des variables transforme toute formule de SL en une formule de SL USL-compatible qui lui est équivalente.

Plongement de SL dans USL. Ici, on établit qu'USL est au moins aussi expressive que SL. On exhibe pour cela un plongement de SL dans USL : ce plongement s'applique aux formules de SL ainsi qu'aux contextes utilisés pour SL.

Pour la définition du plongement $\mathrm{us}|$ des formules de SL dans des formules d'USL, on fait les observations suivantes :

- Le plongement est défini sur les formules de SL qui sont USL-compatibles (cf. Def. 5.9) ;
- Pour toute formule φ de SL, pour toute sous formule $\varphi' = (a,x)\varphi''$ de φ, on traduit l'opérateur (a,x) par :
  - $(a \rhd x)$ s'il n'y a pas d'autre liaison (a,y) au dessus de φ' dans φ
  - $(a \not\rhd y)(a \rhd x)$ sinon.

  La traduction des sous formules est donc paramétrée par, pour chaque agent a, l'unique variable de stratégie (si elle existe) à laquelle a est lié dans un contexte utilisé pour l'évaluation de φ. On traite cet aspect en introduisant dans la traduction en paramètre une fonction partielle f des agents dans les variables ;

– Pour toute telle fonction $f : Ag \nrightarrow X$, pour tout agent a et pour toute variable x, on note $f[a \mapsto x]$ la fonction partielle de domaine $dom(f) \cup \{a\}$ t.q. pour tout $a' \in dom(f[a \mapsto x])$,

$$f[a \mapsto x](a') = \begin{cases} f(a) & \text{si } a' \neq a \\ x & \text{sinon} \end{cases}$$

– Les quantificateurs de SL sont simulés par les quantificateurs sur les stratégies $\langle\langle x \rangle\rangle_d$ et $[\![x]\!]_d$ définis dans la Section 5.1.2.2 ;

– Dans SL, les quantificateurs peuvent s'appliquer à toutes les formules de chemin alors que, dans USL, pour toute formule $\langle\langle x \rangle\rangle \varphi$ ou $[\![x]\!]\varphi$, φ est une formule d'état. Pour que notre plongement satisfasse cette contrainte, on donne un traitement différent des formules de SL $\langle\langle x \rangle\rangle \varphi$ ou $[\![x]\!]\varphi$, selon que φ est une formule d'état ou non (cf. Définition 5.8) : si φ n'est pas une formule d'état, on insère $[\![x_\varnothing]\!](\varnothing \rhd x_\varnothing)$ dans le plongement (ici, $\varnothing$ est la coalition vide et $x_\varnothing$ est une variable fraîche dans la formule transformée). Cette insertion permet de transformer toute formule qui n'est pas une formule d'état en une formule d'état, en quantifiant universellement sur les exécutions qui sont possibles dans le contexte courant. En effet, on peut vérifier que la sémantique d'USL est telle que pour tous $\mathcal{G}, \chi, s$ et pour toute formule φ qui n'est pas une formule d'état, $\mathcal{G}, \chi, s \models_{\text{USL}} [\![x_\varnothing]\!](\varnothing \rhd x_\varnothing)\varphi$ ssi pour tout $\lambda \in out(\chi, s), \mathcal{G}, \chi, \lambda \models_{\text{USL}} \varphi$.

Définition 5.10 (Plongement de SL dans USL)**.** *Soit φ une formule de SL et $f : Ag \nrightarrow X$. On définit* $\mathsf{usl}_f(\varphi)$ *par récurrence structurelle sur φ :*

– $\mathsf{usl}_f(p) = p$

– $\mathsf{usl}_f(\neg\psi) = \neg\, \mathsf{usl}_f(\psi)$

– $\mathsf{usl}_f(\psi_1 \wedge \psi_2) = \mathsf{usl}_f(\psi_1) \wedge \mathsf{usl}_f(\psi_2)$

– $\mathsf{usl}_f(\psi_1 \vee \psi_2) = \mathsf{usl}_f(\psi_1) \vee \mathsf{usl}_f(\psi_2))$

– $\mathsf{usl}_f(\langle\langle x \rangle\rangle \psi) = \begin{cases} \langle\langle x \rangle\rangle_d([\![x_\varnothing]\!](\varnothing \rhd x_\varnothing)\, \mathsf{usl}_f(\psi)) \textit{ si } \psi \textit{ n'est pas une formule d'état} \\ \qquad \textit{(où } x_\varnothing \textit{ est fraîche dans } \varphi) \\ \langle\langle x \rangle\rangle_d\, \mathsf{usl}_f(\psi) \textit{ si } \psi \textit{ est une formule d'état} \end{cases}$

– $\mathsf{usl}_f([\![x]\!]\psi) = \begin{cases} [\![x]\!]_d([\![x_\varnothing]\!](\varnothing \rhd x_\varnothing)\, \mathsf{usl}_f(\psi)) \textit{ si } \psi \textit{ n'est pas une formule d'état} \\ \qquad \textit{(où } x_\varnothing \textit{ est fraîche dans } \varphi) \\ [\![x]\!]_d\, \mathsf{usl}_f(\psi) \textit{ si } \psi \textit{ est une formule d'état} \end{cases}$

– $\mathsf{usl}_f((a,x)\psi) = \begin{cases} (a \not\rhd f(a))(a \rhd x)\, \mathsf{usl}_{f[a \mapsto x]}(\psi) \textit{ si } a \in dom(f) \\ (a \rhd x)\, \mathsf{usl}_{f[a \mapsto x]}(\psi) \textit{ sinon} \end{cases}$

– $\mathsf{usl}_f(\mathsf{X}\psi) = \mathsf{X}\, \mathsf{usl}_f(\psi)$

– $\mathsf{usl}_f(\psi_1 \,\mathsf{U}\, \psi_2) = \mathsf{usl}_f(\psi_1)\, \mathsf{U}\, \mathsf{usl}_f(\psi_2)$

– $\mathsf{usl}_f(\psi_1 \,\mathsf{R}\, \psi_2) = \mathsf{usl}_f(\psi_1)\, \mathsf{R}\, \mathsf{usl}_f(\psi_2)$

On note alors $\mathsf{usl}(\varphi)$ *pour* $\mathsf{usl}_{f_\varnothing}(\varphi)$*, où $f_\varnothing$ est la fonction $f_\varnothing : Ag \nrightarrow X$ de domaine vide.*

On considère également une traduction des contextes pour SL en des contextes pour USL. On fait d'abord les remarques suivantes :

Remarque 5.2.

– *Pour toute CGS $\mathcal{G}$, pour tout énoncé φ de SL qui est USL-compatible, pour tout contexte pour SL $\overline{\chi}$ utilisé dans l'évaluation selon $\models_{SL}$ de φ dans $\mathcal{G}$, $\overline{\chi}$ est tel que pour tout agent a dans $dom(\overline{\chi})$, il y a une variable de stratégie x dans $dom(\overline{\chi})$ telle que $\overline{\chi}(a) = \overline{\chi}(x)$.*

– *Pour toute CGS $\mathcal{G}$, pour tout énoncé φ de SL qui est USL-compatible, pour tout contexte pour SL $\overline{\chi}$ utilisé dans l'évaluation selon $\models_{SL}$ de φ dans $\mathcal{G}$, si $\overline{\chi}$ est utilisé dans l'état s à une étape pour une sous-formule φ' de φ qui n'est pas une formule d'état, alors $\overline{\chi}$ est un contexte complet et s-total. (cf. Section 2.3.2, Définition 2.24).*

Dans la définition qui suit, chaque contexte de stratégies $\overline{\chi}$ pour SL est transformé en un *ensemble de contextes* pour USL. En effet, un contexte de stratégies $\overline{\chi}$ pour SL associe directement des agents et des variables à des stratégies, alors qu'un contexte χ pour USL associe des agents à des variables qui sont elles-mêmes associées à des multi-stratégies.

Ainsi, on transforme $\overline{\chi}$ en des contextes pour USL t.q. chaque variable x dans le domaine de $\overline{\chi}$ est associée à $\overline{\chi}(x)$, et t.q. chaque agent a dans le domaine de $\overline{\chi}$ est associé à une variable x t.q. $\overline{\chi}(x) = \overline{\chi}(a)$. Ceci assure que a est engagé envers la stratégie $\overline{\chi}(a)$. Maintenant, supposons qu'il y a différentes variables dans le domaine de $\overline{\chi}$ ayant la même image que a. Alors il y a différentes manières de transformer $\overline{\chi}$ en un contexte pour USL. C'est pourquoi la transformation peut définir plusieurs contextes pour USL.

Définition 5.11 (Plongement de l'ensemble des contextes pour SL dans l'ensemble des contextes pour USL)**.** *Soit $\overline{\chi}$ un contexte de stratégie pour SL, alors* $\mathsf{usl}(\overline{\chi})$ *est l'ensemble des contextes pour USL $\chi = \langle \alpha, \gamma \rangle$ tels que :*

– *$dom(\alpha) = dom(\overline{\chi}) \cap X$ et pour tout $x \in dom(\alpha)$, $\alpha(x) = \overline{\chi}(x)$*

– *$dom(\gamma) = dom(\overline{\chi}) \cap Ag$ et pour tout $a \in dom(\gamma)$, $\gamma(a)$ est un singleton $\{x\}$, $x \in dom(\alpha)$ et $\alpha(x) = \overline{\chi}(a)$.*

Remarque 5.3. *On remarque que pour toute CGS $\mathcal{G}$, pour tout contexte de stratégie $\overline{\chi}$ dans $\mathcal{G}$, pour tout $\chi, \chi' \in \mathsf{usl}(\overline{\chi})$, et pour tout état s dans $\mathcal{G}$, $out(\chi, s) = out(\chi', s)$. De plus, si $\overline{\chi}$ est complet et s-total, alors $out(\chi, s)$ est le singleton formé par l'unique exécution λ déterminée par $\overline{\chi}$ et s : $out(\chi, s) = jeu^{(\overline{\chi}, s)}$.*

Alors, pour tout contexte de stratégie $\overline{\chi}$ et pour toute formule φ d'USL, si φ ne contient pas de quantificateur sur des variables dans $dom(\overline{\chi})$ (*i.e.* pour toute formule φ t.q. pour toute sous-formule $\langle\langle x \rangle\rangle \psi$ ou $[\![x]\!]\psi$ de φ, $x \notin dom(\overline{\chi})$) et pour tout $\chi, \chi' \in \mathsf{usl}(\overline{\chi})$, on a : $\mathcal{G}, \chi, s \models_{\mathrm{USL}} \varphi$ ssi $\mathcal{G}, \chi', s \models_{\mathrm{USL}} \varphi$.

Lemme 5.5. *Pour toute formule φ de SL, pour toute CGS $\mathcal{G}$ et pour tout contexte de stratégie $\overline{\chi}$ tel que $\overline{\chi}$ peut-être utilisé dans l'évaluation de φ dans $\mathcal{G}$, pour tout $\chi \in \mathsf{usl}(\overline{\chi})$, et pour tout état s :*

– *si φ est une formule d'état alors :*

$$\mathcal{G}, \overline{\chi}, s \models_{SL} \varphi \text{ ssi } \mathcal{G}, \chi, s \models_{USL} \mathsf{usl}(\varphi)$$

– *si φ n'est pas une formule d'état alors :*

$$\mathcal{G}, \overline{\chi}, s \models_{SL} \varphi \text{ ssi l'unique exécution } \Lambda \in out(\chi, s) \text{ est t.q. } \mathcal{G}, \chi, \Lambda \models_{USL} \mathsf{usl}(\varphi)$$

Démonstration. Par induction structurelle sur φ. On considère que les éléments nécessaires pour les cas $\varphi = \langle\langle x \rangle\rangle \psi$ et $\varphi = [\![x]\!]\psi$ ont été donnés dans les paragraphes précédents et dans la Section 5.1.2.2 : ces formules sont traduites grâce aux quantificateurs d'USL sur les stratégies

$\langle\langle x\rangle\rangle_d$, et $[\![x_\varnothing]\!](\varnothing \rhd x_\varnothing)$ est inséré si nécessaire, pour transformer ψ en une formule d'état. Le traitement du cas $\varphi = (a,x)\psi$ consiste en une simple réécriture des définitions. On présente ici les cas pour la négation et l'opérateur X :

- Si $\varphi = \neg\psi$:
  - Si ψ est une formule d'état, alors $\mathcal{G},\overline{\chi},s \models_{\mathrm{SL}} \neg\psi$ ssi (par hypothèse d'induction) ce n'est pas vrai que $\mathcal{G},\chi,s \models_{\mathrm{USL}} \mathsf{usl}(\psi)$, ce qui est la définition de $\mathcal{G},\chi,s \models_{\mathrm{USL}} \neg\,\mathsf{usl}(\psi)$ et est vrai ssi $\mathcal{G},\chi,s \models_{\mathrm{USL}} \mathsf{usl}(\neg\psi)$.
  - Si ψ n'est pas une formule d'état, alors $\mathcal{G},\overline{\chi},s \models_{\mathrm{SL}} \neg\psi$ ssi ce n'est pas vrai que $\mathcal{G},\overline{\chi},s \models_{\mathrm{SL}} \psi$. Par hypothèse d'induction, ceci est équivalent à : ce n'est pas vrai que $\mathcal{G},\chi,\Lambda \models_{\mathrm{USL}} \mathsf{usl}(\psi)$, ce qui est la définition de $\mathcal{G},\chi,\Lambda \models_{\mathrm{USL}} \neg\,\mathsf{usl}(\psi)$ et est vrai ssi $\mathcal{G},\chi,\Lambda \models_{\mathrm{USL}} \mathsf{usl}(\neg\psi)$.
- Si $\varphi = \mathsf{X}\psi$: $\mathcal{G},\overline{\chi},s \models_{\mathrm{SL}} \mathsf{X}\psi$ ssi $\mathcal{G},(\overline{\chi},s)^1 \models_{\mathrm{SL}} \psi$, où $(\overline{\chi},s)^1 = (\overline{\chi}^{s\cdot s^1}, s^1)$ est t.q. s^1 est le successeur de s déterminé par $\overline{\chi}$ (voir Section 2.3.2, Définition 2.27) : $s^1 = tr(s,\delta)$, où δ est la décision t.q. pour tout $a \in Ag, \delta(a) = \overline{\chi}(a)(s)$. Par hypothèse d'induction et par la Définition 5.11, ceci est équivalent à $\mathcal{G},\chi^{s\cdot s^1}, s^1 \models_{\mathrm{USL}} \mathsf{usl}(\psi)$. Grâce à la Remarque 5.3, ceci est vrai ssi $\mathcal{G},\chi^{\Lambda_0\cdot\Lambda_1},\Lambda_{\geqslant 1} \models_{\mathrm{USL}} \mathsf{usl}(\psi)$, et donc ssi $\mathcal{G},\chi,\Lambda \models_{\mathrm{USL}} \mathsf{X}\,\mathsf{usl}(\psi)$ (alors, si ψ est une formule d'état, c'est équivalent à $\mathcal{G},\chi,\Lambda_0 \models_{\mathrm{USL}} \mathsf{usl}(\mathsf{X}\psi)$). □

On peut désormais prouver qu'USL est au moins aussi expressive que SL :

Lemme 5.6. *SL ⪯ USL.*

Démonstration. En appliquant le Lemme 5.5 au cas particulier où $\overline{\chi}$ est le contexte de stratégie vide et s est l'état initial de la CGS, on a que pour toute CGS $\mathcal{G}$ et pour tout énoncé $\varphi, \mathcal{G} \models_{\mathrm{SL}} \varphi$ ssi $\mathcal{G} \models_{\mathrm{USL}} \mathsf{usl}(\varphi)$. □

Corollaire 5.7. *Le problème de satisfiabilité pour USL n'est pas décidable.*

Démonstration. L'indécidabilité du problème de satisfiabilité pour SL est établie dans [MMPV11]. Si USL avait une procédure de décision pour ce problème, par le Lemme 5.6 elle apporterait aussi une procédure de décision pour SL. □

SL est strictement moins expressive qu'USL. On démontre la supériorité stricte de l'expressivité d'USL grâce à une nouvelle caractérisation, il s'agit du *pouvoir de distinction* :

Définition 5.12 (Pouvoir de distinction)**.** *Soient $\mathscr{L}$ et $\mathscr{L}'$ deux logiques intrerprétées par des CGSs et soient $\mathcal{G}_1$ et $\mathcal{G}_2$ deux CGSs :*

- *On dit que $\mathscr{L}$ distingue entre $\mathcal{G}_1$ et $\mathcal{G}_2$ ssi il existe une formule φ de $\mathscr{L}$ t.q. $\mathcal{G}_1 \models_{\mathscr{L}} \varphi$ et $\mathcal{G}_2 \not\models_{\mathscr{L}} \varphi$.*
- *On dit que $\mathscr{L}$ a un pouvoir de distinction au moins égal à celui de $\mathscr{L}'$, et on note $\mathscr{L}' \preceq_{dis} \mathscr{L}$, ssi pour toute paire de CGSs $\{\mathcal{G}_1,\mathcal{G}_2\}$, si $\mathscr{L}'$ distingue entre $\mathcal{G}_1$ et $\mathcal{G}_2$, alors $\mathscr{L}$ distingue entre $\mathcal{G}_1$ et $\mathcal{G}_2$.*
- *On dit que $\mathscr{L}$ a un pouvoir de distinction strictement supérieur à celui de $\mathscr{L}'$, et on note $\mathscr{L}' \prec_{dis} \mathscr{L}$, ssi $\mathscr{L}' \preceq_{dis} \mathscr{L}$ et ce n'est pas le cas que $\mathscr{L} \preceq_{dis} \mathscr{L}'$.*

On a le lemme suivant, qui permet de comparer les relations $\preceq$ et $\preceq_{dis}$:

Lemme 5.8. *Soient $\mathscr{L}$ et $\mathscr{L}'$ deux logiques interprétées par des CGSs. Alors si $\mathscr{L}' \preceq \mathscr{L}$, on a $\mathscr{L}' \preceq_{dis} \mathscr{L}$.*

Démonstration. Si $\mathscr{L}'$ Distingue deux CGSs $\mathscr{G}_1$ et $\mathscr{G}_2$, alors il y a une formule φ de $\mathscr{L}'$ t.q. $\mathscr{G}_1 \models_{\mathscr{L}'} \varphi$ et $\mathscr{G}_2 \not\models_{\mathscr{L}'} \varphi$. De plus, si $\mathscr{L}' \preceq \mathscr{L}$ alors il y a une formule de $\mathscr{L}$ équivalente à φ. On la désigne par $\mathscr{L}(\varphi)$. On a donc $\mathscr{G}_1 \models_{\mathscr{L}} \mathscr{L}(\varphi)$ et $\mathscr{G}_2 \not\models_{\mathscr{L}'} \mathscr{L}(\varphi)$. Ceci implique que $\mathscr{L}' \preceq_{dis} \mathscr{L}$. □

La possibilité de définir un prédicat d'égalité entre les ensembles d'actions pour USL induit la possibilité de *compter* les actions disponibles et en particulier de distinguer des modèles dont les actions ont exactement les mêmes conséquences. Cela permet de distinguer les pouvoirs de distinction d'USL et SL à l'aide d'exemples simples.

Lemme 5.9. *USL $\not\preceq_{dis}$ SL*

Démonstration. Les modèles $\mathscr{G}_1$ et $\mathscr{G}_2$ de la Figure 5.7 représentent un agent a et une proposition atomique p. Ces deux modèles ont un unique état s. Dans la figure, toute action depuis tout état est représentée par une flèche. Chaque flèche depuis chaque état conduit au successeur de cet état si a joue l'action correspondante. Ces deux modèles ne diffèrent donc que par le nombre d'actions équivalentes pour a. Dans le modèle $\mathscr{G}_1$, a dispose d'une unique action qui fait boucler l'exécution en s. Dans le modèle $\mathscr{G}_2$, a dispose de deux actions équivalentes. En utilisant l'opérateur $\langle\langle !x \rangle\rangle$ de la Section 5.1.2.1, on a alors :

- $\mathscr{G}_1, s \models_{\text{USL}} \langle\langle !x \rangle\rangle (a \rhd x) \mathsf{X} p$
- $\mathscr{G}_2, s \not\models_{\text{USL}} \langle\langle !x \rangle\rangle (a \rhd x) \mathsf{X} p$
- Alors que, pour toute formule φ de SL, $\mathscr{G}_1, s \models_{\text{SL}} \varphi$ ssi $\mathscr{G}_2, s \models_{\text{SL}} \varphi$. □

$\mathscr{G}_1$ $\mathscr{G}_2$

FIGURE 5.7 – Illustration de la comparaison entre USL et SL : $\mathscr{G}_1$ and $\mathscr{G}_2$

On conclut cette section avec le résultat d'expressivité attendu :

Théorème 5.10. *SL $\prec$ USL*

Démonstration. Par les Lemmes 5.8 et 5.9, on a : USL $\not\preceq$ SL. On obtient le Théorème 5.10 en combinant avec le Lemme 5.6. □

5.2.2.2 USL^{lp} et USL

On montre maintenant qu'USL est strictement plus expressive qu'USL^{lp}. On définit d'abord un plongement d'USL^{lp} dans USL, puis on montre qu'il n'y a pas de plongement réciproque d'USL dans USL^{lp}.

Plongement d'USL^{lp} dans USL. On définit ici un plongement d'USL^{lp} dans USL. Il repose sur la possibilité dans USL de simuler la règle pour la composition des engagements de la sémantique d'USL^{lp} (la priorité à gauche). Soit $\overline{X}$ une séquence finie non vide de variables. On définit ci-dessous la formule $x = lp(\overline{X})$. Cette formule est vraie ssi x se comporte comme la composition des multi-stratégies de $\overline{X}$ selon la règle de priorité à gauche :

Définition 5.13 (Simulation de la règle de priorité à gauche)**.** *Soient $\overline{X} \in X^*$, $y \in X$ et $a \in Ag$, alors la formule $x = lp(\overline{X})$ est définie par récurrence structurelle sur $\overline{X}$, comme suit :*

- *Si $\overline{X} = y$, alors :*
$$x = lp(\overline{X}) \triangleq x = y$$
- *Si $\overline{X} = \overline{X'} \cdot y$. Alors soit x' une multi-stratégie telle que $x' = lp(\overline{X'})$. Conformément à la Définition 4.17, une multi-stratégie x telle que $x = lp(\overline{X' \cdot y})$ est telle que pour tout $\tau \in Track$:*
  - *si $x'(\tau) \cap y(\tau) \neq \varnothing$ alors $x(\tau) = x'(\tau) \cap y(\tau)$*
  - *si $x'(\tau) \cap y(\tau) = \varnothing$ alors $x(\tau) = x'(\tau)$*

  Formellement, on a donc :

$$\begin{aligned} x = lp(\overline{X} \cdot y) \triangleq [\![x']\!](x' = lp(\overline{X}) \rightarrow \Box((((a \rhd x')(a \rhd y)) \mathsf{X} \top) \rightarrow \\ [\![z]\!](((a \rhd x')(a \rhd y)(a \rhd z)) \mathsf{X} \top) \leftrightarrow ((a \rhd x)(a \rhd z)) \mathsf{X} \top))) \\ \wedge (((a \rhd x)(a \rhd y)) \mathsf{X} \neg \top) \rightarrow x =_{\mathsf{X}} x'))) \end{aligned}$$

où a est un agent quelconque du langage.

Cette formule utilise un agent a quelconque : il ne sert que pour exprimer les tests d'intersection vide entre les ensembles d'actions indiqués par différentes multi-stratégies.

Grâce à ces formules, on peut définir un plongement d'USL^{lp} dans USL. Pour traduire chaque lieur $(A \rhd x)$, on lie chaque agent a dans A, non pas directement à x, mais à la multi-stratégie qui simule l'engagement courant pour a composé avec x par la priorité à gauche. Pour cette opération, il faut auparavant que a révoque sa multi-stratégie courante. La traduction du délieur $(A \not\rhd x)$ procède de manière similaire.

La transformation d'une formule φ de USL^{lp} en une formule $\mathsf{usl}(\varphi)$ de USL utilise deux paramètres :

- Une fonction partielle $f : Ag \nrightarrow X$, qui indique les liaisons des agents dans la formule image. À toute étape dans l'évaluation de $\mathsf{usl}(\varphi)$, chaque agent est en effet lié à au plus une multi-stratégie : c'est la multi-stratégie simulant la composition par la priorité à gauche des différentes multi-stratégies auxquelles le même agent est lié dans la procédure d'évaluation pour φ, à la même étape.
- Un engagement γ, qui indique les liaisons des agents dans l'évaluation de φ. Il est utilisé pour paramétrer l'usage de $lp(\overline{X})$ fait dans la traduction des opérateurs $(A \rhd x)$ et $(A \not\rhd x)$.

Définition 5.14 (Plongement d'USL^{lp} dans USL)**.** *Soient φ une formule de SL et $f : Ag \nrightarrow X$ et γ un engagement. On définit $\mathsf{usl}_{f,\gamma}(\varphi)$ par induction structurelle sur les sous-formules de φ :*

- $\mathsf{usl}_{f,\gamma}(p) = p$, *pour toute proposition atomique p*
- $\mathsf{usl}_{f,\gamma}(\neg\psi) = \neg\, \mathsf{usl}_{f,\gamma}(\psi)$
- $\mathsf{usl}_{f,\gamma}(\psi_1 \wedge \psi_2) = \mathsf{usl}_{f,\gamma}(\psi_1) \wedge \mathsf{usl}_{f,\gamma}(\psi_2)$

- $\mathsf{usl}_{f,\gamma}(\mathsf{X}\psi) = \mathsf{X}\,\mathsf{usl}_{f,\gamma}(\psi)$
- $\mathsf{usl}_{f,\gamma}(\psi_1 \mathsf{U} \psi_2) = \mathsf{usl}_{f,\gamma}(\psi_1) \mathsf{U}\, \mathsf{usl}_{f,\gamma}(\psi_2)$
- $\mathsf{usl}_{f,\gamma}(\langle\langle x\rangle\rangle\psi) = \langle\langle x\rangle\rangle\, \mathsf{usl}_{f,\gamma}(\psi)$
- $\mathsf{usl}_{f,\gamma}((A \rhd x)\psi) = [\![x_{1\psi}]\!] \ldots [\![x_{n,\psi}]\!](\bigwedge_{i\in I_A}(x_{i,\psi} = lp(\gamma[A \oplus x](a))) \rightarrow [a_1 \rhd x_{1,\psi}]_f \ldots [a_n \rhd x_{n,\psi}]_f\, \mathsf{usl}_{f,\gamma[A\oplus x]}(\psi)))$
- $\mathsf{usl}_{f,\gamma}((A \not\rhd x)\psi) = [\![x_{1\psi}]\!] \ldots [\![x_{n,\psi}]\!](\bigwedge_{i\in I_A}(x_{i,\psi} = lp(\gamma[A \ominus x](a))) \rightarrow ([a_1 \rhd x_{1,\psi}]_f \ldots [a_n \rhd x_{n,\psi}]_f\, \mathsf{usl}_{f,\gamma[A\ominus x]}(\psi)))$

où, pour tous $a \in Ag, x \in X, f : Ag \nrightarrow X$ et $\psi \in USL$, $[a \rhd x]_f\psi$ est une abbréviation pour :

- $(a \not\rhd f(a))(a \rhd x)\psi$ *si* $a \in dom(f)$
- $(a \rhd x)\psi$ *sinon*

On note alors $\mathsf{usl}(\varphi)$ *pour* $\mathsf{usl}_{f,\gamma}(\varphi)$.

On peut dès lors formuler le lemme suivant :

Lemme 5.11. *Pour tout énoncé φ d'USL^{lp}, pour toute CGS $\mathcal{G}$:*

$$\mathcal{G} \models_{USL^{lp}} \varphi \text{ ssi } \mathcal{G} \models_{USL} \mathsf{usl}(\varphi)$$

Démonstration. (Résumé) La preuve s'obtient par récurrence. L'hypothèse d'induction est la suivante : Pour toute CGS $\mathcal{G}$, pour tout état s de $\mathcal{G}$, pour tout contexte $\chi = \langle\alpha, \gamma\rangle$ de $\mathcal{G}$ tel que $dom(\alpha = \{a_1, \ldots, a_n\})$ et pour toute sous-formule ψ de φ : $\mathcal{G}, \chi, s \models_{\text{USL}^{lp}} \varphi$ ssi

$$\mathcal{G}, \chi_\varnothing[x_1 \mapsto eS(\chi, a_1)] \ldots [x_n \mapsto eS(\chi, a_n)][a_1 \mapsto x_1] \ldots [a_n \mapsto x_n] \models_{\text{USL}} \mathsf{usl}_{f,\gamma}(\psi)$$

où f est la fonction de domaine $dom(f) = dom(\alpha)$ telle que pour tout $a_k \in dom(f)$, $f(a) = x_k$. □

Par le Lemme 5.11, on a qu'USL est au moins aussi expressive qu'USL^{lp} :

Lemme 5.12. *USL$^{lp} \preceq$ USL.*

USL^{lp} est strictement moins expressive qu'USL. On prouve la supériorité stricte de l'expressivité d'USL sur USL^{lp} de la même manière que dans la comparaison avec SL :

Lemme 5.13. *USL $\not\preceq$ USL^{lp}*

Démonstration. On reprend les modèles $\mathcal{G}_1$ et $\mathcal{G}_2$ de la Figure 5.7, et on observe que pour toute formule φ d'USL^{lp}, $\mathcal{G}_1, s \models_{\text{USL}^{lp}} \varphi$ ssi $\mathcal{G}_2, s \models_{\text{USL}^{lp}} \varphi$, ce qui amène USL $\not\preceq_{dis}$ USL^{lp} et donc le résultat du Lemme. □

On a alors le résultat d'expressivité suivant :

Théorème 5.14. *SL $\prec$ USL*

Démonstration. Immédiat avec les Lemmes 5.12 et 5.13. □

5.3 *Model-checking*

Dans cette section, on s'intéresse au *model-checking* des différentes sémantiques pour USL. Les bornes de complexité qui sont discutées ici sont données en termes de longueur d'une formule et de taille d'un modèle. On définit d'abord la taille d'un modèle :

Définition 5.15 (Taille d'un modèle)**.** *Soit une CGS* $\mathcal{G} = \langle Ag, G, s_0, At, val, Ac, tr\rangle$*. La* taille *de* $\mathcal{G}$*, notée* $|\mathcal{G}|$*, est donnée par* $|G| + |Ac|$*.*

Étant donnée une logique $\mathscr{L}$, on note $\mathsf{MC}(\mathscr{L})$ son problème de *model-checking* : soient φ une formule de $\mathscr{L}$ et $\mathcal{G}$ un modèle tel que $|\mathcal{G}| \in \mathbb{N}$, il s'agit de déterminer si $\mathcal{G} \models_{\mathscr{L}} \varphi$ ou non. Pour les sémantiques USL, USL^{lp} et USL^{cf}, on rejoint des résultats de complexité similaires à ceux pour ATL^{*}_{sc} et SL : le problème de *model-checking* est décidable en temps NONELÉMENTAIRE sur la longueur des formules et POLYNOMIAL sur la taille des modèles. Cependant, le problème de *model-checking* pour les versions positionnelles de ces sémantiques est considérablement plus simple. Il est en effet PSPACE complet, ce qui rejoint le résultat pour ATL^{*0}_{sc} [DCL11].

On s'appuie sur les résultats d'intertraductibilité de nos sémantiques. Pour les sémantiques à mémoire et positionnelles, on mène les preuves :

- des résultats de difficulté sur les sémantiques les moins expressives, c'est-à-dire USL^{lp} et $\text{USL}^{lp,0}$.
- des résultats de complétude sur les sémantiques les plus expressives c'est-à-dire USL^{cf} et $\text{USL}^{cf,\,0}$. Plus précisément, on définit une quatrième sémantique USL^{∞} (et sa version positionnelle $\text{USL}^{\infty,o}$), qui est équivalente à USL^{cf} (respectivement à $\text{USL}^{cf,\,0}$) et sur laquelle on prouve ces résultats.

On présente donc d'abord USL^{∞} et $\text{USL}^{\infty,o}$, ainsi que les réductions de $\mathsf{MC}(\text{USL}^{cf})$ en $\mathsf{MC}(\text{USL}^{\infty})$ et de $\mathsf{MC}(\text{USL}^{cf,\,0})$ en $\mathsf{MC}(\text{USL}^{\infty,o})$ (Section 5.3.1.1). On présente ensuite les résultats de complexité de *model-checking* pour les sémantiques à mémoire (Section 5.3.2), puis pour les sémantiques positionnelles (Section 5.3.3).

5.3.1 Encodage de l'arrêt des exécutions dans la syntaxe du langage

5.3.1.1 USL^{∞}

Sous USL et USL^{cf}, quand un agent est engagé v.a.v. de multi-stratégies qui indiquent des ensembles d'actions en intersection vide après un préfixe d'exécution, alors l'exécution s'arrête après ce préfixe d'exécution. USL^{∞} est une adaptation de cette sémantique t.q. une exécution ne s'arrête jamais. Dans USL^{∞}, on distingue une proposition atomique spécial que l'on écrit $At_{\perp}$, par lequel on simule l'arrêt d'une exécution : $At_{\perp}$ est vrai seulement dans un état spécial $\overline{\varnothing}$, qui est nécessairement rejoint quand un agent est engagé v.a.v. d'ensembles d'actions contradictoires. Par ailleurs, une exécution qui parvient dans l'état $\overline{\varnothing}$ n'en ressort pas. Conséquemment, USL^{∞} est interprétée sur la sous-classe des CGSs dans lesquelles $At_{\perp}$ et $\overline{\varnothing}$ sont interprétés de cette manière. Elles sont appelées *CGSs pour* USL^{∞}, on écrit $\text{CGS}^{\infty}s$:

Définition 5.16 ($\text{CGS}^{\infty}s$)**.** *Une CGS* $\mathcal{G} = \langle Ag, G, s_0, At, val, Ac, tr\rangle$ *est une* CGS^{∞} *ssi :*

- $\overline{\varnothing} \in G$
- $At_{\perp} \in At$
- $At_{\perp} \in val(\overline{\varnothing})$

– *pour tout* $s \in G \setminus \{\overline{\varnothing}\}, At_\perp \notin val(s)$

– *pour toute* $dc \in Dc, tr(\overline{\varnothing}, dc) = \overline{\varnothing}$

Pour USL$^\infty$, les successeurs d'un préfixe d'exécution τ sous un contexte χ sont écrits $Msucc^\infty_\chi(\tau)$. Ils sont définis de la même manière que pour USL^{cf}, excepté le fait que $\overline{\varnothing}$ est désormais un état à part entière du modèle. L'état $\overline{\varnothing}$ est inclus dans l'ensemble des successeurs en cas d'actions conflictuelles de la part des différents acteurs. Il remplace par ailleurs $\varnothing$ comme image de $Msucc^\infty_\chi$ dans les deux cas suivants :

– soit le contexte χ indique des ensembles d'actions contradictoires pour au moins un agent,

– soit la fonction de transition n'est pas définie pour l'état courant et la décision prise en compte.

Définition 5.17 (Fonction de successeurs)**.** *Soient une CGS*$^\infty$ $\mathcal{G} = \langle Ag, G, s_o, At, val, Ac, tr\rangle$ *et un contexte* $\chi = \langle \alpha, \gamma\rangle$*. Alors la fonction de* successeurs $Msucc^\infty_\chi : Track \to \mathcal{P}(G)$ *déterminée par* χ *est définie, pour tout préfixe d'exécution* τ*, par :*

$$Msucc^\infty_\chi(\tau) = \begin{cases} Msucc^{cf}_\chi(\tau) & \text{si ce n'est pas l'ensemble vide} \\ \{\overline{\varnothing}\} & \text{sinon} \end{cases}$$

L'ensemble des issues d'un contexte χ pour USL$^\infty$ et d'un préfixe d'exécution τ ($out^\infty(\chi, \tau)$) est défini de manière similaire à la définition pour USL^{cf}, excepté le fait que son image n'est faite que d'exécutions infinies :

Définition 5.18 (Issues)**.** *Soient une CGS*$^\infty$ $\mathcal{G} = \langle Ag, G, s_o, At, val, Ac, tr\rangle$*, un préfixe d'exécution* τ *et un contexte* $\chi = \langle \alpha, \gamma\rangle$*. On définit l'ensemble des* issues *de* χ *et* τ *dans* $\mathcal{G}$ *comme l'ensemble* $out^\infty(\chi, \tau)$ *des exécutions infinies* λ *dans* $\mathcal{G}$ *t.q.* $\lambda_0 = last(\tau)$ *et pour tout* $i \in \mathbb{N}$, $\lambda_{i+1} \in Msucc^\infty_{\chi^{\lambda_{\leqslant i}}}(\lambda_{\leqslant i})$.

On ne détaille pas la définition de la *relation de satisfaction* $\models_{\text{USL}^\infty}$. Elle est définie par induction structurelle sur les formules, de la même manière que $\models_{\text{USL}^{cf}}$, excepté le fait que la fonction d'issues utilisée est out^∞ à la place de out^{cf}. Par ailleurs, comme dans USL$^\infty$ toutes les exécutions dans les issues d'un contexte et d'un préfixe d'exécution sont infinies, il n'est pas nécessaire de vérifier la longueur des exécutions dans l'interprétation des opérateurs temporels.

5.3.1.2 Réduction de MC(USL^{cf}) à MC(USL$^\infty$)

La réduction de MC(USL^{cf}) à MC(USL$^\infty$) consiste en une double transformation des CGS^{cf}*s* $\mathcal{G}$ en CGS$^\infty$*s* $\mathcal{G}^\infty$, et des formules φ d'USL^{cf} en formules φ^∞ d'USL$^\infty$, telle que :

– $\mathcal{G}^\infty$ est une adaptation de $\mathcal{G}$ qui répond aux contraintes de la Définition. 5.16.

– on ajoute des occurences de la sous-formule $\neg At_\perp$ dans φ^∞, de manière à ce que l'exécution n'aille pas dans l'état $\overline{\varnothing}$ si la satisfaction de la formule n'est pas déjà assurée par les états déjà visités.

On définit d'abord la transformation des CGS^{cf}*s*

Définition 5.19 (Tranformation d'une CGS^{cf} en CGS$^\infty$)**.** *Soit une CGS*^{cf} $\mathcal{G} = \langle Ag, G, s_0, At, val, Ac, tr\rangle$*. On note* $\mathcal{G}^\infty$ *la CGS*$^\infty$ $\langle Ag, G^\infty, s_0, At^\infty, val^\infty, Ac, tr^\infty\rangle$ *définie par :*

– $G^\infty = G \cup \{\overline{\varnothing}\}$

- $At^\infty = At \cup \{At_\perp\}$
- *Pour tout* $s \in G, val^\infty(s) = val(s)$ *et* $val(\overline{\varnothing}) = At^\infty$
- *Pour tout* $(s, dc) \in dom(trans), tr^\infty(s, dc) = tr(s, dc)$
- *Pour tout* $(s, dc) \notin dom(trans), tr^\infty(s, dc) = \overline{\varnothing}$

Remarque 5.4. *Pour la valuation dans* $\overline{\varnothing}$*, la seule chose nécessaire est que* $At_\perp \in val(\overline{\varnothing})$*, ce qui est imposé par la Définition 5.16. Par ailleurs, pour rendre la transformation déterministe, on doit spécifier pour toute proposition atomique* p *dans At, si* $p \in val(\overline{\varnothing})$ *ou non. Parmi n'importe quelle autre valeur de vérité pour At, le choix fait de rendre vrais tous les atomes dans* $\overline{\varnothing}$ *est complètement arbitraire.*

On en vient à la transformation des formules d'USL^{cf}. Dans le traitement des opérateurs temporels, cette transformation spécifie explicitement que l'exécution n'est pas envoyée dans l'état qui satisfait $At_\perp$ (*i.e.* dans $\overline{\varnothing}$).

Définition 5.20 (Transformation des formules d'USL^{cf} en formules d'USL$^\infty$)**.** *La transformation d'une formule* φ *d'USL*^{cf} *en une formule* φ^∞ *d'USL*$^\infty$ *est définie par induction structurelle sur* φ *:*

- $p^\infty = p$
- $(\neg\varphi)^\infty = \neg\varphi^\infty$
- $(\varphi_1 \wedge \varphi_2)^\infty = \varphi_1^\infty \wedge \varphi_2^\infty$
- $(\langle\langle x\rangle\rangle\varphi)^\infty = \langle\langle x\rangle\rangle\varphi^\infty$
- $((A \rhd x)\varphi)^\infty = (A \rhd x)\varphi^\infty$
- $((A \not\rhd x)\varphi)^\infty = (A \not\rhd x)\varphi^\infty$
- $(\mathsf{X}\varphi)^\infty = \mathsf{X}(\varphi^\infty \wedge \neg At_\perp)$
- $(\varphi_1 \mathsf{U} \varphi_2)^\infty = (\varphi_1^\infty \wedge \neg At_\perp) \mathsf{U} (\varphi_2^\infty \wedge \neg At_\perp)$

On a alors la correspondance suivante :

Proposition 5.15. *Pour toute CGS* $\mathcal{G}$ *et pour toute formule* φ *d'USL*^{cf} *:*

$$\mathcal{G} \models_{USL^{cf}} \varphi \text{ ssi } \mathcal{G}^\infty \models_{USL^\infty} \varphi^\infty$$

Démonstration. (Résumé) Immédiate, par induction structurelle sur φ. □

La version positionnelle d'USL$^\infty$, USL$^{\infty,o}$, est obtenue avec les définitions précédentes, en remplaçant l'usage des multi-stratégies par les multi-stratégies positionnelles. On vérifie que la réduction de MC(USL^{cf}) à MC(USL$^\infty$) est transposable pour réduire MC(USL^{cf,0}) à MC(USL$^{\infty,o}$). Avec les notations introduites ci-dessus, on vérifie la proposition suivante :

Proposition 5.16. *Pour toute CGS* $\mathcal{G}$ *et pour toute formule* φ *de*

$$USL^{cf,0}, \mathcal{G} \models_{USL^{cf,0}} \varphi \text{ ssi } \mathcal{G}^\infty \models_{USL^{\infty,o}} \varphi^\infty$$

Démonstration. (Résumé) À nouveau immédiate, par induction structurelle sur φ. □

5.3.2 Sémantiques à mémoire

On s'intéresse ici aux problèmes de *model-checking* pour les sémantiques USL, USL^{lp}, USL^{cf} et USL$^{\infty}$. On montre d'abord qu'ils n'ont pas de borne élémentaire (Section 5.3.2.1), l'analyse est menée sur USL^{lp}. Puis on montre que ces problèmes sont effectivement décidables en temps non-élémentaire (Section 5.3.2.2). La procédure est adaptée de celle pour SL exposée dans [MMPV11]. Elle est ici décrite pour USL$^{\infty}$.

5.3.2.1 Borne inférieure

Comme pour SL [MMPV11], on se sert d'une réduction du problème de satisfiabilité pour la logique temporelle propositionnelle quantifiée (QPTL [Sis83]) en instances de MC(USL^{lp}). QPTL est une extension de LTL au moyen des quantificateurs $\exists p$ et $\forall p$ sur les séquences de valuations pour une proposition atomique p. Soit p une proposition atomique, alors, intuitivement, $\exists p.\psi$ et $\forall p.\psi$ veulent respectivement dire : il y a une séquence d'évaluations de p telle que φ est vraie et pour toute séquence d'évaluations de p, φ est vraie. Les opérateurs temporels employés dans QPTL sont $\mathsf{X}, \Diamond$ et $\Box$:

Définition 5.21 (Syntaxe de QPTL)**.** *Soit At un ensemble de propositions atomiques. La syntaxe de QPTL est donnée par :*

$$\varphi := p|\neg\varphi|\varphi\wedge\varphi|\varphi\vee\varphi|\mathsf{X}\varphi|\Diamond\varphi|\Box\varphi|\exists p.\varphi|\forall p.\varphi$$

où $p \in At$.

Les formules de QPTL sont interprétées sur des exécutions valuées infinies :

Définition 5.22 (Satisfaction sémantique pour QPTL)**.**
Soit λ une exécution, alors :

- $\lambda \models_{QPTL} p$ *ssi* $p \in val(\lambda_0)$, *pour toute proposition atomique* p
- $\lambda \models_{QPTL} \varphi_1 \wedge \varphi_2$ *ssi* $\lambda \models_{QPTL} \varphi_1$ *et* $\lambda \models_{QPTL} \varphi_2$
- $\lambda \models_{QPTL} \varphi_1 \vee \varphi_2$ *ssi* $\lambda \models_{QPTL} \varphi_1$ *ou* $\lambda \models_{QPTL} \varphi_2$
- $\lambda \models_{QPTL} \neg\varphi_1$ *ssi* $\lambda \not\models_{QPTL} \varphi_1$
- $\lambda \models_{QPTL} \mathsf{X}\varphi_1$ *ssi* $\lambda_{\geqslant 1} \models_{QPTL} \varphi_1$
- $\lambda \models_{QPTL} \Diamond\varphi_1$ *ssi il existe* $i \geqslant 0$ *t.q.* $\lambda_{\geqslant i} \models_{QPTL} \varphi_1$
- $\lambda \models_{QPTL} \Box\varphi_1$ *ssi pour tout* $i \geqslant 0, \lambda_{\geqslant i} \models_{QPTL} \varphi_1$
- $\lambda \models_{QPTL} \exists p.\varphi_1$ *ssi il existe* λ' *t.q. pour tout* $n \in \mathbb{N}$ *et pour tout* $p' \in At \setminus \{p\}, p \in val(\lambda'_n)$ *ssi* $p' \in val(\lambda_n)$, *et* $\lambda' \models_{QPTL} \varphi_1$
- $\lambda \models_{QPTL} \forall p.\varphi_1$ *ssi pour tout* λ' *t.q. pour tout* $n \in \mathbb{N}$ *et pour tout* $p' \in At \setminus \{p\}, p \in val(\lambda'_n)$ *ssi* $p' \in val(\lambda_n), \lambda' \models_{QPTL} \varphi_1$

On dit d'une formule de QPTL qu'elle est en *forme normale* si elle est écrite sous la forme

$$Q_1 p_1 Q_2 p_2 \ldots Q_k p_k \psi$$

où chaque Q_i est soit $\forall$ soit $\exists$ et ψ est une formule de QPTL sans quantificateur. On sait par [Sis83] que toute formule de QPTL a une écriture en forme normale. On sait aussi par [Sis83] que le

problème de satisfiabilité pour la classe des formules de QPTL en forme normale telles que $Q_1 = \exists$ et il y a $k-1$ alternance de quantificateurs est k-EXPSPACE complète.

On réduit le problème de satisfiabilité pour QPTL (QPTLSAT) à une instance de *model-checking* pour USL^{lp} dans le modèle $\mathcal{G}_{\text{QPTL}}$, qui est décrit ci-dessous et représenté dans la Figure 5.8. On en conclura que MC(USL^{lp}) est k-EXPSPACE difficile, pour tout $k \in \mathbb{N}$. Le modèle $\mathcal{G}_{\text{QPTL}}$ est identique à celui utilisé dans [MMPV11]. Il a deux états s_0 (l'état initial) et s_1. La valuation est définie sur une proposition atomique p vraie seulement dans l'état s_1. L'unique agent a dispose de deux actions t et f. Depuis chacun des états, a mène l'exécution en s_1 s'il joue t et en s_0 s'il joue f.

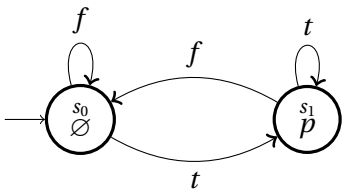

FIGURE 5.8 – Le modèle $\mathcal{G}_{\text{QPTL}}$: illustration de QPTLSAT en tant qu'instance de *model-checking* pour USL^{lp}.

Il nous reste à définir une transformation telle que chaque formule φ de QPTL est transformée en une formule $\mathsf{usl}^{lp}(\varphi)$ d'USL^{lp} telle que φ est satisfiable ssi $\mathcal{G}_{\text{QPTL}}, s_0 \models_{\text{USL}^{lp}} \mathsf{usl}^{lp}(\varphi)$

Cette transformation utilise une formule avec une variable de multi-stratégie libre $Det_q(x)$, qui permet de spécifier, dans $\mathcal{G}_{\text{QPTL}}$, que la multi-stratégie x est instanciée par une stratégie. Le modèle $\mathcal{G}_{\text{QPTL}}$ est en effet tel que, depuis chacun des états, il y a une transition qui assure p et une transition qui assure $\neg p$. Par ailleurs, chaque transition est décidée seulement par l'agent a. Une multi-stratégies peut donc être spécifiée pour être déterministe : c'est une multi-stratégie qui, depuis tout état, si elle est jouée par a, soit assure $\mathsf{X}\, p$ soit assure $\mathsf{X} \neg p$. Le prédicat $Det_q(x)$ est défini comme suit :

$$Det_q(x) \triangleq \Box((a \rhd x)\mathsf{X}\, q \vee (a \rhd x)\mathsf{X} \neg q)$$

On peut maintenant transformer les énoncés de QPTL :

Définition 5.23 (Transformation des énoncés de QPTL en formules d'USL^{lp})**.** *La transformation des formules de QPTL est définie en deux temps : on définit d'abord, pour toute formule φ de QPTL en forme normale, la formule* $\mathsf{usl}^{lp^*}(\varphi)$*, par récurrence structurelle sur φ :*

- $\mathsf{usl}^{lp^*}(q) = (a \not\rhd x_a)(a \rhd x_q)\mathsf{X}\, p$ *pour toute proposition atomique q.*
- $\mathsf{usl}^{lp^*}(\neg\varphi) = \neg\, \mathsf{usl}^{lp^*}(\varphi)$
- $\mathsf{usl}^{lp^*}(\varphi_1 \wedge \varphi_2) = \mathsf{usl}^{lp^*}(\varphi_1) \wedge \mathsf{usl}^{lp^*}(\varphi_2)$
- $\mathsf{usl}^{lp^*}(\varphi_1 \vee \varphi_2) = \mathsf{usl}^{lp^*}(\varphi_1) \vee \mathsf{usl}^{lp^*}(\varphi_2)$
- $\mathsf{usl}^{lp^*}(\mathsf{X}\varphi_1) = \mathsf{X}\, \mathsf{usl}^{lp^*}(\varphi_1)$
- $\mathsf{usl}^{lp^*}(\Diamond\varphi_1) = \Diamond\, \mathsf{usl}^{lp^*}(\varphi_1)$
- $\mathsf{usl}^{lp^*}(\Box\varphi_1) = \Box\, \mathsf{usl}^{lp^*}(\varphi_1)$
- $\mathsf{usl}^{lp^*}(\exists q.\varphi) = \langle\langle x_q \rangle\rangle (Det_q(x_q) \wedge \mathsf{usl}^{lp^*}(\varphi))$
- $\mathsf{usl}^{lp^*}(\forall q.\varphi) = [\![x_q]\!] (Det_q(x_q) \rightarrow \mathsf{usl}^{lp^*}(\varphi))$

On obtient pour φ une formule de la forme

$$Q_1(\dots Q_k(\mathsf{usl}^{lp^*}(\psi))\dots)$$

où pour tout $1 \leqslant j \leqslant k$, pour toute formule $\psi, Q_j(\psi) = \langle\langle x_q^i \rangle\rangle(Det_q(x_q^i) \wedge \psi)$ ou $Q_j(\psi) = [\![x_q^i]\!](Det_q(x_q^i) \rightarrow \psi)$ et $\mathsf{usl}^{lp^*}(\psi)$ est une formule sans quantificateur. On obtient alors l'énoncé $\mathsf{usl}^{lp}(\varphi)$ en préfixant $\mathsf{usl}^{lp^*}(\varphi)$ par la liaison de a à la stratégie existentiellement quantifiée x_a :

$$\mathsf{usl}^{lp}(\varphi) = \langle\langle x_a \rangle\rangle(Det_q(x_a) \wedge (a \rhd x_a)\, \mathsf{usl}^{lp^*}(\varphi))$$

On peut alors poser le théorème suivant :

Théorème 5.17. *La complexité du* model-checking *pour USL^{lp} n'a pas de borne inférieure élémentaire.*

Démonstration. (Résumé) Pour tout énoncé φ de QPTL définie sur l'ensemble d'atomes At, φ est satisfiable ssi $\mathscr{G}_{\text{QPTL}} \models_{\text{USL}^{lp}} \mathsf{usl}^{lp}(\varphi)$. La preuve en est donnée pour SL dans [MMPV11] : chaque stratégie x_q est en fait utilisée comme la séquence linéaire des actions ac_q^k indiquée par x_q après k transitions faites en jouant la stratégie courante. En interprétant $(ac_q^k)_{k\in\mathbb{N}}$ comme une exécution infinie valuée sur la proposition $q : ac_q^k \models q$ ssi $ac_q^k = t$, on retrouve exactement les termes de la sémantique de QPTL. Le problème de la satifiabilité de φ pour QPTL se réduit donc à $\mathsf{MC}(\text{USL}^{lp})$ dans $\mathscr{G}_{\text{QPTL}}$. Alors, pour tout $k \in \mathbb{N}$, $\mathsf{MC}(\text{USL}^{lp})$ est $k-$EXPSPACE difficile. Donc la complexité de $\mathsf{MC}(\text{USL}^{lp})$ n'admet pas de borne inférieure élémentaire. □

Corollaire 5.18. $\mathsf{MC}(USL)$ *et* $\mathsf{MC}(USL^{cf})$ *n'admettent pas de borne inférieure élémentaire.*

Démonstration. On obtient ces résultats grâce aux résultats d'expressivité donnés dans la Section 5.2.

- Supposons qu'il y ait une borne élémentaire $\mathscr{B}(|\varphi|)$ pour $\mathsf{MC}(\text{USL})$. Alors, pour toute formule φ d'USL^{lp}, le *model-checking* de φ est décidable en $\mathscr{B}(|\mathsf{usl}(\varphi)|)$, où $\mathsf{usl}(\varphi)$ est obtenue par la Définition 5.14. On vérifie que $|\mathsf{usl}(\varphi)| \in (2^{O|\varphi|})$: dans chaque item de la Définition 5.14, le nombre de sous-formules ajoutées est dans $O(|\mathsf{Var}(\varphi)| \times |Ag|)$, qui est elle-même dans $O(|\varphi|)$. Ainsi, le *model-checking* pour φ est décidable en $\mathscr{B}(O(|\varphi|^2))$. Donc $\mathsf{MC}(\text{USL}^{lp})$ a une borne inférieure élémentaire $\mathscr{B}(O(|\varphi|^2))$, en contradiction avec le Théorème 5.17.
- Le résultat pour USL^{cf} est immédiat, comme $\mathsf{MC}(\text{USL}^{cf})$ englobe $\mathsf{MC}(\text{USL})$.

□

5.3.2.2 Borne supérieure

On montre ici que $\mathsf{MC}(\text{USL}^{\infty})$ est décidable en temps NONELÉMENTAIRE sur la longueur de la formule et POLYNOMIAL sur la taille du modèle. On en conclut le résultat similaire pour $\mathsf{MC}(\text{USL}^{cf})$, $\mathsf{MC}(\text{USL})$ et $\mathsf{MC}(\text{USL}^{lp})$:

Proposition 5.19. *Le problème* $\mathsf{MC}(USL^{\infty})$*, de décider la satisfaction d'un énoncé φ d'USL$^{\infty}$ dans une CGS$^{\infty}$ $\mathscr{G}$, peut être traité en temps* NONELÉMENTAIRE *sur la longueur de φ et* POLYNOMIAL *sur la taille de $\mathscr{G}$.*

Démonstration. (Résumé) La preuve est adaptée de [MMPV11]. $\varphi \in \text{USL}^\infty$, pour toute CGS^∞ $\mathcal{G}$, on construit un *automate d'arbres alternant avec condition d'acceptation paritaire* $\mathcal{A}_\mathcal{G}[\varphi, \varnothing]$. Cet automate accepte exactement les objets (des arbres étiquetés) qui encodent les couples d'un état s de $\mathcal{G}$ et du contexte vide $\chi_\varnothing$ qui sont tels que $\mathcal{G}, \chi_\varnothing, s \models_{\text{USL}^\infty} \varphi$ (on appelle un tel objet un *déroulement contexte-état* pour s et $\chi_\varnothing$). S'il n'est pas vrai que $\mathcal{G}, \chi_\varnothing, s \models_{\text{USL}^\infty} \varphi$, alors le langage accepté par $\mathcal{A}_\mathcal{G}[\varphi, \varnothing]$ est vide.

La construction de $\mathcal{A}_\mathcal{G}[\varphi, \varnothing]$ est récursive sur la structure de φ. Pour tout énoncé $\varphi \in \text{USL}^\infty$, pour toute occurence ψ^i d'une sous-formule ψ de φ, on écrit $X^\varphi_{\psi^i}$ l'ensemble des variables x telles que ψ^i est dans la portée d'un quantificateur $\langle\langle x \rangle\rangle$ ou $[\![x]\!]$ dans φ. Alors, on construit un *automate d'arbres alternant avec condition d'acceptation paritaire* $\mathcal{A}_\mathcal{G}[\psi, X^\varphi_{\psi^i}]$ t.q. pour tout état s dans G, et pour tout contexte χ t.q. $\chi = \langle \alpha, \gamma \rangle$ et $dom(\alpha) = X^\varphi_{\psi^i}$, $\mathcal{G}, \chi, s \models_{\text{USL}^\infty} \psi$ ssi $\mathcal{T}_{\chi,s} \in \mathcal{L}(\mathcal{A}_\mathcal{G}[\psi, X^\varphi_{\psi^i}])$, où $\mathcal{T}_{\chi,s}$ est le *déroulement contexte-état* pour s et χ.

Cette preuve est détaillées dans l'Annexe B. Il y a principalement trois adaptations à faire sur la preuve donnée pour SL dans [MMPV11], pour la construction de $\mathcal{A}_\mathcal{G}[\psi, X^\varphi_{\psi^i}]$:

- La définition des déroulements contexte-état doit prendre en compte la définition particulière des contextes pour USL^∞.
- Le traitement inductif de l'opérateur $\langle\langle x \rangle\rangle$ consiste à introduire une disjonction dans la fonction de transition de l'automate. Pour SL la portée de cette disjonction est l'ensemble des actions dans la CGS^∞ (Ac). Pour USL^∞ c'est l'ensemble des sous-ensembles non vides de Ac ($\mathcal{P}^{<\omega}_{>0}(Ac)$). Cette adaptation n'accroît ni la *taille* ni l'*indice* de l'automate. Elle n'affecte donc pas la complexité générale de l'algorithme.
- Une étape inductive est ajoutée pour le cas $(A \not\rhd x)\psi^i$. On obtient l'automate $\mathcal{A}_\mathcal{G}[(A \not\rhd x)\psi, X^\varphi_{(A \not\rhd x)\psi^i}]$ à partir de $\mathcal{A}_\mathcal{G}[\psi, X^\varphi_{\psi^i}]$ en changeant sa fonction de transition : les transitions données par la nouvelle fonction de transition sont celles données par l'ancienne pour les mêmes entrées excepté le fait que les choix pour les agents dans A indiqués par x dont retirés. □

Grâce à la réduction de $\text{MC}(\text{USL}^{cf})$ à $\text{MC}(\text{USL}^\infty)$, ceci amène des résultats similaires pour USL^{cf}, USL et USL^{lp}

Corollaire 5.20.

- *Le problème* $\text{MC}(USL^{cf})$*, de vérifier la satisfaction d'un énoncé d'*USL^{cf} φ *dans une* CGS^{cf} $\mathcal{G}$ *peut être traité en temps* NONELÉMENTAIRE *sur la longueur de* φ *et* POLYNOMIAL *sur la taille de* $\mathcal{G}$.
- *Le problème* $\text{MC}(USL)$ *(resp.* $\text{MC}(USL^{lp})$*), de vérifier la satisfaction d'un énoncé d'USL (resp. de* $\text{MC}(USL^{lp})$*)* φ *dans une CGS* $\mathcal{G}$ *peut également être traité en temps* NONELÉMENTAIRE *sur la longueur de* φ *et* POLYNOMIAL *sur la taille de* $\mathcal{G}$.

Démonstration.

- Soit φ un énoncé d'USL^{cf} et soit $\mathcal{G}$ une CGS^{cf}. On note φ^∞ l'énoncé d'USL^∞ obtenue de φ en appliquant la transformation décrite dans la Définition 5.20, et $\mathcal{G}^\infty$ la CGS^∞ obtenue de $\mathcal{G}$ en appliquant la Définition 5.19. On vérifie facilement que $|\varphi^\infty|$ est dans $O(|\varphi|)$ et $|\mathcal{G}^\infty|$ est dans $O(|\mathcal{G}|)$. Ainsi, les deux bornes de complexité sont préservées par les transformations.
- Le résultat pour USL est direct, comme $\text{MC}(\text{USL}^{cf})$ englobe $\text{MC}(\text{USL})$

– De manière similaire, le résultat pour MC(USL) est obtenu à partir de celui pour MC(USL^{cf}) grâce à la transformation des énoncés φ d'USL^{lp} en des énoncés usl(φ) d'USL. Comme $|\text{usl}(\varphi)| \in (2^{O(|\varphi|)})$, la transformation engendre un saut exponentiel mais n'affecte pas la complexité générale de l'algorithme.

□

5.3.3 Les sémantiques positionnelles

On s'intéresse maintenant aux problèmes de *model-checking* pour les sémantiques USL^{0}, USL^{lp,0} et USL$^{cf,\,0}$. Ils sont PSPACE complets. On montre d'abord qu'ils sont PSPACE difficiles (Section 5.3.3.1). L'analyse est menée sur USL^{lp}. Puis, on montre que ces problèmes sont effectivement décidables en espace POLYNOMIAL sur la taille de leurs entrées (Section 5.3.3.2). Les preuves de cette section sont adaptées de celles pour ATL$^{*0}_{sc}$ décrites dans [BBGK07, DCL11].

5.3.3.1 Borne inférieure

La preuve pour la borne inférieure est inspirée de [BBGK07, DCL11]. Comme pour ATL$^{*0}_{sc}$, on se sert d'une réduction du problème de satisfiabilité pour les formules Booléennes Quantifiées (QBF) en instances de MC(USL^{lp,0}). QBF est une extension de la logique booléenne au moyen des quantificateurs $\exists p$ et $\forall p$ sur les propositions atomiques. Soit p une proposition atomique. Alors, intuitivement, $\exists p.\psi$ et $\forall p.\psi$ veulent respectivement dire : il y a une évaluation de p telle que φ est vraie et : pour toute évaluation de p, φ est vraie. Toute formule de QBF admet une *forme normale*

$$\exists X_1 \forall X_2 \exists X_3 \ldots Q_k X_k \; \varphi(X_1, X_2, X_3, \ldots X_k)$$

où pour tout $1 \leqslant i \leqslant k, Q_i$ est $\forall$ si i est pair, et $\exists$ sinon, et où φ est une formule booléenne en Forme Normale Conjonctive (FNC) : $\varphi = \bigwedge_{j=1,\ldots,n} C_j$, telle que chaque C_j est une clause disjonctive sur l'ensemble de variables propositionnelles $\{X_1, X_2, X_3, \ldots, X_k\}$. Par ailleurs, pour tout $k \in \mathbb{N}$, on note QBF$_k$ l'ensemble des formule de QBF dont l'écriture en forme normale nécessite k quantificateurs, et QBFSAT$_k$ le problème de la satisfiabilité pour les formules de QBF$_k$. Ce problème est Σ_k-difficile [Pap94]. On prouve le lemme suivant :

Lemme 5.21. *Pour tout $k \geqslant 0$,* QBFSAT$_k$ *se réduit à* MC($USL^{lp,0}$).

Démonstration. À partir d'une instance $\mathscr{I}$ de QBFSAT$_k$, on construit la CGS $\mathscr{G}_{\text{QBF}} = \langle Ag_I, \text{G}_I, u_1, At_I, val_I, ac_I, tr_I \rangle$ représentée sans valuation dans la Figure 5.9 : elle modélise un jeu à tour : depuis tout état u_ℓ tel que $\ell \leqslant k$, la transition est décidée par le seul agent a_ℓ, entre $\neg x_{\ell+1}$ et $x_{\ell+1}$ et depuis $\neg x_\ell$ ou x_ℓ la transition mène nécessairement à u_ℓ. Depuis l'état u_{k+1}, toute transition boucle sur u_{k+1}. $\mathscr{G}_{\text{QBF}}$ est donc définie par :

- $Ag_I = \{a_j\}_{1 \leqslant k}$
- $\text{G}_I = \bigcup_{0 \leqslant \ell \leqslant k} \{x_\ell, \neg x_\ell, u_\ell\} \cup \{u_{k+1}\}$
- At_I est l'ensemble des clauses C_j de la formule évaluée.
- La valuation sur les états est faite de manière à rendre vrais, dans chaque état x_ℓ ou $\neg x_\ell$, exactement les ensembles de clauses $C_j \in \varphi$ compatibles avec la valeur de vérité correspondante pour X_ℓ :
  - $\forall 1 \leqslant \ell \leqslant k,\ val\,(x_\ell) = \{C_j \mid X_\ell \in C_j\}$
  - $\forall 1 \leqslant \ell \leqslant k,\ val\,(\neg x_\ell) = \{C_j \mid \neg X_\ell \in C_j\}$

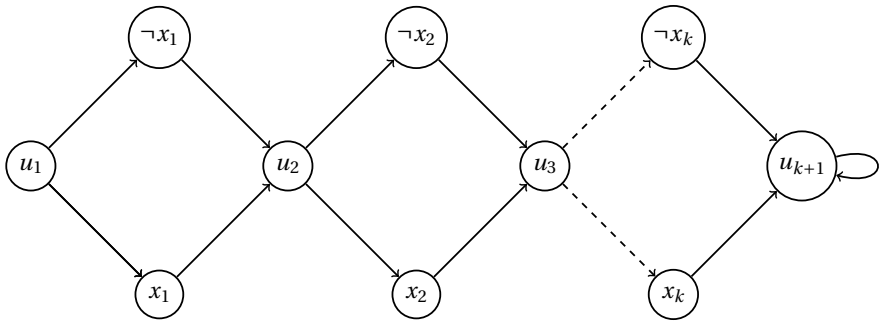

FIGURE 5.9 – Le modèle $\mathcal{G}_{\text{QBF}}$: illustration de QBFSAT en tant qu'instance de *model-checking* pour USL^{lp,0}

- $Ac_I = \{ac_0, ac_1\}$
- Pour toute décision $dc \in Dc$:
  - pour tout $\ell \in [A, \dots k]$:
    - $tr(u_\ell, dc) = \neg x_\ell$ si $dc(a_\ell) = ac_0$
    - $tr(u_\ell, dc) = x_\ell$ si $dc(a_\ell) = ac_1$
    - $tr(x_\ell, dc) = tr(\neg x_\ell, dc) = u_{\ell+1}$
  - $tr(u_{k+1}, dc) = u_{k+1}$

Comme chaque joueur ne joue de manière efficace qu'une fois et comme il joue en décidant si X_ℓ est vraie ou non, pour tout joueur a_ℓ, une stratégie correspond à une valeur de vérité pour la variable X_ℓ. En effet, la formule $\langle\langle x_\ell \rangle\rangle (a_\ell \rhd x) \Diamond \psi$ (resp. $\neg\langle\langle x_\ell \rangle\rangle (a_\ell \rhd x) \neg\Diamond \psi$) signifie qu'il existe une valeur de vérité pour X_ℓ t.q. (resp. pour toute valeur de vérité pour X_ℓ,) ψ est vraie.

Considérons maintenant la formule d'USL^{lp,0} suivante :

$$\varphi_I \triangleq \langle\langle x_1 \rangle\rangle (a_1 \rhd x_1)(\neg\langle\langle x_2 \rangle\rangle (a_2 \rhd x_2) \neg(\\ \langle\langle x \rangle\rangle (a_3 \rhd x)(\dots(\neg^k \langle\langle x_k \rangle\rangle (a_k \rhd x_k) \neg^k (\bigwedge_{j \in 1,\dots,J} \Diamond C_j))\dots)$$

où, pour tout entier ℓ tel que $1 \leqslant \ell \leqslant k$, $\neg^\ell$ est égal à $\neg$ si ℓ est impair et est l'opérateur identité si ℓ est pair. Cette formule est vraie en u_1 ssi $\exists X_1 \forall X_2 \exists X_3 \dots Q_k X_k \varphi(X_1, X_2, X_3, \dots X_n)$ est satisfiable. On a donc le résultat du Lemme 5.21. □

On peut alors avancer le théorème suivant :

Théorème 5.22. MC($USL^{lp,0}$) *est* PSPACE*-difficile.*

Démonstration. Comme la version non bornée de QBFSAT est PSPACE complète, on obtient la borne inférieure correspondante pour le *model-checking* d'USL^{lp,0} dans son ensemble. □

Les résultats similaires pour MC(USL^0) et MC(USL$^{cf,\,0}$) s'obtiennent de la même manière :

Corollaire 5.23. MC(USL^0) *et* MC($USL^{cf,\,0}$) *sont tous les deux* PSPACE*-difficiles.*

Démonstration. Soit $\mathcal{I}$ une instance de QBFSAT. Avec les notations de la preuve qui précèdent, on vérifie que $\mathcal{G}_{\text{QBF}} \models_{\text{USL}^{lp,0}} \varphi_I$ ssi $\mathcal{G}_{\text{QBF}} \models_{\text{USL}^0} \varphi_I$ ssi $\mathcal{G}_{\text{QBF}} \models_{\text{USL}^{cf,\,0}} \varphi_I$ □

5.3.3.2 Complétude

Les résultats de complétude sont obtenus par adaptation de la preuve pour le résultat équivalent pour ATL^{*0}_{sc}, développée dans [DCL11].

Lemme 5.24. $\mathsf{MC}(USL^{\infty,o})$ *est* PSPACE*-Complet.*

Démonstration. On décrit un algorithme récursif de décision pour $\mathsf{MC}(\mathrm{USL}^{\infty,o})$. Cet algorithme énumère les multi-stratégies positionnelles possibles. Il utilise ensuite, pour chacune de ces multi-stratégies positionnelles, une restriction de la CGS^{∞} dans laquelle le problème est posé, qui est envisagée comme un modèle de Kripke. L'algorithme fait alors appel au *model-checking* de LTL dans des modèles de Kripke.

On introduit d'abord une notation qui permet de traiter une formule de chemin d'$\mathrm{USL}^{\infty,o}$ comme une formule de LTL dont les atomes sont les sous-formules d'états de la formule de départ. Pour une formule φ donnée, on écrit $\mathcal{Q}(\varphi)$ l'ensemble de ses sous-formules d'états maximales (les sous formules d'états ψ de φ telles que ψ n'est pas sous-formule d'état propre d'une autre sous-formule propre de φ). On écrit également $\mathrm{LTL}(\varphi)$ la formule obtenue de φ en y remplaçant chaque sous-formule ψ de $\mathcal{Q}(\varphi)$ par un nouvel atome $\overline{\psi}$.

Soient maintenant $\mathcal{G}$ une CGS^{∞}, χ un contexte pour $\mathcal{G}$, s un état de $\mathcal{G}$, et φ une formule de LTL. Alors $\mathsf{MC}(\mathrm{LTL}(\mathcal{G},\chi,s,\varphi))$ désigne le *model-checking* pour la satisfaction $\models_{\mathrm{LTL}}$ de φ depuis l'état s du modèle de Kripke $\mathcal{G}_{\chi}$ défini par :

- L'ensemble des états G et la fonction de valuation sont ceux de $\mathcal{G}$.
- La relation de transition R est : pour tout $s,s' \in \mathrm{G}, R(s,s')$ ssi $s' \in Msucc^{\infty}_{\chi}(s)$

Avec ces notations, la procédure est décrite dans l'Algorithme 1. Il calcule, pour tous $\mathrm{CGS}^{\infty}\mathcal{G}$, contexte χ, état s et formule φ, la décision $\mathsf{MC}(\mathrm{USL}^{\infty,o}(\mathcal{G},\chi,s,\varphi))$ de $\mathcal{G},\chi,s \models_{\mathrm{USL}^{\infty,o}} \varphi$. Par ailleurs, une multi-stratégie peut être stockée dans un espace de taille $\bigcirc(|\mathrm{G}| \times |Ac|)$. Comme la valuation des états est linéaire en fonction de la taille de φ et comme le problème utilisé comme oracle dans l'Algorithme 1 ($\mathsf{MC}(\mathrm{LTL})$) est lui-même PSPACE-complet [SC85], l'Algorithme 1 répond également en espace polynomial.

□

On en déduit le résultat suivant :

Théorème 5.25. $\mathsf{MC}(USL^{cf,\,0})$ *est* PSPACE*-Complet.*

Démonstration. (Résumé) Immédiat, grâce au Lemme 5.24 et à la réduction de $\mathsf{MC}(\mathrm{USL}^{cf,\,0})$ à $\mathsf{MC}(\mathrm{USL}^{\infty,o})$ décrite dans la Section 5.3.1.2.

□

Les problèmes $\mathsf{MC}(\mathrm{USL}^{0})$ et $\mathsf{MC}(\mathrm{USL}^{lp,0})$ donnent lieu à des résultats similaires :

Théorème 5.26. *Les problèmes* $\mathsf{MC}(USL^{0})$ *et* $\mathsf{MC}(USL^{lp,0})$ *sont* PSPACE*-complets.*

Démonstration. Le résultat pour $\mathsf{MC}(\mathrm{USL}^{0})$ vient directement. Comme $\mathsf{MC}(\mathrm{USL}^{cf,\,0})$ englobe $\mathsf{MC}(\mathrm{USL}^{0})$. Pour $\mathsf{MC}(\mathrm{USL}^{lp,0})$, on ne peut pas s'appuyer sur le plongement dans USL^{0} de la Section 5.2.2.2, comme celui-ci génère un accroissement exponentiel de la taille de la formule. L'algorithme décrit ci-dessus est cependant aisément adaptable à $\mathrm{USL}^{lp,0}$: il suffit de remplacer l'appel à $Msucc^{\infty}_{\chi}$ par un appel à $Msucc^{lp}_{\chi}$ dans la construction de $\mathcal{G}_{\chi}$. □

Algorithme 1 $\mathsf{MC}(\mathrm{USL}^{\infty,o}(\mathscr{G},\chi,s,\varphi))$

requiert: $\mathscr{G}$, une CGS$^\infty$, $\chi=\langle\alpha,\gamma\rangle$, un contexte positionnel pour $\mathscr{G}$, $s_0\in \mathrm{G}$ et φ une formule de chemins d'USL^0.
assure: OUI ssi $\mathscr{G},\chi,s\models_{\mathrm{USL}^o}\varphi$
  $\mathscr{G}'=\mathscr{G}_\chi$
  pour tout $\psi\in\mathscr{Q}(\varphi)$ **faire**
    pour tout $s'\in \mathrm{G}$ **faire**
      si $\psi=\langle\langle x\rangle\rangle\psi'$ **alors**
        pour tout $\sigma\in \mathit{Strat}^0$ **faire**
          si $\mathsf{MC}(\mathrm{USL}^0(\mathscr{G},\langle\alpha,\gamma[x\to\sigma]\rangle,s',\psi))$ **alors**
            étiqueter s' avec $\overline{\psi}$
          fin si
        fin pour tout
      sinon si $\psi=(A\rhd x)\psi'$ **alors**
        si $\mathsf{MC}(\mathrm{USL}^0(\mathscr{G},\langle\alpha,\gamma[A\oplus x]\rangle,s,\psi))$ **alors**
          étiqueter s' avec $\overline{\psi}$
        fin si
      sinon si $\psi=(A\not\rhd x)\psi'$ **alors**
        si $\mathsf{MC}(\mathrm{USL}^{\infty,o}(\mathscr{G},\langle\alpha,\gamma[A\ominus x]\rangle,s,\psi))$ **alors**
          étiqueter s' avec $\overline{\psi}$
        fin si
      fin si
    fin pour tout
  fin pour tout
  renvoie $\mathsf{MC}(\mathrm{LTL}(\mathscr{G}',s,\mathrm{LTL}(\varphi)))$

Chapitre 6

Formalisation de KHI et des critères de correction pour l'assignation

Sommaire

Dans ce chapitre, on utilise les apports des Chapitres 4 et 5 pour formaliser les instances de KHI et les critères de correction sur ces instances. On introduit ainsi une variante d'USL^0, USL_{KHI}. On décrit alors la modélisation d'une instance de KHI et d'une affectation de ses variables des moyens en un modèle d'USL_{KHI} (Section 6.1). On donne ensuite, dans la Section 6.2, la formalisation des critères de correction décrits dans la Section 3.5. On l'illustre également à l'aide de variantes du modèle $\mathsf{Kh}_{C,A}$ (Section 6.3).

6.1 Modélisation d'une instance de KHI en USL_{KHI}

Dans le Chapitre 3, on a décrit la manière dont les éléments du langage des exigences dans une instance de KHI étaient formalisés en LTL. On modélise ici les moyens d'une instance de KHI en un modèle sémantique, étant donnée une affectation. Dès lors, on pourra représenter le problème de l'assignation comme un problème de satisfaction de formules d'USL_{KHI} dans ce modèle sémantique. On présente d'abord la syntaxe d'USL_{KHI} (Section 6.1.1), puis une procédure

pour modéliser une instance Kh de KHI, avec une affectation, en modèles d'USL$_{\text{KHI}}$ (Section 6.1.2). La sémantique pour USL$_{\text{KHI}}$ est alors donnée dans la Section 6.1.3.

6.1.1 Le langage USL$_{\text{KHI}}$

Rappelons que, dans la *Synthèse de l'état de l'art et introduction aux chapitres suivants* intercalée entre les Chapitres 2 et 3, nous avons décrit les buts à remplir par KHI et USL$_{\text{KHI}}$. Ces buts sont décrits en particulier dans la Figure de la page 56 : les exigences de *contextes de stratégies* et *distinction liaison-révocation* imposent d'utiliser une version d'USL. Pour satisfaire l'exigence de *capacités conflictuelles*, il faut une sémantique de type USL^{cf} interprétée sur les CGS^{cf}*s*. Par ailleurs, les atomes de ce langage sera dans Cond.

Enfin, l'exigence de *model-checking simple* impose de faire les choix qui économisent le plus de ressources dans la procédure de *model-checking*. Les propriétés que nous formaliserons avec USL$_{\text{KHI}}$ sont toutes des propriétés de capacités concurrentes de coalitions à jouer des rôles. Or la formalisation d'un rôle est une instance du schéma suivant :

$$Form(\rho) \triangleq \bigwedge_{i \in I} \Box \bigwedge_{j \in \{1,2,3\}} (\varphi_{i,j} \rightarrow \mathsf{X} \psi_{i,j})$$

où $\varphi_{i,j}$ et $\psi_{i,j}$ sont des formules de *Cond*. Il s'agit donc toujours, pour vérifier que les coalitions peuvent jouer des rôles, de savoir si, étant données les propositions $\varphi_{i,j}$ que satisfait l'état courant, ces coalitions sont en mesure d'assurer les conditions $\psi_{i,j}$ pour le prochain état. Les capacités des acteurs sont toujours évaluées selon l'état courant seulement. L'utilisation de multi-stratégies positionnelles est donc suffisante pour la modélisation de KHI.

Comme mentionné ci-dessus, on cherche à modéliser la correction d'une assignation sous une affectation donnée des moyens. On a la définition suivante pour les formules de USL$_{\text{KHI}}$:

Définition 6.1 (USL$_{\text{KHI}}$ (Kh))**.** *Soit Kh une instance de* KHI *avec l'ensemble d'acteurs Acteur et dont le langage des exigences est caractérisé par l'ensemble de variables Var(E). Alors, USL$_{\text{KHI}}$ (Kh) est un langage USL (Ag, At, Var(E)) où*

- *Ag = Acteur*
- *At = Cond(Var(E))*

6.1.2 KHI et les CGS$^{cf}_{\text{KHI}}$*s*

On interprète USL$_{\text{KHI}}$ dans des modèles dérivés des CGS^{cf}*s* pour l'interprétation de KHI. On les appelle CGS$^{cf}_{\text{KHI}}$*s*. Une CGS$^{cf}_{\text{KHI}}$ est un modèle $\mathscr{G} = \langle Ag, \mathrm{G}, S_o, At, val, Ac, tr \rangle$ défini comme une CGS^{cf}, à la différence que l'état initial s_0 dans une CGS^{cf} est remplacé par un ensemble d'états initiaux S_o dans une CGS$^{cf}_{\text{KHI}}$. Par ailleurs, la construction d'une CGS$^{cf}_{\text{KHI}}$ est particulièrement spécifiée pour une instance de KHI et une affectation.

À partir d'une instance Kh de KHI et d'une affectation Aff, la Section 6.1.2 décrit donc ces modèles en deux étapes : une première CGS$^{cf}_{\text{KHI}}$, $\mathscr{G}_{\mathsf{Kh,Aff}} = \langle Ag, \mathrm{G}, S_o, At, val, Ac, tr \rangle$, est directement engendrée par une description des états possibles du système dans Cond(Var(E)). On construit ensuite une seconde CGS$^{cf}_{\text{KHI}}$, $\mathscr{G}_{\mathsf{Kh,Aff},f} = \langle Ag, \mathrm{G}_f, S_{o_f}, At, val_f, Ac, tr_f \rangle$. On l'obtient par un quotient des états de $\mathscr{G}_{\mathsf{Kh,Aff}}$. Elle est de taille finie et rend donc possible l'utilisation de la procédure de *model-checking* décrite dans la Section 5.3.3.

Dans ces deux $\text{CGS}^{cf}_{\text{KHI}}$, les actions des agents sont décrites à travers des *choix*, qui sont les ensembles de successeurs qu'une action jouée par un agent depuis un état laisse possibles. L'ensemble des actions de la $\text{CGS}^{cf}_{\text{KHI}}$ est ensuite dérivé de cette première définition.

L'ensemble des états de $\mathcal{G}_{\mathsf{Kh,Aff}}$ correspond à l'ensemble des assignations possibles pour les variables dans Var(E). On obtient l'ensemble des états de $\mathcal{G}_{\mathsf{Kh,Aff},f}$ en fusionnant, parmi les états de $\mathcal{G}_{\mathsf{Kh,Aff}}$, ceux qui satisfont les mêmes propositions parmi celles utilisées dans Kh sous l'affectation Aff.

Par leurs choix, les agents décrivent directement le sous-ensemble des potentiels états successeurs qu'ils laissent possible. L'action des agents dans Kh consiste en effet à déterminer éventuellement la ou les valeurs d'une ou plusieurs variables. Chaque choix pour un agent a est donc représenté par l'ensemble des états compatibles avec ces valeurs. À chaque état, l'ensemble des choix disponibles pour un agent est clos par intersection, de manière à ce qu'il puisse éventuellement y agir selon une ou plusieurs de ses capacités.

La fonction de transition identifie le successeur d'un état en fonction de l'état courant et de l'ensemble des choix exprimés par les agents. Sauf choix conflictuels ou engagement contradictoire de la part d'un acteur, ce successeur est toujours un élément de l'intersection des choix. Le choix de formalisation qui est fait pour le déterminer au sein de cet ensemble offre une réponse au problème du cadre *(frame problem) :* en l'absence d'un choix explicite exprimé en ce sens par les acteurs, la valeur d'une variable (ou la satisfaction d'une condition) n'est pas modifiée au cours de l'exécution. En cas de choix conflictuels ou d'engagement contradictoire pour un acteur, la fonction de transition n'est pas définie.

Pour $\mathcal{G}_{\mathsf{Kh,Aff}}$ comme pour $\mathcal{G}_{\mathsf{Kh,Aff},f}$, on définit successivement un ensemble d'agents, un ensemble d'états, un ensemble de propositions atomiques, une valuation et des fonctions de choix jouables et de transitions.

6.1.2.1 Un modèle infini : $\mathcal{G}_{\mathbf{Kh},\mathsf{Aff}}$

On définit d'abord les différents éléments qui constituent le modèle $\mathcal{G}_{\mathsf{Kh,Aff}}$, pour une instance Kh de KHI et une affectation Aff des moyens de Kh.

Tout d'abord, *l'ensemble des agents* dans $\mathcal{G}_{\mathsf{Kh,Aff}}$ est l'ensemble des acteurs dans Kh qui ont un ensemble de capacités non vides :

Définition 6.2 (Ensemble des agents *Ag* dans $\mathcal{G}_{\mathsf{Kh,Aff}}$)**.**

$$Ag = \{a \in Acteur \mid a.peut \neq \varnothing\}$$

L'ensemble des états dans $\mathcal{G}_{\mathsf{Kh,Aff}}$, G, peut dès lors être défini : il s'agit d'un ensemble en bijection avec l'ensemble des assignations possibles des variables des exigences qui sont compatibles avec les propriétés statiques du contexte transformées par Aff.

Définition 6.3 (Ensemble des états G)**.**

$$G \triangleq \{\overline{v} \mid v \in dom(Var(E))^{Var(E)} \text{ et pour toute } pdc \in propContexte_s, \overline{v} \models_{Cond} Form(pdc)[\mathsf{Aff}]\}$$

Les états initiaux sont ceux qui satisfont les propriétés initiales du domaine transformées par Aff :

Définition 6.4 (États initiaux S_0)**.**

$$S_0 = \{\overline{s} \text{ pour toute } pdc \in propContexte_i, s \models_{Cond} Form(pdc)[\mathsf{Aff}]\}$$

Les propositions atomiques de $\mathcal{G}_{\mathsf{Kh,Aff}}$ sont des propositions de $\mathrm{LTL}_{\mathrm{KHI}}(\mathrm{Var(E)})$. L'adéquation avec le formalisme des CGSs telles qu'introduites dans la Définition 2.7 exige d'utiliser un ensemble fini de propositions atomiques. On choisit celles qui sont utilisées dans Kh pour décrire les rôles et les pré-conditions de capacité des acteurs transformées par Aff. On remarque que ceci limite le langage utilisable dans $\mathcal{G}_{\mathsf{Kh,Aff}}$ aux formules de $\mathrm{USL}_{\mathrm{KHI}}$ dont les atomes sont dans cet ensemble. Les formules que nous évaluerons dans $\mathcal{G}_{\mathsf{Kh,Aff}}$ sont en effet celles pour la caractérisation des critères de correction de l'assignation, dont les atomes sont ceux des rôles. En toute rigueur, il n'est donc même pas nécessaire d'inclure les atomes pour les pré-conditions de capacité des acteurs. Mais l'ensemble At incluant ces atomes sera également utilisé pour définir l'ensemble des états de $\mathcal{G}_{\mathsf{Kh,Aff},f}$. On l'utilise donc dès maintenant par souci d'économie de définitions.

Soit une formule φ de Cond(Var(E)). On note $AtSf(\varphi)$ l'ensemble de ses atomes, *i.e.* l'ensemble de ses sous-formules qui sont de la forme $x \sim n$, $x - y \sim n$ ou $x + y \sim n$, où $x, y \in X$, $lab(x) = lab(y) = lab$, n est une constante dans $dom(l)$, et $\sim \in \{<, >, =, \leqslant, \geqslant\}$. Cette notation s'étend à un ensemble de formules Γ : $AtSf(\Gamma) = \bigcup_{\varphi \in \Gamma} AtSf(\varphi)$. On peut alors définir l'ensemble de propositions atomiques At :

Définition 6.5 (Ensemble des propositions atomiques At)**.**

$$At = AtSf(\{Form(\rho)\}_{\rho \in Rôle} \cup Acteur.\mathrm{peut}.préCond[\mathsf{Aff}])$$

où l'on rappelle que pour toute formule φ et pour toute affectation Aff, $\varphi[\mathsf{Aff}]$ *désigne l'ensemble des formules obtenues de φ en y remplaçant toute occurrence de variable x par une variable dans* $\mathsf{Aff}(x)$ *(Définition 3.4).*

La fonction de valuation est donnée par la satisfaction sémantique pour *Cond* :

Définition 6.6 (Valuation *val*)**.**
Soient $\overline{v} \in G$ et $\varphi \in Cond(Var(E))$, alors $\varphi \in val(\overline{v})$ ssi $v \models_{Cond} \varphi$.

Dans KHI, les acteurs expriment leurs choix en décidant la valeur de certaines variables. Afin de définir l'ensemble des actions de $\mathcal{G}_{\mathrm{KHI,Aff}}$, on modélise d'abord ces choix dans les $\mathrm{CGS}^{cf}_{\mathrm{KHI}}s$. On définit ainsi une fonction qui, pour chaque agent et chaque état, indique l'ensemble des *choix jouables* en cet état par cet agent.

Pour modéliser les choix dans le modèle, on définit la notion de *fibre au dessus d'une valuation partielle*. Les fibres sont des ensembles de valuations qui ont en commun la valeur pour une ou plusieurs variables.

Définition 6.7 (Fibre au-dessus d'une valuation partielle)**.** *Soit v une valuation sur un ensemble de variables X_2 et soit X_1 un ensemble de variables tel que $X_2 \subseteq X_1$. On note alors $fb_{X_1}(v)$, et on appelle* fibre sur X_1 au dessus de v*, l'ensemble des valuations sur X_1 qui étendent v :*

$$fb_{X_1}(v) = \{v' \in dom(X_1)^{X_1} \mid \text{pour tout } x_i \in X_2, v'(x_i) = v(x_i)\}$$

On note alors $v = v'_{\restriction X_2}$. On appelle par ailleurs *la base de fb*, et on note *base(fb)*, l'ensemble X_2 des variables sur lequel une fibre *fb* est définie. On note encore par la suite $\overline{fb_{X_1}(v)}$ l'ensemble des états de $\mathcal{G}_{\mathrm{KHI,Aff}}$ qui correspondent à des valuations dans $fb_{X_1}(v)$: $\overline{fb_{X_1}(v)} = \{\overline{w} \mid w \in fb_{X_1}(v)\}$.

Nous arrivons maintenant à la définition de la fonction de choix jouables. Nous définissons d'abord les choix qui correspondent à l'exercice d'une capacité pour l'agent a : dans l'état $\overline{v}$, si

a a une capacité c telle que $c.pr\acute{e}Cond[\mathsf{Aff}]$ est satisfaite en $\overline{v}$, alors il peut donner aux variables dans $c.fen\hat{e}tre[\mathsf{Aff}]$ n'importe quelle valeur qui satisfait $c.fen\hat{e}tre[\mathsf{Aff}]$, en sélectionnant la fibre correspondante. L'ensemble de ces choix est ensuite clos par intersection, de manière à ce qu'un agent puisse jouer en même temps selon plusieurs de ses capacités. On considère également qu'un agent peut choisir de ne pas intervenir sur le système en un état donné. Il n'affecte alors, dans cet état, la valeur d'aucune variable : on modélise cette possibilité en intégrant pour chaque agent et chaque état, l'ensemble G parmi les choix disponibles.

Définition 6.8 (Fonction de choix jouables C: $(Ag \times \mathrm{G}) \to \mathscr{P}(\mathscr{P}(\mathrm{G}))$)**.** *Soit $G' \subseteq G$, pour tout agent a et pour tout état $\overline{v}$, $G' \in C(a,\overline{v})$ ssi*

- *$G' = G$*
- *ou il existe une capacité $(pr\acute{e}Cond, fen\hat{e}tre)$ pour a telle que $v \models_{Cond} pr\acute{e}Cond[\mathsf{Aff}]$, une assignation $w \in dom(\mathsf{Var}(fen\hat{e}tre[\mathsf{Aff}]))^{\mathsf{Var}(fen\hat{e}tre[\mathsf{Aff}])}$ telle que $G' = \overline{fb_{Var(E)}(w)}$ et pour tout $w' \in G'$, $\overline{w'} \models_{Cond} fen\hat{e}tre[\mathsf{Aff}]$*
- *ou il existe deux choix $G_1, G_2 \in C(a,\overline{v})$ pour a dans l'état $\overline{v}$ et $G = G_1 \cap G_2$. Autrement dit, $C(a,\overline{v})$ est clos par intersection.*

La convergence avec le formalisme des CGSs tel que présenté dans la Définition 2.7 impose de définir, plutôt qu'un ensemble de choix particulier par agent et par état, un unique ensemble d'*actions* jouables par tout agent depuis chaque état.

En utilisant les choix jouables introduits dans la Définition 6.8, on définit l'ensemble Ac, qui satisfait ces contraintes. On définit également une fonction Ch qui permet de retrouver, à partir d'un agent, d'un état et d'une action, le choix auquel cette action correspond. Ces définitions ont pour seul but de fournir une description des $\mathrm{CGS}^{cf}_{\mathrm{KHI}}s$ qui entre dans le cadre du formalisme des CGSs. Par la suite, on continuera cependant d'utiliser les choix jouables. La définition des ensembles d'actions et de la fonction Ch est suivie d'une remarque sur la part d'indéterminisme et d'arbitraire qu'elle met en œuvre.

Définition 6.9 (Ensemble d'actions Ac et fonction Ch)**.** *Pour tout agent a et pour tout état $\overline{v}$, on ordonne l'ensemble de choix $C(a,\overline{v})$: $C(a,\overline{v}) = G_{a,\overline{v},1}, \dots G_{a,\overline{v},k_{a,\overline{v}}}$. Soit alors $k = max(k_{a,\overline{v}} \mid a \in Ag, \overline{v} \in G)$. On définit l'ensemble des actions dans $\mathscr{G}_{Kh,\mathsf{Aff}}$ par : $Ac = [1,\dots,k]$. On a alors la fonction de choix jouables Ch, qui prend en argument un agent, un état et une action, et renvoie un choix jouable par cet agent en cet état : pour tous $a \in Ag, \overline{v} \in G, ac \in Ac$,*

$$Ch(a,\overline{v},ac) = \begin{cases} G_{a,\overline{v},ac} & si\ ac \leqslant k_{a,\overline{v}} \\ G_{a,\overline{v},k_{a,\overline{v}}} & sinon \end{cases}$$

La notation $Ch(a,\overline{v},ac)$ est étendue aux ensembles d'actions : pour tout $Ac' \subseteq Ac$,

$$Ch(a,\overline{v},Ac') = \{Ch(a,\overline{v},ac)\}_{ac \in Ac'}$$

Remarque 6.1. *La Définition 6.9 comporte une part d'indéterminisme et d'arbitraire :*

- *Pour tout agent a et pour tout état $\overline{v}$, l'ordre utilisé sur l'ensemble de choix $C(a,\overline{v})$ n'est pas défini.*
- *L'ensemble Ac est fixé pour tout état et pour tout agent. Or ce n'est pas le cas pour les choix disponibles, dont le nombre varie selon les agents et les états. Dans le cas général donc, il y a plus d'éléments dans Ac que dans $C(a,\overline{v})$. La fonction Ch doit seulement être une fonction injective de $C(a,\overline{v})$ dans Ac. Cette exigence d'injectivité étant remplie, le choix de la Définition 6.9 est complètement arbitraire.*

On peut maintenant définir la fonction de transition du modèle :

Définition 6.10 (Fonction de transition *tr*)**.** *La fonction de transition est définie par :*

$$dom(tr) = \{(\overline{v}, dc) \mid \bigcap_{a \in Ag} Ch(a, \overline{v}, dc(a)) \neq \emptyset\}$$

et, pour tout $(\overline{v}, dc) \in dom(tr)$,

$$tr(\overline{v}, dc) = \bigcap_{a \in Ag} Ch(a, \overline{v}, dc(a)) \cap \overline{fb_X(v_{\restriction X_{\text{Pass}}})}$$

avec $X_{\text{Pass}} = \{x_k \in X \mid \forall a \in Ag(x_k \notin base(dc(a)))\}$.

Intuitivement, $\overline{fb_X(v_{\restriction X_{Pass}})}$ représente les assignations pour lesquelles les variables non affectées par les choix des agents (les variables dans X_{Pass}) ont la même valeur que dans l'assignation v, correspondant à l'état courant.

Le modèle $\mathscr{G}_{\mathsf{Kh,Aff}}$ permet de modéliser les capacités d'actions des acteurs sur le système. C'est cependant un modèle de cardinal infini. Il ne permet donc pas d'utiliser de procédure finie de *model-checking*.

Afin de rendre utilisable la procédure décrite dans la Section 5.3.3, nous définissons donc un modèle fini, $\mathscr{G}_{\mathsf{Kh,Aff},f}$. Elle est obtenue par le quotient de $\mathscr{G}_{\mathsf{Kh,Aff}}$ par la relation d'équivalence obtenue de la satisfaction des sous-ensembles de propositions atomiques.

6.1.2.2 Un modèle fini : $\mathscr{G}_{\mathsf{Kh,Aff},f}$

Les ensembles des agents et de propositions atomiques pour $\mathscr{G}_{\mathsf{Kh,Aff},f}$ sont identiques à ceux pour $\mathscr{G}_{\mathsf{Kh,Aff}}$. De même que pour $\mathscr{G}_{\mathsf{Kh,Aff}}$, nous définissons pour $\mathscr{G}_{\mathsf{Kh,Aff},f}$ l'ensemble des états G_f, l'ensemble des états initiaux S_{0_f} la valuation val_f, la fonction C_f de choix disponibles par agent et par état et la fonction de transition tr_f.

La définition de l'ensemble d'états G_f nécessite un quotient de l'ensemble G par la relation d'équivalence sémantique modulo les éléments de *At* du modèle.

On définit ainsi la relation d'équivalence $\equiv_f$ sur G. Elle associe deux états ssi ils satisfont les mêmes atomes dans *At*.

$$\forall \overline{v}, \overline{v'} \in \mathrm{G}(\overline{v} \equiv_f \overline{v'} \leftrightarrow (\forall p \in At((\overline{v} \models_{\text{Cond}} p \leftrightarrow (\overline{v'} \models_{\text{Cond}} p)))))$$

Définition 6.11 (L'ensemble des états G_f)**.** *L'ensemble des états de* $\mathscr{G}_{\mathsf{Kh,Aff},f}$*, noté* G_f*, est le quotient de G par* $\equiv_f$

L'ensemble G_f est donc en bijection avec un sous-ensemble de $\mathscr{P}(At)$: à chaque état $\overline{s}$ de G_f est associé un élément s de $\mathscr{P}(At)$ qui est le plus grand ensemble de conditions de *At* vraies dans toute assignation de $\overline{s}$.

À nouveau, les états initiaux, les états de S_{0_f}, sont les états qui satisfont les propriétés initiales du domaine. Avec la notation utilisée ci-dessus on a la définition suivante :

Définition 6.12 (Etats initiaux S_{0_f})**.** *Les états initiaux sont les classes d'états* s_{0_f} *de G t.q.*

$$\textit{pour tout } \overline{v} \in s_{0_f},\ \textit{pour toute } pdc \in propContexte_i, v \models_{Cond} Form(pdc)$$

La *valuation* val_f est donnée de manière immédiate par la correspondance mentionnée ci-dessus entre un état $\overline{s}$ et un ensemble $s \subseteq At$: pour tout $p \in At$ et $\overline{s} \in \mathrm{G}_f$, $p \in val_f(\overline{s})$ ssi $p \in s$.

La *fonction de choix jouables* C_f est obtenue en utilisant $\equiv_f$.

Définition 6.13 (Fonction de choix jouables $C_f : Ag \times \mathrm{G}_f \to \mathscr{P}(\mathscr{P}(\mathrm{G}_f))$)**.** *Soit $G'_f \subseteq G_f$, alors pour tout agent a et pour tout état $\overline{s}$, $G'_f \in C_f(a,\overline{s})$ ssi il y a $\overline{v} \in s$ et $G' \in C(a,\overline{v})$ tels que $G'_f = [G']_{\equiv_f}$.*

L'ensemble d'actions est le même que pour $\mathscr{G}_{\mathsf{Kh},\mathsf{Aff}}$, soit l'ensemble Ac de la Définition 6.9. De même que pour $\mathscr{G}_{\mathsf{Kh},\mathsf{Aff}}$, on définit une fonction qui permet d'associer un choix à une action jouée par un agent dans un état donné. On la note Ch_f : pour tout agent a, pour tout état $\overline{s}$ de G' et pour toute action ac, elle définit un choix $Ch_f(a,\overline{s},ac)$. Elle est définie de manière analogue à la fonction Ch pour $\mathscr{G}_{\mathsf{Kh},\mathsf{Aff}}$ et elle est également étendue aux ensembles d'actions (cf. définition 6.9).

La *fonction de transition* tr_f est également définie de manière analogue à tr. Le successeur d'un état est pris dans l'intersection des choix des agents. Au sein de cette intersection, s'il reste des atomes dont la valeur de vérité n'est pas fixée, on leur donne la même valeur que dans l'état courant.

Définition 6.14 (Fonction de transition tr_f)**.** *La fonction de transition est définie par :*

$$dom(tr_f) = \{(\overline{s}, dc) \mid \bigcap_{a \in Ag} Ch_f(a,\overline{s},dc(a)) \neq \varnothing\}$$

et, pour tout $(\overline{s}, dc) \in dom(tr_f)$,

$$tr(\overline{s}, dc) = \bigcap_{a \in Ag} Ch_f(a,\overline{s},dc(a)) \cap P_{\mathrm{Pass}}(\overline{s}, dc)$$

où :

$$P_{\mathrm{Pass}}(\overline{s}, dc) = \{\overline{t} \mid \forall p \in At((\exists \overline{u}, \overline{v} \in \bigcap_{a \in Ag} Ch_f(a,\overline{s},dc(a)) \\ ((\overline{u} \models_{Cond(At)} p \wedge \overline{v} \not\models_{Cond(At))} p)) \to ((p \in s) \leftrightarrow (p \in t))\}$$

Intuitivement, $P_{Pass}(\overline{s}, dc)$ est l'ensemble des états qui partagent avec $\overline{s}$ la valuation sur l'ensemble des propositions atomiques dont la valeur de vérité n'est pas fixée par dc. C'est donc l'équivalent, pour des valuations booléennes, de la fibre de base les variables non fixées par dc et dont toutes les valuations sont équivalentes à s sur ces variables. C'est donc la contrepartie de l'ensemble $\overline{fb_X(v_{\restriction X_{Pass}})}$ dans la Définition 6.10.

Le modèle $\mathscr{G}_{\mathsf{Kh},\mathsf{Aff},f}$ donne ainsi une interprétation de Kh, Aff dans un modèle fini et ouvre la voie pour la résolution du problème de l'assignation.

On peut désormais formuler l'équivalence, pour les formules avec atomes dans At, entre les deux modèles étudiés dans la Section 6.1.2.

Théorème 6.1. *Soient Kh un modèle de* KHI *et* Aff *une affectation des moyens de Kh. Soient aussi $\mathscr{G}_{Kh,\mathsf{Aff}} = \langle Ag, G, At, val, Ac, tr, s_o\rangle$ et $\mathscr{G}_{Kh,\mathsf{Aff},f} = \langle Ag, G_f, At, val_f, Ac, tr_f, s_{o_f}\rangle$ les deux modèles issus de Kh comme décrit ci dessus, soient $\overline{v} \in G$ et $[\overline{v}]_f$ la classe d'équivalence de $\overline{v}$ par $\equiv_f$, et φ une formule d'états d'USL$_{\mathrm{KHI}}$ dont les atomes sont dans At. Alors :*

$$\mathscr{G}_{Kh,\mathsf{Aff}}, \overline{v} \models_{USL_{\mathrm{KHI}}} \varphi \text{ ssi } \mathscr{G}_{Kh,\mathsf{Aff},f}, [\overline{v}]_f \models_{USL_{\mathrm{KHI}}} \varphi$$

Démonstration. Immédiate, par récurrence structurelle sur φ.

□

6.1.3 Sémantique pour USL_{KHI}

La logique USL_{KHI} emprunte à $\text{USL}^{cf,\,0}$ la définition pour la relation de satisfaction d'une formule dans un état d'un modèle. On fait d'abord l'observation suivante pour la sémantique de $\text{USL}^{cf,\,0}$:

Remarque 6.2. *Pour toute* CGS^{cf}_{KHI} $\mathcal{G} = \langle Ag, G, S_o, At, val, Ac, tr\rangle$, *pour tous* $s_1, s_2, s' \in G$, *en appelant* $\mathcal{G}_{s_i}$ *la* CGS^{cf} $\mathcal{G}_{s_i} = \langle Ag, G, s_i, At, val, Ac, tr\rangle$ *pour* $i \in \{1,2\}$, *pour toute formule* φ *de* $USL^{cf,\,0}$, $\mathcal{G}_1, s' \models_{USL^{cf,\,0}} \varphi$ *ssi* $\mathcal{G}_2, s' \models_{USL^{cf,\,0}} \varphi$.

Cette observation est immédiate : la définition pour la satisfaction d'une formule de $\text{USL}^{cf,\,0}$ dans un état d'un modèle est indépendante de l'état initial de ce modèle.

La satisfaction d'une formule de USL_{KHI} dans un modèle est définie comme la satisfaction de cette formule dans chacun des états initiaux de ce modèle. On a donc la définition suivante :

Définition 6.15 (Satisfaction sémantique pour USL_{KHI})**.**

Soit $\mathcal{G} = \langle Ag, G, S_o, At, val, Ac, tr\rangle$ *une* CGS^{cf}_{KHI}. *Alors, la relation de satisfaction* $\models_{USL_{\text{KHI}}}$ *est définie, pour tout état* $s \in G$ *et pour toute formule* φ *de* USL_{KHI} *par :*

$$\mathcal{G} \models_{USL_{\text{KHI}}} \varphi \text{ ssi pour tout } s' \in G, \mathcal{G}_{s'} \models_{USL^{cf,\,0}} \varphi$$

où $\mathcal{G}_{s'}$ *est la* CGS^{cf} $\mathcal{G}_{s'} = \langle Ag, G, s_i, At, val, Ac, tr\rangle$. *Étant donnée une formule* φ *de* USL_{KHI}, *on écrit* $\mathcal{G} \models_{USL_{\text{KHI}}} \varphi$ *ssi pour tout* $s_0 \in S_0, \mathcal{G}, s_0 \models_{USL_{\text{KHI}}} \varphi$.

6.2 Formalisation des critères de correction pour l'assignation

On dispose à ce stade d'une méthode pour modéliser le domaine d'une instance de KHI dans des $\text{CGS}^{cf}_{\text{KHI}}$s, ainsi que d'une formalisation des rôles. On donne ici la formalisation des critères de correction pour l'assignation formulée dans la Section 3.5. On définit auparavant des définitions relatives aux affectations (Section 6.2.1). On définit ensuite formellement la correction locale (Section 6.2.2), la correction globale (Section 6.2.3), la collaboration (Section 6.2.4) et la contribution (Section 6.2.5).

6.2.1 Définitions préliminaires

Les formalisations de cette section utilisent les notions d'affectations d'agents et de coalitions, ainsi que celle d'unions entre des affectations des moyens. On commence donc par donner les définitions correspondantes.

Définition 6.16 (Affectations d'agents et de coalitions)**.** *Soit Kh une instance de* KHI, *a un agent de Kh et A une coalition d'agents de Kh. On appelle* affectation de *a (resp.* affectation de *A) une affectation de domaine* $\mathsf{Var}(a)$ *(resp. affectation de domaine* $\mathsf{Var}(A)$*) dans Var(E).*

On définit maintenant l'union entre deux affectations :

Définition 6.17 (Union d'affectations)**.** *: soient* Aff_1 *et* Aff_2 *deux affectations. On note* $\mathsf{Aff}_1 \cup \mathsf{Aff}_2$ *l'affectation de domaine dom*$(\mathsf{Aff}_1 \cup \mathsf{Aff}_2)$ *et définie, pour toute* $x_i \in dom(\mathsf{Aff}_1 \cup \mathsf{Aff}_2)$, *par :*

$$\mathsf{Aff}_1 \cup \mathsf{Aff}_2(x_i) = \begin{cases} \mathsf{Aff}_1(x_i) & si\ x_i \notin dom(\mathsf{Aff}_2) \\ \mathsf{Aff}_2(x_i) & si\ x_i \notin dom(\mathsf{Aff}_1) \\ \mathsf{Aff}_1(x_i) \cup \mathsf{Aff}_2(x_i) & sinon \end{cases}$$

Par ailleurs, la définition des CGSs pour USL_{KHI} ne prend pas en compte les propriétés dynamiques du contexte. Celles-ci ne sont pas prises en compte non plus dans la définition de la satisfaction $\models_{\text{USL}_{\text{KHI}}}$. On les intègre syntaxiquement dans la définition des critères de correction. Dans l'ensemble de ces critères, pour exprimer qu'une coalition d'acteurs assure un rôle ρ étant donné un ensemble de propriétés dynamiques du contexte $propContexte_d$, on formalise le fait que ces acteurs jouent des multi-stratégies dont les issues

- sont cohérentes avec $propContexte_d$ (il y a au moins une exécutions dans ces issues qui satisfait $propContexte_d$) et
- sont telles que toute exécution possiblement finie qui satisfait $propContexte_d$ du contexte est aussi modèle de $Form(\rho)$.

On dit alors que les acteurs assurent la satisfaction des rôles *modulo* les propriétés du contexte. On introduit une notation pour cet aspect :

Notation 6.1.

- *Soient a un agent, x une variable de multi-stratégie et φ une formule de USL_{KHI}. Alors, pour tout ensemble Γ de formules de USL_{KHI}, $(a \rhd x)_\Gamma \varphi$ désigne la formule*

$$\neg(a \rhd x)\neg(\bigwedge_{\psi\in\Gamma}\psi) \wedge (a \rhd x)((\bigwedge_{\psi\in\Gamma}\psi) \rightarrow \varphi).$$

- *Cette définition s'étend aux abbréviations introduites dans la Notation 4.1 : pour un vecteur de variables indicé par les agents du langage $\overrightarrow{x}$ et une coalition d'agents $A = \{a_1, \dots a_n\}$, $\overrightarrow{x}_A$, $(A \rhd \overrightarrow{x_A})_\Gamma$ désigne la séquence $(a_1 \rhd x_{a_1})_\Gamma, \dots, (a_n \rhd x_{a_n})_\Gamma$.*
- *Étant donnée une instance Kh de* KHI*, on désigne par ailleurs par PDC l'ensemble $\{Form(pdc)\}_{pdc\in propContexte_s}$ des formules pour les propriétés dynamiques du contexte.*

Dans les définitions formelles des critères de correction qui suivent, toutes les multi-stratégies jouées par les agents respectent les propriétés du contexte, transformées par une affectation Aff. Les liaisons de multi-stratégies utilisent donc exclusivement l'opérateur $(a \rhd x)_{PDC[\text{Aff}]}$.

6.2.2 Correction locale

On définit d'abord formellement le critère de correction locale, tel que donné informellement dans la Définition 3.16. Cette propriété énonce que pour chacun des rôles, il existe une affectation des moyens telle que la coalition à laquelle ce rôle est assigné est en mesure de l'assurer. On a donc la définition suivante :

Définition 6.18 (CL, définition formelle)**.** *Soit Kh un modèle de* KHI*, Kh satisfait* CL *(Rôle) ssi pour tout rôle $\rho \in$ Rôle il y a une affectation Aff_i de Var(M) dans Var(E) telle que :*

$$\mathcal{G}_{Kh,\text{Aff}_i} \models_{USL_{\text{KHI}}} \langle\langle \overrightarrow{x} \rangle\rangle (\rho.peut \rhd \overrightarrow{x_{\rho.peut}})_{PDC[\text{Aff}_i]} Form(\rho)$$

Pour illustrer ces définitions, on vérifie la correction locale de l'assignation donnée dans la Figure 3.5 :

Proposition 6.2. *Le modèle $Kh_{C,A}$ satisfait* CL *(Rôle).*

Démonstration. (Résumé) Les détails pour la construction du modèle $\mathscr{G}_{\mathsf{Kh}_{C,A}}$, ainsi que pour la satisfaction des rôles pour les rôles *segment embarqué carto* et *segment embarqué armée*, sont donnés dans l'Annexe C.5. Il s'agit de montrer qu'il y a des affectations Aff_c et Aff_{arm} de Var(M) telles que :

$$\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_c} \models_{\mathrm{USL}_{\mathrm{KHI}}} \langle\langle \overrightarrow{x} \rangle\rangle (\{S1,S2\} \rhd \overrightarrow{x_{\{S1,S2\}}})_{PDC[\mathsf{Aff}_c]} \mathit{Form}(\textit{segment embarqué carto})$$

et

$$\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_{arm}} \models_{\mathrm{USL}_{\mathrm{KHI}}} \langle\langle \overrightarrow{x} \rangle\rangle (\{S2,S3\} \rhd \overrightarrow{x_{\{S2,S3\}}})_{PDC[\mathsf{Aff}_{arm}]} \mathit{Form}(\textit{segment embarqué armée})$$

L'illustration procède comme suit :

1. On définit les affectations Aff_c et Aff_{act}. Dans Aff_c, les variables de capacités pour *S1* sont affectées aux variables qui permettent de répondre à $ordre_c^1$ (on dit que *S1* est affecté à $ordre_c^1$), *S2* est affecté à $ordre_c^2$ et *S3* est affecté à $ordre_{arm}^1$. Dans Aff_{arm}, *S1* est affecté à $ordre_c^1$, *S2* est affecté à $ordre_{arm}^1$ et *S3* est affecté à $ordre_{arm}^2$.
2. On construit les $\mathrm{CGS}^{cf}_{\mathrm{KHI}} s\, \mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_c}$ et $\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_{arm}}$.
3. On écrit la formalisation de chacun des rôles *segment embarqué carto* et *segment embarqué armée*. Pour chacun de ces rôles, la formule *Form*(*segment embarqué act*) s'écrit sous la forme $\Psi^1_{act} \wedge \Psi^2_{act}$.
4. Pour chacun de ces rôles, on écrit également des *spécifications de multi-stratégies* $specStrat^1_{act}$ et $specStrat^2_{act}$. Pour $i \in \{1,2\}$, la spécification $specStrat^i_{act}$ décrit sémantiquement les issues obtenues par un agent a qui joue une multi-stratégie σ suffisante pour assurer la satisfaction de Ψ^i_{act}, modulo *PDC*.
5. On observe que les satellites peuvent jouer une multi-stratégie qui respecte $specStrat^i_{act}$, selon leur affectation :
   - Sous Aff_c, *S1* peut jouer d'une manière qui respecte $specStrat^1_c$, *S2* d'une manière qui respecte $specStrat^2_c$ et *S3* d'une manière qui respecte $specStrat^1_{arm}$.
   - Sous Aff_{arm}, *S1* peut jouer d'une manière qui respecte $specStrat^1_c$, *S2* d'une manière qui respecte $specStrat^1_{arm}$ et *S3* d'une manière qui respecte $specStrat^2_{arm}$.
6. On observe que, pour $act \in \{c, arm\}$ et $i \in \{1,2\}$, toute exécution λ infinie issue d'un contexte dans lequel un agent joue une multi-stratégie qui respecte $specStrat^i_{act}$ est telle que $\lambda \models_{\mathrm{LTL}_{\mathrm{KHI}}} \Psi^i_{act}$.
7. On conclut la preuve.

La satisfaction pour les rôles *segment sol carto* et *segment sol armée* est vérifiée de manière analogue. □

6.2.3 Correction globale

La correction globale utilise une unique affectation et un unique ensemble d'une multi-stratégie par acteur pour vérifier la satisfaction de tous les rôles du modèle. Elle est définie formellement comme suit :

Définition 6.19 (CG, définition formelle)**.** *Soit Kh un modèle de* KHI*, Kh satisfait* CG *(Rôle) ssi il y a une affectation* Aff *de Var(M) dans Var(E) telle que :*

$$\mathcal{G}_{Kh,\mathsf{Aff}} \models_{USL_{\mathrm{KHI}}} \langle\langle \overrightarrow{x} \rangle\rangle \bigwedge_{\rho \in Rôle} (\rho.peut \rhd \overrightarrow{x_{\rho.peut}})_{PDC[\mathsf{Aff}]} Form(\rho)$$

L'assignation donnée dans la Figure 3.5 n'est pas correcte selon ce critère. En effet, le satellite *S2* ne peut pas jouer en même temps de manière efficace et simultanée pour chacune des deux coalitions desquelles il fait partie bien que, comme observé dans la Proposition 6.2, il puisse jouer pour chacune d'elle de manière séparée :

Proposition 6.3. *Le modèle $Kh_{C,A}$ ne satisfait pas* CG *(Rôle).*

Démonstration. (Résumé)

On s'en persuade en considérant que les seules affectations pertinentes pour évaluer CG sont des affectations Aff telles que les variables pour chacun des satellites sont affectées, de manière globale, à un ensemble de variables pour la réponse à un ordre.

On cherche alors une solution pour affecter trois satellites aux réponses à quatre ordres. La seule possibilité est d'affecter au moins un des satellites à au moins deux ordres. Considérons par exemple une affectation dans laquelle *S2* est affectée à un ordre $ordre^i_c$ pour *segment embarqué carto* et à un ordre $ordre^j_{arm}$ pour *segment embarqué armée*. L'affectation Aff sera alors, en particulier, telle que $\mathsf{Aff}(tâcheSatellite^2) = \{tâcheEnAttente^i_c, tâcheEnAttente^j_{arm}\}$. Supposons alors que ces deux ordres $ordre^i_c$ et $ordre^j_{arm}$ soient émis en même temps, et $ordre^i_c \neq ordre^j_{arm}$. Alors *S2* doit jouer de manière à donner à $tâcheEnAttente^i_c$ la valeur de $ordre^i_c$ et à $tâcheEnAttente^j_{arm}$ la valeur de $ordre^j_{arm}$. Par la définition de Aff, il ne peut agir sur chacune des deux variables $tâcheEnAttente^i_c$ et $tâcheEnAttente^j_{arm}$ qu'en modifiant la valeur de la variable $tâcheSatellite^2$. Il affecte alors simultanément la même valeur à $tâcheEnAttente^i_c$ et $tâcheEnAttente^j_{arm}$, ce qui contredit la satisfaction des rôles comme $ordre^i_c \neq ordre^j_{arm}$. □

Le critère de correction globale apparaît plus pertinent que la version locale. Avec la correction globale en effet, on envisage les actions concurrentes et respectives de toutes les coalitions.

On a par ailleurs une implication entre la satisfaction de la correction globale et la satisfaction de la correction locale :

Théorème 6.4. *Soit Kh un modèle de* KHI*. Si Kh satisfait* CG*(Rôle), alors Kh satisfait* CL*(Rôle).*

Démonstration. Soit Kh un modèle de KHI tel que Kh satisfait CG(Rôle). Alors il y a une affectation Aff de Var(M) dans Var(E) telle que :

$$\mathcal{G}_{\mathsf{Kh},\mathsf{Aff}} \models_{\mathrm{USL_{KHI}}} \langle\langle \overrightarrow{x} \rangle\rangle \bigwedge_{\rho \in \mathrm{Rôle}} (\rho.\mathrm{peut} \rhd \overrightarrow{x_{\rho.\mathrm{peut}}})_{PDC[\mathsf{Aff}]} Form(\rho)$$

Donc pour tout rôle ρ, Aff est une affectation telle que :

$$\mathcal{G}_{\mathsf{Kh},\mathsf{Aff}} \models_{\mathrm{USL_{KHI}}} \langle\langle \overrightarrow{x} \rangle\rangle (\rho.\mathrm{peut} \rhd \overrightarrow{x_{\rho.\mathrm{peut}}})_{PDC[\mathsf{Aff}]} Form(\rho)$$

Le modèle Kh satisfait donc CL(Rôle). □

L'implication réciproque n'est pas vraie dans le cas général, comme l'illustre le modèle $\mathsf{Kh}_{C,A}$.

6.2.4 Collaboration

On donne ici les définitions pour les relations de collaboration introduites dans la Section 3.5.4.1. On définit d'abord la collaboration entre deux rôles :

Définition 6.20 (Collaboration entre deux rôles, définition formelle)**.** *Soit Kh un modèle de* KHI *et soient deux rôles ρ_1 et ρ_2 dans Kh.Le modèle Kh satisfait* $\mathsf{Col}(\rho_1,\rho_2)$ *ssi il y a une affectation* Aff *de Var(M) dans Var(E) telle que :*

$$\mathcal{G}_{Kh,\mathsf{Aff}} \models_{USL_{\text{KHI}}} \langle\langle \overrightarrow{x} \rangle\rangle (\rho_1.peut \rhd \overrightarrow{\rho_1.peut})_{PDC[\mathsf{Aff}]} (\rho_1 \wedge \langle\langle \overrightarrow{y} \rangle\rangle (\rho_2.peut \rhd \overrightarrow{x_{\rho_2.peut}})_{PDC[\mathsf{Aff}]} \rho_2)$$

Les relations de collaboration entre deux ensembles de rôles, introduites de manière informelle dans la Définition3.19, sont formalisées comme suit :

Définition 6.21 (Collaboration entre deux ensembles de rôles, définition formelle)**.** *Soit Kh une instance de* KHI*, et soit $Rôle_1$ et $Rôle_2$ deux ensembles de rôles dans Kh.*

- *Le modèle Kh satisfait* $\mathsf{Col}_L(Rôle_1, Rôle_2)$ *ssi il y a une affectation* Aff *de Var(M) dans Var(E) et une assignation de variables α : Var(E) $\nrightarrow$ MStrat^0 telles que $dom(\alpha) = \{x_a\}_{a \in Rôle_1.peut}$,*

  $$\mathcal{G}_{Kh,\mathsf{Aff}}, \langle \alpha, \gamma_\varnothing \rangle \models_{USL_{\text{KHI}}} \bigwedge_{\rho \in Rôle_1} ((\rho.peut \rhd \overrightarrow{x_{\rho.peut}})_{PDC[\mathsf{Aff}]} \rho)$$

  et pour tout rôle $\rho' \in Rôle_2$ il y a une affectation $\mathsf{Aff}_{\rho'}$ *telle que*

  $$\mathcal{G}_{Kh,\mathsf{Aff}\cup\mathsf{Aff}_{\rho'}}, \langle \alpha, \gamma_\varnothing \rangle \models_{USL_{\text{KHI}}} (Rôle_1.peut \rhd \overrightarrow{x_{Rôle_1.peut}})_{PDC[\mathsf{Aff}\cup\mathsf{Aff}_{\rho'}]} (\langle\langle \overrightarrow{y} \rangle\rangle (\rho'.peut \rhd \overrightarrow{y})_{PDC[\mathsf{Aff}\cup\mathsf{Aff}_{\rho'}]} Form(\rho'))$$

- *Le modèle Kh satisfait* $\mathsf{Col}_G(Rôle_1, Rôle_2)$ *ssi il y a une affectation* Aff *de Var(M) dans Var(E) telle que :*

  $$\mathcal{G}_{Kh,\mathsf{Aff}} \models_{USL_{\text{KHI}}} \langle\langle \overrightarrow{x} \rangle\rangle (\bigwedge_{\rho \in \rho_1} ((\rho.peut \rhd \overrightarrow{x_{\rho.peut}})_{PDC[\mathsf{Aff}]} \rho) \wedge \bigwedge_{\rho' \in \rho_2} ((\rho'.peut \rhd \overrightarrow{x_{\rho'.peut}})_{PDC[\mathsf{Aff}]} \rho'))$$

6.2.5 Contribution

On donne maintenant les définitions formelles pour les relations de contribution introduites dans la Section 3.5.4.2 :

Définition 6.22 (Contribution entre deux rôles, définition formelle)**.** *Soit Kh un modèle de* KHI *et soient deux rôles ρ_1 et ρ_2 dans Kh. Le modèle Kh satisfait* $\mathsf{Contr}(\rho_1,\rho_2)$ *ssi pour toute affectation* Aff_1 *de* $\mathsf{Var}(\rho_1.joue)$ *dans Var(E) et pour toute assignation de variables α : Var(E) $\nrightarrow$ MStrat^0 telle que $dom(\alpha) = \{x_a\}_{a \in Rôle_1.peut}$ et*

$$\mathcal{G}_{Kh,\mathsf{Aff}_1}, \langle \alpha, \gamma_\varnothing \rangle \models_{USL_{\text{KHI}}} (\rho_1.peut \rhd \overrightarrow{x_{\rho_1.peut}})_{PDC[\mathsf{Aff}_1]} Form(\rho_1)$$

il y a une affectation Aff_2 *de* $\mathsf{Var}(\rho_2.peut)$ *dans Var(E) telle que :*

$$\mathcal{G}_{Kh,\mathsf{Aff}_1\cup\mathsf{Aff}_2}, \langle \alpha, \gamma_\varnothing \rangle \models_{USL_{\text{KHI}}} \langle\langle \overrightarrow{x} \rangle\rangle (\rho_2.peut \rhd \overrightarrow{x_{\rho_2.peut}})_{PDC[\mathsf{Aff}_1\cup\mathsf{Aff}_1]} Form(\rho_2)$$

Les relations de contribution entre deux ensembles de rôles, introduites de manière informelle dans la Définition 3.21, sont formalisées comme suit :

Définition 6.23 (Contribution entre deux ensembles de rôles, définition formelle)**.** *Soit Kh une instance de* KHI *et soit* $R\hat{o}le_1$ *et* $R\hat{o}le_2$ *deux ensembles de rôles dans Kh.*

– *Le modèle Kh satisfait* $\mathsf{Contr}_L(R\hat{o}le_1, R\hat{o}le_2)$ *ssi pour toute affectation* Aff *de* $\mathsf{Var}(Role_1.peut)$ *dans Var(E), pour toute assignation de variables* α : *Var(E)* $\nrightarrow$ *MStrat*^{0} *telle que* $dom(\alpha) = \{x_a\}_{a \in R\hat{o}le_1.peut}$ *et*

$$\mathcal{G}_{\mathsf{Aff}_1}, \langle \alpha, \gamma_\varnothing \rangle \models_{USL_{\mathrm{KHI}}} \bigwedge_{\rho \in R\hat{o}le_1} (\rho.peut \rhd \overrightarrow{x_{\rho.peut}})_{PDC[\mathsf{Aff}_1]} Form(\rho))$$

pour tout $\rho' \in R\hat{o}le_2$ *il y a une affectation* $\mathsf{Aff}_{\rho'}$ *de* $\mathsf{Var}(\rho'.peut)$ *dans Var(E) telle que :*

$$\mathcal{G}_{\mathsf{Aff}_1 \cup \mathsf{Aff}_{\rho'}}, \langle \alpha, \gamma_\varnothing \rangle \models_{USL_{\mathrm{KHI}}}$$
$$(R\hat{o}le_1.peut \rhd \overrightarrow{x_{R\hat{o}le_1.peut}})_{PDC[\mathsf{Aff}_1 \cup \mathsf{Aff}_{\rho'}]} (\langle\langle \overrightarrow{y} \rangle\rangle (\rho'.peut \rhd \overrightarrow{y_{\rho'.peut}})_{PDC[\mathsf{Aff}_1 \cup \mathsf{Aff}_{\rho'}]} Form(\rho'))$$

– *Le modèle Kh satisfait* $\mathsf{Contr}_G(R\hat{o}le_1, R\hat{o}le_2)$ *ssi pour toute affectation* Aff *de* $\mathsf{Var}(Role_1.peut)$ *dans Var(E), pour toute assignation* α : *Var(E)* $\nrightarrow$ *MStrat*^{0} *telle que* $dom(\alpha) = \{x_a\}_{a \in R\hat{o}le_1.peut}$ *et*

$$\mathcal{G}_{\mathsf{Aff}_1}, \langle \alpha, \gamma_\varnothing \rangle \models_{USL_{\mathrm{KHI}}} \bigwedge_{\rho \in R\hat{o}le_1} ((\rho.peut \rhd \overrightarrow{x_{\rho.peut}})_{PDC[\mathsf{Aff}_1]} Form(\rho))$$

il y a une affectation $\mathsf{Aff}_{\rho'}$ *de domaine* $\mathsf{Var}(R\hat{o}le_2.peut)$ *dans Var(E) telle que :*

$$\mathcal{G}_{\mathsf{Aff}_1 \cup \mathsf{Aff}_{\rho'}}, \langle \alpha, \gamma_\varnothing \rangle \models_{USL_{\mathrm{KHI}}} (R\hat{o}le_1.peut \rhd \overrightarrow{x_{R\hat{o}le_1.peut}})_{PDC[\mathsf{Aff}_1 \cup \mathsf{Aff}_{\rho'}]} (\langle\langle \overrightarrow{y} \rangle\rangle \bigwedge_{\rho' \in R\hat{o}le_2} ((\rho'.peut \rhd \overrightarrow{y_{\rho'.peut}})_{PDC[\mathsf{Aff}_1 \cup \mathsf{Aff}_{\rho'}]} Form(\rho'))))$$

Remarque 6.3. *La résolution du* model-checking *dans un modèle pour* USL_{KHI} *appelle une modification de l'Algorithme 1. Il faut en effet énumérer l'ensemble des états initiaux du modèle pour vérifier qu'une formule est satisfaite dans chacun de ces états initiaux. Par ailleurs, les définitions formelles pour les critères de correction ne sont pas, à proprement parler, données dans* USL_{KHI}*. On y insère en effet des quantifications sur les affectations* Aff *et les assignations* α*. Dès lors, l'Algorithme 1 doit être également adapté pour traiter ces quantifications. On le fait en traitant successivement les cas de* $\mathsf{MC}(USL_{\mathrm{KHI}}(\mathcal{G}_{\mathsf{Aff}}, \langle \alpha, \gamma_\varnothing \rangle, s, \varphi))$*, où* Aff *varie dans l'ensemble des affectations et* α *varie dans l'ensemble des assignations. Comme les états initiaux, les affectations et les assignations sont des objets de taille polynomiales sur la taille des entrées, cette énumération n'affecte pas la borne polynomiale de l'espace nécessaire pour la vérification.*

6.3 Illustration des critères de correction à l'aide de variantes du modèle $\mathsf{Kh}_{C,A}$

On utilise ici différentes variantes du modèle $\mathsf{Kh}_{C,A}$ pour illustrer les critères de correction de l'assignation. Dans une première variante, $\mathsf{Kh}_{S_{mem}}$, un nouveau satellite est introduit dans l'ensemble des acteurs. Il est capable de traiter simultanément deux ordres. On s'intéresse à la correction globale et à la relation de contribution dans ce modèle. L'ensemble des rôles pour l'*agence*

cartographique ne contribue pas globalement à l'ensemble des rôles pour l'*armée* (Section 6.3.1). Cette contribution est obtenue dans le modèle Kh_{Secur}, par l'introduction d'une exigence de sécurité dans le modèle de buts pour l'*armée* (Section 6.3.2). Enfin, dans le modèle $\mathsf{Kh}_{C,A,G}$, on introduit un troisième modèle de buts. On peut ainsi illustrer les relations de collaboration locale et globale (Section 6.3.3).

Les détails techniques pour la description de ces différents modèles et les éléments de preuve pour les propositions avancées dans cette section sont donnés dans l'Annexe C.6.

6.3.1 Le modèle $\mathsf{Kh}_{S_{mem}}$: utilisation de satellites à mémoire étendue

Dans cette première variante, on introduit un nouveau satellite, S_{mem}. Ce satellite est différent de ceux utilisés dans le modèle $\mathsf{Kh}_{C,A}$: il peut stocker deux ordres de photographies ainsi que deux images, qu'il peut renvoyer au sol sur deux signaux distincts. On considère alors l'assignation : $assign_{\mathsf{Kh}_{S_{mem}}}$ représentée dans la Figure 6.1. Dans cette assignation, S_{mem} remplace

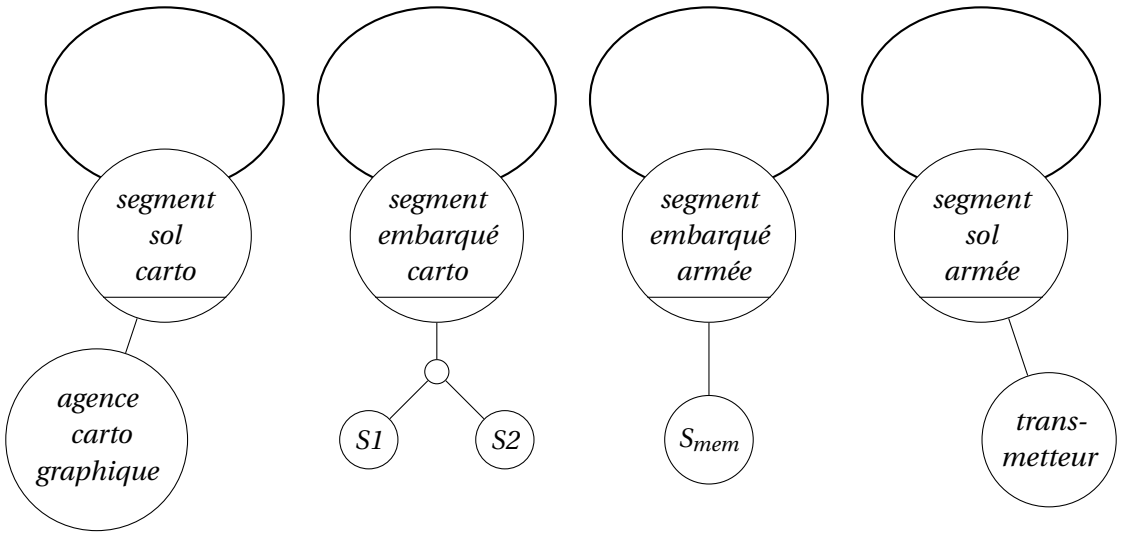

FIGURE 6.1 – Assignation pour le modèle $\mathsf{Kh}_{S_{mem}}$

S3 dans le rôle *segment embarqué armée*. Par ailleurs, *S2* ne participe plus à la coalition pour *segment embarqué carto*.

On a la proposition suivante :

Proposition 6.5. *Le modèle* $Kh_{S_{mem}}$ *satisfait* CG(*Rôle*).

L'assignation $assign_{\mathsf{Kh}_{S_{mem}}}$ est donc correcte, au sens de la correction globale, pour le modèle $\mathsf{Kh}_{S_{mem}}$. C'est-à-dire qu'il y a une manière d'affecter globalement les variables des acteurs et d'utiliser globalement leurs capacités de manière à ce que chacun des rôles ρ soit assuré par la coalition $assign_{\mathsf{Kh}_{S_{mem}}}(\rho)$. Un ingénieur peut donc spécifier le comportement de ces acteurs de manière à garantir l'ensemble des rôles. Comme mentionné dans la Section 3.5.4.2, l'utilisation de ce critère n'a cependant de pertinence que s'il existe effectivement un unique ingénieur qui peut contrôler le comportement de chacun des acteurs (ou s'il y a un accord sur ces comportements). Dans le cas inverse, on s'intéressera à la satisfaction de critères de contribution pour chacun des ingénieurs. On cherche en particulier à savoir si les rôles pour l'*agence cartographique* peuvent

contribuer globalement aux rôles pour l'*armée*. Dans le sens négatif, il s'agit de savoir si les acteurs des coalitions pour l'*agence cartographique* peuvent agir de manière à satisfaire les rôles *segment embarqué carto* et *segment sol carto* tout en empêchant les acteurs *S2* et *transmetteur* d'assurer la satisfaction des rôles *segment embarqué armée* et *segment sol armée*. Or c'est le cas, on vérifie en effet la proposition suivante :

Proposition 6.6. *Le modèle* $Kh_{S_{mem}}$ *ne satisfait pas*

$$\mathsf{Contr}_G(\{\textit{segment embarqué carto}, \textit{segment sol carto}\}, \{\textit{segment embarqué armée}, \textit{segment sol armée}\})$$

Dans le modèle $\mathsf{Kh}_{S_{mem}}$ donc, les acteurs dans les coalitions pour les rôles de l'*armée* dépendent des affectations de variables et des actions des acteurs dans les coalitions pour les rôles de l'*agence cartographique* pour satisfaire leurs rôles : comme les variables de signaux dans les exigences pour l'*armée* et dans celles pour l'*agence cartographique* portent les mêmes étiquettes, une affectation peut en effet traiter de manière uniforme ces différentes variables. Dans cette modélisation, les variables $\textit{réponse}^1_{arm}$ et $\textit{réponse}^2_{arm}$ sont modifiables par n'importe quel acteur ayant une capacité d'action sur une variable étiquetée **image**. Il n'y a donc aucune protection des signaux échangés entre le segment sol et le segment embarqué pour l'*armée*. On corrige ce problème en introduisant, dans le modèle de but pour l'*armée*, un but de *sécurité*. On obtient ainsi le modèle Kh_{Secur}, développé dans la section suivante.

6.3.2 Le modèle Kh_{Secur} : introduction d'un but de sécurité pour *armée*

Dans cette variante, l'*armée* a besoin de sécuriser les échanges d'informations entre son segment sol et son segment embarqué. Un but de *sécurité* est donc ajouté à son modèle de but (cf. Figure 6.3.2). Ce but est raffiné en les exigences *confidentialité* (les messages échangés entre le

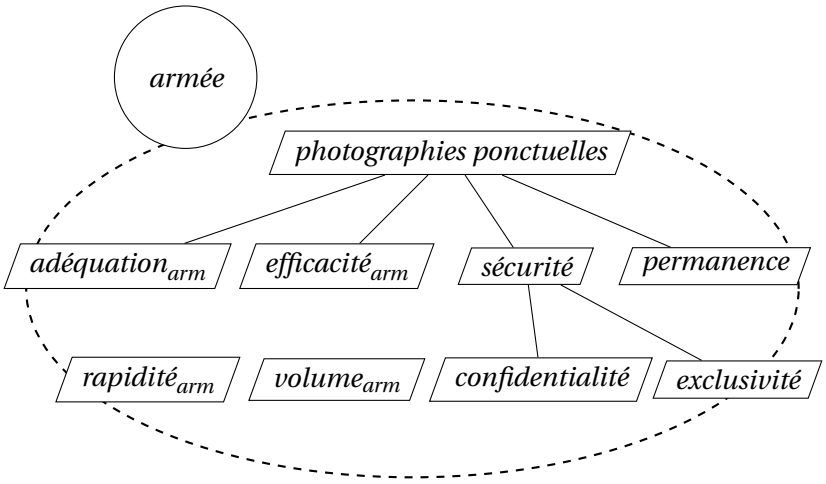

FIGURE 6.2 – Modèle de buts pour *armée* pour le modèle Kh_{Secur}

segment sol et le segment embarqué ne peuvent pas être lus par un tiers) et *exclusivité* (seuls les messages des acteurs capables de s'identifier sont pris en compte). On réalise ces exigences grâce à un cryptage des messages contenus dans les signaux. Pour une variable x représentant un *ordre* ou une *réponse*, le fait de contenir une valeur cryptée est représenté par un booléen $x.crypt$:

Formellement on introduit trois nouvelles étiquettes de variables :

– une étiquette **booléen**, de domaine $\{1,2\}$.

– deux étiquettes pour les valeurs cryptées : une pour les positions et une pour les images. On les appelle respectivement **positionCryp** et **imageCryp**. Les variables étiquetées respectivement **positionCryp** et **imageCryp** sont axiomatisées comme celles étiquetées **position** et **image**. Pour chaque variable x étiquetée **positionCryp** ou **imageCryp**, on introduit donc une variable $x.crypt$ étiquetée **booléen**.

La *confidentialité* assure que tous les messages qui circulent sont cryptés. Pour l'*exclusivité*, on considère que, pour chaque signal, le cryptage vaut comme identification de l'agent. Le but *exclusivité* requiert donc que chaque acteur refasse le cryptage pour envoyer un message. Pour cela, le marqueur de cryptage restera initialisé à la valeur 0 tant que la variable correspondante ne contient pas de message. La satisfaction de ces exigences implique des nouvelles opérations de cryptage et de décryptage des variables $ordre^i_{arm}$ et $réponse^i_{arm}$.

Les acteurs dans cette variante sont ceux du modèle $\mathsf{Kh}_{C,A}$, ainsi que deux appareils de cryptage, $crypteur^1$ et $crypteur^2$. Ces appareils sont capables d'effectuer les opérations de cryptage et de décryptage. L'assignation pour le modèle Kh_{Secur} est représentée dans la Figure 6.3.2 : elle est similaire à celle du modèle $\mathsf{Kh}_{S_{mem}}$, à ceci près que les crypteurs $crypteur^1$ et $crypteur^2$ sont ajoutés respectivement aux coalitions pour les rôles *segment embarqué armée* et *segment sol armée*.

Le modèle Kh_{Secur} est décrit dans l'Annexe C.6.2. On peut vérifier la proposition suivante :

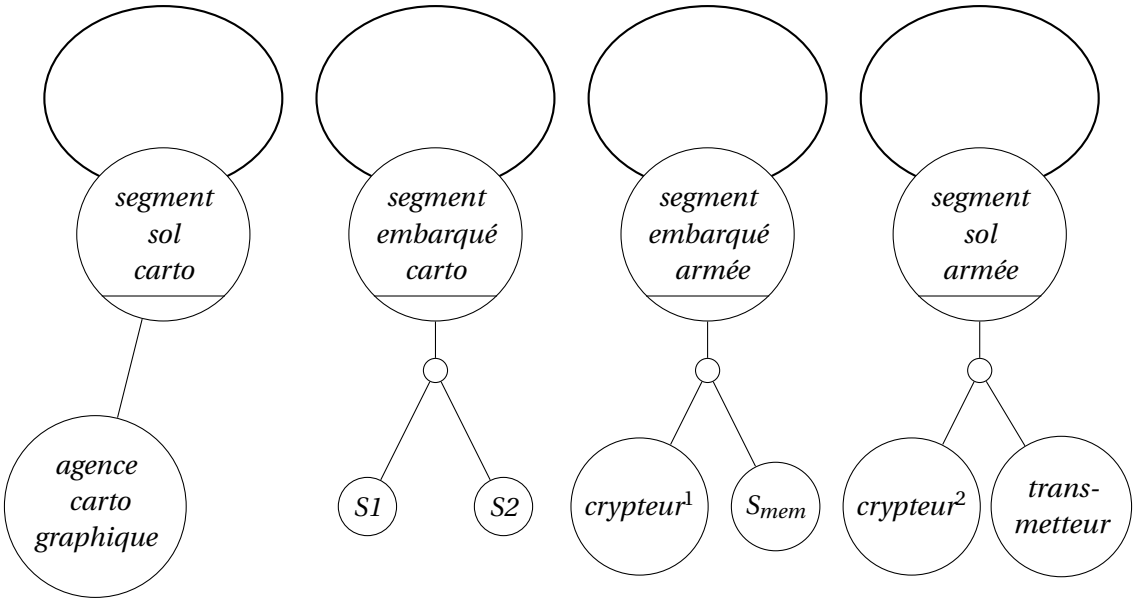

FIGURE 6.3 – Assignation pour le modèle Kh_{Secur}

Proposition 6.7. *Le modèle Kh_{Secur} satisfait*

$\mathsf{Contr}_G(\{$*segment embarqué carto, segment sol carto*$\}$,
$\{$*segment embarqué armée, segment sol armée*$\})$

6.3.3 Le modèle $\mathsf{Kh}_{C,A,G}$: introduction d'un troisième modèle de buts

Cette dernière variante de notre modèle prend en charge les buts pour une troisième mission, *GPS*, qui a également besoin d'une couverture terrestre régulière de la surface de la terre par images satellites. On illustre, grâce à ce modèle, la différence entre les propriétés de collaborations locale et globale. Le modèle de buts et les exigences pour *GPS*, de même que la description des rôles induits par ce nouveau modèle de buts, sont similaires à ceux pour l'*agence cartographique*.

On s'intéresse aux trois coalitions pour les segments embarqués, selon l'assignation représentée dans la Figure 6.3.3 (dans la figure on a représenté exclusivement les segments embarqués) et à leurs interactions : le satellite S_{mem} participe aux coalitions pour chacun des trois rôles de segments embarqués. Chacune de ces coalitions contient par ailleurs un satellite *S1*, *S2* ou *S3*.

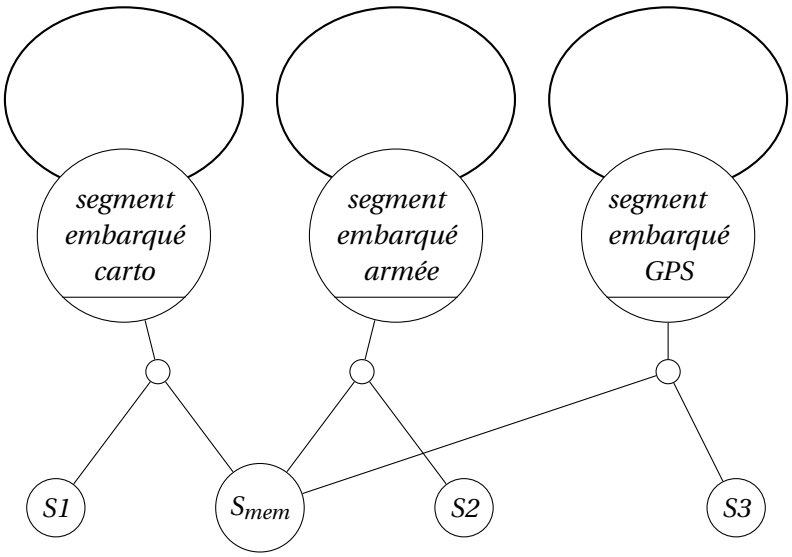

FIGURE 6.4 – Assignation pour le modèle $\mathsf{Kh}_{C,A,G}$ (segments embarqués)

Chacun des satellite *S1*, *S2* et *S3* est en mesure d'assurer la réponse à un ordre, respectivement pour les rôles *segment embarqué carto*, *segment embarqué armée* et *segment embarqué GPS*. Le satellite S_{mem} peut alors agir pour compléter deux des rôles au plus. En particulier, si la coalition pour *segment embarqué armée* assure son rôle, alors, une et une seule des deux coalitions pour *segment embarqué carto* et *segment embarqué GPS* est en mesure d'assurer son rôle. Le rôle *segment embarqué armée* peut donc collaborer localement avec *segment embarqué carto* et *segment embarqué GPS*, mais ne peut pas collaborer globalement avec cet ensemble de rôles :

Proposition 6.8.

– *Le modèle* $Kh_{C,A,G}$ *satisfait*

$$\mathsf{Col}_L(\{\textit{segment embarqué armée}\}, \{\textit{segment embarqué carto}, \textit{segment embarqué GPS}\})$$

– *Le modèle* $Kh_{C,A,G}$ *ne satisfait pas*

$$\mathsf{Col}_G(\{\textit{segment embarqué armée}\}, \{\textit{segment embarqué carto}, \textit{segment embarqué GPS}\})$$

Conclusion et perspectives

Sommaire

Bilan

Le but de ces travaux était de proposer des outils pour formaliser et résoudre le problème de l'assignation en ingénierie des exigences. Notre proposition a consisté d'abord dans le développement de deux éléments :

- Un langage de modélisation, KHI. Il s'agit d'un langage de modélisation pour l'ingénierie des exigences. Il réunit dans un langage unique des possibilités procurées dans l'état de l'art mais développées dans des langages distincts. Ces possibilités sont d'une part un formalisme pour une description comportementale des exigences d'un système à élaborer, d'autre part un cadre conceptuel pour poser le problème de l'assignation. Dans KHI, ce problème est exprimé comme celui de la satisfaction de différents critères de correction pour la relation d'assignation de rôles à des coalitions d'acteurs.
- Une logique temporelle multi-agents, USL. En utilisant des stratégies non déterministes (des *multi-stratégies*), elle permet de formaliser des raffinements de multi-stratégies et donc d'exprimer des situations ou un même agent participe à des coalitions auxquelles sont assignés différents rôles. Par ailleurs, l'utilisation des exécutions possiblement finies dans USL, ainsi que l'extension des CGSs aux CGS^{cf}s, permettent de formaliser les notions à la fois d'engagements contradictoires pour un agent et de capacités conflictuelles pour un ensemble d'agents.

  Nous avons par ailleurs poursuivi le développement de USL et sa comparaison avec les formalismes multi-agents pré-existants. La comparaison avec SL a en particulier été menée : USL est à la fois plus expressive et de complexité équivalente (pour les problèmes de *model-checking* et de satisfiabilité). Ce nouveau formalisme permet en particulier de formaliser la notion de *pérennité* pour les capacités des agents. Il introduit également des prédicats d'inclusion et d'égalité entre les multi-stratégies. L'exploitation de cette possibilité à été amorcée. On a notamment pu caractériser des notions de dépendance et dépendance forte entre multi-stratégies.

Nous avons également adapté USL pour un usage destiné à KHI. Nous avons ainsi défini USL_{KHI}, une version de USL qui permet de formaliser l'ensemble des concepts de KHI. À notre connaissance, KHI associé à USL_{KHI} est aujourd'hui le seul langage de modélisation en ingénierie

des exigences doté d'un moyen de formaliser des exigences comportementales et les capacités d'actions des acteurs sur le système. L'usage d'USL_{KHI} permet en particulier de formaliser le problème de l'assignation et de le résoudre en espace polynomial.

Enfin, nous avons illustré de manière exhaustive l'ensemble des concepts définis et utilisés par KHI et USL_{KHI}, à l'aide d'un cas d'étude concernant des missions d'observation par satellites.

Limitations et perspectives

Certains aspects de ces travaux appellent cependant des enrichissements ou ouvrent des pistes pour de futurs travaux. Dans la Section 3.2.2, nous avons introduit des variables étiquetées. Par ailleurs, dans le cas d'étude des missions d'observations nous avons mis des variables en relation les unes avec les autres selon leurs étiquettes, en les liant par des schémas d'axiomes.

Une version plus approfondie de KHI utiliserait un langage plus riche pour traiter ces variables, en définissant des structures de données. Un langage de spécifications algébriques pourrait par exemple remplacer les propositions de *Cond*.

Dans le prolongement direct de ces travaux, la pertinence de nouveaux critères de correction de l'assignation pourrait être interrogée : les deux prédicats de correction locale et globale de l'assignation correspondent à deux modes de composition différents entre les rôles. Dans les deux cas, on s'intéresse à la conjonction de la satisfiabilité des différents rôles moyennant une quantification existentielle sur les affectations et les multi-stratégies.

- Selon la correction locale, pour chacun des rôles il existe une affectation et un vecteur de multi-stratégies pour satisfaire ce rôle. La quantification sur les affectations et les multi-stratégies est donc dans la portée de la conjonction sur les différents rôles.
- Selon la correction globale, il existe une unique affectation et un vecteur de multi-stratégies pour satisfaire l'ensemble des rôles. C'est donc cette fois la composition des différents rôles qui se trouve dans la portée de la quantification sur les affectations et les multi-stratégies.

Si l'on considère les capacités des différents acteurs comme des ressources dont ils disposent pour réaliser des actions sur le système, alors la correction locale est une conjonction avec duplication des ressources, et la correction globale une conjonction avec partage des ressources pour les différents rôles.

Cet aspect rejoint notamment certains aspects de la logique linéaire [Gir87, Gir06]. Dans ce formalisme en effet, la conjonction $\wedge$ est remplacée par deux opérateurs : *avec*, &, qui est interprété comme une somme directe et le *tenseur*, $\otimes$, interprété comme un produit des ressources disponibles. Une direction à ouvrir serait de généraliser les relations entre les rôles dans les corrections locale et globale, pour les utiliser comme des opérateurs dans un langage formel dont les atomes représenteraient la satisfaction des rôles. Un tel langage permettrait d'étendre encore et d'affiner les critères exprimables et vérifiables à l'aide de KHI. La similarité avec les opérateurs de la logique linéaire pourrait alors être mieux comprise. Ceci permettrait encore d'exprimer des collaborations ou contributions imbriquées.

Sur le plan des logiques multi-agents, plusieurs pistes d'approfondissements pourraient également être ouvertes. La recherche de fragments d'USL avec des meilleurs résultats de complexité devrait être menée. Dans [MMPV12], les auteurs définissent un fragment de SL avec un *model-checking* 2-EXPTIME complet. Il serait intéressant d'examiner si un fragment correspondant existe pour USL, avec la même complexité et un plus grand pouvoir expressif.

L'étude du pouvoir expressif de USL pourrait également être poursuivie. En premier lieu, des preuves pour les deux conjectures selon lesquelles USL^{lp} et SL sont incomparables (Conjectures 5.3 et 5.4) devraient être recherchées.

L'utilisation de l'opérateur de révocation offre par ailleurs des perspectives intéressantes : dans ce mémoire nous avons utilisé l'opérateur de révocation pour qu'un agent révoque sa multi-stratégie seulement immédiatement avant d'être lié à une autre. Un usage plus libre de l'opérateur de révocation pourrait être analysé. Dans ce mémoire également, nous avons donné des éléments d'amorce pour des caractérisations de relation de dépendance et d'uniformité des multi-stratégies. D'éventuels moyens donnés par USL pour poursuivre cette analyse devraient aussi être recherchés.

Des formalismes tels que CTL quantifié (QCTL [LM13]) ou le μ-calcul quantifié [Pin07] permettent de quantifier sur les valuations possibles dans des modèles de Kripke. Ils permettent également de caractériser des stratégies ou des multi-stratégies pour des agents. Une direction d'étude est donc ouverte, qui consiste à caractériser USL comme un fragment de ces formalismes.

Dans le Chapitre 4, les différentes sémantiques que nous avons présentées traduisent la composition de deux multi-stratégies en prenant l'intersection des ensembles de choix exprimées par chacune d'elles. D'autres modes de composition pourraient être envisagés, comme une composition par union des ensembles de choix exprimés par les multi-stratégies. Une telle étude permettrait de poursuivre l'étude de la relation de composition entre multi-stratégies et de mieux la comprendre. Elle pourrait également définir les fragments de QCTL ou du μ-calcul quantifié correspondant aux différents modes de composition des multi-stratégies.

Bibliographie

[ABG+07a] Y. Asnar, R. Bonato, P. Giorgini, F. Massacci, V. Meduri, C. Riccucci et A. Saydane, Secure and Dependable Patterns in Organizations : An Empirical Approach., in *15th IEEE International Requirements Engineering Conference (RE '07)*, New Delhi, India, 15/10/2007 2007.

[ABG07b] Y. Asnar, V. Bryl et P. Giorgini, Using Risk Analysis to Evaluate Design Alternatives., in *Agent-Oriented Software Engineering VII, Post-proceedings (invited paper)*, Springer, 2007.

[AG06] Y. Asnar et P. Giorgini, Modelling Risk and Identifying Countermeasure in Organizations., in *1st International Workshop on Critical Information Infrastructures Security (CRITIS '06)*, Samos Island, Greece, 2006.

[AHK97] R. Alur, T. A. Henzinger et O. Kupferman, Alternating-Time Temporal Logic, in *COMPOS*, pages 23–60, 1997.

[AHK02] R. Alur, T. A. Henzinger et O. Kupferman, *Alternating-time temporal logic*, J. ACM 49(5), 672–713 (2002).

[BBGK07] C. Baier, T. Brazdil, M. Grosser et A. Kucera, Stochastic Game Logic, in *Quantitative Evaluation of Systems.*, pages 227 –236, 2007.

[BDCLLM09] T. Brihaye, A. Da Costa Lopes, F. Laroussinie et N. Markey, *ATL with strategy contexts and bounded memory*, Logical Foundations of Computer Science , 92–106 (2009).

[BMM+07] V. Bryl, M. Montali, P. Mello, P. Torroni et N. Zannone, B-Tropos. Agent-oriented requirements engineering meets computational logic for declarative business process modeling and verification., in *8th Workshop on Computational Logic in Multi-Agent Systems (CLIMA-VIII)*, Porto, Portugal, 10/09/2007 2007.

[BPG+04] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia et J. Mylopoulos, *Tropos : An agent-oriented software development methodology*, Autonomous Agents and Multi-Agent Systems , 203–236 (2004).

[CBC] C. Chareton, J. Brunel et D. Chemouil, A Logic with Revocable and Refinable Strategies, Soumis.

[CBC11] C. Chareton, J. Brunel et D. Chemouil, A Formal Treatment of Agents, Goals and Operations Using Alternating-Time Temporal Logic, in *Brazilian Symposium on Formal Methods (SBMF)*, pages 188–203, 2011.

[CBC12] C. Chareton, J. Brunel et D. Chemouil, Vers une sémantique des jeux pour un langage d'ingénierie des exigences par buts et agents, in *Approches Formelles dans l'Assistance au Développement de Logiciels (AFADL)*, 2012.

[CBC13] C. Chareton, J. Brunel et D. Chemouil, Towards an Updatable Strategy Logic, in *Proc. 1st International Workshop on Strategic Reasoning SR*, 2013.

[CDGM10a] A. Chopra, F. Dalpiaz, P. Giorgini et J. Mylopoulos, Modeling and reasoning about service-oriented applications via goals and commitments, in *Advanced Information Systems Engineering*, pages 113–128, Springer, 2010.

[CDGM10b] A. Chopra, F. Dalpiaz, P. Giorgini et J. Mylopoulos, Reasoning about agents and protocols via goals and commitments, in *Proc. of the 9th Intl Conf. on Autonomous Agents and Multiagent Systems-Volume 1*, pages 457–464, 2010.

[CE82] E. M. Clarke et E. A. Emerson, Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic, in *Logic of Programs, Workshop*, pages 52–71, 1982.

[CHP10] K. Chatterjee, T. A. Henzinger et N. Piterman, *Strategy logic*, Inf. & Comp. 208(6), 677–693 (2010).

[CS09] A. Chopra et M. Singh, Multiagent commitment alignment, in *Proc. of The 8th Intl Conf. on Autonomous Agents and Multiagent Systems-Volume 2*, pages 937–944, 2009.

[CvL13] A. Cailliau et A. van Lamsweerde, *Assessing requirements-related risks through probabilistic goals and obstacles*, Requir. Eng. 18(2), 129–146 (2013).

[DCL11] A. Da Costa Lopes, *Propriétés de jeux multi-agents*, Phd thesis, École normale supérieure de Cachan, September 2011.

[DLLM10] A. Da Costa Lopes, F. Laroussinie et N. Markey, ATL with strategy contexts : Expressiveness and Model Checking, in *IARCS Annual Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 8, pages 120–132, 2010.

[EFH+03] C. Eisner, D. Fisman, J. Havlicek, Y. Lustig, A. McIsaac et D. V. Campenhout, Reasoning with Temporal Logic on Truncated Paths, in *Computer Aided Verification, 15th International Conference (CAV)*, pages 27–39, 2003.

[Gir87] J.-Y. Girard, *Linear Logic*, Theor. Comput. Sci. 50, 1–102 (1987).

[Gir06] J.-Y. Girard, *Le point aveugle : cours de logique. Vers la perfection*, Le point aveugle : cours de logique, Editions Hermann, 2006.

[Jac01] M. Jackson, *Problem frames : analyzing and structuring software development problems*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[KV98] O. Kupferman et M. Y. Vardi, Weak alternating automata and tree automata emptiness, in *ACM Symposium on Theory of computing*, pages 224–233, 1998.

[Let02] E. Letier, *Reasoning about Agents in Goal-Oriented Requirements Engineering*, PhD thesis, Université Catholique de Louvain, 2002.

[LM13] F. Laroussinie et N. Markey, Quantified CTL : expressiveness and complexity, Research Report LSV-13-07, Laboratoire Spécification et Vérification, ENS Cachan, France, April 2013, 41 pages.

[LVL02] E. Letier et A. Van Lamsweerde, Deriving operational software specifications from system goals, in *Proc. of the 10th ACM SIGSOFT symposium on Foundations of software engineering*, page 128, ACM, 2002.

[MGL09] A. Matoussi, F. Gervais et R. Laleau, Expressing KAOS Goal Refinement Patterns with Event-B, in *Rodin User and Developer Workshop 2009 - Extended Abstracts*, edited by M. Butler, S. Hallerstede et L. Voisin, Technical Report, DEPLOY Project, Electronics and Computer Science, University of Southampton, 2009.

[MGL11] A. Matoussi, F. Gervais et R. Laleau, A Goal-Based Approach to Guide the Design of an Abstract Event-B Specification, in *16th International Conference on Engineering of Complex Computer Systems*, IEEE, 2011.

[MMPV11] F. Mogavero, A. Murano, G. Perelli et M. Y. Vardi, *Reasoning About Strategies : On the Model-Checking Problem*, CoRR abs/1112.6275 (2011).

[MMPV12] F. Mogavero, A. Murano, G. Perelli et M. Y. Vardi, What Makes Atl* Decidable? A Decidable Fragment of Strategy Logic., in *CONCUR*, edited by M. Koutny et I. Ulidowski, volume 7454 of *Lecture Notes in Computer Science*, pages 193–208, Springer, 2012.

[MMV10] F. Mogavero, A. Murano et M. Y. Vardi, Reasoning about strategies, in *IARCS Annual Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 8, pages 133–144, 2010.

[MS87] D. E. Muller et P. E. Schupp, *Alternating automata on infinite trees*, "Theoretical Computer Science" 54 (October 1987).

[MS95] D. E. Muller et P. E. Schupp, *Simulating alternating tree automata by nondeterministic automata : New results and new proofs of the theorems of Rabin, McNaughton and Safra*, Theoretical Computer Science 141, 69 – 107 (1995).

[MS06] A. Mallya et M. Singh, *Incorporating commitment protocols into Tropos*, Agent-Oriented Software Engineering VI , 69–80 (2006).

[Pap94] C. H. Papadimitriou, *Computational complexity*, Addison-Wesley, 1994.

[Pin07] S. Pinchinat, Quantified mu-calculus with decision modalities for concurrent game structures, Technical report, Dept. of Computer Science, Faculty of Engineering and Information Technology, Australian National University, 2007.

[SC85] A. P. Sistla et E. M. Clarke, *The Complexity of Propositional Linear Temporal Logics*, J. ACM 32, 733–749 (1985).

[Sis83] A. P. Sistla, *Theoretical issues in the design and verification of distributed systems*, PhD thesis, Harvard University, Cambridge, MA, USA, 1983.

[SvdHed] J. Sack et W. van der Hoek, *A modal logic for mixed strategies*, Studia Logica (accepted).

[vL03] A. van Lamsweerde, From System Goals to Software Architecture, in *Formal Methods for Software Architectures*, pages 25–43, 2003.

[vL04] A. van Lamsweerde, Elaborating Security Requirements by Construction of Intentional Anti-Models, in *ICSE*, pages 148–157, 2004.

[vL09] A. van Lamsweerde, *Requirements Engineering - From System Goals to UML Models to Software Specifications*, Wiley, 2009.

[vLL98] A. van Lamsweerde et E. Letier, Integrating Obstacles in Goal-Driven Requirements Engineering, in *ICSE*, pages 53–62, 1998.

[Yu96] E. S.-K. Yu, *Modelling strategic relationships for process reengineering*, PhD thesis, University of Toronto, Toronto, Ont., Canada, Canada, 1996, UMI Order No. GAXNN-02887 (Canadian dissertation).

[Yu09] E. S. K. Yu, Social Modeling and i*, in *Conceptual Modeling : Foundations and Applications*, pages 99–121, 2009.

[ÅGJ07] T. Ågotnes, V. Goranko et W. Jamroga, Alternating-time temporal logics with irrevocable strategies, in *Theoretical aspects of rationality and knowledge*, pages 15–24, 2007.

[ÅGJ08] T. Ågotnes, V. Goranko et W. Jamroga, Strategic Commitment and Release in Logics for Multi-Agent Systems, in *8th Conference on Logic and the Foundations of Game and Decision Theory*, 2008.

Annexe A

Exposition détaillée de la Conjecture 5.3

Sommaire

On détaille ici les différentes étapes du raisonnement synthétisé dans la Section 5.2.1.1 pour étayer la Conjecture 5.3. On définit d'abord un fragment de SL, le fragment SL_1^1 (Section A.1). On prouve ensuite que s'il y a une formule de SL équivalente à la formule d'USL^{lp} $Form(\mathsf{ContP}(a,p))$, alors cette formule n'est pas équivalente à une formule de SL_1^1 (Section A.2). La Conjecture 5.3 repose elle-même sur la conjecture selon laquelle si cette formule existe, alors elle est équivalente à une formule de SL_1^1 (Section A.3).

A.1 Un fragment de SL : SL_1^1

On définit d'abord SL_1^1 : c'est le fragment de $SL(\{a\}, At, X)$ dans lequel :

- Les opérateurs dérivés $\vee$ et R sont introduits
- la négation est réduite aux atomes
- le quantificateur universel de stratégie $[\![x]\!]$ n'est pas utilisé.

Formellement :

Définition A.1. *SL_1^1 est engendré par la grammaire suivante :*

$$\varphi := p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \varphi \,\mathsf{U}\, \varphi \mid \varphi \,\mathsf{R}\, \varphi \mid \langle\langle x\rangle\rangle\varphi \mid (a,x)\varphi$$

où $a \in Ag$, $p \in At$ et $x \in X$.

A.2 Si $\mathsf{ContP}(a,p)$ est caractérisable dans SL, alors elle n'est pas caractérisable dans SL_1^1

On prouve ici qu'aucune formule de SL_1^1 ne caractérise $\mathsf{ContP}(a,p)$. On procède pour ceci à un plongement de SL_1^1 dans Σ_1^1, le fragment de la logique de second ordre qui n'utilise pas le quantificateur universel d'ordre 2. Ce plongement est valable dans une classe de modèles, $\mathscr{C}$, qui est axiomatisable en logique de premier ordre FOL. On peut donc y appliquer un argument de compacité.

A.2.1 La classe $\mathscr{C}$

On définit d'abord la classe $\mathscr{C}$: elle est caractérisée par l'arbre infini d'états s_I avec $I \in \{s_0, s_1\}^*$ tels que depuis chaque état s_I, a peut jouer l'action ac_1 et entraîner l'exécution dans l'état $s_{I\cdot 1}$ ou jouer l'action ac_0 et entraîner l'exécution dans l'état $s_{I\cdot 0}$. On définit ainsi d'abord un modèle de Kripke $\mathscr{C}^*$: la relation d'accessibilité entre deux états s et s' y est définie comme l'existence d'une action qui permet à a de forcer une transition de s à s'.

La classe $\mathscr{C}$ est exactement l'ensemble des CGSs définies par le choix d'une valuation de p sur les états de $\mathscr{C}^*$. Le modèle $\mathscr{C}^*$ est représenté dans la Figure A.1. En interprétant les transitions

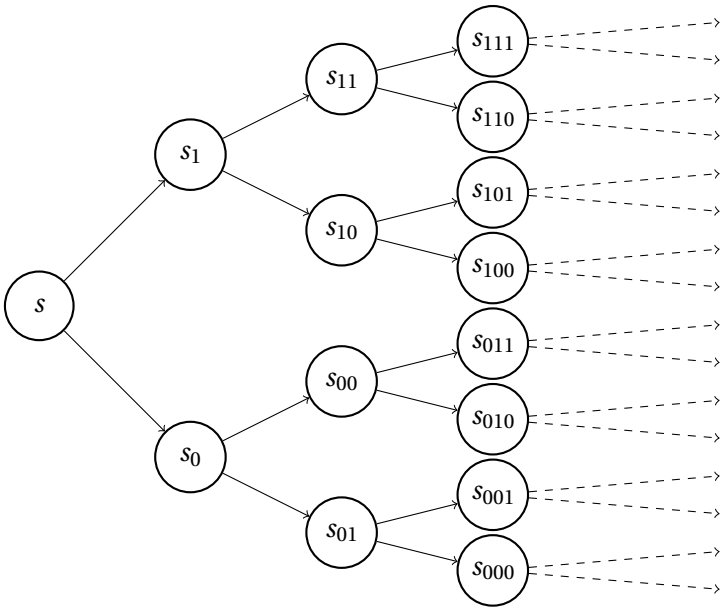

FIGURE A.1 – La classe de CGSs $\mathscr{C}$

possibles comme une relation entre les états, la classe $\mathscr{C}$ est aussi une classe de modèles de Tarski pour la logique classique. On l'axiomatise comme suit en logique de premier ordre (FOL) :

– Tout état a exactement deux transitions sortantes :

$$\mathrm{Ax}_1 \triangleq \forall y(\exists y', y''(R(y,y') \wedge R(y,y'') \wedge y' \neq y'' \wedge \forall z(R(y,z) \rightarrow (z = y' \vee z = y''))))$$

– Aucun état n'est en relation avec lui-même :

$$\mathrm{Ax}_2 \triangleq \forall y \neg R(y,y)$$

– Aucun état n'a de transition sortante vers s_0 :

$$\mathrm{Ax}_3 \triangleq \forall y(\neg R(y, s_0))$$

– Tout autre état a une seule transition entrante :

$$\mathrm{Ax}_4 \triangleq \forall y(y \neq s_0 \rightarrow (\exists y'(R(y', y) \wedge (\forall y''(R(y'', y) \rightarrow y'' = y')))))$$

– Définition de R^*, la clôture transitive de la relation R :

$$\mathrm{Ax}_5 \triangleq \forall y, y'(R^*(y, y') \leftrightarrow (R(y, y') \vee \exists s''(R(y, y'') \wedge R^*(y'', y'))))$$

– Tout état est accessible depuis s_0 :

$$\mathrm{Ax}_6 \triangleq \forall y(R^*(s_0, y))$$

– L'ensemble des antécédents d'un état par R^* est totalement ordonné :

$$\mathrm{Ax}_7 \triangleq \forall y, y', y''((R^*(y', y) \wedge R^*(y'', y)) \rightarrow (R^*(y', y'') \vee R^*(y', y'') \vee y' = y''))$$

On vérifie que la classe $\mathscr{C}$ est incluse dans l'ensemble des modèles de ces axiomes : Pour tout modèle $\mathscr{G} \in \mathscr{C}, \mathscr{G} \models_{\mathrm{FOL}} \mathrm{Ax}_{\mathscr{C}}$, où $\mathrm{Ax}_{\mathscr{C}} \triangleq \bigwedge_{i\in[1\ldots7]} \mathrm{Ax}_i$.

On vérifie par ailleurs le Lemme suivant :

Lemme A.1. *L'axiomatique $Ax_{\mathscr{C}}$ est complète pour le langage de FOL avec l'égalité et les symboles de relations R et R^* (on note FOL$(=, R, R^*)$) : tous ses modèles de cardinal $\aleph_0$ sont isomorphes.*

Démonstration. Soit $\mathscr{G}$ et $\mathscr{G}$' deux modèles de $\mathrm{Ax}_{\mathscr{C}}$ de cardinal $\aleph_0$ et soit $\mathscr{G}_\omega(\mathscr{G}, \mathscr{G}')$ le jeu d'Ehrenfeucht-Fraïssé associé à $\mathscr{G}$ et $\mathscr{G}$'. On décrit une stratégie gagnante pour II dans $\mathscr{G}_\omega(\mathscr{G}, \mathscr{G}')$. Dans ce qui suit, on indice par $\mathscr{G}$ ou $\mathscr{G}$' les interprétations des symboles du langage dans chacun de ces modèles. Soit la suite de choix $c_1, c_2, \ldots c_n$ pour I et soit S_n le plus petit ensemble tel que :

– $\{c_1, c_2, \ldots c_n\} \subseteq S_n, s \in S_n$
– pour tout $s', s'' \in S_n$, si t est tel que $R^*_{\mathscr{G}}(t, s'), R^*_{\mathscr{G}}(t, s'')$ et $\forall t'(R^*_{\mathscr{G}}(t, t') \rightarrow \neg R^*_{\mathscr{G}}(t, s') \vee \neg R^*_{\mathscr{G}}(t, s''))$ (t est le dernier prédécesseur commun de s et s' par R^*), alors $t \in S_n$.

Au long du jeu, le joueur II définit progressivement une fonction partielle des états de $\mathscr{G}$ dans les états de $\mathscr{G}$'. On note e cette fonction. À chaque tour de jeu n, le domaine de e est S_n. La fonction est enrichie :

– $S_0 = \{s_{\mathscr{C}}\}$ et $e(s_{\mathscr{C}}) = s_{\mathscr{G}}$
– Si $n \geqslant 1$, II complète e sur S_n de sorte que pour tout $s', s'' \in Sn$, pour tout $i \in \mathbb{N}, R^i_{\mathscr{G}}(e(s'), e(s''))$ ssi $R^i_{\mathscr{C}}(s', s'')$.

À chaque tour de jeu n, II joue $e(c_n)$. On vérifie facilement que la fonction e ainsi définie constitue une stratégie gagnante pour II.

□

L'axiomatique $\mathrm{Ax}_{\mathscr{C}}$ est donc $\aleph_0-$catégorique pour FOL$(=, R, R^*)$: tout modèle de cardinal $\aleph_0$ de $\mathrm{Ax}_{\mathscr{C}}$ est isomorphe à $\mathscr{C}^*$. Elle caractérise donc également l'ensemble des modèles étendant $\mathscr{C}^*$ par une valuation de p. On a donc le lemme suivant :

Lemme A.2. *Pour tout modèle $\mathscr{M}$ de FOL$(=, R, R^*, p)$,*

$$\mathscr{M} \models_{FOL(=,R,R^*,p)} Ax_{\mathscr{C}} \text{ ssi } \mathscr{M} \in \mathscr{C}$$

Démonstration. Immédiate □

A.2.2 Les formules $\mathsf{sl}(\mathsf{Cont}(a,p,n))$ et les modèles $\{\mathscr{C}_n\}_{n\in\mathbb{N}^*}$

La formule $Form(\mathsf{ContP}(a,p))$ énonce l'existence d'une multi-stratégie qui permet à a, à tout moment, de choisir entre p et $\neg p$ tout en retrouvant cette capacité intacte après ce choix. On peut donc énoncer $\mathsf{ContP}(a,p)$ comme la limite, quand n tend vers l'infini, de la propriété suivante : *a a une stratégie qui lui permet à tout moment de choisir entre p et $\neg p$ et d'exercer n fois ce choix*. On écrit cette propriété des CGSs $\mathsf{Cont}(a,p,n)$. Pour tout n, cette dernière propriété est caractérisable en SL, par une formule $\mathsf{sl}(\mathsf{Cont}(a,p,n))$. On définit $\mathsf{sl}(\mathsf{Cont}(a,p,n))$ par induction sur n.

Définition A.2 $(\mathsf{sl}(\mathsf{Cont}(a,p,n)), n\in\mathbb{N}^*)$**.**

- *La formule* $\mathsf{sl}(\mathsf{Cont}(a,p,1))$ *indique simplement que a dispose d'une multi-stratégie telle qu'à tout moment en suivant cette multi-stratégie, a peut choisir entre p et $\neg p$. Il s'agit donc de la formule 5.1 de la Section 5.1.1.1, dans laquelle* accès *est remplacé par p :*

$$\mathsf{sl}(\mathsf{Cont}(a,p,1)) \triangleq \langle\langle x_2\rangle\rangle(a,x_2)\Box(\langle\langle x_1\rangle\rangle(a,x_1)\mathsf{X}\,p \wedge \langle\langle x_1\rangle\rangle(a,x_1)\mathsf{X}\neg p)$$

- *Pour des raisons de clarté, on détaille ici le cas $n=2$. Pour satisfaire* $\mathsf{sl}(\mathsf{Cont}(a,p,2))$*, a doit avoir une multi-stratégie telle qu'à tout moment en suivant cette multi-stratégie, a peut choisir entre p et $\neg p$ et se garantir en même temps la capacité exprimée par* $\mathsf{sl}(\mathsf{Cont}(a,p,1))$ *:*

$$\begin{aligned}\mathsf{sl}(\mathsf{Cont}(a,p,2)) \triangleq \langle\langle x_3\rangle\rangle(a,x_3)\Box(&\\ &\langle\langle x_2\rangle\rangle(a,x_2)\mathsf{X}(p\wedge\Box(\langle\langle x_1\rangle\rangle(a,x_1)\mathsf{X}\,p\wedge\langle\langle x_1\rangle\rangle(a,x_1)\mathsf{X}\neg p))\\ &\quad\wedge\langle\langle x_2\rangle\rangle(a,x_2)\mathsf{X}(\neg p\wedge\Box(\langle\langle x_1\rangle\rangle(a,x_1)\mathsf{X}\,p\wedge\langle\langle x_1\rangle\rangle(a,x_1)\mathsf{X}\neg p)))\end{aligned}$$

- *De manière générale, pour $n\in\mathbb{N}$,* $\mathsf{sl}(\mathsf{Cont}(a,p,n))$ *utilise la formule ψ_n définie par récurrence sur n :*
  - $\psi_0 \triangleq \top$
  - *pour tout* $n\in\mathbb{N}$, $\psi_{n+1} \triangleq \Box(\langle\langle x_{n+1}\rangle\rangle(a,x_{n+1})\mathsf{X}(p\wedge\psi_n)\wedge\langle\langle x_{n+1}\rangle\rangle(a,x_{n+1})\mathsf{X}(\neg p\wedge\psi_n))$

  On a alors, pour tout $n\in\mathbb{N}$*,*

$$\mathsf{sl}(\mathsf{Cont}(a,p,n)) \triangleq \langle\langle x_{n+1}\rangle\rangle(a,x_{n+1})\psi_n$$

On remarque que pour tout $n\in\mathbb{N}$, $\mathsf{sl}(\mathsf{Cont}(a,p,n))\in \mathrm{SL}^1_1$.

On montre maintenant que pour tout $n\in\mathbb{N}^*$, $\mathsf{sl}(\mathsf{Cont}(a,p,n))$ et $\neg\mathsf{sl}(\mathsf{Cont}(a,p,n))$ sont toutes les deux satisfiables dans $\mathscr{C}$. Pour ceci, on définit un ensemble de modèles $\{\mathscr{C}_n\}_{n\in\mathbb{N}^*}$ dans $\mathscr{C}$ tels que pour tout $n\in\mathbb{N}^*, \mathscr{C}_n, s \models_{\mathrm{SL}} \mathsf{sl}(\mathsf{Cont}(a,p,n))$ et $\mathscr{C}_n, s \not\models_{\mathrm{SL}} \mathsf{sl}(\mathsf{Cont}(a,p,n+1))$. Il y a plusieurs moyens de définir un tel ensemble de modèles. On décrit ici notre choix. Il est guidé exclusivement par des motifs de simplicité.

On fait d'abord l'observation qu'un état s_I d'un modèle $\mathscr{G}$ de $\mathscr{C}$ est tel que

$$\mathscr{G}, s_I \models \langle\langle x\rangle\rangle(a,x)\mathsf{X}(p)\wedge\langle\langle x\rangle\rangle(a,x)\mathsf{X}(\neg p)$$

ssi p est vrai dans exactement un des successeurs de s_I : $p\in val(s_{I\cdot 1})$ ssi $p\notin val(s_{I\cdot 0})$. On dit d'un tel état qu'il est *libre*.

Le modèle $\mathscr{C}_1$ comporte un chemin partant de s et composé d'états libres. Pour chaque état de ce chemin, un de ses successeurs est également libre et l'autre est la racine d'un arbre sans état libre.

Le modèle $\mathscr{C}_1$ est représenté dans la Figure A.2 : le nom des états n'est pas écrit pour des raisons de place : depuis tout état s_I, la flèche vers la droite représente la transition vers $s_{I\cdot 1}$ et la flèche vers la gauche représente la transition vers $s_{I\cdot 0}$. Un état est représenté en noir ssi il satisfait la propriété p. Les états libres sont ceux atteints par une succession d'actions ac_1 : pour tout état s_I, $p \in val_1(s_I)$ ssi $I \in 1^* \cdot 0$.

On a donc :

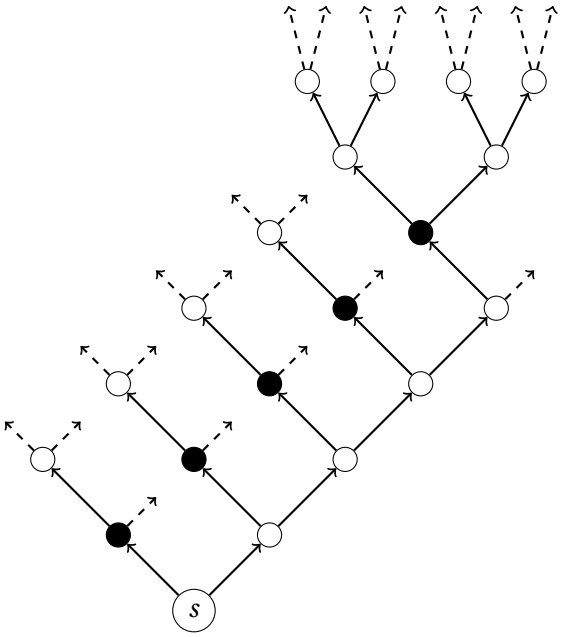

FIGURE A.2 – Le modèle $\mathscr{C}_1$

On généralise maintenant la construction à $\mathscr{C}_{n+1}$, pour $n \in \mathbb{N}$. On appelle s_n l'état initial de $\mathscr{C}_n$. Depuis l'état s_{n+1} de $\mathscr{C}_{n+1}$, il y a un chemin d'états libres $(s_{1^k})_{k\in\mathbb{N}}$ où p n'est pas satisfaite et tel que pour tout $k \in \mathbb{N}$, $p \in val(s_{1^k\cdot 0})$ et les successeurs de s_{1^k} satisfont les mêmes formules que l'état s_n de $\mathscr{C}_n$. La Figure A.3 représente $\mathscr{C}_{n+1}$. Depuis chacun des états dans $\{s_{1^k\cdot 0}\}_{k\in\mathbb{N}}$:

- une transition conduit à une copie de $\mathscr{C}_n$,
- une transition conduit à une copie de $\mathscr{C}_{n,(p)}$, le modèle issu de $\mathscr{C}_n$ en changeant la valuation en s_n, soit tel que $val(s_n) = \{p\}$.

On a donc, pour tout état s_I, $p \in val_n(s_I)$ ssi $I \in \{(1^* \cdot 0)^k\}_{k\leqslant n}$.

On définit également le modèle $\mathscr{C}_\infty$ dans $\mathscr{C}$. $\mathscr{C}_\infty$ est la limite de la suite $\mathscr{C}_n$ quand n tend vers l'infini : Pour tout état s_I, $p \in val_i(s_I)$ ssi $I \in (1^* \cdot 0)^*$. Ceci équivaut à : p est vrai exactement dans tous les états s_I tels que I se termine par 0.

Lemme A.3. *On vérifie facilement les assertions suivantes :*

1. *Pour tout* $n \in \mathbb{N}, \mathscr{C}_n \models_{SL} \mathsf{sl}(\mathsf{Cont}(a, p, n))$
2. *Pour tout* $n \in \mathbb{N}, \mathscr{C}_n \not\models_{SL} \mathsf{sl}(\mathsf{Cont}(a, p, n+1))$

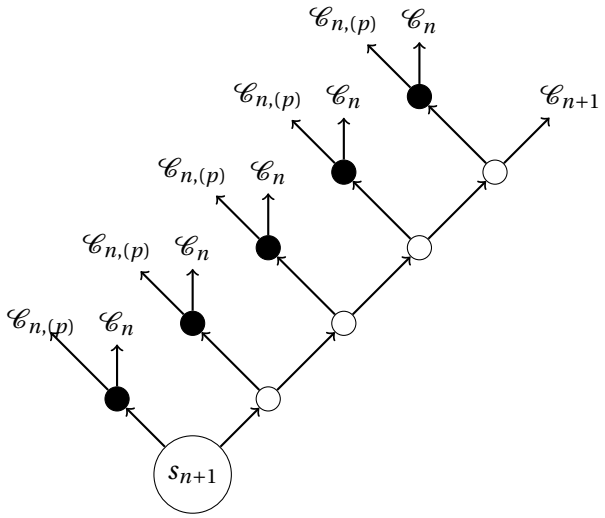

FIGURE A.3 – Le modèle $\mathscr{C}_{n+1}$, pour $n \in \mathbb{N}$

3. *Pour tout* $n \in \mathbb{N}, \mathscr{C}_n \not\models_{USL^{lp}} Form(\mathsf{ContP}(a,p))$
4. *Pour tout* $n \in \mathbb{N}, \mathscr{C}_\infty \models_{SL} \mathsf{sl}(\mathsf{Cont}(a,p,n))$
5. $\mathscr{C}_\infty \models_{USL^{lp}} Form(\mathsf{ContP}(a,p))$
6. *Dans* $\mathscr{C}$ *,* $Form(\mathsf{ContP}(a,p))$ *est équivalent à l'ensemble des* $\mathsf{sl}(\mathsf{Cont}(a,p,n))$ *pour* $n \in \mathbb{N}$.

Démonstration. (Résumé) Seul le dernier item de la proposition nécessite des éléments de preuve. Dans $\mathscr{C}$, $Form(\mathsf{ContP}(a,p))$ est vraie dans un état s_I ssi, pour toute fonction f de $\mathbb{N}$ dans $\{0,1\}$, a peut forcer l'exécution λ depuis s_I à satisfaire p exactement dans les états λ_i tels que $f(i) = 1$.

Et pour tout $n \in \mathbb{N}$, pour tout modèle $\mathscr{G} \in \mathscr{C}$, pour tout état s_I de $\mathscr{G}$, s_I satisfait $\mathsf{sl}(\mathsf{Cont}(a,p,n))$ ssi, pour toute fonction f de $\mathbb{N}$ dans $\{0,1\}$ telle qu'il y a au plus n valeurs i dans $\mathbb{N}$ telle que $f_n(i) = 1$, a peut forcer l'exécution λ depuis s_I à satisfaire p exactement dans les états λ_i tels que $f_n(i) = 1$.

Donc , pour tout modèle $\mathscr{G} \in \mathscr{C}$, pour tout état s_I de $\mathscr{G}$, s_I satisfait l'ensemble des formules $\mathsf{sl}(\mathsf{Cont}(a,p,n))$ pour $n \in \mathbb{N}$ ssi, pour toute fonction f de $\mathbb{N}$ dans $\{0,1\}$, a peut forcer l'exécution λ depuis s_I à satisfaire p exactement dans les états λ_i tels que $f(i) = 1$. □

On s'intéresse maintenant au fragment SL_1^1 de SL.

A.2.3 Plongement de SL_1^1 dans Σ_1^1 dans la classe $\mathscr{C}$

Dans cette section, on opère une traduction de SL_1^1 dans Σ_1^1, qui préserve l'équivalence dans les modèles de $\mathscr{C}$. Pour tout énoncé φ de SL_1^1, on définit une formule $\Sigma_1^1(\varphi)(y)$ de Σ_1^1, à une variable libre y et telle que pour tout $\mathscr{G} \in \mathscr{C}, \mathscr{G}, s \models_{\mathrm{SL}} \varphi$ ssi $\mathscr{G}, [y \rightarrowtail s] \models_{\Sigma_1^1} \Sigma_1^1(\varphi)(y)$. Pour cette traduction, on définit d'abord une propriété sur les ensembles d'états : cette propriété a en paramètre une stratégie S et un état y. Elle caractérise l'ensemble des états qui est parcouru si a joue la stratégie S à partir de l'état y.

Cette caractérisation utilise donc la propriété intermédiaire *représenter une stratégie*. Dans SL, une stratégie pour a est une fonction de *Track* dans *Ac*. Dans la classe $\mathscr{C}$, comme d'une part chaque état détermine un unique préfixe d'exécution à partir de l'état s (car $\mathscr{C}$ est un arbre) et comme d'autre part, une action de la part de a détermine un successeur depuis chaque état, on

peut caractériser une stratégie comme une relation binaire : une stratégie est une relation binaire incluse dans R et telle que chaque état est lié à exactement un autre état. Formellement :

Définition A.3 (Caractérisation des stratégies dans $\mathscr{C}$)**.** *Soit S un prédicat binaire dans un langage de Σ^1_1, la formule Strat(S) est définie comme suit :*

$$\mathit{Strat}(S) \triangleq \forall y, y'(S(y,y') \rightarrow R(y,y')) \wedge \forall y \exists y'(S(y,y') \wedge \forall y''(S(y,y'') \rightarrow y' = y''))$$

On définit maintenant la formule pour *représenter l'unique exécution issue de y quand a joue la stratégie représentée par S* : pour un prédicat unaire E, il s'agit de caractériser un ensemble qui contient une suite infinie d'états du modèle commençant par y, dont la relation de succession est incluse dans S et qui est totalement ordonné par R^*. On formalise cette propriété comme suit :

Définition A.4 (Caractérisation des issues d'une stratégie dans $\mathscr{C}$)**.** *Soit S un prédicat binaire et E un prédicat unaire dans un langage de Σ^1_1, la formule Issue(E,S,y) est définie comme suit :*

$$\begin{aligned}\mathit{Issue}(E,S,y) \triangleq\ & E(y) \wedge \forall z(E(z) \rightarrow \exists z'(S(z,z') \wedge E(z'))) \\ & \wedge \forall z, z', ((E(z) \wedge E(z')) \rightarrow (R^*(z,z') \vee R^*(z',z))) \\ & \wedge \forall z(E(z) \rightarrow (\exists z'(E(z') \wedge R(z',z)) \leftrightarrow z \neq y))\end{aligned}$$

Ces deux caractérisations sont valables modulo l'axiomatique $\mathrm{Ax}_{\mathscr{C}}$, *i.e.* dans la classe de modèle $\mathscr{C}$. On peut maintenant donner le plongement de SL^1_1 dans Σ^1_1. On ne l'utilisera que modulo $\mathrm{Ax}_{\mathscr{C}}$, *i.e.* dans la classe de modèles $\mathscr{C}$ (voir le Lemme A.4). Il utilise en paramètre un ensemble de symboles de relations binaires St, qui encode une assignation de stratégies, et un symbole de relation unaire E, qui encode l'engagement de a v.a.v. d'une stratégie.

Définition A.5 (Plongement de SL^1_1 dans Σ^1_1)**.** *Soient St un ensemble de symboles de relations binaires, E un symbole de relation unaire et y une variable. Alors pour toute formule φ dans SL^1_1, la formule $\Sigma^1_1(\varphi)_{St,E}(y)$ est donnée dans le Tableau A.1 :*

$$\begin{array}{rcl}
\Sigma^1_1(p)_{St,E}(y) & \triangleq & p(y) \\
\Sigma^1_1(\neg p)_{St,E}(y) & \triangleq & \neg p(y) \\
\Sigma^1_1(\varphi_1 \vee \varphi_2)_{St,E}(y) & \triangleq & \Sigma^1_1(\varphi_1)_{St,E}(y) \vee \Sigma^1_1(\varphi_2)_{St,E}(y) \\
\Sigma^1_1(\varphi_1 \wedge \varphi_2)_{St,E}(y) & \triangleq & \Sigma^1_1(\varphi_1)_{St,E}(y) \wedge \Sigma^1_1(\varphi_2)_{St,E}(y) \\
\Sigma^1_1(\langle\langle x\rangle\rangle\varphi)_{St,E}(y) & \triangleq & \exists S_x(\mathit{Strat}(St_x) \wedge \Sigma^1_1(\varphi)_{S\cup\{S_x\},E}(y)) \\
\Sigma^1_1((a,x)\varphi)_{St,E}(y) & \triangleq & \exists E_{S_x,y}\, \mathit{Issue}(E_{S_x,y},S,y) \wedge \Sigma^1_1(\varphi)_{St_x,E_{S_x,y}}(y) \\
\Sigma^1_1(\mathsf{X}\varphi)_{St,E}(y) & \triangleq & \forall y'((E(y') \wedge R(y,y')) \rightarrow \Sigma^1_1(\varphi)_{St,E}(y')) \\
\Sigma^1_1(\varphi_1 \mathsf{U} \varphi_2)_{St,E}(y) & \triangleq & \exists y'(E(y') \wedge \Sigma^1_1(\varphi_2)_{St,E}(y') \\
 & & \wedge(\forall y''((E(y'') \wedge R^*(y,y'') \wedge R^*(y',y'')) \\
 & & \rightarrow \Sigma^1_1(\varphi_1)_{St,E}(y'')))) \\
\Sigma^1_1(\Box\varphi)_{St,E}(y) & \triangleq & \forall y'((E(y') \wedge R^*(y,y')) \rightarrow \Sigma^1_1(\varphi)_{St,E}(y')) \\
\Sigma^1_1(\varphi_1 \mathsf{R} \varphi_2)_{St,E}(y) & \triangleq & \Sigma^1_1(\Box\varphi_2)_{St,E}(y) \\
 & & \vee(\Sigma^1_1(\Box\varphi_2 \mathsf{U} (\varphi_2 \wedge \varphi_1))_{St,E})
\end{array}$$

TABLE A.1 – Plongement de SL^1_1 dans Σ^1_1

En particulier, si φ est un énoncé, la traduction $\Sigma^1_1(\varphi)_{St,E}(y)$ ne dépend pas de St ni de E. On peut alors écrire $\Sigma^1_1(\varphi)(y)$ pour $\Sigma^1_1(\varphi)_{St,E}(y)$, pour n'importe quels St et E.

On remarque par ailleurs que pour toute formule φ de SL_1^1, aucune occurence de quantificateur du second ordre dans $\Sigma_1^1(\varphi)(y)$ ne se trouve dans la portée d'une négation $\neg$ ni en antécédent d'une flèche $\rightarrow$. $\Sigma_1^1(\varphi)(y)$ est donc bien une formule de Σ_1^1. On a alors le lemme suivant :

Lemme A.4. *Pour toute formule φ de SL_1^1, pour toute CGS $\mathscr{G} \in \mathscr{C}$ et pour tout état s de $\mathscr{G}$:*

$$\mathscr{G}, s \models_{SL} \varphi \text{ ssi } \mathscr{G}, [y \mapsto s] \models_{\Sigma_1^1} \Sigma_1^1(\varphi)(y)$$

Démonstration. Immédiate, par récurrence sur la complexité de φ. □

A.2.4 $\mathsf{sl}(Form(\mathsf{ContP}(a,p))) \notin \mathbf{SL}_1^1$

On peut maintenant poser la proposition principale de cette section :

Proposition A.5. *Il n'y a pas de formule de SL_1^1 équivalente à $Form(\mathsf{ContP}(a,p))$.*

Démonstration. On procède par l'absurde en utilisant un argument de compacité. Supposons qu'il y ait une formule $\mathsf{sl}(Form(\mathsf{ContP}(a,p)))$ équivalente à $Form(\mathsf{ContP}(a,p))$ dans SL_1^1. Alors il y a une formule $\Sigma_1^1(\mathsf{sl}(Form(\mathsf{ContP}(a,p))))(y)$ de Σ_1^1 qui lui est équivalente. Soit maintenant $N \subset \mathbb{N}$ un ensemble fini d'indice de $\mathbb{N}$ et soit $\{\mathsf{sl}(\mathsf{Cont}(a,p,n))\}_{n\in N}$. N a un élément maximal *sup(N)* et, d'après le Lemme A.3, pour tout $n \in N$:

$$\mathscr{C}_{sup(N)}[y \mapsto s] \models_{\Sigma_1^1} \Sigma_1^1(\mathsf{sl}(\mathsf{Cont}(a,p,n)))(y)$$

et

$$\mathscr{C}_{sup(N)}, [y \mapsto s] \models_{\Sigma_1^1} \neg\Sigma_1^1(Form(\mathsf{ContP}(a,p)))(y)$$

On a donc que, pour tout ensemble de formules $\{\mathsf{sl}(\mathsf{Cont}(a,p,n))\}_{n\in N}$ tel que $n \in \mathscr{P}^{<\omega}\mathbb{N}$.

$$\{\mathrm{Ax}_{\mathscr{C}}\} \cup \{\Sigma_1^1(\mathsf{sl}(\mathsf{Cont}(a,p,n)))(y)\}_{n\in N\subset\mathbb{N}} \cup \{\neg\Sigma_1^1(\mathsf{sl}(Form(\mathsf{ContP}(a,p))))(y)\}$$

est satisfiable. Alors, par compacité pour la satisfiabilité des formules de Σ_1^1 :

$$\{\mathrm{Ax}_{\mathscr{C}}\} \cup \{\Sigma_1^1(\mathsf{sl}(\mathsf{Cont}(a,p,n)))(y)\}_{n\in\mathbb{N}} \cup \{\neg\Sigma_1^1(\mathsf{sl}(Form(\mathsf{ContP}(a,p))))(y)\}$$

est satisfiable. Par le Lemme A.2, cette formule a donc un modèle dans $\mathscr{C}$, ce qui est en contradiction avec le Lemme A.3. □

A.3 Conjecture : si $\mathsf{ContP}(a,p)$ est caractérisable dans SL, alors elle est caractérisable dans $\mathbf{SL}_1^1$

On conjecture que s'il y a une formule de SL équivalente à $Form(\mathsf{ContP}(a,p))$, alors cette formule est également équivalente à une formule de SL_1^1 (on simplifie en écrivant $\mathsf{sl}(Form(\mathsf{ContP}(a,p))) \in \mathrm{SL}_1^1$). On s'appuie pour ceci sur la notion d'expansivité, définie ci-après.

A.3.1 Simulations et expansivité

On appelle *expansive* une propriété de modèles de Kripke qui est stable par *simulation*. On définit d'abord cette notion :

Définition A.6 (Simulation)**.** *Une simulation d'un modèle de Kripke $\mathcal{K} = \langle K, s_0, R, val\rangle$ dans un autre modèle de Kripke $\mathcal{K}' = \langle G', s'_0, R', val'\rangle$ est une fonction sim des états de $\mathcal{K}$ dans les états de $\mathcal{K}'$ qui est telle que $sim(s_0) = s'_0$ et qui satisfait l'*harmonie atomique *et la clause* zig*, qui permet un traçage des exécutions de $\mathcal{K}$ dans $\mathcal{K}'$:*

- *Harmonie atomique : si s et s' sont deux états de $\mathcal{K}$ et $\mathcal{K}'$ et si $sim(s) = s'$ alors s et s' satisfont les mêmes propositions atomiques.*
- *Zig : si s et s' sont deux états de $\mathcal{K}$ et $\mathcal{K}'$ et si $sim(s) = s'$ alors pour tout état s_1 de $\mathcal{K}$ t.q. $R(s, s_1)$ il existe un état s'_1 de $\mathcal{K}'$ t.q. $sim(s_1, s'_1)$ et $R'(s', s'_1)$.*

On dit alors que $\mathcal{K}'$ simule *$\mathcal{K}$.*

Le langage SL($\{a\}$) est très proche de CTL*. La principale différence est que SL($\{a\}$) quantifie sur des stratégies au lieu de quantifier sur des exécutions. Or dans SL($\{a\}$), une stratégie appliquée à partir d'un état définit de manière unique une exécution. On peut donc considérer des CGSs pour SL($\{a\}$) comme des modèles de Kripke. Similairement, on peut interpréter SL($\{a\}$) dans des modèles de Kripke $\mathcal{K} = \langle K, R, val\rangle$ en posant : pour tous états s et s' de $\mathcal{K}$, $R(s, s')$ ssi il existe une action ac telle que si a joue ac à partir de s alors l'exécution se poursuit en s'. On peut donc parler de simulations d'une CGS pour SL($\{a\}$) par une autre.

On définit alors la notion de propriété *expansive* :

Définition A.7. *Soit P une propriété sur les modèles de Kripke. On dit de P que c'est une propriété* expansive *ssi pour tout modèle $\mathcal{K}$ qui a cette propriété, tout modèle $\mathcal{K}'$ qui simule $\mathcal{K}$ a aussi cette propriété.*

A.3.2 Conjecture : $\mathsf{sl}(Form(\mathsf{ContP}(a, p))) \in \mathbf{SL}^1_1$

On observe que la propriété $Form(\mathsf{ContP}(a, p))$ est expansive. En effet, tout modèle $\mathcal{K}$ est modèle de $Form(\mathsf{ContP}(a, p))$ ssi :

- Soit $\mathcal{K}$ simule $\mathscr{C}_\infty$
- Soit $\mathcal{K}$ simule le modèle $\mathscr{C}_{\infty,(p)}$ obtenu de $\mathscr{C}_\infty$ par la modification suivante : la proposition p est vraie dans l'état initial.

La Conjecture 5.3 repose alors sur la conjecture suivante :

Conjecture A.6. *Soit φ une formule de SL* ($\{a\}$)*. Si φ caractérise une propriété expansive, alors φ est équivalente à une formule de SL^1_1.*

La Conjecture A.6 repose elle-même sur l'intuition que pour toute formule φ de SL, pour toute occurence de sous-formule ψ^i de φ qui est universellement quantifiée dans φ, si $\neg\psi$ est satisfiable, alors on peut étendre tout modèle de ψ en lui *greffant* un contre-modèle de ψ. Les tentatives pour exploiter cette idée dans une preuve formelle de la Conjecture A.6 menées dans le cadre de nos travaux n'ont cependant pas abouti.

Annexe B

Décision du *model-checking* pour USL$^\infty$: preuve détaillée

Sommaire

On prouve ici l'existence d'un algorithme de décision pour le problème de *model-checking* d'USL$^\infty$. La preuve est adaptée de [MMPV11, BDCLLM09, DCL11]. Comme pour SL, cet algorithme tourne en temps NONELÉMENTAIRE sur la longueur de la formule et POLYNOMIAL sur la taille du modèle.

Pour tout énoncé $\varphi \in \text{USL}^\infty$, pour toute CGS$^\infty$ $\mathcal{G}$, on construit un *automate d'arbres alternant avec condition d'acceptation paritaire* $\mathcal{A}_{\mathcal{G}}[\varphi, \varnothing]$. Cet automate accepte exactement les objets (des arbres étiquetés) qui encodent les couples du contexte $\chi_\varnothing = \langle \alpha_\varnothing, \gamma_\varnothing \rangle$ et d'un état s de $\mathcal{G}$ tels que $\mathcal{G}, \chi_\varnothing, s \models_{\text{USL}^\infty} \varphi$ (on appelle un tel objet un *déroulement contexte-état* pour $\chi_\varnothing$ et s). S'il n'est pas vrai que $\mathcal{G}, \chi_\varnothing, s \models_{\text{USL}^\infty} \varphi$, alors le langage accepté par $\mathcal{A}_{\mathcal{G}}[\varphi, \varnothing]$ est vide.

La preuve suit les étapes suivantes :

- Dans la Section B.1, on définit les *arbres étiquetés* et on introduit des notations les concernant.
- Les *automates d'arbres alternants* sont introduits dans la Section B.2,
- La Section B.3 introduit les *conditions d'acceptation paritaire* sur ces automates. Ces trois première étapes donnent le matériel nécessaire pour la construction de l'automate $\mathcal{A}_{\mathcal{G}}[\varphi, \varnothing]$. Il reste à le construire effectivement.
- Pour ce faire, on introduit d'abord des lemmes préliminaires, qui définissent des transformations utiles sur ces automates (Section B.4).
- Ensuite, on introduit les *déroulements contexte-état* (Section B.5).
- L'algorithme de construction pour $\mathcal{A}_{\mathcal{G}}[\varphi, \varnothing]$ est donné dans la Section B.6.

– Grâce à cet automate, on définit un *automate d'arbre non-déterministe à condition d'acceptation paritaire* $\mathcal{N}_{\mathcal{G}}[\varphi]$ t.q. $\mathcal{G} \models_{\text{USL}^\infty} \varphi$ ssi $\mathcal{L}(\mathcal{N}_{\mathcal{G}}[\varphi]) = \varnothing$, où $\mathcal{L}(\mathcal{N}_{\mathcal{G}}[\varphi])$ est le langage accepté par $\mathcal{N}_{\mathcal{G}}[\varphi]$. On calcule ensuite le temps d'exécution pour le test du vide sur $\mathcal{L}(\mathcal{N}_{\mathcal{G}}[\varphi])$ et on conclut la preuve dans la Section B.7.

B.1 Arbres étiquetés

On définit d'abord la notion d'*arbres étiquetés*, et on apporte les notations nécessaires.

Définition B.1 (Arbres étiquetés)**.** *Soient deux ensembles Eti et G. Un* G-arbre *Eti*-étiqueté *est un couple* $\mathcal{T} = \langle T, l \rangle$ *t.q. :*

– $T \subseteq G^*$ *est un ensemble non vide de mots sur l'alphabet G. Il satisfait la propriété suivante : pour tout mot non vide* $n = m \cdot s \in T$ *t.q.* $m \in G^*$ *et* $s \in G$, $m \in T$*. Autrement dit, T est clos par toute opération sélectionnant des préfixes de ses éléments.*

– *l est une fonction d'étiquetage, qui associe une étiquette dans Eti à chacun des éléments de T.*

Un ensemble non vide de mots finis qui satisfait la première des conditions de cette définition est aussi appelé un G-arbre.

Soit un arbre étiqueté $\mathcal{T} = \langle T, l \rangle$. Pour tout $n \in T$, on appelle *ensemble des directions depuis n dans* $\mathcal{T}$, et on écrit $\text{dir}_n(T)$, l'ensemble des éléments s dans G t.q. $n \cdot s$ est dans T : $\text{dir}_n(T) = \{s \in \text{G} \mid n \cdot s \in T\}$.

L'*ensemble des séquences infinies dans T* est l'ensemble

$$Séq_{\mathcal{T}} = \{s_0 \cdot s_1 \cdots \in \text{G}^\omega \mid \forall i \in \mathbb{N}, s_0 \cdot s_1 \cdots s_i \in T\}$$

Soit $\rho = (s_i)_{i \in \mathbb{N}}$ une de ces séquences, on écrit $l(\rho)$ la séquence infinie sur *Eti* donnée par la fonction d'étiquetage $\mathcal{T}$: $l(\rho) = (l(s_i))_{i \in \mathbb{N}}$. On écrit aussi $\text{Inf}(l(\rho))$ pour l'ensemble des éléments dans G qui apparaissent infiniment souvent dans $l(\rho)$.

Soient maintenant deux ensembles d'étiquettes disjoints Eti_1 et Eti_2, soit $Eti = Eti_1 \times Eti_2$, et soit $\mathcal{T} = \langle T, l \rangle$ un G−arbre *Eti*-étiqueté. On écrit $l(n) = (l_1(n), l_2(n))$ où $l_1(n)$ est le composant de $l(n)$ dans Eti_1 et $l_2(n)$ est le composant de $l(n)$ dans Eti_2. Pour tout $i \in \{1, 2\}$, on désigne le G−arbre Eti_i-étiqueté $\mathcal{T}_i = \langle T, l_i \rangle$ par $\text{Proj}_{Eti_i}(\mathcal{T})$.

B.2 Automates d'arbres alternants

Les automates d'arbres alternants sont une généralisation des automates d'arbres non-déterministes (précisément, l'ensemble des automates d'arbres alternants est la clôture de l'ensemble des automates d'arbres non-déterministes par l'opération de complémentation). Pour définir la fonction de transition de tels automates, on a besoin de définir les *formules booléennes positives* sur un ensemble de propositions atomiques :

Définition B.2 (Formules booléennes positives)**.** *Soit un ensemble de propositions atomiques At, l'ensemble des* formules booléennes positives *sur At est généré par la grammaire suivante :*

$$\varphi := p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \top \mid \bot$$

où $p \in At$.

La satisfaction d'une formule booléenne positive est définie de manière classique sur les valuations $val : At \to \{\top, \bot\}$. On dit d'un sous-ensemble At' de At qu'il satisfait une formule booléenne positive φ ssi la valuation $val_{At'}$, définie par $val_{At'}(p) = \top$ ssi $p \in At'$, satisfait φ.

Définition B.3 (Automate d'arbres alternant)**.** *Soient deux ensembles finis G et Eti. Un* G-automate d'arbres alternant sur *Eti (on écrit aussi $\langle G, Eti\rangle - ATA$, ou ATA quand il n'y a pas d'ambiguité), est un* 4*-uplet $\mathscr{A} = \langle Q, q_0, \tau, Acc\rangle$ t.q. :*

- *Q est un ensemble fini d'états,*
- *$q_0 \in Q$ est l'état initial,*
- *$\tau : (Q \times Eti) \to PBF(G \times Q)$ est la fonction de transitions,*
- *$Acc : Q^\omega \to \{\top, \bot\}$ est une* condition d'acceptation*. Elle est définie comme une fonction de l'ensemble des exécutions dans $\{\top, \bot\}$ (voir la Section B.3).*

Sous ces notation, G est appelé l'ensemble des directions *de l'automate $\mathscr{A}$. L'ensemble des automates d'arbres non déterministes sur Eti (on écrit aussi $\langle G, Eti\rangle - NTAs$, ou NTAs) peut être défini comme un sous-ensemble des $\langle G, Eti\rangle - ATAs$: un $\langle G, Eti\rangle - NTA$ est un $\langle G, Eti\rangle - ATA$ dans lequel la fonction de transition ne définit que des formules qui, quand elles sont écrites en forme normale disjonctive, associent exactement une transition à chaque direction dans G dans chacun de ses composants disjonctifs. Formellement, pour tout état q et pour toute $d \in Eti$, la transition a la forme suivante :*

$$\tau(q, d) = \bigvee_{i \in I(q)} \left(\bigwedge_{s \in G} (s, q_{i,s}) \right)$$

où $I(q)$ est l'ensemble fini des indices et les $q_{i,s}$ sont des états dans Q.

B.3 Exécutions et conditions d'acceptation paritaire

On traite ici des exécutions d'un *ATA* ou d'un *NTA*, et on introduit les *conditions d'acceptations paritaires.*

Définition B.4 (Exécutions)**.** *Soit un $\langle G, Eti\rangle - ATA\ \mathscr{A} = \langle Q, q_0, \tau, Acc\rangle$, et soit $\mathscr{T} = \langle T, l\rangle$ un $G-$arbre Eti-étiqueté. Une* exécution *de $\mathscr{A}$ sur $\mathscr{T}$ est un $G \times Q-$arbre $\mathscr{E}$ t.q. pour tout nœud $e = (s, q) = (s_0 s_1 \dots s_n, q_0 q_1 \dots q_n)$ de $\mathscr{E}$, on a :*

- *$s \in T$,*
- *L'ensemble $dir_e(E) = \{(s^0_{n+1}, q^0_{n+1}), (s^1_{n+1}, q^1_{n+1}), \dots, (s^k_{n+1}, q^k_{n+1})\} \subseteq G \times Q$ satisfait $\tau(q_n, l(s_n))$.*

Une exécution $\mathscr{E}$ est acceptante *ssi pour toutes les séquences infinies $\lambda \in (G \times Q)^\omega$ dans $Séq_{\mathscr{E}}$, on a $Acc(\lambda) = \top$. Un arbre $\mathscr{T}$ est acceptant pour $\mathscr{A}$ ssi il y a une exécution acceptante de $\mathscr{A}$ sur $\mathscr{T}$. Cette définition vaut en particulier pour les NTAs.*

Dans le reste de cette preuve on utilise des *conditions d'acceptation paritaires* :

Définition B.5 (Conditions d'acceptation paritaires)**.** *Une condition d'acceptation paritaire pour un automate $\mathscr{A}$ est identifiée par une chaîne de sous-ensembles inclus dans Q, l'ensemble des états de $\mathscr{A}$: $F_1 \subseteq \dots \subseteq F_k = Q$, t.q. $k \in \mathbb{N}$. La condition d'acceptation est alors donnée, sur une séquence infinie $\lambda \in (G \times Q)^\omega$, par : $Acc(\lambda) = \top$ ssi,*

$$min\{l \leqslant k \mid F_k \cap Inf(Proj_Q(\lambda)) \neq \varnothing \text{ est pair}\}$$

*Sous cette notation, le nombre k est appelé l'*indice *de l'automate (on note $id_{\mathscr{A}}$). Et la taille de $\mathscr{A}$, $|\mathscr{A}|$, est donnée par son nombre d'états. Un ATA avec une telle condition d'acceptation est appelé un* automate d'arbres alternant paritaire *(un APT). Similairement, un NTA avec une condition d'acception paritaire est un* automate d'arbres non-déterministe paritaire *(et on écrit NPT).*

B.4 Lemmes préliminaires

Ces définitions étant données, avant d'en venir aux déroulements de contextes et d'états comme entrées pour ces automates, on présente deux lemmes sur les *ATAs* qui seront utilisés plus loin dans la preuve. Tout d'abord, la classe des langages qui sont reconaissables par des *ATAs* est stable par les opérations d'intersection et de complémentation :

Lemme B.1 (Intersection et complémentation)**.** *[MS87, MS95]. Soient $\mathscr{A}$ un $\langle G, Eti\rangle - APT$ qui accepte le langage $\mathscr{L}(\mathscr{A})$ et $\mathscr{B}$ un $\langle G, Eti\rangle - APT$ qui accepte le langage $\mathscr{L}(\mathscr{B})$. Alors :*

- *Il y a un $\langle G, Eti\rangle - APT$ $\mathscr{C}$ qui accepte le langage $\mathscr{L}(\mathscr{A}) \cap \mathscr{L}(\mathscr{B})$. $|\mathscr{C}|$ est bornée par $|\mathscr{A}| + |\mathscr{B}| + 1$ et $id_{\mathscr{C}}$ est borné par $max(id_{\mathscr{A}}, id_{\mathscr{B}})$.*
- *Il y a un $\langle G, Eti\rangle - APT$ $\mathscr{D}$ qui accepte le langage $\overline{\mathscr{L}(\mathscr{A})}$, le complément du langage $\mathscr{L}(\mathscr{A})$. $\mathscr{D}$ a la même taille que $\mathscr{A}$ et pour indice $id_{\mathscr{D}} = id_{\mathscr{C}} + 1$.*

Le lemme suivant établit que pour tout *APT*, il y a un *NPT* équivalent.

Lemme B.2 (Non-déterminisation)**.** *[MS95] Soit $\mathscr{A}$ un $\langle G, Eti\rangle - APT$. Alors il y a un $\langle G, Eti\rangle - NPT$ $\mathscr{N}$ qui accepte le même langage que $\mathscr{A}$ et tel que $|\mathscr{N}| = 2^{\bigcirc(|\mathscr{A}|.id_{\mathscr{A}}.log(|\mathscr{A}|))}$ et $id_{\mathscr{N}} = \bigcirc(|\mathscr{A}|.id_{\mathscr{A}}.log(|\mathscr{A}|))$.*

Ce dernier lemme est utilisé à deux reprises :

- La procédure de construction pour $\mathscr{A}_{\mathscr{G}}[\varphi, \varnothing]$ est définie par récurrence sur la structure de φ : pour chacune des occurences de sous-formules ψ^i de φ, elle construit un automate.Dans cette construction, le Lemme B.2 est utilisé dans l'étape pour $\langle\langle x\rangle\rangle$.
- Suite à la construction de $\mathscr{A}_{\mathscr{G}}[\varphi, \varnothing]$, l'opération qui permet d'obtenir $\mathscr{N}_{\mathscr{G}}[\varphi]$ à partir de $\mathscr{A}_{\mathscr{G}}[\varphi, \varnothing]$ est appelée la *projection sur une direction.* Elle ne peut être appliquée que sur des *NPT*. Elle nécessite donc au préalable une application du Lemme B.2 sur $\mathscr{A}_{\mathscr{G}}[\varphi, \varnothing]$.

B.5 Entrées des automates utilisés

Pour tout énoncé φ et pour toute occurrence de sous-formule ψ^i de φ, les entrées de $\mathscr{A}_{\mathscr{G}}[\varphi, X^{\varphi}_{\psi^i}]$ sont des *déroulements contexte-état* pour un état s et un contexte χ t.q. $\chi = (\alpha, \gamma)$ et $dom(\alpha)$ est l'ensemble $X^{\varphi}_{\psi^i}$ des variables quantifiées *au-dessus de* ψ^i dans φ. On commence par définir formellement cet ensemble de variables :

Définition B.6 (Ensemble des variables quantifiées au-dessus d'une occurence de sous-formule dans un énoncé)**.** *Soit φ un énoncé d'USL$^{\infty}$. Alors :*

- $X^{\varphi}_{\varphi} = \varnothing$
- *si* $\neg\psi^i \in Sf(\varphi)$ *alors* $X^{\varphi}_{\psi^i} = X^{\varphi}_{\neg\psi^i}$
- *si* $(a \rhd x)\psi^i \in Sf(\varphi)$ *alors* $X^{\varphi}_{\psi^i} = X^{\varphi}_{(a\rhd x)\psi^i}$

– *si* $(a \not\triangleright x)\psi^i \in Sf(\varphi)$ *alors* $X^{\varphi}_{\psi^i} = X^{\varphi}_{(a\not\triangleright x)\psi^i}$

– *si* $\mathsf{X}\psi^i \in Sf(\varphi)$ *alors* $X^{\varphi}_{\psi^i} = X^{\varphi}_{\mathsf{X}\psi^i}$

– *si* $\psi_1^i \wedge \psi_2^j \in Sf(\varphi)$ *alors* $X^{\varphi}_{\psi_1^i} = X^{\varphi}_{\psi_2^j} = X^{\varphi}_{\psi_1^i \wedge \psi_2^j}$

– *si* $\psi_1^i \,\mathsf{U}\, \psi_2^j \in Sf(\varphi)$ *alors* $X^{\varphi}_{\psi_1^i} = X^{\varphi}_{\psi_2^j} = X^{\varphi}_{\psi_1^i \mathsf{U} \psi_2^j}$

– *si* $\langle\langle x\rangle\rangle\psi^i \in Sf(\varphi)$ *alors* $X^{\varphi}_{\psi^i} = X^{\varphi}_{\langle\langle x\rangle\rangle\psi^i} \cup \{x\}$

Un *déroulement contexte-état*, pour un contexte χ et un état s d'une CGS$^\infty$ $\mathscr{G}$, est un arbre. Il représente les ensembles d'actions correspondants aux différentes multi-stratégies données par χ le long des exécutions possibles depuis s. Ses nœuds sont des états de G, l'ensemble des états de χ. L'étiquette de chaque nœud contient la donnée de l'état courant et les ensembles d'actions pour les agents qui sont indiqués par χ. Ce second type de données est composé par une *assignation d'ensembles d'actions*, et un engagement. On définit d'abord les *assignations d'ensembles d'actions* :

Définition B.7 (Assignation d'ensembles d'actions pour $\mathscr{G}$)**.** *Soit* $\mathscr{G} = \langle Ag, G, s_0, At, val, ac, tr\rangle$ *une CGS*$^\infty$*. On apelle* assignation d'ensembles d'actions pour $\mathscr{G}$ *une fonction partielle* α_0*, de X dans* $\mathscr{P}_{>0}(\mathscr{P}_{>0}(Ac))$*. Ainsi, une assignation d'ensembles d'actions associe un ensemble d'ensembles d'actions aux variables dans son domaine. C'est l'encodage de l'information donnée par une assignation de multi-stratégies appliquée à un préfixe d'exécution donné. On note MAc l'ensemble des assignations d'ensembles d'actions.*

Soit maintenant un ensemble de variables X' et soit Ag un ensemble d'agents. On note $Eng(X')$ l'ensemble des engagements d'agents de Ag envers des variables dans X'. Pour tout énoncé φ et pour toute occurrence de sous-formule ψ^i de φ, les étiquettes des arbres qui sont utilisés comme entrées pour $\mathscr{A}_{\mathscr{G}}[\varphi, X^{\varphi}_{\psi^i}]$ sont des éléments de $G \times MAc \times Eng(X^{\varphi}_{\psi^i})$. Plus précisément, une étiquette sur $X^{\varphi}_{\psi^i}$ est un élément $(s, \alpha\_0, \gamma) \in G \times MAc \times Eng(X^{\varphi}_{\psi^i})$ tel que les variables dans l'image de γ sont toutes dans le domaine de α_0. Soit un ensemble de variables X', on écrit $Eti(X')$ l'ensemble des étiquettes sur X'.

Une étiquette lab détermine un ensemble de successeurs $Msucc_{MAc}(lab)$ depuis l'état dans cette étiquette. C'est l'ensemble des successeurs possibles quand chaque agent joue conformément aux ensembles d'actions donnés par l'assignation d'ensembles d'actions pour les variables envers lesquelles il est engagé. Ainsi, la définition est similaire à celle pour $Msucc^{\infty}_{\chi}$ (voir définition 5.17) :

Définition B.8. *Soient un ensemble de variables* X' *et une étiquette* $lab = (s, \alpha\_0, \gamma) \in Eti(X')$*. Alors,* $Msucc_{MAc}(lab) =$

–
$$\{tr(s, dc) \mid \forall \langle a, x\rangle \in \gamma, dc(a) \in \alpha\_0(x)(\tau)\}$$
si cet ensemble est non vide

– $\{\overline{\varnothing}\}$ *sinon*

On peut désormais définir un *déroulement contexte-état* :

Définition B.9. *Soit une CGS*$^\infty$ $\mathscr{G} = \langle Ag, G, s_0, At, Ac, tr\rangle$*, soit un état s dans G et soit* χ *un contexte pour* $\mathscr{G}$*. Alors, un G-arbre* $G \times MAc \times Eng(X')$*-étiqueté* $\mathscr{T} = \langle T, l\rangle$ *est un* déroulement contexte-état *pour* χ *et s ssi pour tout* $t \in T$ *il y a* $t' \in G^*$ *t.q.* $t = s \cdot t'$ *et* $l(t) = (last(t), \alpha\_0, \gamma)$*, où pour*

tout $x \in dom(\alpha\_0), \alpha\_0(x) = \alpha(x)(t)$. *Si* T *est un déroulement contexte-état pour* s *et* χ, *on apelle* déroulement de contexte *pour* χ *la projection* $Proj_{MAc \times Eng(X')}(\mathcal{T})$.

B.6 Évaluation d'un déroulement contexte-état

On peut désormais formuler et prouver le principal lemme de cette preuve. Il affirme que, étant données une CGS$^\infty$ $\mathcal{G}$ et un énoncé φ d'USL$^\infty$, pour toute ocurrence de sous-formule ψ^i de φ, on peut construire un automate qui accepte exactemement les arbres qui sont des déroulements contexte-état pour un contexte $\chi = (\alpha, \gamma)$ et un état s tels que $dom(\alpha) = X^\varphi_{\psi^i}$ et $\mathcal{G}, \chi, s \models_{\text{USL}^\infty} \psi^i$.

Lemme B.3. *Soit une CGS*$^\infty$ $\mathcal{G} = \langle Ag, G, s_0, At, Ac, tr, s_0\rangle$, *et soit* φ *un énoncé d'USL*$^\infty$. *Alors pour toute ocurrence de sous-formule* ψ^i *de* φ, *il y a un* $\langle G, Eti(X^\varphi_{\psi^i})\rangle$*-APT* $\mathcal{A}_\mathcal{G}[\varphi, X^\varphi_{\psi^i}]$ *tel que pour tout état* s *dans* G, *et pour tout contexte* χ *t.q.* $\chi = \langle \alpha, \gamma\rangle$ *et* $dom(\alpha) = X^\varphi_{\psi^i}, \mathcal{G}, \chi, s \models_{USL^\infty} \psi^i$ *ssi* $\mathcal{T}_{\chi,s} \in \mathcal{L}(\mathcal{A}_\mathcal{G}[\varphi, X^\varphi_{\psi^i}])$, *où* $\mathcal{T}_{\chi,s}$ *est le déroulement contexte-état pour* χ *et* s.

Démonstration. On prouve ce lemme en décrivant une procédure de construction pour $\mathcal{A}_\mathcal{G}[\psi^i, X^\varphi_{\psi^i}]$. La construction est menée par récurrence structurelle sur la longueur de ψ^i :

- Si ψ^i est une proposition atomique, la seule chose à vérifier est que l'état dans l'étiquette à la racine de l'arbre donné en entrée satisfait ψ^i. Ainsi, $\mathcal{A}_\mathcal{G}[\psi^i, X^\varphi_{\psi^i}] = \langle\{\overline{\psi^i}\}, \overline{\psi^i}, \tau_{\psi^i}, (\{\}\{\psi^i\})\rangle$ où, pour tout $lab = (s, \alpha\_0, \gamma) \in Eti(X^\varphi_{\psi^i}), \tau_{\psi^i}(\overline{\psi^i}, lab) = \top$ si $\psi^i \in val(s)$ et $\tau_{\psi^i}(\overline{\psi^i}, lab) = \bot$ sinon.
- Si $\psi^i = \neg\psi^j_1$, alors $\mathcal{A}_\mathcal{G}[\psi^i, X^\varphi_{\psi^i}]$ est donné par le lemme B.1 : si $\mathcal{A}_\mathcal{G}[\psi^j_1, X^\varphi_{\psi^j_1}] = \langle Q_{\psi^j_1}, q_{0_{\psi^j_1}}, \tau_{\psi^j_1}, Acc_{\psi^j_1}\rangle$, alors $\mathcal{A}_\mathcal{G}[\psi^i, X^\varphi_{\psi^i}] = \langle Q_{\psi^j_1}, q_{0_{\psi^j_1}}, \tau_{\psi^i}, Acc_{\psi^i}\rangle$ où :
  - pour tout $lab \in Eti(X^\varphi_{\psi^i})$, pour tout $q \in Q_{\psi^j}$, $\tau_{\psi^i}(q, lab) = \neg\tau_{\psi^j_1}(q, lab)$
  - si $Acc_{\psi^j_1} = (F_1, \dots F_k)$ alors $Acc_{\psi^i} = (\varnothing, F_1, \dots F_k)$
- Si $\psi^i = \psi^j_1 \wedge \psi^k_2$, alors si $\mathcal{A}_\mathcal{G}[\psi^j_1, X^\varphi_{\psi^j_1}] = \langle Q_{\psi^j_1}, q_{0_{\psi^j_1}}, \tau_{\psi^j_1}, Acc_{\psi^j_1}\rangle$ et $\mathcal{A}_\mathcal{G}[\psi^k_2, X^\varphi_{\psi^k_2}] = \langle Q_{\psi^k_2}, q_{0_{\psi^k_2}}, \tau_{\psi^k_2}, Acc_{\psi^k_2}\rangle$, alors $\mathcal{A}_\mathcal{G}[\psi^i, X^\varphi_{\psi^i}] = \langle Q_{\psi^i}, q_{o_{\psi^i}}, \tau_{\psi^i}, Acc_{\psi^i}\rangle$ où :
  - $Q_{\psi^i} = \{q_{o_{\psi^i}}\} \cup Q_{\psi^j_1} \cup Q_{\psi^k_2}$ et $q_{o_{\psi^i}} \notin Q_{\psi^j_1} \cup Q_{\psi^k_2}$
  - pour tout $lab \in Eti(X^\varphi_{\psi^i}), \tau_{\psi^i}(q_{0_{\psi^i}}, lab) = \tau_{\psi^j_1}(q_{0_{\psi^j_1}}, lab) \wedge \tau_{\psi^k_2}(q_{0_{\psi^k_2}}, lab)$
  - pour tout $q \in Q_{\psi^j_1} \cup Q_{\psi^j_1}$ et pour tout $lab \in Eti(X^\varphi_{\psi^i}), \tau_{\psi^i}(q, lab) =$
    - $\tau_{\psi^j_1}(q, lab)$ si $q \in Q_{\psi^j_1}$
    - $\tau_{\psi^k_2}(q, lab)$ si $q \in Q_{\psi^k_2}$
  - si $Acc_{\psi^j_1} = (F^1_1, \dots F^1_{k_1})$ et $Acc_{\psi^k_2} = (F^2_1, \dots F^2_{k_2})$, $min(\{k_1, k_2\}) = k_l$ et $max(\{k_1, k_2\}) = k_m$ alors $Acc_{\psi^i} = (F^1_1 \cup F^2_1, \dots, F^1_{k_l} \cup F^2_{k_l}, F^m_{k_l+1}, \dots F^m_{k_m-1}, Q_{\psi^i})$.
- Pour la résolution du cas $\psi^i = \mathsf{X}\psi^j_1$, on doit tester l'automate $\mathcal{A}_\mathcal{G}[\psi^j_1, X^\varphi_{\psi^j_1}]$ sur les successeurs de la racine de l'arbre qui est donné en entrée. Ces successeurs sont donnés par la fonction $Msucc_{MAc}$ appliquée à l'étiquette de la racine. Concrètement, si $\mathcal{A}_\mathcal{G}[\psi^j_1, X^\varphi_{\psi^j_1}] = \langle Q_{\psi^j_1}, q_{0_{\psi^j_1}}, \tau_{\psi^j_1}, Acc_{\psi^j_1}\rangle$ alors $\mathcal{A}_\mathcal{G}[\psi^i, X^\varphi_{\psi^i}] = \langle Q_{\psi^i}, q_{0_{\psi^i}}, \tau_{\psi^i}, Acc_{\psi^i}\rangle$ où :

- $Q_{\psi^i} = \{q_{o_{\psi^i}}\} \cup Q_{\psi_1^j}$ et $q_{o_{\psi^i}} \notin Q_{\psi_1^j}$
- pour tout $lab \in Eti(X^{\varphi}_{\psi^i})$:

$$\tau_{\psi^i}(q_{0_{\psi^i}}, lab) = \bigwedge_{s \in Msucc_{MAc}(lab)} (s, q_{0_{\psi^i}})$$

- pour tout $q \in Q_{\psi^i} \neq q_{0_{\psi^i}}$ et pour toute $lab \in Eti(X^{\varphi}_{\psi^i}), \tau_{\psi^i}(q, lab) = \tau_{\psi_1^j}(q, lab)$
- si $Acc_{\psi_1^j} = (F_1, \ldots F_k)$, alors $Acc_{\psi^i} = (F_1, \ldots F_k \cup \{q_{0_{\psi^i}}\})$.

– Le cas pour $\psi^i = \psi_1^j \cup \psi_2^k$ est résolu en utilisant le cas précédent et l'équivalence $(\psi_1 \cup \psi_2) \leftrightarrow \psi_1 \vee (\psi_2 \wedge \mathsf{X}(\psi_1 \cup \psi_2))$. Étant donnés les automates $\mathscr{A}_{\mathscr{G}}[\psi_1^j, X^{\varphi}_{\psi_1^j}]$ et $\mathscr{A}_{\mathscr{G}}[\psi_2^k, X^{\varphi}_{\psi_2^k}]$, pour obtenir $\mathscr{A}_{\mathscr{G}}[\psi^i, X^{\varphi}_{\psi^i}]$, on construit la fonction de transition sur cette équivalence. De plus, la satisfaction de la sous-formule $\mathsf{X}(\psi_1 \cup \psi_2)$ implique qu'il y a une transition qui boucle sur l'état initial. Cependant, l'automate doit assurer que l'entrée ne contient pas de branche qui reste indéfiniment dans l'état initial. Pour éviter cela, l'état initial est inclus dans le premier des ensembles de la condition de parité. Précisément, si $\mathscr{A}_{\mathscr{G}}[\psi_1^j, X^{\varphi}_{\psi_1^j}] = \langle Q_{\psi_1^j}, q_{0_{\psi_1^j}}, \tau_{\psi_1^j}, Acc_{\psi_1^j}\rangle$ et $\mathscr{A}_{\mathscr{G}}[\psi_2^k, X^{\varphi}_{\psi_2^k}] = \langle Q_{\psi_2^k}, q_{0_{\psi_2^k}}, \tau_{\psi_2^k}, Acc_{\psi_2^k}\rangle$ alors $\mathscr{A}_{\mathscr{G}}[\psi^i, X^{\varphi}_{\psi^i}] = \langle Q_{\psi^i}, q_{0_{\psi^i}}, \tau_{\psi^i}, Acc_{\psi^i}\rangle$ où :

- $Q_{\psi^i} = \{q_{o_{\psi^i}}\} \cup Q_{\psi_1^j} \cup Q_{\psi_2^k}$ et $q_{o_{\psi^i}} \notin Q_{\psi_1^j} \cup Q_{\psi_2^k}$
- pour toute $lab \in Eti(X^{\varphi}_{\psi^i})$:

$$\tau_{\psi^i}(q_{0_{\psi^i}}, lab) = \tau_{\psi_2^k}(q_{0_{\psi_2^k}}, lab) \vee (\tau_{\psi_1^j}(q_{0_{\psi_1^j}}, lab \wedge \bigwedge_{s \in Msucc_{MAc}(lab)} (s, q_{0_{\psi^i}}))$$

- pour tout $q \in Q_{\psi_1^j} \cup Q_{\psi_2^k}$ et pour toute $lab \in Eti(X^{\varphi}_{\psi^i}), \tau_{\psi^i}(q, lab) =$
  - $\tau_{\psi_1^j}(q, lab)$ si $q \in Q_{\psi_1^j}$
  - $\tau_{\psi_2^k}(q, lab)$ si $q \in Q_{\psi_2^k}$
- si $Acc_{\psi_1^j} = (F_1^1, \ldots F_{k_1}^1)$ et $Acc_{\psi_2^k} = (F_1^2, \ldots F_{k_2}^2)$, $min(\{k_1, k_2\}) = k_l$ et $max(\{k_1, k_2\}) = k_m$ alors $Acc_{\psi^i} = (\{q_{0_{\psi^i}}\} \cup F_1^1 \cup F_1^2, \ldots, \{q_{0_{\psi^i}}\} \cup F_{k_l}^1 \cup F_{k_l}^2, \{q_{0_{\psi^i}}\} \cup F_{k_l+1}^m, \ldots \{q_{0_{\psi^i}}\} \cup F_{k_m-1}^m, Q_{\psi^i})$.

– Traiter le cas $\psi^i = (A \rhd x)\psi_1^j$ consiste à réécrire la fonction de transition de l'automate pour ψ_1^j, pour l'adapter à la transformation du contexte. Chaque transition dans $\mathscr{A}_{\mathscr{G}}[\psi^i, X^{\varphi}_{\psi^i}]$ doit être égale à la transition donnée par $\mathscr{A}_{\mathscr{G}}[\psi_1^j, X^{\varphi}_{\psi_1^j}]$ pour la même entrée, à laquelle seraient ajoutés les choix pour les agents dans A selon x. Si $\mathscr{A}_{\mathscr{G}}[\psi_1^j, X^{\varphi}_{\psi_1^j}] = \langle Q_{\psi_1^j}, q_{0_{\psi_1^j}}, \tau_{\psi_1^j}, Acc_{\psi_1^j}\rangle$, alors $\mathscr{A}_{\mathscr{G}}[\psi^i, X^{\varphi}_{\psi^i}] = \langle Q_{\psi_1^j}, q_{0_{\psi_1^j}}, \tau_{\psi^i}, Acc_{\psi_1^j}\rangle$ où pour tout $q \in Q_{\psi_1^j}$ et pour tout $(s, \alpha\_0, \gamma) \in Eti(X^{\varphi}_{\psi^i})$,

$$\tau_{\psi^i}(q, (s, \alpha\_0, \gamma)) = \tau_{\psi_1^j}(q, (s, \alpha\_0, \gamma[A \oplus x]))$$

– À nouveau, traiter le cas pour $\psi^i = (A \not\rhd x)\psi_1^j$ consiste en une réécriture de la fonction de transition de l'automate pour ψ^i. Cette réécriture est telle que la transition donnée par la nouvelle fonction est égale à celle donnée par l'ancienne pour la même entrée, excepté le fait que les choix pour les agents dans A selon x seraient retirés. Si $\mathscr{A}_{\mathscr{G}}[\psi_1^j, X^{\varphi}_{\psi_1^j}] = \langle Q_{\psi_1^j}, q0_{\psi_1^j}, \tau_{\psi_1^j}, Acc_{\psi_1^j}\rangle$, alors $\mathscr{A}_{\mathscr{G}}[\psi^i, X^{\varphi}_{\psi^i}] = \langle Q_{\psi_1^j}, q0_{\psi_1^j}, \tau_{\psi^i}, Acc_{\psi_1^j}\rangle$ où pour tout $q \in Q_{\psi_1^j}$ et pour tout $(s, \alpha\_0, \gamma) \in Eti(X^{\varphi}_{\psi^i})$,

$$\tau_{\psi^i}(q,(s,\alpha\_0,\gamma)) = \tau_{\psi_1^j}(q,(s,\alpha\_0,\gamma[A \ominus x]))$$

– Pour le cas $\psi^i = \langle\langle x\rangle\rangle\psi_1^j$, la fonction de transition de $\mathscr{A}_{\mathscr{G}}[\psi^i, X^{\varphi}_{\psi^i}]$ doit donner la disjonction de tous les ensembles de transitions possibles correspondants aux ensembles d'actions instanciant x. Il nous faut d'abord non-déterminiser l'automate $\mathscr{A}_{\mathscr{G}}[\psi_1^j, X^{\varphi}_{\psi_1^j}]$: si $\mathscr{A}_{\mathscr{G}}[\psi_1^j, X^{\varphi}_{\psi_1^j}] = \langle Q_{\psi_1^j}, q0_{\psi_1^j}, \tau_{\psi_1^j}, Acc_{\psi_1^j}\rangle$ alors (Lemme B.2), il y a un $\langle \mathrm{G}, Eti(X^{\varphi}_{\psi_1^j})\rangle - NPT$ $\mathscr{N}_{\mathscr{G}}[\psi_1^j, X^{\varphi}_{\psi_1^j}] = \langle Q'_{\psi_1^j}, q'0_{\psi_1^j}, \tau'_{\psi_1^j}, Acc'_{\psi_1^j}\rangle$ qui accepte le même langage. Alors, $\mathscr{A}_{\mathscr{G}}[\psi^i, X^{\varphi}_{\psi^i}]$ est défini comme le $\langle \mathrm{G}, Eti(X^{\varphi}_{\psi^i})\rangle - NPT\mathscr{A}_{\mathscr{G}}[\psi^i, X^{\varphi}_{\psi^i}] = \langle Q'_{\psi_1^j}, q'0_{\psi_1^j}, \tau_{\psi^i}, Acc'_{\psi_1^j}\rangle$ t.q. pour tout $q \in Q'_{\psi_1^j}$ et pour tout $(s, \alpha\_0, \gamma) \in Eti(X^{\varphi}_{\psi^i} \cup \{x\})$,

$$\tau_{\psi^i}(q,(s,\alpha\_0,\gamma)) = \bigvee_{ac^* \in \mathscr{P}(Ac)} \tau'_{\psi_1^j}(q,(s,\alpha\_0[x \to ac^*],\gamma),$$

□

En particulier, dans le cas où ψ^i est la formule φ elle-même, alors $X^{\varphi}_{\psi^i} = \varnothing$. Donc, $\mathscr{A}_{\mathscr{G}}[\varphi, X^{\varphi}_{\psi^i}] = \mathscr{A}_{\mathscr{G}}[\varphi, \varnothing]$ est un $\langle \mathrm{G}, Eti(\varnothing)\rangle$-$APT$. Les entrées de $\mathscr{A}_{\mathscr{G}}[\varphi, \varnothing]$ sont des déroulements contexte-état pour des contextes $\chi = \langle \alpha, \gamma\rangle$ tels que $dom(\alpha) = \varnothing$. Par la définition des contextes (Définition 4.9) il est évident que le seul contexte $\chi = \langle \alpha, \gamma\rangle$ t.q. $dom(\alpha) = \varnothing$ est le contexte vide $\chi_\varnothing$. Ainsi, les entrées de $\mathscr{A}_{\mathscr{G}}[\varphi, \varnothing]$ sont des déroulements contexte-état pour un état s et $\chi_\varnothing$.

B.7 Conclusion de la preuve

Un dernier lemme est nécessaire pour conclure la preuve. Il assure que, dans l'arbre en entrée, le composant "état" dans l'étiquette des nœuds correspond toujours à l'état qui est le nœud lui-même :

Lemme B.4 (Projection sur une direction [MMPV11]). *soit $\mathscr{N}$ un $\langle G, G\times Eti\rangle$-NPT et soit s_0 un état dans G. Alors il y a un $\langle G, Eti\rangle$-NPT $\mathscr{N}_{s_0}$ t.q. pour tout G-arbre Eti-étiqueté $\mathscr{T} = \langle T, v\rangle$, $\mathscr{T} \in \mathscr{L}(\mathscr{N}_{s_0})$ ssi $\mathscr{T}' \in \mathscr{L}(\mathscr{N})$, où $\mathscr{T}' = \langle T', v'\rangle$ est le G-arbre $G\times$ Eti-étiqueté t.q. $v'(t) = (last(s_0.t), v(t))$, pour tout $t \in T$. de plus, $|\mathscr{N}_{s_0}| = |Eti|.|\mathscr{N}|$ et $id_{\mathscr{N}_{s_0}} = id_{\mathscr{N}}$.*

On peut finalement prouver que l'existence de $\mathscr{A}_{\mathscr{G}}[\varphi, \varnothing]$, pour tout énoncé φ d'USL$^\infty$, fournit une procédure de décision pour MC(USL$^\infty$) :

Théorème B.5. *Le problème* MC(*USL*$^\infty$)*, de vérifier la satisfaction d'un énoncé φ d'USL$^\infty$ dans une CGS$^\infty$ $\mathscr{G}$, peut être traité en temps* NonElémentaire *sur la longueur de φ et* Polynomial *sur la taille de $\mathscr{G}$.*

Démonstration. Comme établi dans le Lemme B.3, pour toute CGS$^\infty$ $\mathcal{G}$, pour tout état s dans $\mathcal{G}$ et pour tout énoncé φ dans USL$^\infty$, le *model-checking* de USL$^\infty$ appliqué à $\mathcal{G}$, s et φ se réduit au problème de l'appartenance de $\mathcal{T}$ dans $\mathcal{L}(\mathcal{A}_{\mathcal{G}}[\varphi,\varnothing])$, où $\mathcal{T}$ est un déroulement contexte-état pour $\chi_\varnothing$ et s. De plus, $\mathcal{A}_{\mathcal{G}}[\varphi,\varnothing]$ est un $\langle \mathrm{G}, \mathrm{G}\times MAc\times Eng(\varnothing)\rangle - APT$. Par le lemme B.2, il y a un $\langle \mathrm{G}, MAc\times Eng(\varnothing)\rangle - NPT$ $\mathcal{N}_{\mathcal{G}}[\varphi,]$ qui accepte le même langage que $\mathcal{A}_{\mathcal{G}}[\varphi,\varnothing]$.

On peut donc appliquer le Lemme B.4 à $\mathcal{N}_{\mathcal{G}}[\varphi,\varnothing]$: on obtient un $\langle \mathrm{G}, MAc\times Eng(\varnothing)\rangle$-$NPT$ $\mathcal{N}_\varphi^{\mathcal{G}}$ qui accepte exactement :

- les déroulements de $\chi_\varnothing$ ssi $\mathcal{G}, s \models_{\mathrm{USL}^\infty} \varphi$,
- aucune entrée sinon.

on a donc : $\mathcal{G}, s \models_{\mathrm{USL}^\infty} \varphi$ ssi $\mathcal{L}(\mathcal{N}_{\mathcal{G}}[\varphi])$, le langage accepté par $\mathcal{N}_{\mathcal{G}}[\varphi]$, est non vide.

Calculons le temps d'exécution pour le test du vide appliqué à $\mathcal{N}_{\mathcal{G}}[\varphi]$: pour un automate non-déterministe de taille n et d'indice k, il est pratiqué en temps $\bigcirc(n^k)$ [KV98]. Chaque étape dans la construction de $\mathcal{A}_{\mathcal{G}}[\varphi, x^{\varphi}_{\psi^i}]$ accroît la taille de l'automate courant $\mathcal{A}$ au plus de $|\mathcal{A}|$ à $2^{\bigcirc(|\mathcal{A}|id_{\mathcal{A}}.log(|\mathcal{A}|))}$. Une étape de cette construction accroît également son indice au plus de $id_{\mathcal{A}}$ à $\bigcirc(|\mathcal{A}|id_{\mathcal{A}}.log(|\mathcal{A}|))$. Il y a au plus $|\varphi|$ étapes pour la construction de $\mathcal{N}_{\mathcal{G}}[\varphi]$. L'application finale du lemme B.4 accroit la taille de l'automate par un facteur $|\mathrm{G}|$, qui est le cardinal de l'ensemble des directions dans l'automate $\mathcal{N}_{\mathcal{G}}[\varphi,\varnothing]$. Alors, $\mathcal{N}_{\mathcal{G}}[\varphi]$ est de taille au plus $|\mathcal{G}|\cdot te(m,2,m)$ et d'indice au plus $te(m,2,m)$, où $m = \bigcirc(|\varphi^{\infty_1}|.log(|\varphi|)$ et $te(n_1,n_2,n_3)$ est la tour d'exponentielle de hauteur n_1, de premier terme n_2 et de facteur n_3 :

- $te(0,n_2,n_3) = n_2$
- pour tout $n_1 \in \mathbb{N}$, $te(n_1+1,n_2,n_3) = n_3^{te(n_1,n_2,n_3)}$

On obtient ainsi le résultat du Théorème B.5 : MC(USL$^\infty$) est décidable en temps NONELÉMENTAIRE sur la longueur de la formule et POLYNOMIAL sur la taille du modèle. □

Annexe C

Une illustration : le cas d'étude des missions d'observation satellite

Sommaire

Dans cette annexe, on détaille le cas d'étude des missions d'observations par satellite. On considère deux modèles de buts : celui pour une agence de cartographie (*agence cartographique*) et celui pour une puissance militaire (*armée*). Chacun a besoin de se procurer des images satellites de la terre. Pour l'*agence cartographique*, il s'agit de réaliser une analyse continuelle de l'état du ciel pour tenir à jour ses bases de données. Pour l'*armée*, il s'agit de mettre en œuvre un dispositif prêt à répondre rapidement à un besoin éventuel pour une image d'une région de la terre.

On analyse d'abord les besoins à remplir pour chacune de ces deux missions, on en dérive les modèles de buts (Section C.1) ainsi qu'une opérationnalisation des exigences (Section C.2). Les capacités des acteurs sont ensuite formalisées (Section C.3) ainsi que les propriétés du domaine (Section C.4). La correction locale du modèle $\mathsf{Kh}_{C,A}$ est prouvée dans la Section C.5. À partir de ce modèle de référence, le modèle $\mathsf{Kh}_{C,A}$, plusieurs variantes sont enfin envisagées afin d'illustrer les différents critères de correction de l'assignation (Section C.6).

C.1 Buts

On identifie ici les besoins à satisfaire dans le modèle $\mathsf{Kh}_{C,A}$. On les modélise dans des modèles de buts et on formalise les exigences qui en découlent.

C.1.1 Identification des besoins à remplir

L'agence de cartographie a besoin d'une couverture régulière de l'état du ciel par des images satellites pour mettre à jour ses bases de données. Le dispositif à mettre en œuvre doit pouvoir identifier les images dont il a besoin au fur et à mesure de la réception de ces images. Un traitement des données est donc nécessaire. Le dispositif doit aussi pouvoir fournir ces images. Le système doit donc mettre en place un cycle de traitements d'images, d'identifications de besoins en nouvelles images et de production de ces images. Ce cycle doit être efficace, c'est-à-dire à la fois se dérouler sur une période courte et permettre une acquisition importante d'images pour chaque période.

Pour l'*armée*, il s'agit de pouvoir se procurer ponctuellement des images pour préparer une intervention dans une zone déterminée en réaction à une crise quelconque. L'identification des besoins ne relève donc pas du système, celui-ci étant précisément destiné à répondre à un besoin quand il survient du domaine environnant. Le dispositif de réponse doit à nouveau être efficace, c'est-à-dire rapide et pouvant répondre à un besoin important. Toute réponse doit par ailleurs permettre une nouvelle utilisation ultérieure du système.

C.1.2 Buts et raffinements

Ces besoins à remplir sont modélisés dans KHI comme des buts à satisfaire (cf. Figure C.1). Pour l'*agence cartographique*, le but de plus haut niveau, *couverture régulière*, est raffiné en une exigence de *planification* (traitements des images reçues pour l'identification de nouveaux besoins d'images) et un but de prise et collecte d'images proprement dites (*photographies*). Ce dernier est lui-même raffiné par les buts d'*efficacité* (qui est encore raffiné par les exigences de *rapidité* et de *volume* d'une manière directement dictée par l'analyse des besoins) et d'*adéquation*. Dans l'analyse des buts ci-dessus, l'*adéquation* était implicite. Elle est rendue explicite dans la modélisation : il s'agit de lier l'exigence de *planification* au but *photographies* en s'assurant que les images produites par le système répondent bien aux besoins identifiés dans la *planification*.

Pour l'*armée*, le but principal du dispositif est de pouvoir assurer des *photographies ponctuelles*, selon des besoins survenus dans l'environnement du système. Le raffinement de ce but est similaire à celui de *photographies* pour l'*agence cartographique*, à la différence près qu'une exigence de *permanence* y est ajoutée : si le système est utilisé, alors il doit fonctionner de manière à être prêt à servir à nouveau après cette utilisation. Il s'agit de garantir que la production d'images par le système se fait d'une manière qui permet de retrouver ses conditions initiales.

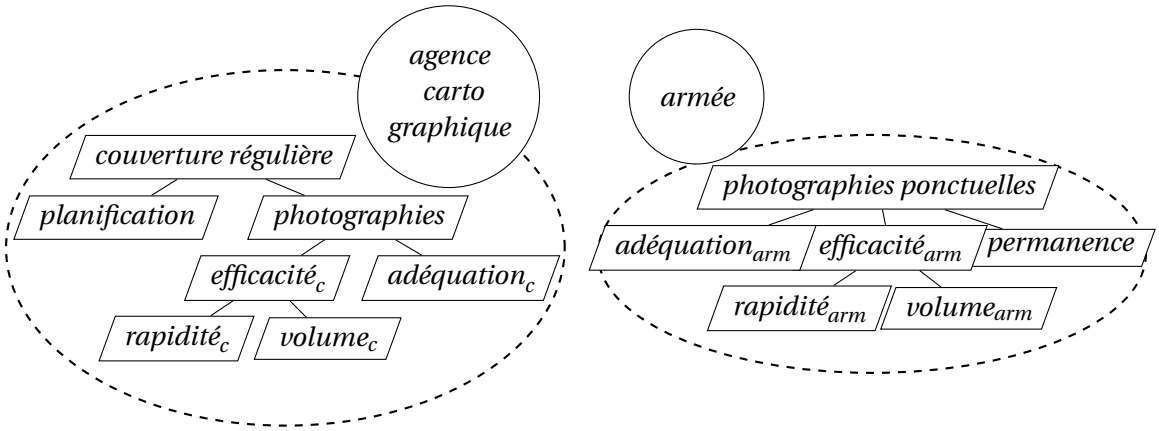

FIGURE C.1 – Modèles de buts pour l'*agence cartographique* et l'*armée*

C.1.3 Étiquettes et variables utilisées pour la formalisation des exigences

On présente ici les variables qui sont directement impliquées dans la formalisation des exigences et leurs étiquettes. Un bilan général des variables utilisées dans le modèle $\mathsf{Kh}_{C,A}$ est par ailleurs donné dans le Tableau C.7, Section C.3.5.

C.1.3.1 Étiquettes

Les variables utilisées dans tout ce modèle portent les étiquettes **image** et **position** :

Une variable étiquetée position encode une coordonnée spatiale. On encode également avec des variables portant cette étiquette les positions terrestres des zones à photographier. C'est encore l'étiquette pour les variables encodant la position correspondant aux valeurs contenues dans les variables étiquetées **image** dans l'intervalle $[0,\dots 5]$ dans $\mathbb{N}$. Le domaine de l'étiquette **position** contient aussi la valeur particulière -1, qui est donnée à

une variable $x.pos$ pour encoder le fait que la variable x étiquetée **image** ne contient pas de valeur. On a donc : $dom(\textbf{position}) = [-1,\dots 5]$. À une variable x étiquetée **position** sont associées deux variables $x.old$ et $x.succ$ étiquetées **position** également : $x.old$ contient la valeur de la variable x dans l'état précédent, et $x.succ$ désigne la position suivante sur la trajectoire des satellites. Elle sert en particulier à la description, dans les moyens, du mouvement des satellites. L'axiomatisation de ces variables est donnée dans le Tableau C.1.

[**position**.*old* : **position**]
$old.ax \triangleq \{\Box(\textbf{position} = k \rightarrow \mathsf{X}(\textbf{position}.old = k))\}_{k\in[-1,\dots 5]}$
[**position**.*succ* : **position**]
$succ.ax \triangleq \{\Box(\textbf{position} = k \rightarrow \textbf{position}.succ = k+1 [\text{mod}6])_{k\in[-1,\dots 5]}$

TABLE C.1 – Variables étiquetées **position** et axiomes associés

On remarque que l'application de l'opération *succ* ne renvoie jamais la valeur -1 : cette dernière valeur en effet n'encode pas une position géographique mais signale une absence de donnée.

Une variable x étiquetée image encode une image transmise ou stockée. On y associe une variable $x.old$ d'étiquette **image** et une variable $x.pos$ étiquetée **position**. Le domaine de l'étiquette **image** est un ensemble fini de valeurs noté $\mathscr{I}$. L'axiomatisation de $x.old$ est similaire à celle pour les variables étiquetées **position** (voir Tableau C.2). La variable $x.pos$ contient, pour une image x, la **position** de laquelle elle a été prise ou doit être prise. Elle est telle que deux images identiques correspondent à la même position.

[**image**.*old* : **image**]
$old.ax \triangleq \{\Box(\textbf{image} = k \rightarrow \mathsf{X}(\textbf{image}.old = k))\}_{k\in[-1,\dots 5]}$
[**image**.*pos* : **position**]
$pos.ax \triangleq \Box(x = y \rightarrow x.pos = y.pos)$

TABLE C.2 – Variables étiquetées **image** et axiomes associés

C.1.3.2 Variables

On présente maintenant les variables utilisées dans la formalisation des exigences. La satisfaction du but *volume*, pour l'*agence cartographique* comme pour *armée*, sera assurée en permettant un traitement des besoins pour de nouvelles photographies qui les prenne en charge deux par deux. Les besoins sont donc des tâches à accomplir, encodées dans les variables $ordre^i_{act}$ étiquetées **position**, avec $i \in \{1,2\}$ et $act \in \{agence\ cartographique, armée\}$. L'usage de cette notation est généralisé pour les variables et les opérations qui nécessitent ces paramètres.

Pour les variables $ordre^i_{act}$, la valeur -1 indique que l'ordre correspondant est désactivé, *i.e.* qu'il ne contient pas d'instruction. Pour chacune de ces variables d'ordres, une variable $imageStockéeSol^i_{act}$, contient l'image donnée comme réponse à cet ordre.

Comme l'*armée* n'attend pas du système qu'il élabore les *ordres* mais transmette les *besoins* perçus dans l'environnement, ses exigences utilisent également les variables $besoin^1$ et $besoin^2$.

Ces dernières variables désignent les besoins tels qu'ils sont perçus par le système dans son environnement.

C.1.4 Formalisation des exigences

Nous pouvons maintenant proposer une formalisation des exigences du modèle $\mathsf{Kh}_{C,A}$. Dans ces formules, X^k abrège une succession de k occurrences de l'opérateur X, et pour toute formule $\varphi(i)$ qui utilise i comme paramètre de ses variable $\wedge_i \varphi(i)$ abrège la formule $\varphi(1) \wedge \varphi(2)$.

***volume**$_{act}$* Les exigences de *volume* requièrent que les ordres soient identifiés ou transmis deux par deux. On les formalise pour chacun des deux modèles de buts, en exigeant que les deux variables $ordre^i_{act}$ soient activées en même temps :

$$Form(volume_{act}) \triangleq \Box((ordre^1_{act} = -1 \wedge \mathsf{X}(ordre^1_{act} \geqslant 0)) \leftrightarrow (ordre^2_{act} = -1 \wedge \mathsf{X}(ordre^2_{act} \geqslant 0))$$

planification La planification, à laquelle seule l'*agence cartographique* est soumise, requiert que, après enregistrement des réponses à une série de deux ordres lancés (transition identifiée par $(\vee_i(imageStockéeSol^i_c.pos = -1) \wedge \mathsf{X}(\wedge_i imageStockéeSol^i_c.pos \geqslant 0)$, *i.e.* : la deuxième image est enregistrée), une nouvelle série d'ordres soit à nouveau lancée dans un intervalle de trois transitions :

$$Form(planification) \triangleq \Box((\vee_i(imageStockéeSol^i_c.pos = -1) \wedge \mathsf{X}(\wedge_i imageStockéeSol^i_c.pos \geqslant 0)) \rightarrow \bigvee_{k<3} \mathsf{X}^{k+1}(\wedge_i(ordre^i_c = -1 \wedge \mathsf{X}\, ordre^i_c \geqslant 0)))$$

***adéquation**$_{act}$* Les exigences d'*adéquation* requièrent que, à partir du moment où les deux ordres sont envoyés, les variables de réponse ne soient remplies que par des réponses adéquates à ces ordres. Donc, soit ces variables restent toujours vides, soit elles le restent jusqu'à prendre une valeur dont la position correspond à l'ordre. Dans la formalisation, on maintient également la valeur de l'ordre tant que la réponse n'est pas enregistrée, afin de pouvoir exprimer et vérifier cette adéquation ($ordre^i_{act} = ordre^i_{act}.old$) :

$$Form(adéquation_{act}) \triangleq \wedge_i(\Box((ordre^i_{act}.old = -1 \wedge ordre^i_{act} \geqslant 0) \rightarrow ((\Box(imageStockéeSol^i_{act}.pos = -1)) \vee (imageStockéeSol^i_{act}.pos = -1 \wedge \mathsf{X}(ordre^i_{act} = ordre^i_{act}.old)) \,\mathsf{U}\, imageStockéeSol^i_{act}.pos = ordre^i_{act}.old)))))$$

***rapidité**$_c$* selon *rapidité*$_c$, à partir du moment où un ordre est lancé, (transition identifiée par $(ordre^i_c = -1 \wedge \mathsf{X}(ordre^i_c \geqslant 0))$, une réponse à cet ordre doit être enregistrée dans un intervalle de temps acceptable. Dans notre modélisation, on a posé la limite au temps écoulé en 16 transitions du système. Ce choix est arbitraire. $Form(rapidité_c)$ est la conjonction de cette condition pour chacun des ordres $ordre^1_c$ et $ordre^2_c$:

$$Form(rapidité_c) \triangleq \wedge_i\Box((ordre^i_c = -1 \wedge ordre^i_c \geqslant 0) \rightarrow \bigvee_{k<16} \mathsf{X}^{k+1}(imageStockéeSol^i_{arm}.pos = -1 \wedge \mathsf{X}(imageStockéeSol^i_{arm}.pos \geqslant 0)))$$

***rapidité**$_{arm}$* l'exigence *rapidité*$_{arm}$ est formalisée presque de la même manière que *rapidité*$_c$. Seulement, la transition de référence n'est pas l'identification des *ordres* à accomplir mais la survenue de nouveaux *besoins* dans l'environnement :

$$Form(\textit{efficacité}_{arm}) \triangleq \Box(\wedge_i((\textit{besoin}^i = -1 \wedge \textit{besoin}^i \geqslant 0) \rightarrow \bigvee_{k<16} \mathsf{X}^{k+1}(\textit{imageStockéeSol}^i_{arm}.pos = -1 \wedge \mathsf{X}(\textit{imageStockéeSol}^i_{arm}.pos \geqslant 0)))$$

permanence la *permanence* requiert que, quand une photo est enregistrée, l'ordre correspondant reste désactivé tant qu'un besoin n'est pas identifié :

$$Form(\textit{permanence}) \triangleq \Box(\wedge_i((\textit{imageStockéeSol}^i_{arm}.pos = -1 \wedge \mathsf{X}(\textit{imageStockéeSol}^i_{arm}.pos \geqslant 0)) \rightarrow ((\Box(\textit{ordre}^i_c = -1)) \vee (\textit{ordre}^i_c = -1 \,\mathsf{U}\, \textit{besoin}^i \geqslant 0))))$$

C.2 Opérations et opérationnalisation

Dans l'état initial du système, toutes les variables de contenu d'image, sauf celles pour les photographies enregistrées, sont initialisées de manière à ce que leur position soit égale à -1. Les ordres et les besoins sont également initialisés à -1.

Pour l'*agence cartographique*, l'exécution du système procède comme suit : l'agence élabore des ordres (opération *élaborationOrdres*). Ces ordres sont reçus par des satellites (opération *réceptionOrdres*$_c$) et des photos qui leur correspondent sont prises (opérations $\textit{prisePhoto}^1_c$ et $\textit{prisePhoto}^2_c$). Les satellites envoient au sol les photos qu'il ont prises (opérations $\textit{envoiPhoto}^1_c$ et $\textit{envoiPhoto}^2_c$). Finalement, l'*agence cartographique* les récupère (opérations $\textit{réceptionPhoto}^1_c$ et $\textit{réceptionPhoto}^2_c$), élabore de nouveaux ordres, et un nouveau cycle démarre.

À ces opérations s'ajoutent trois mécanismes pour maintenir les valeurs pour des variables. Il s'agit, entre le moment où une variable reçoit une certaine valeur et le moment où cette valeur est utilisée, de la conserver. Pour chacune de ces variables, le mécanisme qui permet cette garantie se modèlise dans le formalisme des opérations, bien qu'il ne s'agisse pas, à strictement parler, d'une opération. Il s'agit d'abord, pour les satellites qui prennent les photos, de conserver la valeur des ordres reçus avant de prendre les photos correspondantes (opérations $\textit{maintienOrdreAtt}^i_{act}$, $act \in \{c, arm\}, i \in \{1,2\}$) et de conserver ensuite en mémoire ces images (opérations $\textit{maintienMémoire}^i_{act}$). Par ailleurs, quand un ordre est envoyé aux satellites, sa valeur est également conservée jusqu'à réception d'une réponse à cet ordre, afin de pouvoir exprimer et vérifier l'*adéquation* de cette réponse (opérations $\textit{maintienOrdre}^i_{act}$).

La mission pour *armée* suit le même processus, à la différence que l'ordre n'est pas élaboré par le système. Il s'agit d'un besoin manifesté dans l'environnement et transmis aux satellites en tant qu'ordre auquel répondre. L'opération *élaborationOrdres* y est donc remplacée par une opération *transmissionBesoin*, qui consiste à transmettre aux satellites les besoins identifiés en tant qu'ordres. Par ailleurs, le but de *permanence* est assuré par des opérations $\textit{maintienÉtatAttente}^1$ et $\textit{maintienÉtatAttente}^2$, déclenchées dès qu'une réponse à un besoin a été apportée, et qui maintiennent le dispositif en attente de nouveaux besoins identifiés.

Dans les paragraphes qui suivent, on décrit en détail les contraintes de chacune de ces opérations. L'opérationnalisation est par ailleurs présentée exhaustivement dans le Tableau C.3.

L'élaboration des ordres pour l'*agence cartographique* consiste à donner une valeur positive aux deux variables $\textit{ordre}^i_c$. Pour l'exigence de *volume*$_c$, les opérations correspondantes pour

chacun des deux ordres sont lancées en même temps. Pour l'*adéquation*, ces opérations doivent vider le contenu de la photo enregistrée, afin de laisser disponible cette variable pour la réponse au nouvel ordre. Pour la *planification*, ces opérations doivent avoir lieu quand et dès que les deux variables de photos enregistrées sont remplies et les variables $ordre_c^i$ sont vides. On a donc cette condition à la fois comme *reqPre* et comme *reqTrig*.

L'opération correspondante pour *armée* est celle de transmission des besoins (opérations *transmissionBesoin*). Pour satisfaire la contrainte de *rapidité*, cette opération est déclenchée quand et seulement quand les deux ordres sont désactivés et les besoins sont exprimés (les variables correspondantes ont une valeur positive). L'exigence d'*adéquation* requiert par ailleurs que les valeurs envoyées comme ordre correspondent bien respectivement aux besoins exprimés. Cette exigence impose également que l'opération soit déclenchée dès que les besoins apparaissent, afin de s'assurer que les ordres envoyés correspondent aux premiers besoins identifiés. Enfin, cette opération doit vider la valeur de la photo enregistrée correspondante. L'exigence de *volume* pour l'*armée* impose à nouveau, par ailleurs, que les deux opérations *transmissionBesoin* soient déclenchées en même temps.

Les opérations de réception des ordres consistent, pour chacune des missions, à ce que les deux ordres envoyés soient reçus comme des tâches à accomplir par les moyens à bord. Il s'agit donc de remplir les deux variables de tâches (*tâcheEnAttente*), en leur donnant les valeurs des derniers ordres émis quand ils l'ont été, pour contribuer au but d'*adéquation*. Pour la *rapidité* du système, l'opération doit être déclenchée dès que cette dernière condition est remplie. C'est également au cours de ces opérations que les variables de transmission d'images des satellites au sol (*réponse*) sont réinitialisées (leur position passe à -1).

Une fois les ordres reçus, le maintien de leur valeur en mémoire est assuré par des opérations *maintienOrdreAtt*, qui consistent simplement à confirmer, au long des transitions, la valeur des variables *tâcheEnAttente*. On s'assure ainsi que ces variables gardent les mêmes valeurs entre leur initialisation comme prise en compte des ordres (opération *réceptionOrdres*), et leur utilisation pour spécifier la prise des photos (opérations *prisePhoto*). Elles sont donc déclenchées continuellement tant que les pré-conditions ne sont pas remplies pour le déclenchement des opérations de prise des photos. Elles sont nécessaires pour satisfaire à la fois les exigences de *rapidité* et d'*adéquation*.

Par les opérations *prisePhoto*, les satellites donnent une valeur aux images stockées à bord (variables *imageStockéeBord*). Pour l'*adéquation* et pour chacun des ordres, il faut que la photo prise soit une photo de la position exprimée dans cet ordre et enregistrée à bord comme une tâche à accomplir. Pour cela, elle doit être prise quand le satellite qui la prend atteint la position indiquée comme tâche à accomplir. Pour l'efficacité, cette photo doit être prise dès que possible, c'est-à-dire dès que cette dernière condition est remplie.

Pour chacune des photos, la permanence de l'image contenue en mémoire est assurée par l'opération *maintienMémoire*. Cette opération est déclenchée à chaque transition jusqu'à ce que les pré-conditions suffisantes de déclenchement (*reqTrig*) pour l'envoi de la photo correspondante soient remplies.

L'envoi des photos consiste à donner une valeur positive aux variables de réponses aux ordres (*réponse*). Pour envoyer un signal à terre, un satellite doit se trouver à une position particulière de communication, identifiée comme le cinquième point sur la trajectoire. Cette contrainte est prise en charge pour les exigences de *rapidité*. Ces exigences requièrent également que la mémoire embarquée soit non vide au moment de l'envoi (en effet, dans le cas où l'ordre courant est une image prise de la cinquième position, le satellite qui la prend en charge passe une première fois à cette position pour prendre la photo. Lors de ce premier passage il déclenche

une opération *prisePhoto* mais ne déclenche pas *envoiPhoto*. Il déclenchera cette opération après une révolution supplémentaire sur sa trajectoire, quand sa mémoire embarquée sera pleine). L'exigence *adéquation* requiert encore que la valeur envoyée dans le signal soit celle de la mémoire embarquée. Par ailleurs, l'acteur qui la déclenche vide sa mémoire embarquée et sa mémoire de tâche à accomplir, afin de satisfaire directement les pré-conditions pour recevoir de nouveaux ordres.

Les contraintes pour les opérations *réceptionPhoto* diffèrent légèrement pour *armée* et pour *agence cartographique*. En effet, pour chacune de ces missions, ces opérations doivent désactiver les ordres courants. Mais dans le cas de l'*agence cartographique*, il s'agit d'une réponse à l'exigence de *planification*, alors que pour l'*armée*, c'est la *permanence* qui exige cette réinitialisation, afin que le système soit à nouveau prêt à réagir directement à l'identification d'un éventuel futur besoin. Par ailleurs, pour les deux missions, l'*adéquation* requiert que ces opérations soient déclenchées quand et dès que la réponse est envoyée, et que la photo enregistrée correspond à cette réponse. L'exigence de *rapidité* nécessite également que ces opérations soient déclenchées dès que la réponse est envoyée.

Les opérations *maintienOrdre* permettent de garder en mémoire, tout au long du cycle, la valeur d'un ordre qui a été envoyé. Ceci permet de vérifier l'*adéquation* de la réponse apportée à un ordre. Pour chaque acteur, on déclenche donc les opérations de maintien des ordres tant que les variables de *réponse* ne sont pas instanciées.

Les opérations *maintienÉtatAttente* permettent enfin d'assurer le maintien des variables *ordre* pour l'*armée* en état d'attente, tant qu'un *besoin* n'est pas identifié. Ceci permet de satisfaire l'exigence de *permanence*.

Ces opérations sont formalisées dans le Tableau C.3.

$élaborationOrdres^i$		$domPre$	$ordre^i_c = -1$
		$domPost$	$ordre^i_c \geqslant 0$
	$adéquation_c$	$reqPost$	$imageStockéeSol^i_c.pos$ = -1
	$planification$	$reqPre$	$imageStockéeSol^i_c.pos \geqslant 0$
			$\wedge imageStockéeSol^{3-i}_c.pos \geqslant 0$
			$\wedge ordre^{3-i}_c = -1$
		$reqPost$	$ordre^{3-i}_c \geqslant 0$
		$reqTrig$	$imageStockéeSol^i_c.pos \geqslant 0$
			$\wedge imageStockéeSol^{3-i}_c.pos \geqslant 0$
			$\wedge ordre^{3-i}_c = -1$
	$volume_c$	$reqPre$	$ordre^{3-i}_c = -1$
		$reqPost$	$ordre^{3-i}_c \geqslant 0$
$transmissionBesoin$		$domPre$	$ordre^i_{arm} = -1$
		$domPost$	$ordre^i_{arm} \geqslant 0$
	$adéquation_{arm}$	$reqPre$	$besoin^i \geqslant 0 \wedge besoin^i.old = -1$
		$reqPost$	$ordre^i_{arm} = besoin^{arm}i.old$
			$\wedge imageStockéeSol^i_{arm}.pos = -1$
		$reqTrig$	$besoin^{arm}i \geqslant 0 \wedge besoin^{arm}i.old = -1$
	$rapidité_{arm}$	$reqPre$	$besoin^i \geqslant 0 \wedge besoin^i.old = -1$
		$reqTrig$	$besoin^{arm}i \geqslant 0 \wedge besoin^{arm}i.old = -1$

	$volume_{arm}$	*reqPre*	$ordre^{3-i}_{arm} = -1$
		reqPost	$ordre^{3-i}_{arm} \geqslant 0$
$réceptionOrdres_{act}$		*domPre*	$tâcheEnAttente^{i}_{act} = -1$
		domPost	$tâcheEnAttente^{i}_{act} \geqslant 0$
	$adéquation_{act}$	*reqPre*	$ordre^{i}_{act} \geqslant 0 \wedge ordre^{i}_{act}.old = -1$
		reqPost	$tâcheEnAttente^{i}_{act} = ordre^{i}_{act}.old$ $\wedge réponse^{i}_{act} = \text{-}1$
		reqTrig	$ordre^{i}_{act} \geqslant 0 \wedge ordre^{i}_{act}.old = -1$
	$rapidité_{act}$	*reqPre*	$ordre^{i}_{act} \geqslant 0 \wedge ordre^{i}_{act}.old = -1$
		reqPost	$réponse^{i}_{act} = \text{-}1$
		reqTrig	$ordre^{i}_{act} \geqslant 0 \wedge ordre^{i}_{act}.old = -1$
$maintienOrdreAtt^{i}_{act}$		*domPre*	$tâcheEnAttente^{i}_{act} \geqslant 0$
		domPost	$tâcheEnAttente^{i}_{act} \geqslant 0$
	$adéquation_{act}$	*reqPre*	$imageStockéeBord^{i}_{act}.pos = -1$ $\vee positionUnitéBord^{i}_{act} < 5$
		reqPost	$tâcheEnAttente^{i}_{act} = tâcheEnAttente^{i}_{act}.old$
		reqTrig	$imageStockéeBord^{i}_{act}.pos = -1$ $\vee positionUnitéBord^{i}_{act} < 5$
	$rapidité_{act}$	*reqPre*	$imageStockéeBord^{i}_{act}.pos = -1$ $\vee positionUnitéBord^{i}_{act} < 5$
		reqPost	$tâcheEnAttente^{i}_{act} = tâcheEnAttente^{i}_{act}.old$
		reqTrig	$imageStockéeBord^{i}_{act}.pos = -1$ $\vee positionUnitéBord^{i}_{act} < 5$
$prisePhoto^{i}_{act}$		*domPre*	$imageStockéeBord^{i}_{act}.pos = \text{-}1$
		domPost	$imageStockéeBord^{i}_{act}.pos \geqslant 0$
	$adéquation_{act}$	*reqPre*	$tâcheEnAttente^{i}_{act} \geqslant 0$ $\wedge tâcheEnAttente^{i}_{act} = positionUnitéBord^{i}_{act}$
		reqPost	$imageStockéeBord^{i}_{act}.pos$ $= tâcheEnAttente^{i}_{act}.old$
	$rapidité_{arm}$	*reqPre*	$tâcheEnAttente^{i}_{act} \geqslant 0$ $\wedge tâcheEnAttente^{i}_{act} = positionUnitéBord^{i}_{act}$
		reqTrig	$tâcheEnAttente^{i}_{act} \geqslant 0$ $\wedge tâcheEnAttente^{i}_{act} = positionUnitéBord^{i}_{act}$
$maintienMémoire^{i}_{act}$		*domPre*	$imageStockéeBord^{i}_{act}.pos \geqslant 0$
		domPost	$imageStockéeBord^{i}_{act}.pos \geqslant 0$
	$adéquation_{act}$	*reqPost*	$imageStockéeBord^{i}_{act}.pos$ $= imageStockéeBord^{i}_{act}.pos.old$
	$rapidité_{act}$	*reqPre*	$positionUnitéBord^{i}_{act} < 5$
		reqTrig	$positionUnitéBord^{i}_{act} < 5$
$envoiPhoto^{i}_{act}$		*domPre*	$réponse^{i}_{act}.pos = \text{-}1$
		domPost	$réponse^{i}_{act}.pos \geqslant 0$
	$adéquation_{act}$	*reqPost*	$réponse^{i}_{act}.pos$ $= imageStockéeBord^{i}_{act}.pos.old$

			$\wedge$ $imageStockéeBord^i_{act}.pos$ = -1
			$\wedge$ $tâcheEnAttente^i_{act}$ =-1
	$rapidité_{act}$	$reqPre$	$imageStockéeBord^i_{act}.pos \geqslant 0$
			$positionUnitéBord^i_{act} = 5$
		$reqTrig$	$imageStockéeBord^i_{act}.pos \geqslant 0$
			$positionUnitéBord^i_{act} = 5$
$réceptionPhoto^i_c$		$domPre$	$imageStockéeSol^i_c.pos$ = -1
		$domPost$	$imageStockéeSol^i_c.pos \geqslant 0$
	$planification$	$reqPre$	$ordre^i_c \geqslant 0$
		$reqPost$	$ordre^i_c = -1$
	$adéquation_c$	$reqPre$	$réponse^i_c.pos \geqslant 0$
			$\wedge réponse^i_c.pos.old$ = -1
		$reqPost$	$imageStockéeSol^i_c.pos$
			$= réponse^i_c.pos.old$
		$reqTrig$	$réponse^i_c.pos \geqslant 0$
			$\wedge réponse^i_c.pos.old$ = -1
	$rapidité_c$	$reqPre$	$réponse^i_c.pos \geqslant 0$
			$\wedge réponse^i_c.pos.old$ = -1
		$reqTrig$	$réponse^i_c.pos \geqslant 0$
			$\wedge réponse^i_c.pos.old$ = -1
$réceptionPhoto^i_{arm}$		$domPre$	$imageStockéeSol^i_{arm}.pos$ = -1
		$domPost$	$imageStockéeSol^i_{arm}.pos \geqslant 0$
	$permanence$	$reqPre$	$ordre^i_{arm} \geqslant 0$
	$adéquation_{arm}$	$reqPre$	$réponse^i_{arm}.pos \geqslant 0$
			$\wedge réponse^i_{arm}.pos.old$ = -1
		$reqPost$	$imageStockéeSol^i_{arm}.pos$
			$= réponse^i_{arm}.pos.old$
		$reqTrig$	$réponse^i_{arm}.pos \geqslant 0$
			$\wedge réponse^i_{arm}.pos.old$ = -1
	$rapidité_{arm}$	$reqPre$	$réponse^i_{arm}.pos \geqslant 0$
			$\wedge réponse^i_{arm}.pos.old$ = -1
		$reqTrig$	$réponse^i_{arm}.pos \geqslant 0$
			$\wedge réponse^i_{arm}.pos.old$ = -1
$maintienOrdre^i_{act}$		$domPre$	$ordre^i_{act}$ >= 0
		$domPost$	$ordre^i_{act}$ >=0
	$adéquation_{act}$	$reqPre$	$réponse^i_{act}.pos = -1$
			$\vee réponse^i_{act}.pos.old \geqslant 0$
		$reqPost$	$ordre^i_{act} = ordre^i_{act}.old$
		$reqTrig$	$réponse^i_{act}.pos = -1$
			$\vee réponse^i_{act}.pos.old \geqslant 0$
$maintienÉtatAttente^i$		$domPre$	$ordre^i_{arm} = -1$
		$domPost$	$ordre^i_{arm} = -1$
	$permanence$	$reqPre$	$besoin^i$ =-1

| | | *reqTrig* | *besoin*^{i} =-1

TABLE C.3 – Opérationnalisation

La correction de cette opérationnalisation pour les exigences du modèle a été prouvée pour LTL à l'aide du *model-checker* NuSMV, les spécifications utilisées sont données dans la Section C.7.

La répartition de ces opérations dans des rôles est représentée dans la Figure C.3. Pour chacune des missions, un segment embarqué prend en charge toutes les opérations qui relèveront des satellites mis en place, et un segment sol gère les images reçues. Pour la mission pour l'*armée*, il s'agit de la transmission des besoins, du maintien en mémoire des ordres ainsi transmis, de la réception des photos et du maintien dans l'état d'attente de nouveaux besoins. Le segment sol pour l'agence météo réalise l'élaboration des ordres, garde mémoire de ces ordres envoyés et reçoit les photos.

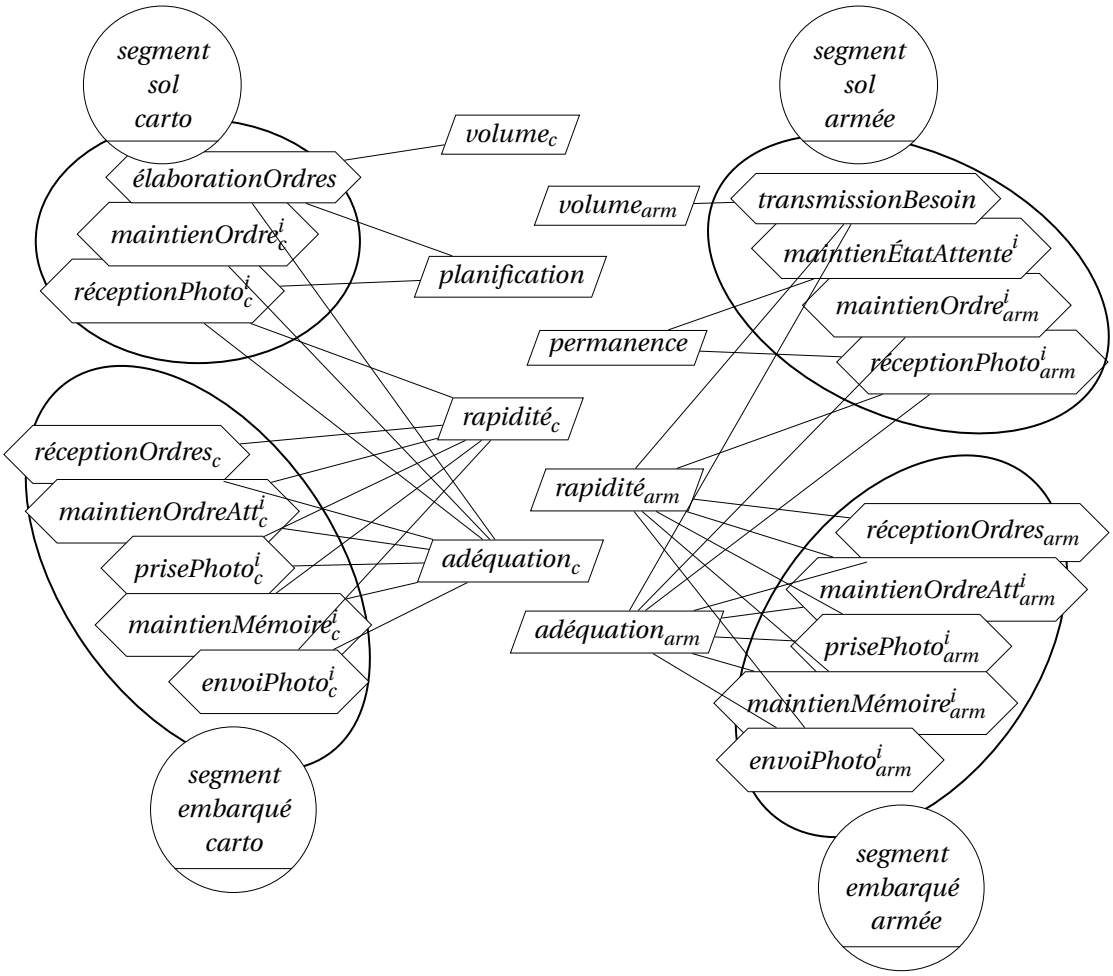

FIGURE C.2 – Opérationnalisation et rôles pour les missions d'observation spatiale

C.3 Acteurs

On s'intéresse maintenant aux acteurs et on décrit leurs capacités. L'*armée* est décrite sans capacité d'action. Les acteurs qui ont un ensemble non vide de capacités sont donc, d'une part, l'*agence cartographique* et un *transmetteur* (l'*agence cartographique* agit directement sur le système et l'*armée* fait appel au *transmetteur* pour transmettre les besoins qu'elle identifie aux satellites), et d'autre part trois satellites (*S1*, *S2* et *S3*). Cette section inclue également une synthèse sur l'ensemble des variables utilisées dans le modèle Kh (Section C.3.5).

C.3.1 Le *transmetteur*

Le *transmetteur* peut transmettre les ordres dans les signaux $signalSol_{arm}$. Il s'agit d'envoyer dans ces signaux les valeurs contenues dans les *besoinPerçu*. Par ailleurs, le *transmetteur* peut enregistrer les réponses aux besoins qu'il a transmis. Les variables de photos enregistrées $mémoireSol_{arm}$ prennent alors les valeurs des réponses reçues dans $signalBord_{arm}$. Dans le langage des pré-conditions et fenêtres de capacités, cette dernière capacité est décrite comme un ensemble de capacités, une pour chaque valeur possible de la réponse. Enfin, le *transmetteur* a la capacité de réinitialiser à -1 les signaux qu'il utilise et les variables de sa mémoire. Ses capacités sont données dans le tableau C.4.

$transmissionOrdre^i$	*préCond*	$signalSol^i_{arm} = -1$
	fenêtre	$signalSol^i_{arm} = besoinPerçu^i.old$
$\{enregistrementPhoto^{i,k}_{arm}\}_{k\in[0,...5]}$	*préCond*	$signalBord^i_{arm}.pos = k$
	fenêtre	$mémoireSol^i_{arm}.pos = k$
$désactivationOrdre^i_{arm}$	*préCond*	$\top$
	fenêtre	$signalSol^i_{arm} = -1$
$libérationMémoireSol^i_{arm}$	*préCond*	$\top$
	fenêtre	$mémoireSol^i_{arm}.pos = -1$

TABLE C.4 – Capacités du *transmetteur*

C.3.2 L'*agence cartographique*

Les capacités de l'*agence cartographique* sont similaires à celles du *transmetteur*. Elles sont données dans le Tableau C.5. Toutefois, l'*agence cartographique* a la capacité de choisir la valeur de l'ordre qu'elle envoie, les capacités $transmissionOrdre^i$ sont donc remplacées par des capacités $envoiOrdre^i$, par lesquelles l'*agence cartographique* peut donner à ses ordres n'importe quelle valeur positive.

C.3.3 Les satellites *S1*, *S2* et *S3*

Les capacités des trois satellites sont similaires entre elles. On factorise leur description en utilisant le paramètre $j \in \{1,2,3\}$ (tableau C.6). Chacun des trois satellites *Sj* peut agir de trois manières : sur sa mémoire du prochain ordre à réaliser ($tâcheSatellite^j$), à laquelle il peut donner à tout moment n'importe quelle valeur, sur sa mémoire embarquée ($mémoireSatellite^j$) qu'il peut remplir en prenant une photographie à tout moment (y correspond donc une capacité par position possible ($positionSatellite^j$) en plus de la capacité à vider cette mémoire) et sur le

$envoiOrdre^c$	*préCond*	$signalSol^i_c = -1$
	fenêtre	$signalSol^i_c \in [0,\dots 5]$
$\{enregistrementPhoto^{i,k}_c\}_{k\in[0,...5]}$	*préCond*	$signalBord^i_c.pos = \mathrm{k}$
	fenêtre	$mémoireSol^i_c.pos = k$
$désactivationOrdre^i_c$	*préCond*	$\top$
	fenêtre	$signalSol^i_c.pos = \text{-}1$
$libérationMémoireSol^i_c$	*préCond*	$\top$
	fenêtre	$mémoireSol^i_c.pos = \text{-}1$

TABLE C.5 – Capacités de l'*agence cartographique*

message envoyé en réponse ($signalSatellite^j$), auquel il peut donner n'importe quelle valeur à tout moment.

$enregistrementOrdre^j$	*préCond*	$\top$
	fenêtre	$tâcheSatellite^j \in [-1,5]$
$\{prisePhoto^j\}_{k\in[0,...,5]}$	*préCond*	$positionSatellite^j = \mathrm{k}$
	fenêtre	$mémoireSatellite^j.pos = k+1[mod6]$
$libérationMémoireSat^j$	*préCond*	$mémoireSatellite^j.pos \geqslant 0$
	fenêtre	$mémoireSatellite^j.pos = -1$
$\{signal^i_{act}\}$	*préCond*	$\top$
	fenêtre	$signalSatellite^j \in \mathscr{E}$

TABLE C.6 – Capacités de *Sj*

C.3.4 Assignation

L'assignation du modèle $\mathsf{Kh}_{C,A}$ est donnée dans la Figure C.3 : l'*agence cartographique* prend en charge elle-même son rôle de *segment sol*, celui pour l'*armée* est confiée au *transmetteur*. Les trois satellites disponibles *S1*, *S2* et *S3*, se répartissent les deux rôles de *segments embarqués* de la manière suivante : *segment embarqué carto* est assigné à la coalition $\{S1,S2\}$ et *segment embarqué arméee* à la coalition $\{S2,S3\}$.

C.3.5 Variables utilisées dans le modèle $\mathsf{Kh}_{C,A}$

On récapitule ici la classification des variables utilisées dans ce modèle (voir Tableau C.7). On note $X_{C,A}$ l'ensemble de ces variables.

Ces variables se répartissent selon trois critères qui sont l'étiquette, le langage (c'est-à-dire la propriété d'être une variables des moyens ou des exigences) et le fait d'être commandé par le sol ou la partie embarquée du système. Dans chaque langage il y a des variables de mémoire à la fois au sol et dans la partie embarquée (variables *imageStockéeBord* et *imageStockéeSol* pour les exigences, *mémoireSatellite* et *mémoireSol* pour les moyens). Ces variables sont étiquetées **image**. Dans chaque langage, des variables étiquetées **image** portent par ailleurs le signal émis de la partie embarquée au sol (il s'agit de la variable *réponse* dans le langage des exigences et des variables *signalSatellite* et *signalBord* dans le langage des moyens). Les variables étiquetées

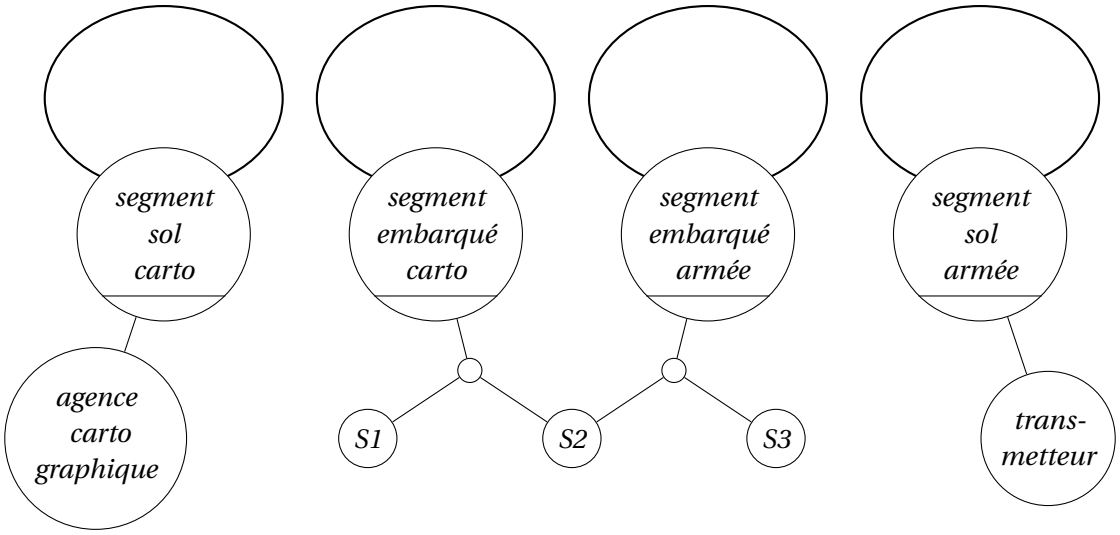

FIGURE C.3 – Assignation pour le modèle $\mathsf{Kh}_{C,A}$

position encodent principalement la position de la prochaine photographie qui doit être prise. Celle-ci se présente d'abord pour *armée* comme un besoin : on utilise les variables *besoin* pour les mentionner dans les exigences, et *besoinPerçu* dans les moyens. Pour les deux missions elle est transmise vers la partie embarquée (un *ordre* du point de vue des exigences, et un *signalSol* dans les moyens) et gardée en mémoire dans la partie embarquée (variables *tâcheEnAttente* et *tâcheSatellite*). S'ajoutent encore l'encodage de la position réelle des satellites (*positionUnitéBord* dans les exigences, et *positionSatellite* dans les moyens).

	exigences		
	$\mathscr{L}$	embarqué	sol
	paramètres	$i \in \{1,2\}, act \in \{c, arm\}$	$i \in \{1,2\}, act \in \{c, arm\}$
$X_{C,A}$	**position**	$positionUnitéBord^i_{act}$ $tâcheEnAttente^i_{act}$	$ordre^i_{act}$ $besoin^i$
	image	$imageStockéeBord^i_{act}$ $réponse^i_{act}$	$imageStockéeSol^i_{act}$
	moyens		
	$\mathscr{L}$	embarqué	sol
	paramètres	$j \in \{1,2,3\}$	$i \in \{1,2\}, act \in \{c, transmetteur\}$
	position	$positionSatellite^j$ $tâcheSatellite^j$	$signalSol^i_{act}$ $besoinPerçu^i$
	image	$mémoireSatellite^j$ $signalSatellite^j$ $signalBord^i_{act}$	$mémoireSol^i_{act}$
pour toute $x \in X_{C,A}$, $x.old$ de même étiquette et même langage pour toute $x \in [x_i : \textbf{position}]$, $x.old$ de même étiquette et même langage			

TABLE C.7 – Tableau des variables utilisées dans le modèle $\mathsf{Kh}_{C,A}$

C.4 Propriétés du domaine

Dans $\mathsf{Kh}_{C,A}$, le domaine est défini à l'aide de trois propriétés : tout d'abord, on initialise les variables de la manière décrite dans la Section C.2. Par ailleurs, le mouvement des satellites est décrit sur la trajectoire schématisée par les entiers de 1 à 5. Enfin, une propriété du domaine indique que les *besoinPerçu* se manifestent toujours en même temps.

C.4.1 Initialisation des variables

L'initialisation des variables est une description de l'état initial du système. La sémantique de $\mathrm{LTL}_{\mathrm{KHI}}$ permet de la décrire avec une formule propositionnelle. Dans l'état initial, les seules variables d'image non vides sont les images $mémoireSol^i_{act}$. Cette initialisation permet d'enclencher le système. De même, les seules variables de position ayant des valeurs positives sont les positions des satellites :

$Form(\textbf{\textit{domPropInit}}) \triangleq$

$$\bigwedge_{x \in X} (x = -1) \wedge \bigwedge_{x \in X'} (x \geqslant 0)$$

où

$$X = \{mémoireSatellite^j.pos\}_{j\in 1,2,3} \cup \{signalSatellite^j.pos\}_{j\in 1,2,3} \cup \{signalBord^i_{act}\}_{act\in\{c,arm\},i\in\{1,2\}} \cup \{signalSol^i_{act}\}_{act\in\{c,arm\},i\in\{1,2\}} \{tâcheSatellite^j\}_{j\in\{1,2,3\}} \cup \{besoinPerçu^i\}_{i\in\{1,2\}}$$

et $X' = \{mémoireSol^i_{act}\}_{act\in\{c,arm\},i\in\{1,2\}} \cup \{positionSatellite^j\}_{j\in\{1,2,3\}}$

C.4.2 Description du mouvement des satellites

On encode le mouvement des satellites par le fait qu'à chaque transition du système, ils avancent d'une unité sur leur trajectoire circulaire :

$Form(\textbf{\textit{domPropMouv}}) \triangleq$

$$\Box(\bigwedge_{j\in\{1,2,3\}} (\mathsf{X}(positionSatellite^j = positionSatellite^j.old + 1 [mod 6])))$$

C.4.3 Encodage du volume des besoins

Notre modélisation considère également que les variables de besoin sont remplies deux par deux.

$Form(\textbf{\textit{domPropBes}}) \triangleq$

$$\Box((besoin^1 = -1 \wedge \mathsf{X}(besoin^1 \geqslant 0)) \leftrightarrow (besoin^2 = -1 \wedge \mathsf{X}(besoin^2 \geqslant 0)))$$

C.5 Correction locale du modèle $\mathsf{Kh}_{C,A}$

On prouve ici la correction locale du modèle $\mathsf{Kh}_{C,A}$. On a en effet la proposition suivante :

Proposition C.1. *Le modèle $Kh_{C,A}$ satisfait* CL *(Rôle).*

Démonstration. La preuve est détaillée dans les sections suivantes : on définit d'abord les affectations Aff_c et Aff_{arm} (Section C.5.1). On en dérive la définition des $\mathrm{CGS}^{cf}_{\mathrm{KHI}}$s $\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_c}$ et $\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_{arm}}$ (Section C.5.2). La formalisation des rôles *segment embarqué carto* et *segment embarqué armée* est donnée dans la Section C.5.3. On décrit dans la Section C.5.4 des spécifications pour des multi-stratégies. Les satellites *S1*, *S2* et *S3* sont en mesure de respecter ces spécifications de multi-stratégies (Section C.5.5). Par ailleurs, ces spécifications ne contredisent pas les propriétés dynamiques du contexte et toute exécution issue d'un contexte où ces spécifications sont respectées est modèle de *Form*(*segment embarqué carto*) et *Form*(*segment embarqué armée*) (Section C.5.6). □

C.5.1 Affectations

On définit d'abord les affectations utilisées pour vérifier la satisfiabilité des rôles *segment embarqué carto* et *segment embarqué armée*. Pour *segment embarqué carto*, on considère l'affectation Aff_c donnée dans le Tableau C.8. Pour *segment embarqué armée*, on utilise l'affectation Aff_{arm} donnée dans le Tableau C.8. Intuivement, dans Aff_c, les variables de capacités pour *S1* sont affectées aux variables qui permettent de répondre à $ordre^1_c$ (on dit que *S1* est affecté à $ordre^1_c$), *S2* est affecté à $ordre^2_c$ et *S3* est affecté à $ordre^1_{arm}$. Dans Aff_{arm}, *S1* est affecté à $ordre^1_c$, *S2* est affecté à $ordre^1_{arm}$ et *S3* est affecté à $ordre^2_{act}$.

Le reste de cette preuve consiste à montrer que :

$$\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_c} \models_{\mathrm{USL}_{\mathrm{KHI}}} \langle\langle \vec{x} \rangle\rangle (\{S1,S2\} \rhd \overrightarrow{x_{\{S1,S2\}}})_{PDC[\mathsf{Aff}_c]} \mathit{Form}(\textit{segment embarqué carto})$$

et

$$\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_{arm}} \models_{\mathrm{USL}_{\mathrm{KHI}}} \langle\langle \vec{x} \rangle\rangle (\{S2,S3\} \rhd \vec{x}_{\{S2,S3\}})_{PDC[\mathsf{Aff}_{arm}]} \mathit{Form}(\textit{segment embarqué armée})$$

C.5.2 Les $\mathrm{CGS}^{cf}_{\mathrm{KHI}}$s $\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_c}$ et $\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_{arm}}$

On définit d'abord les $\mathrm{CGS}^{cf}_{\mathrm{KHI}}$s utilisées pour ces vérifications. Pour *segment embarqué carto*, on utilise la $\mathrm{CGS}^{cf}_{\mathrm{KHI}}$ $\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_c}$ décrite dans la Section C.5.2.1. La $\mathrm{CGS}^{cf}_{\mathrm{KHI}}$ pour la vérification de *segment embarqué armée*, $\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_{arm}}$ est décrite dans la Section C.5.2.2.

C.5.2.1 Le modèle $\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_c}$

On détaille d'abord les différents éléments du modèle

$$\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_c} = \langle Ag_{C,A}, \mathrm{G}_{C,A}, S_{C,A}, At_{C,A}, val_{C,A}, Ac_{C,A,c}, tr_{C,A,c} \rangle :$$

$\boldsymbol{Ag}_{C,A}$ L'ensemble d'agents du modèle est composé des satellites, de l'*agence cartographique* et du *transmetteur* :

$$Ag_{C,A} = \{S1, S2, S3, \textit{agence cartographique}, \textit{transmetteur}\}$$

$\mathbf{G}_{C,A}$ Les états de $\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_{met}}$ sont les valuations possibles pour les variables dans Var(E) qui vérifient les propriétés statiques du contexte :

$$\mathrm{G}_{C,A} = \{\overline{v} \mid v \in dom(\mathrm{Var(E)})^{\mathrm{Var(E)}}, v \models \Gamma \cup \Gamma'\}$$

où

$$
\begin{array}{lcl}
\mathsf{Aff}_c(\mathit{positionSatellite}^1) & = & \{\mathit{positionUnitéBord}_c^1\} \\
\mathsf{Aff}_c(\mathit{positionSatellite}^2) & = & \{\mathit{positionUnitéBord}_c^2\} \\
\mathsf{Aff}_c(\mathit{positionSatellite}^3) & = & \{\mathit{positionUnitéBord}_{arm}^1\} \\
\mathsf{Aff}_c(\mathit{tâcheSatellite}^1) & = & \{\mathit{tâcheEnAttente}_c^1\} \\
\mathsf{Aff}_c(\mathit{tâcheSatellite}^2) & = & \{\mathit{tâcheEnAttente}_c^2\} \\
\mathsf{Aff}_c(\mathit{tâcheSatellite}^3) & = & \{\mathit{tâcheEnAttente}_{arm}^1\} \\
\mathsf{Aff}_c(\mathit{mémoireSatellite}^1) & = & \{\mathit{imageStockéeBord}_c^1\} \\
\mathsf{Aff}_c(\mathit{mémoireSatellite}^2) & = & \{\mathit{imageStockéeBord}_c^2\} \\
\mathsf{Aff}_c(\mathit{mémoireSatellite}^3) & = & \{\mathit{imageStockéeBord}_{arm}^1\} \\
\mathsf{Aff}_c(\mathit{signalSatellite}^1) & = & \{\mathit{réponse}_c^1\} \\
\mathsf{Aff}_c(\mathit{signalSatellite}^2) & = & \{\mathit{réponse}_c^2\} \\
\mathsf{Aff}_c(\mathit{signalSatellite}^3) & = & \{\mathit{réponse}_{arm}^1\} \\
 & & \\
\mathsf{Aff}_c(\mathit{signalSol}_c^1) & = & \{\mathit{ordre}_c^1\} \\
\mathsf{Aff}_c(\mathit{signalSol}_c^2) & = & \{\mathit{ordre}_c^2\} \\
\mathsf{Aff}_c(\mathit{signalSol}_{arm}^1) & = & \{\mathit{ordre}_{arm}^1\} \\
\mathsf{Aff}_c(\mathit{signalSol}_{arm}^2) & = & \{\mathit{ordre}_{arm}^2\} \\
\mathsf{Aff}_c(\mathit{besoinPerçu}^1) & = & \{\mathit{besoin}^1\} \\
\mathsf{Aff}_c(\mathit{besoinPerçu}^2) & = & \{\mathit{besoin}^2\} \\
\mathsf{Aff}_c(\mathit{mémoireSol}_c^1) & = & \{\mathit{imageStockéeSol}_c^1\} \\
\mathsf{Aff}_c(\mathit{mémoireSol}_c^2) & = & \{\mathit{imageStockéeSol}_c^2\} \\
\mathsf{Aff}_c(\mathit{mémoireSol}_{arm}^1) & = & \{\mathit{imageStockéeSol}_{arm}^1\} \\
\mathsf{Aff}_c(\mathit{mémoireSol}_{arm}^2) & = & \{\mathit{imageStockéeSol}_{arm}^2\} \\
\mathsf{Aff}_c(\mathit{signalBord}_c^1) & = & \{\mathit{réponse}_c^1\} \\
\mathsf{Aff}_c(\mathit{signalBord}_c^2) & = & \{\mathit{réponse}_c^2\} \\
\mathsf{Aff}_c(\mathit{signalBord}_{arm}^1) & = & \{\mathit{réponse}_{arm}^1\} \\
\mathsf{Aff}_c(\mathit{signalBord}_{arm}^2) & = & \{\mathit{réponse}_{arm}^2\}
\end{array}
$$

TABLE C.8 – L'affectation Aff_c

- $\Gamma = \{\Box(X = k \rightarrow \mathsf{X}(X.\mathit{old} = k))\}$ avec
  - $k \in [-1, \ldots 5]$,
  - $X \in \{\mathit{positionUnitéBord}_{act}^i, \mathit{ordre}_{act}^i, \mathit{tâcheEnAttente}_{act}^i, \mathit{besoin}^i\}_{act \in \{c,arm\}, i \in \{1,2\}}$
- $\Gamma' = \{\Box(X = Y \rightarrow X.\mathit{pos} = Y.\mathit{pos})\}$ avec

$$X, Y \in \{\mathit{imageStockéeBord}_{act}^i, \mathit{imageStockéeSol}_{act}^i, \mathit{réponse}_{act}^i\}_{act \in \{c,arm\}, i \in \{1,2\}}$$

$S_{C,A}$ Les états initiaux de $\mathscr{G}_{\mathsf{Kh}_{C,A}, \mathsf{Aff}_{met}}$ sont ceux $\mathrm{G}_{C,A}$ qui satisfont les formules données par $\mathit{propDomaine}_i = \{\mathit{domPropInit}\}$:

$$S_{C,A} = \{s \mid\models_{\mathrm{LTL}_{\mathrm{KH}}} \bigwedge_{x \in X[\mathsf{Aff}_c]} (x = -1) \wedge \bigwedge_{x \in X'[\mathsf{Aff}_c]} (x \geqslant 0)\}$$

où les ensembles de variables X et X' sont tels que dans la Section C.4.1.

$\boldsymbol{At}_{C,A}$ L'ensemble des propositions atomiques est donné directement par la Définition 6.5, on n'en donne pas les détails ici :

$$\mathit{At}_{C,A} = \mathit{AtSf}(\{\mathit{Form}(\rho)\}_{\rho \in \mathrm{Rôle}} \cup \mathit{Acteur.peut.préCond}[\mathsf{Aff}_c])$$

$$
\begin{array}{lcl}
\mathsf{Aff}_{arm}(positionSatellite^1) & = & \{positionUnitéBord_c^1\} \\
\mathsf{Aff}_{arm}(positionSatellite^2) & = & \{positionUnitéBord_{arm}^1\} \\
\mathsf{Aff}_{arm}(positionSatellite^3) & = & \{positionUnitéBord_{arm}^2\} \\
\mathsf{Aff}_{arm}(tâcheSatellite^1) & = & \{tâcheEnAttente_c^1\} \\
\mathsf{Aff}_{arm}(tâcheSatellite^2) & = & \{tâcheEnAttente_{arm}^1\} \\
\mathsf{Aff}_{arm}(tâcheSatellite^3) & = & \{tâcheEnAttente_{arm}^2\} \\
\mathsf{Aff}_{arm}(mémoireSatellite^1) & = & \{imageStockéeBord_c^1\} \\
\mathsf{Aff}_{arm}(mémoireSatellite^2) & = & \{imageStockéeBord_{arm}^1\} \\
\mathsf{Aff}_{arm}(mémoireSatellite^3) & = & \{imageStockéeBord_{arm}^2\} \\
\mathsf{Aff}_{arm}(signalSatellite^1) & = & \{réponse_c^1\} \\
\mathsf{Aff}_{arm}(signalSatellite^2) & = & \{réponse_{arm}^1\} \\
\mathsf{Aff}_{arm}(signalSatellite^3) & = & \{réponse_{arm}^2\} \\
 & & \\
\mathsf{Aff}_{arm}(signalSol_c^1) & = & \{ordre_c^1\} \\
\mathsf{Aff}_{arm}(signalSol_c^2) & = & \{ordre_c^2\} \\
\mathsf{Aff}_{arm}(signalSol_{arm}^1) & = & \{ordre_{arm}^1\} \\
\mathsf{Aff}_{arm}(signalSol_{arm}^2) & = & \{ordre_{arm}^2\} \\
\mathsf{Aff}_{arm}(besoinPerçu^1) & = & \{besoin^1\} \\
\mathsf{Aff}_{arm}(besoinPerçu^2) & = & \{besoin^2\} \\
\mathsf{Aff}_{arm}(mémoireSol_c^1) & = & \{imageStockéeSol_c^1\} \\
\mathsf{Aff}_{arm}(mémoireSol_c^2) & = & \{imageStockéeSol_c^2\} \\
\mathsf{Aff}_{arm}(mémoireSol_{arm}^1) & = & \{imageStockéeSol_{arm}^1\} \\
\mathsf{Aff}_{arm}(mémoireSol_{arm}^2) & = & \{imageStockéeSol_{arm}^2\} \\
\mathsf{Aff}_{arm}(signalBord_c^1) & = & \{réponse_c^1\} \\
\mathsf{Aff}_{arm}(signalBord_c^2) & = & \{réponse_c^2\} \\
\mathsf{Aff}_{arm}(signalBord_{arm}^1) & = & \{réponse_{arm}^1\} \\
\mathsf{Aff}_{arm}(signalBord_{arm}^2) & = & \{réponse_{arm}^2\}
\end{array}
$$

TABLE C.9 – L'affectation Aff_{arm}

val$_{C,A}$ De la même manière, la fonction de valuation est donnée par la Définition 6.6 : soit $\overline{v} \in \mathrm{G}_{C,A}$ et $\varphi \in At_{C,A}$, alors $\varphi \in val(\overline{v})$ ssi $v \models_{\mathrm{Cond}} \varphi$

Ac$_{C,A,c}$ On décrit l'ensemble des actions à travers les choix disponibles. L'ordonnancement des choix disponibles, pour tout agent et tout état et la définition de la fonction $Ch_{C,A,c}$ ne sont pas traités ici. On a donc, pour tout agent $a \in Ag_{C,A}$, pour tout état $\overline{v} \in \mathrm{G}_{C,A}$, que $C_c(a,\overline{v})$ est le plus petit ensemble clos par intersection et tel que,

- si $a =$ *transmetteur* alors,
  - si $ordre_{arm}^i = -1$ et $besoin^i = k$, alors $fb_{\mathrm{Var(E)}}(\overline{ordre_{arm}^i \mapsto k,}) \in C_c(a,\overline{v})$, pour $i \in \{1,2\}$ et $k \in [-1,\ldots 5]$,
  - si $réponse_{arm}^i.pos = k$, alors $fb_{\mathrm{Var(E)}}(\overline{imageStockéeSol_{arm}^i.pos \mapsto k,}) \in C_c(a,\overline{v})$, pour $i \in \{1,2\}$ et $k \in [-1,\ldots 5]$,
  - $ordre_{arm}^i = -1 \in C_c(a,\overline{v})$, pour $i \in \{1,2\}$,
  - $imageStockéeSol_{arm}^i.pos = -1 \in C_c(a,\overline{v})$, pour $i \in \{1,2\}$,
- si $a =$ *agence cartographique* alors,

- si $ordre_c^i = -1$, alors $fb_{\mathrm{Var(E)}}(\overline{ordre_c^i \mapsto k,}) \in C_c(a,\overline{v})$, pour $i \in \{1,2\}$ et $k \in [-1,\dots 5]$,
- si $réponse_c^i.pos = k$, alors $fb_{\mathrm{Var(E)}}(\overline{imageStockéeSol_c^i.pos \mapsto k,}) \in C_c(a,\overline{v})$, pour $i \in \{1,2\}$ et $k \in [-1,\dots 5]$,
- $ordre_c^i = -1 \in C_c(a,\overline{v})$, pour $i \in \{1,2\}$,
- $imageStockéeSol_c^i.pos = -1 \in C_c(a,\overline{v})$, pour $i \in \{1,2\}$,

– si $a = S1$ alors,
- $fb_{\mathrm{Var(E)}}(\overline{tâcheEnAttente_c^1 \mapsto k,}) \in C_c(a,\overline{v})$, pour tout $k \in [-1,5]$
- si $positionUnitéBord_c^1 = k$, alors

$$fb_{\mathrm{Var(E)}}(\overline{imageStockéeBord_c^1.pos = k+1[mod6]}) \in C_c(a,\overline{v})$$

- si $imageStockéeBord_c^1 \geqslant 0$, alors

$$fb_{\mathrm{Var(E)}}(\overline{imageStockéeBord_c^1.pos = -1}) \in C_c(a,\overline{v})$$

- $fb_{\mathrm{Var(E)}}(\overline{réponse_c^1 = j}) \in C_c(a,\overline{v})$, pour $j \in \mathscr{E}$

– si $a = S2$ alors,
- $fb_{\mathrm{Var(E)}}(\overline{tâcheEnAttente_c^2 \mapsto k,}) \in C_c(a,\overline{v})$, pour tout $k \in [-1,5]$
- si $positionUnitéBord_c^2 = k$, alors

$$fb_{\mathrm{Var(E)}}(\overline{imageStockéeBord_c^2.pos = k+1[mod6]}) \in C_c(a,\overline{v})$$

- si $imageStockéeBord_c^2 \geqslant 0$, alors

$$fb_{\mathrm{Var(E)}}(\overline{imageStockéeBord_c^2.pos = -1}) \in C_c(a,\overline{v})$$

- $fb_{\mathrm{Var(E)}}(\overline{réponse_c^2 = j}) \in C_c(a,\overline{v})$, pour $j \in \mathscr{E}$

– si $a = S3$ alors,
- $fb_{\mathrm{Var(E)}}(\overline{tâcheEnAttente_{arm}^1 \mapsto k,}) \in C_c(a,\overline{v})$, pour tout $k \in [-1,5]$
- si $positionUnitéBord_{arm}^1 = k$, alors

$$fb_{\mathrm{Var(E)}}(\overline{imageStockéeBord_{arm}^1.pos = k+1[mod6]}) \in C_c(a,\overline{v})$$

- si $imageStockéeBord_{arm}^1 \geqslant 0$, alors

$$fb_{\mathrm{Var(E)}}(\overline{imageStockéeBord_{arm}^1.pos = -1}) \in C_c(a,\overline{v})$$

- $fb_{\mathrm{Var(E)}}(\overline{réponse_{arm}^1 = j}) \in C_c(a,\overline{v})$, pour $j \in \mathscr{E}$

tr$_{C,A,c}$ La fonction de transition est donnée de manière immédiate, par application de la Définition 6.10 :

$$dom(tr)_{C,A,c} = \{(\overline{v},dc) \mid \bigcap_{a \in Ag_{C,A}} C_{c,C,A,c}(a,\overline{v},dc(a))\} \neq \varnothing$$

et, pour tout $(\overline{v},dc) \in dom(tr_{C,A})$,

$$tr_{C,A}(\overline{v},dc) = \bigcap_{a \in Ag_{C,A}} Ch_{C,A}(a,\overline{v},dc(a)) \cap \overline{fb_X(v_{\restriction X_{Pass}})}$$

avec $X_{Pass} = \{x_k \in X \mid \forall\, a \in Ag_{C,A}\, x_k \notin \mathrm{base}(dc(a))\}$.

C.5.2.2 Le modèle $\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_{arm}}$

Le modèle $\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_{arm}} = \langle Ag_{C,A}, \mathrm{G}_{C,A,arm}, S_{C,A,arm}, At_{C,A}, val_{C,A}, Ac_{C,A,arm}, tr_{C,A,arm}\rangle$ est obtenu de manière similaire à $\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_{arm}}$: l'ensemble d'agents et l'ensemble des propositions atomiques et la valuation sont identiques. L'ensemble des états et l'ensemble des états initiaux sont obtenus de manière similaire, en utilisant Aff_{arm} à la place de Aff_c pour interpréter les variables des moyens. On décrit à nouveau l'ensemble des actions à travers les choix disponibles pour les acteurs. Ils ne différent de ceux de $\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_{arm}}$ que pour les agents *S2* et *S3* :

- si $a = S2$ alors,
  - $fb_{\mathrm{Var(E)}}(\overline{tâcheEnAttente^1_{arm} \mapsto k,}) \in C_{arm}(a, \overline{v})$, pour tout $k \in [-1,5]$
  - si $positionUnitéBord^1_{arm} = k$, alors

$$fb_{\mathrm{Var(E)}}(\overline{imageStockéeBord^1_{arm}.pos = k+1[mod6]}) \in C_{arm}(a, \overline{v})$$

  - si $imageStockéeBord^1_{arm} \geqslant 0$, alors

$$fb_{\mathrm{Var(E)}}(\overline{imageStockéeBord^1_{arm}.pos = -1}) \in C_{arm}(a, \overline{v})$$

  - $fb_{\mathrm{Var(E)}}(\overline{réponse^2_c = j}) \in C_{arm}(a, \overline{v})$, pour $j \in \mathscr{E}$
- si $a = S3$ alors,
  - $fb_{\mathrm{Var(E)}}(\overline{tâcheEnAttente^2_{arm} \mapsto k,}) \in C_{arm}(a, \overline{v})$, pour tout $k \in [-1,5]$
  - si $positionUnitéBord^2_{arm} = k$, alors

$$fb_{\mathrm{Var(E)}}(\overline{imageStockéeBord^2_{arm}.pos = k+1[mod6]}) \in C_{arm}(a, \overline{v})$$

  - si $imageStockéeBord^2_{arm} \geqslant 0$, alors

$$fb_{\mathrm{Var(E)}}(\overline{imageStockéeBord^2_{arm}.pos = -1}) \in C_{arm}(a, \overline{v})$$

  - $fb_{\mathrm{Var(E)}}(\overline{réponse^2_{arm} = j}) \in C_{arm}(a, \overline{v})$, pour $j \in \mathscr{E}$

La fonction de transition $tr_{C,A,arm}$ est définie de manière similaire à $tr_{C,A,c}$, en utilisant la fonction C_{arm} à la place de C_c.

C.5.3 Formalisation des rôles *segment embarqué carto* et *segment embarqué armée*

Pour écrire la formalisation des rôles *segment embarqué carto* et *segment embarqué armée*, on définit d'abord la formule Ψ^i_{act}, pour $act \in \{c, arm\}$ et $i \in \{1,2\}$:

$$\Psi^{i}_{act} \triangleq$$
$$\Box(\mathit{tâcheEnAttente}^{i}_{act} = -1 \rightarrow ((\mathit{ordre}^{i}_{act} \geqslant 0 \wedge \mathit{ordre}^{i}_{act}.\mathit{old} = -1)$$
$$\leftrightarrow (\mathsf{X}((\mathit{tâcheEnAttente}^{i}_{act} \geqslant 0) \wedge (\mathit{tâcheEnAttente}^{i}_{act} = \mathit{ordre}^{i}_{act}.\mathit{old}) \wedge (\mathit{réponse}^{i}_{act} = -1)))))$$
$$\wedge \Box$$
$$(\mathit{tâcheEnAttente}^{i}_{act} \geqslant 0 \rightarrow ((\mathit{imageStockéeBord}^{i}_{act} = -1 \vee \mathit{positionUnitéBord}^{i}_{act} < 5)$$
$$\leftrightarrow (\mathsf{X}(\mathit{tâcheEnAttente}^{i}_{act} = \mathit{tâcheEnAttente}^{i}_{act}.\mathit{old}))))$$
$$\wedge \Box$$
$$(\mathit{imageStockéeBord}^{i}_{act}.\mathit{pos} = -1 \rightarrow$$
$$((\mathit{tâcheEnAttente}^{i}_{act} \geqslant 0 \wedge \mathit{tâcheEnAttente}^{i}_{act} = \mathit{positionUnitéBord}^{i}_{act})$$
$$\leftrightarrow (\mathsf{X}((\mathit{imageStockéeBord}^{i}_{act}.\mathit{pos} \geqslant 0) \wedge (\mathit{imageStockéeBord}^{i}_{act}.\mathit{pos} = \mathit{tâcheEnAttente}^{i}_{act}.\mathit{old})))))$$
$$\wedge \Box$$
$$(\mathit{imageStockéeBord}^{i}_{act}.\mathit{pos} \geqslant 0 \rightarrow ((\mathit{positionUnitéBord}^{i}_{act} < 5)$$
$$\leftrightarrow (\mathsf{X}((\mathit{imageStockéeBord}^{i}_{act}.\mathit{pos} \geqslant 0) \wedge$$
$$(\mathit{imageStockéeBord}^{i}_{act}.\mathit{pos} = \mathit{imageStockéeBord}^{i}_{act}.\mathit{pos}.\mathit{old})))))$$
$$\wedge \Box$$
$$(\mathit{réponse}^{i}_{arm}.\mathit{pos} = -1 \rightarrow (((\mathit{imageStockéeBord}^{i}_{act}.\mathit{pos} \geqslant 0) \wedge (\mathit{positionUnitéBord}^{i}_{act} = 5))$$
$$\leftrightarrow (\mathsf{X}((\mathit{réponse}^{i}_{act}.\mathit{pos} \geqslant 0) \wedge (\mathit{réponse}^{i}_{act}.\mathit{pos} = \mathit{imageStockéeBord}^{i}_{act}.\mathit{pos}.\mathit{old})$$
$$\wedge (\mathit{imageStockéeBord}^{i}_{act}.\mathit{pos} = -1) \wedge (\mathit{tâcheEnAttente}^{i}_{act} = -1)))))$$

Avec les spécifications d'opérations données dans le Tableau C.3 de l'Annexe C, on vérifie que

$$\mathit{Form}(\mathit{segment\ embarqué\ carto}) \leftrightarrow \Psi^{1}_{c} \wedge \Psi^{2}_{c}$$

et

$$\mathit{Form}(\mathit{segment\ embarqué\ armée}) \leftrightarrow \Psi^{1}_{arm} \wedge \Psi^{2}_{arm}$$

C.5.4 Spécifications des multi-stratégies pour les satellites

On écrit également ci-dessous des *spécifications de multi-stratégies* $\mathit{specStrat}^{j}_{act,i}$, pour $\mathit{act} \in \{c, arm\}$, $i \in \{1,2\}$ et $j \in \{1,2,3\}$. La spécification $\mathit{specStrat}^{j}_{act,i}$ décrit les issues possibles quand le satellite *Sj* joue une multi-stratégie qui la respecte. On verra que pour $\mathit{act} \in \{c, arm\}$, $i \in \{1,2\}$ et $j \in \{1,2,3\}$, si *Sj* joue une multi-stratégie qui respecte $\mathit{specStrat}^{j}_{act,i}$, alors la satisfaction de Ψ^{i}_{act} est assurée modulo *PDC*.

Les affectations Aff_{c} et Aff_{arm} transforment de la même manière les propriétés dynamiques du contexte : $\mathit{PDC}[\mathsf{Aff}_{c}]$ et $\mathit{PDC}[\mathsf{Aff}_{arm}]$. On note $\mathit{Ext}(\mathit{PDC})$ l'ensemble des exécutions possiblement finies qui sont modèles de ces formules propriétés dynamiques du contexte. Il est formé exclusivement d'exécutions infinies et il est défini formellement par : pour toute exécution λ, $\lambda \in \mathit{Ext}(\mathit{PDC})$ ssi pour tout $n \in \mathbb{N}$,

– pour toute variable x de Var(E), si $\lambda_n \models_{\mathit{Cond}} x = k$, alors $\lambda_{n+1} \models_{\mathit{Cond}} x.\mathit{old} = k$

– pour tout $i \in \{1,2\}, act \in \{c, arm\}$, pour tout $k \in [0,\dots,5$, si $\lambda_n \models_{Cond} positionUnitéBord^i_{act} = k$ alors $\lambda_{n+1} \models_{Cond} positionUnitéBord^i_{act} = k$

Définition C.1. *Un acteur $a \in \{S1, S2, S3\}$ remplit la spécification de multi-stratégie $specStrat^j_{act,i}$ avec la multi-stratégie σ ssi σ est telle que pour tout état $\overline{v}$, $Ch(a, \overline{v}, \sigma(\overline{v})) \cap Ext(PDC) \neq \varnothing$ et $Ch(a, \overline{v}, \sigma(\overline{v})) \cap Ext(PDC) \subseteq$*

– $fb_{Var(E)}(\overline{tâcheEnAttente^i_{act} \mapsto k, réponse^i_{act} \mapsto -1})$ *pour* $k \in [0,\dots 5]$, *si*

$$\overline{v} \models_{Cond} tâcheEnAttente^i_{act} = -1 \wedge ordre^i_{act} = k \wedge ordre^i_{act}.old = -1$$

– $fb_{Var(E)}(\overline{tâcheEnAttente^i_{act} \mapsto k})$ *pour* $k \in [0,\dots 5]$, *si*

$$\overline{v} \models_{Cond} tâcheEnAttente^i_{act} = k \wedge (imageStockéeBord^i_{act} = -1 \vee positionUnitéBord^i_{act} < 5)$$

– $fb_{Var(E)}(\overline{imageStockéeBord^i_{act}.pos \mapsto k})$ *pour* $k \in [0,\dots 5]$, *si*

$$\overline{v} \models_{Cond} imageStockéeBord^i_{act}.pos = -1 \wedge tâcheEnAttente^i_{act} = positionUnitéBord^i_{act} = k$$

– $fb_{Var(E)}(\overline{imageStockéeBord^i_{act}.pos \mapsto k})$ *pour* $k \in [0,\dots 5]$, *si*

$$\overline{v} \models_{Cond} imageStockéeBord^i_{act}.pos = k \wedge positionUnitéBord^i_{act} < 5$$

– $fb_{Var(E)}(\overline{réponse^i_{act}.pos \mapsto k, imageStockéeBord^i_{act}.pos \mapsto -1, tâcheEnAttente^i_{act} \mapsto -1}))$ *pour* $k \in [0,\dots 5]$, *si*

$$\overline{v} \models_{Cond} réponse^i_{arm}.pos = -1 \wedge imageStockéeBord^i_{act}.pos = k \wedge positionUnitéBord^i_{act} = 5)$$

C.5.5 Les satellites peuvent jouer des multi-stratégies qui respectent les spécifications

On observe que :

– Sous Aff_c :

  – *S1* peut jouer une multi-stratégie $\sigma^1_{c,c,1}$, qui respecte $specStrat^1_c$: on le vérifie en comparant, item par item les choix disponibles pour *S1* dans $\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_c}$, décrits dans la Section C.5.2.1 avec les termes de la Définition C.1 appliqués au cas $act = c, i = 1$ et $j = 1$. De la même manière :

  – *S2* peut jouer une multi-stratégie $\sigma^2_{c,c,2}$, qui respecte $specStrat^2_c$

– Sous Aff_{arm},

  – *S2* peut jouer une multi-stratégie $\sigma^2_{arm,arm,1}$, qui respecte $specStrat^1_{arm}$

  – *S3* peut jouer une multi-stratégie $\sigma^3_{arm,arm,2}$, qui respecte $specStrat^2_{arm}$

C.5.6 Les exécutions issues de ces multi-stratégies satisfont les rôles

On observe que le respect de ces spécifications dans une exécution infinie assure la satisfaction des rôles de segments bords. Plus précisément, pour $act \in \{c, arm\}$ et $i \in \{1,2\}$, toute exécution λ infinie issue d'un contexte dans lequel un satellite Sj joue une multi-stratégie qui respecte $specStrat^{j}_{act,i}$ est telle que $\lambda \models_{\mathrm{LTL}_{\mathrm{KHI}}} \Psi^{i}_{act}$. Formellement, pour tout $s_0 \in S_{C,A}$:

- $out^{cf}(\langle(x_1 \mapsto \sigma^{1}_{c,c,1}), \langle S1, x_1\rangle\rangle, s_0) \cap Ext(PDC) \neq \varnothing$ et

$$\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_c}, \langle(x_1 \mapsto \sigma^{1}_{c,c,1}), \langle S1, x_1\rangle\rangle \models_{\mathrm{USL}_{\mathrm{KHI}}} \Psi^{1}_{c}$$

- $out^{cf}(\langle(x_2 \mapsto \sigma^{2}_{c,c,2}), \langle S2, x_2\rangle\rangle, s_0) \cap Ext(PDC) \neq \varnothing$ et

$$\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_c}, \langle(x_2 \mapsto \sigma^{2}_{c,c,2}), \langle S2, x_2\rangle\rangle \models_{\mathrm{USL}_{\mathrm{KHI}}} \Psi^{2}_{c}$$

- $out^{cf}(\langle(x_2 \mapsto \sigma^{2}_{arm,arm,1}), \langle S2, x_2\rangle\rangle, s_0) \cap Ext(PDC) \neq \varnothing$ et

$$\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_{arm}}, \langle(x_2 \mapsto \sigma^{2}_{arm,arm,1}), \langle S2, x_2\rangle\rangle \models_{\mathrm{USL}_{\mathrm{KHI}}} \Psi^{1}_{arm}$$

- $out^{cf}(\langle(x_3 \mapsto \sigma^{3}_{arm,arm,2}), \langle S3, x_3\rangle\rangle, s_0) \cap Ext(PDC) \neq \varnothing$ et

$$\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_{arm}}, \langle(x_3 \mapsto \sigma^{3}_{arm,arm,2}), \langle S3, x_3\rangle\rangle \models_{\mathrm{USL}_{\mathrm{KHI}}} \Psi^{2}_{arm}$$

On observe enfin que les acteurs dans $\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_{arm}}$ n'ont pas de capacité conflictuelle. Toutes les exécutions du système sont donc infinies. On a finalement :

$$\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_c} \models_{\mathrm{USL}_{\mathrm{KHI}}} \langle\langle \vec{x} \rangle\rangle(\{S1, S2\} \rhd \overrightarrow{x_{\{S1,S2\}}})_{PDC[\mathsf{Aff}_c]} Form(\textit{segment embarqué carto})$$

et

$$\mathscr{G}_{\mathsf{Kh}_{C,A},\mathsf{Aff}_{arm}} \models_{\mathrm{USL}_{\mathrm{KHI}}} \langle\langle \vec{x} \rangle\rangle(\{S2, S3\} \rhd \overrightarrow{x_{\{S2,S3\}}})_{PDC[\mathsf{Aff}_{arm}]} Form(\textit{segment embarqué armée})$$

C.6 Variantes du modèle $\mathsf{Kh}_{C,A}$

On utilise ici différentes variantes du modèle $\mathsf{Kh}_{C,A}$ pour illustrer les critères de correction de l'assignation. Dans une première variantes, $\mathsf{Kh}_{S_{mem}}$, un nouveau satellite est introduit dans les acteurs, capable de traiter simultanément deux ordres. On s'intéresse à la correction globale et à la relation de contribution dans ce modèle. L'ensemble des rôles pour l'*agence cartographique* ne contribue pas globalement à l'ensemble des rôles pour l'*armée* (Section C.6.1). Cette contribution est obtenue dans le modèle Kh_{Secur}, par l'introduction d'une exigence de sécurité dans le modèle de buts pour l'*armée* (Section C.6.2). Enfin, on introduit un troisième modèle de buts dans le modèle $\mathsf{Kh}_{C,A,G}$. On peut ainsi illustrer les relations de collaboration locale et globale (Section C.6.3).

C.6.1 Le modèle $\mathsf{Kh}_{S_{mem}}$: utilisation de satellites à mémoire étendue

Dans cette première variante, on introduit un nouveau satellite, S_{mem}. Ce satellite est différent des satellites utilisés dans le modèle $\mathsf{Kh}_{C,A}$: il peut stocker deux ordres de photographies ainsi que deux images, qu'ils peut renvoyer au sol sur deux signaux distincts. Les capacités de

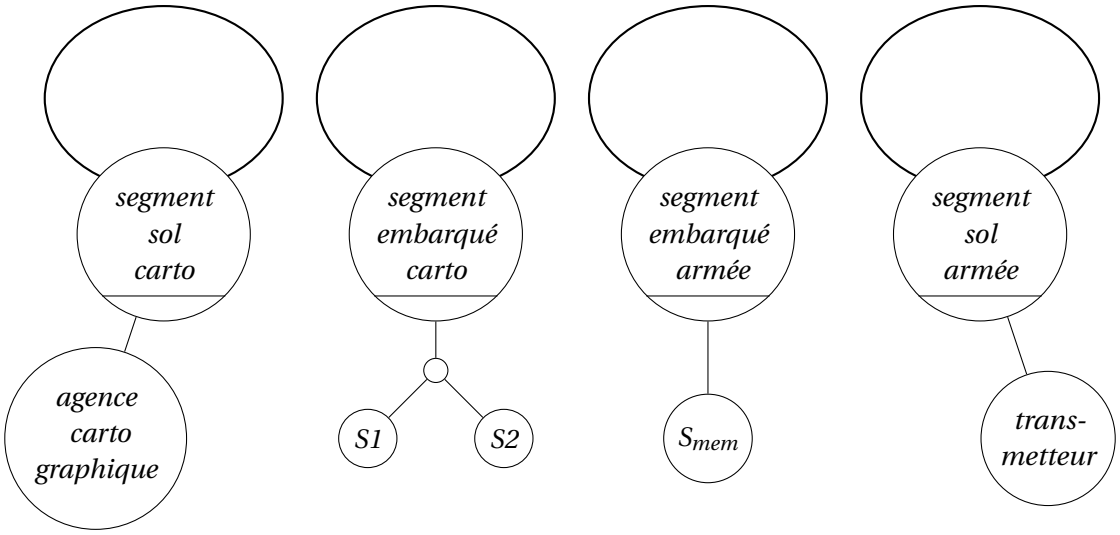

FIGURE C.4 – Assignation pour le modèle $\mathsf{Kh}_{S_{mem}}$

$enregistrementOrdre_1^{S_{mem}}$	*préCond*	$\top$
	fenêtre	$tâcheSatellite_1^{S_{mem}} \in [-1,\ldots,5]$
$enregistrementOrdre_2^{S_{mem}}$	*préCond*	$\top$
	fenêtre	$tâcheSatellite_2^{S_{mem}} \in [-1,\ldots,5]$
$\{prisePhoto_1^{S_{mem}}\}_{k\in[0,\ldots,5]}$	*préCond*	$positionSatellite_j^{S_{mem}} = \mathrm{k}$
	fenêtre	$mémoireSatellite_1^{S_{mem}} = k+1\,[mod 6]$
$\{prisePhoto_2^{S_{mem}}\}_{k\in[0,\ldots,5]}$	*préCond*	$positionSatellite_j^{S_{mem}} = \mathrm{k}$
	fenêtre	$mémoireSatellite_2^{S_{mem}} = k+1\,[mod 6]$
$libérationMémoireSat_1^{S_{mem}}$	*préCond*	$mémoireSatellite_j^{S_{mem}}.pos \geqslant 0$
	fenêtre	$mémoireSatellite_1^{S_{mem}}.pos = -1$
$libérationMémoireSat_2^{S_{mem}}$	*préCond*	$mémoireSatellite_j^{S_{mem}}.pos \geqslant 0$
	fenêtre	$mémoireSatellite_2^{S_{mem}}.pos = -1$
$signal^1$	*préCond*	$\top$
	fenêtre	$signalSatellite_1^{S_{mem}} \in \mathscr{E}$
$signal^2$	*préCond*	$\top$
	fenêtre	$signalSatellite_2^{S_{mem}} \in \mathscr{E}$

TABLE C.10 – Capacités de S_{mem}

S_{mem} sont données dans le Tableau C.10. On ne les commente pas ici : leur description dans le Tableau C.10 est similaire à celle pour les capacités des autres satellites. On considère alors l'assignation : $assign_{\mathsf{Kh}_{S_{mem}}}$ représentée dans la Figure C.4. Dans cette assignation, S_{mem} remplace *S3* dans le rôle *segment embarqué armée*. Par ailleurs, *S2* ne participe plus à la coalition pour *segment embarqué carto*.

On observe que le modèle $\mathsf{Kh}_{S_{mem}}$ satisfait CG(Rôle) mais ne satisfait pas Contr (*segment embarqué carto*, *segment embarqué armée*) :

Proposition C.2. *Le modèle $Kh_{S_{mem}}$ satisfait* CG *(Rôle).*

Résumé. Les détails de la preuve ne sont pas donnés ici. Intuitivement, la disposition de deux variables de mémoire pour S_{mem} lui permet d'assurer à lui seul la satisfaction du rôle *segment embarqué armée*. Dès lors, les capacités de *S2* peuvent être consacrées à sa participation à la coalition pour *segment embarqué carto*. L'affectation $\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}}}$ est donnée dans le Tableau C.11.

$$
\begin{array}{lcl}
\mathsf{Aff}_c(positionSatellite^1) & = & \{positionUnitéBord_c^1\} \\
\mathsf{Aff}_c(positionSatellite^2) & = & \{positionUnitéBord_c^2\} \\
\mathsf{Aff}_c(positionSatellite_1^{S_{mem}}) & = & \{positionUnitéBord_{arm}^1\} \\
\mathsf{Aff}_c(positionSatellite_2^{S_{mem}}) & = & \{positionUnitéBord_{arm}^2\} \\
\mathsf{Aff}_c(tâcheSatellite^1) & = & \{tâcheEnAttente_c^1\} \\
\mathsf{Aff}_c(tâcheSatellite^2) & = & \{tâcheEnAttente_c^2\} \\
\mathsf{Aff}_c(enregistrementOrdre_1^{S_{mem}}) & = & \{tâcheEnAttente_{arm}^1\} \\
\mathsf{Aff}_c(enregistrementOrdre_2^{S_{mem}}) & = & \{tâcheEnAttente_{arm}^2\} \\
\mathsf{Aff}_c(mémoireSatellite^1) & = & \{imageStockéeBord_c^1\} \\
\mathsf{Aff}_c(mémoireSatellite^2) & = & \{imageStockéeBord_c^2\} \\
\mathsf{Aff}_c(mémoireSatellite_1^{S_{mem}}) & = & \{imageStockéeBord_{arm}^1\} \\
\mathsf{Aff}_c(mémoireSatellite_2^{S_{mem}}) & = & \{imageStockéeBord_{arm}^2\} \\
\mathsf{Aff}_c(signalSatellite^1) & = & \{réponse_c^1\} \\
\mathsf{Aff}_c(signalSatellite^2) & = & \{réponse_c^2\} \\
\mathsf{Aff}_c(signalSatellite_1^{S_{mem}}) & = & \{réponse_{arm}^1\} \\
\mathsf{Aff}_c(signalSatellite_2^{S_{mem}}) & = & \{réponse_{arm}^2\} \\
 & & \\
\mathsf{Aff}_c(signalSol_c^1) & = & \{ordre_c^1\} \\
\mathsf{Aff}_c(signalSol_c^2) & = & \{ordre_c^2\} \\
\mathsf{Aff}_c(signalSol_{arm}^1) & = & \{ordre_{arm}^1\} \\
\mathsf{Aff}_c(signalSol_{arm}^2) & = & \{ordre_{arm}^2\} \\
\mathsf{Aff}_c(besoinPerçu^1) & = & \{besoin^1\} \\
\mathsf{Aff}_c(besoinPerçu^2) & = & \{besoin^2\} \\
\mathsf{Aff}_c(mémoireSol_c^1) & = & \{imageStockéeSol_c^1\} \\
\mathsf{Aff}_c(mémoireSol_c^2) & = & \{imageStockéeSol_c^2\} \\
\mathsf{Aff}_c(mémoireSol_{arm}^1) & = & \{imageStockéeSol_{arm}^1\} \\
\mathsf{Aff}_c(mémoireSol_{arm}^2) & = & \{imageStockéeSol_{arm}^2\} \\
\mathsf{Aff}_c(signalBord_c^1) & = & \{réponse_c^1\} \\
\mathsf{Aff}_c(signalBord_c^2) & = & \{réponse_c^2\} \\
\mathsf{Aff}_c(signalBord_{arm}^1) & = & \{réponse_{arm}^1\} \\
\mathsf{Aff}_c(signalBord_{arm}^2) & = & \{réponse_{arm}^2\}
\end{array}
$$

TABLE C.11 – L'affectation $\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}}}$

Le reste de la preuve procède de manière similaire à celle pour la Proposition C.5 : on définit la $\mathrm{CGS}^{cf}_{\mathrm{KHI}}\ \mathscr{G}_{\mathsf{Kh}_{S_{mem}},\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}}}}$, on spécifie les multi-stratégies qui doivent être jouées par les différents acteurs pour que les rôles soient assurés. On vérifie que les acteurs auxquels chacun des rôles est assigné peuvent jouer en même temps des multi-stratégies respectant les spécifications correspondantes modulo les propriétés dynamiques du contexte. Enfin, on vérifie que les acteurs

n'ont pas de capacité conflictuelle dans $\mathscr{G}_{\mathsf{Kh}_{S_{mem}},\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}}}}$. On vérifie ainsi que :

$$\begin{aligned}
&\mathscr{G}_{\mathsf{Kh}_{S_{mem}},\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}}}} \models_{\mathrm{USL}_{\mathrm{KH1}}} \langle\langle \overrightarrow{x} \rangle\rangle (\\
&\quad ((\mathit{agence\ cartographique} \rhd \overrightarrow{x_{\{\mathit{agence\ cartographique}\}}})_{PDC[\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}}}]} \mathit{Form}(\mathit{segment\ sol\ carto}))\\
&\qquad \wedge ((\{S1,S2\} \rhd \overrightarrow{x_{\{S1,S2\}}})_{PDC[\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}}}]} \mathit{Form}(\mathit{segment\ embarqué\ carto}))\\
&\qquad \wedge ((\{S2,S_{mem}\} \rhd \overrightarrow{x_{\{S2,S_{mem}\}}})_{PDC[\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}}}]} \mathit{Form}(\mathit{segment\ embarqué\ armée}))\\
&\qquad \wedge ((\mathit{transmetteur} \rhd \overrightarrow{x_{\{\mathit{transmetteur}\}}})_{PDC[\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}}}]} \mathit{Form}(\mathit{segment\ sol\ armée}))
\end{aligned}$$

□

Le modèle $\mathsf{Kh}_{S_{mem}}$ est donc correct, au sens de la correction globale. C'est-à-dire qu'il y a une manière d'affecter globalement les variables des acteurs et d'utiliser globalement leurs capacités en leur fournissant des multi-stratégies, de manière à ce que chacun des rôles ρ soit assuré par la coalition $\mathit{assign}_{\mathsf{Kh}_{S_{mem}}}(\rho)$. Un ingénieur peut donc spécifier le comportement de ces acteurs de manière à garantir l'ensemble des rôles. L'utilisation de ce critère n'a cependant de pertinence que s'il existe effectivement un ingénieur unique qui puisse spécifier le comportement de tous les acteurs (ou s'il y a un accord sur ces comportements entre les différents ingénieurs). Dans le cas inverse, on s'intéressera à la satisfaction de critères de contribution pour chacun des ingénieurs. Par exemple, les acteurs pour les rôles liés à l'*armée* dispose d'une certaine autonomie si les rôles pour l'*agence cartographique* contribuent à ceux pour l'*armée* : dans ce cas seulement un ingénieur contrôlant les acteurs dans les coalitions pour les rôles pour l'*armée* peut s'assurer de leur satisfaction, pourvu que les les acteurs dans les coalitions pour les rôles pour l'*agence cartographique* recherchent en effet la satisfaction de ces rôles.

On cherche par exemple à savoir si les acteurs qui composent les coalitions pour l'*agence cartographique* peuvent agir de manière à satisfaire les rôles *segment embarqué carto* et *segment sol carto* tout en empêchant les acteurs *S2* et *transmetteur* d'assurer la satisfaction des rôles *segment embarqué armée* et *segment sol armée*. Or c'est le cas, on vérifie en effet la proposition suivante :

Proposition C.3. *Le modèle $Kh_{S_{mem}}$ ne satisfait pas*

$$\mathsf{Contr}_G(\{\mathit{segment\ embarqué\ carto}, \mathit{segment\ sol\ carto}\}, \{\mathit{segment\ embarqué\ armée}, \mathit{segment\ sol\ armée}\})$$

Démonstration. La contribution n'a pas lieu entre les rôles pour l'*agence cartographique* et ceux pour l'*armée*, en particulier parce qu'il y a des assignations par lesquelles les agents dans les coalitions pour l'*agence cartographique* agissent en même temps sur les variables des rôles pour l'*agence cartographique* et sur celles des rôles pour l'*armée*.

Observons par exemple l'affectation $\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}$ donnée dans le Tableau C.12,

$$\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(\mathit{signalSatellite}^1) = \{\mathit{réponse}^1_c, \mathit{réponse}^1_{arm}, \mathit{réponse}^2_{arm}\}.$$

Dans cette affectation, quand il envoie des réponses aux ordres ordre^1_c en remplissant la variable $\mathit{réponse}^1_c$, *S1* émet également la même réponse sur les variables $\mathit{réponse}^1_{arm}$ et $\mathit{réponse}^2_{arm}$. Ce faisant, *S1* empêche la coalition pour le rôle *segment embarqué armée* de contrôler ces deux variables et donc de remplir son rôle.

$$
\begin{array}{lcl}
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(positionSatellite^1) & = & \{positionUnitéBord_c^1\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(positionSatellite^2) & = & \{positionUnitéBord_c^2\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(positionSatellite_1^{S_{mem}}) & = & \{positionUnitéBord_{arm}^1\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(positionSatellite_2^{S_{mem}}) & = & \{positionUnitéBord_{arm}^2\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(tâcheSatellite^1) & = & \{tâcheEnAttente_c^1\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(tâcheSatellite^2) & = & \{tâcheEnAttente_c^2\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(enregistrementOrdre_1^{S_{mem}}) & = & \{tâcheEnAttente_{arm}^1\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(enregistrementOrdre_2^{S_{mem}}) & = & \{tâcheEnAttente_{arm}^2\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(mémoireSatellite^1) & = & \{imageStockéeBord_c^1\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(mémoireSatellite^2) & = & \{imageStockéeBord_c^2\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(mémoireSatellite_1^{S_{mem}}) & = & \{imageStockéeBord_{arm}^1\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(mémoireSatellite_2^{S_{mem}}) & = & \{imageStockéeBord_{arm}^2\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(signalSatellite^1) & = & \{réponse_c^1, réponse_{arm}^1, réponse_{arm}^2\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(signalSatellite^2) & = & \{réponse_c^2\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(signalSatellite_1^{S_{mem}}) & = & \{réponse_{arm}^1\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(signalSatellite_2^{S_{mem}}) & = & \{réponse_{arm}^2\} \\
 & & \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(signalSol_c^1) & = & \{ordre_c^1\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(signalSol_c^2) & = & \{ordre_c^2\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(signalSol_{arm}^1) & = & \{ordre_{arm}^1\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(signalSol_{arm}^2) & = & \{ordre_{arm}^2\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(besoinPerçu^1) & = & \{besoin^1\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(besoinPerçu^2) & = & \{besoin^2\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(mémoireSol_c^1) & = & \{imageStockéeSol_c^1\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(mémoireSol_c^2) & = & \{imageStockéeSol_c^2\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(mémoireSol_{arm}^1) & = & \{imageStockéeSol_{arm}^1\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(mémoireSol_{arm}^2) & = & \{imageStockéeSol_{arm}^2\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(signalBord_c^1) & = & \{réponse_c^1\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(signalBord_c^2) & = & \{réponse_c^2\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(signalBord_{arm}^1) & = & \{réponse_{arm}^1\} \\
\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}(signalBord_{arm}^2) & = & \{réponse_{arm}^2\}
\end{array}
$$

TABLE C.12 – L'affectation $\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}$

On a alors que ce n'est pas le cas que pour toute assignation de variables α de domaine un vecteur de variable $\{x_{agence\ cartographique}, x_{S1}, x_{S2}\}$ telle que

$$
\begin{array}{l}
\mathscr{G}_{\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}}, \alpha \models_{\mathrm{USL}_{\mathrm{KHI}}} (S1 \rhd x_{S1})_{PDC[\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}]} (S2 \rhd x_{S2})_{PDC[\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}]} \\
\qquad (agence\ cartographique \rhd x_{agence\ cartographique})_{PDC[\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}]} \\
\qquad\qquad (Form(segment\ embarqué\ carto) \wedge Form(segment\ sol\ carto))
\end{array}
$$

il y a une affectation $\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},3}$ de domaine $\mathsf{Var}(S_{mem}, \textit{transmetteur})$ telle que :

$$\mathscr{G}_{\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}} \cup \mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},3}, \alpha \models_{\mathrm{USL}_{\mathrm{KH}}} (S1 \rhd x_{S1})_{PDC[\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}]} (S2 \rhd \mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2})_{PDC[x_{S2}]}$$
$$(\textit{agence cartographique} \rhd x_{\textit{agence cartographique}})_{PDC[\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}]} (\langle\langle x_{S2}\rangle\rangle \langle\langle x_{S_{mem}}\rangle\rangle$$
$$(S2 \rhd x_{S2})_{PDC[\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}]} (S_{mem} \rhd x_{S_{mem}})_{PDC[\mathsf{Aff}_{\mathsf{Kh}_{S_{mem}},2}]} \textit{Form}(\textit{segment embarqué armée}))$$

□

Dans le modèle $\mathsf{Kh}_{S_{mem}}$ donc, les acteurs dans les coalitions pour les rôles de l'*armée* dépendent des affectations de variables des acteurs dans les coalitions pour les rôles de l'*agence cartographique* : comme les variables de signaux dans les exigences pour l'*armée* et dans celles pour l'*agence cartographique* portent les mêmes étiquettes, une affectation peut en effet traiter de manière uniforme ces différentes variables. Dans cette modélisation, les variables $\textit{réponse}^1_{arm}$ et $\textit{réponse}^2_{arm}$ sont modifiables par n'importe quel acteur ayant une capacité d'action sur une variable étiquetée **image**. Il n'y a donc aucune protection des signaux échangés entre le segment sol et le segment embarqué pour l'*armée*. On corrige ce problème en introduisant un but de sécurité dans le modèle de but pour l'*armée*. On obtient ainsi le modèle Kh_{Secur}, développé dans la section suivante.

C.6.2 Le modèle Kh_{Secur} : introduction d'un but de sécurité pour l'*armée*

Dans cette variante, l'*armée* a besoin de sécuriser les échanges d'informations entre son segment sol et son segment embarqué. Un but de *sécurité* est donc ajouté à son modèle de but (cf. Figure C.5). Ce but est raffiné par les exigences *confidentialité* (les messages échangés entre le segment sol et le segment embarqué ne peuvent pas être lus par un tiers) et *exclusivité* (seuls les messages des acteurs capables de s'identifier sont pris en compte). On réalise ces exigences grâce à un cryptage des messages contenus dans les signaux. Pour une variable x représentant un *ordre* ou une *réponse*, le fait de contenir une valeur cryptée est représenté par un booléen $x.crypt$:

Formellement on introduit trois nouvelles étiquettes de variables :

- une étiquette **booléen**, de domaine $\{1,2\}$.
- deux étiquettes pour les valeurs cryptées : une pour les positions et une pour les images. On les appelle respectivement **positionCryp** et **imageCryp**. Les variables étiquetées respectivement **positionCryp** et **imageCryp** sont axiomatisées comme celles étiquetées **position** et **image**. Pour chaque variable x étiquetée **positionCryp** ou **imageCryp**, on introduit donc une variable $x.crypt$ étiquetée **booléen**.

Les nouvelles exigences sont encodées de la manière suivante :

Form(***confidentialité***)

$$\Box(\wedge_i(\textit{ordre}^i_{arm} \geqslant 0 \rightarrow \textit{ordre}^i_{arm}.crypt = 1) \wedge (\textit{réponse}^i_{arm} \geqslant 0 \rightarrow \textit{réponse}^i_{arm}.crypt = 1))$$

Form(***exclusivité***)

$$\Box(\wedge_i(\textit{ordre}^i_{arm} = -1 \rightarrow \textit{ordre}^i_{arm}.crypt = 0) \wedge$$
$$(\textit{réponse}^i_{arm} = -1 \rightarrow \textit{réponse}^i_{arm}.crypt = 0))$$

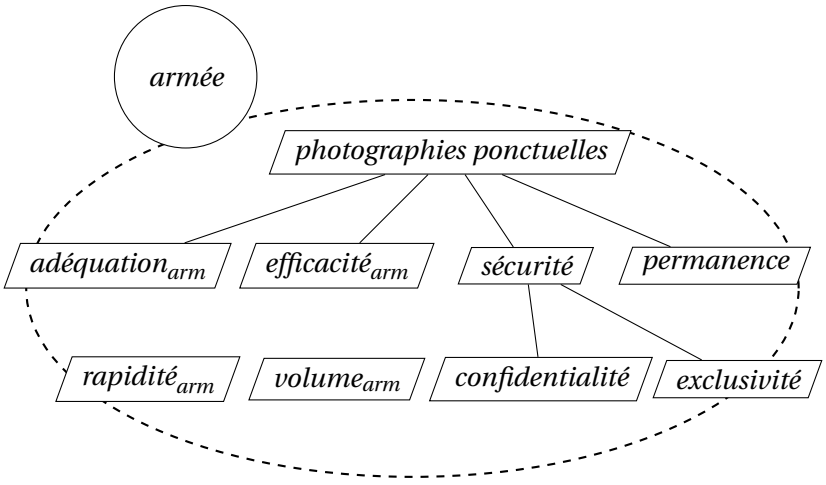

FIGURE C.5 – Modèle de buts pour *armée* pour le modèle Kh_{Secur}

La *confidentialité* assure qu'aucun message non crypté ne circule. Pour l'*exclusivité*, on considère que, pour chaque signal, le cryptage vaut comme identification de l'agent. Le but *exclusivité* requiert donc que chaque acteur refasse le cryptage pour envoyer un message. Pour cela, le marqueur de cryptage restera initialisé à la valeur 0 tant que la variable correspondante ne contient pas de message.

La satisfaction de ces buts implique des nouvelles opérations de cryptage et de décryptage des variables $ordre^i_{arm}$ et $réponse^i_{arm}$. Elles sont spécifiées de manière à ce que pour chacune de ces variables, son cryptage soit déclenché exactement quand elle reçoit une valeur positive, et son décryptage exactement quand elle est réinitialisée à -1 (voir Tableau C.13). Précisément, les opérations $cryptageOrdre^i$ et *transmissionBesoin* doivent être déclenchées en même temps, de même les opérations $cryptageRéponse^i$ et $envoiPhoto^i_{arm}$, $décryptageOrdre^i$ et $réceptionPhoto^i_{arm}$ et $décryptageRéponse^i$ et $réceptionOrdres_{arm}$. Dans le Tableau C.13, seules les nouvelles opérations sont représentées. Toutes les spécifications du tableau C.3 sont par ailleurs maintenues dans cette variante. Le Tableau C.14 fait le bilan des variables utilisées dans le modèle Kh_{Secur}. Les opérations $cryptageOrdre^i$ et $décryptageOrdre^i$ sont ajoutées au rôle *segment sol armée*, et $cryptageRéponse^i$ et $décryptageRéponse^i$ sont ajoutées à *assign*.

Les acteurs dans cette variante sont ceux du modèle $\mathsf{Kh}_{C,A}$, ainsi que deux appareils de cryptage, $crypteur^1$ et $crypteur^2$. Par simplicité, les capacités de $crypteur^1$ et $crypteur^2$ ne sont pas pré-conditionnées par la position des crypteurs. Elles sont décrites dans le Tableau C.15.

L'assignation pour la variante Kh_{Secur} est décrite dans la Figure C.6.2 : elle est similaire à celle du modèle $\mathsf{Kh}_{S_{mem}}$, à ceci près que les crypteurs $crypteur^1$ et $crypteur^2$ sont ajoutés respectivement aux coalitions pour les rôles *segment embarqué armée* et *segment sol armée*.

On peut alors vérifier la proposition suivante :

Proposition C.4. *Le modèle Kh_{Secur} satisfait*

$$\mathsf{Contr}_G(\{\textit{segment embarqué carto}, \textit{segment sol carto}\}, \{\textit{segment embarqué armée}, \textit{segment sol armée}\})$$

Résumé. Similaire à la preuve pour la Proposition C.1. □

$cryptageOrdre^i$		$domPre$	$ordre^i_{arm}.crypt = 0$
		$domPost$	$ordre^i_{arm}.crypt = 1$
	$confidentialité$	$reqPre$	$\wedge_i (besoin^i \geqslant 0)$
			$\wedge \wedge_i (ordre^i_{arm} = -1)$
		$reqTrig$	$\wedge_i (besoin^i \geqslant 0)$
			$\wedge \wedge_i (ordre^i_{arm} = -1)$
$cryptageRéponse^i$		$domPre$	$réponse^i_{arm}.crypt = 0$
		$domPost$	$réponse^i_{arm}.crypt = 1$
	$exclusivité$	$reqPre$	$réponse^i_{arm}$= -1
			$\wedge positionUnitéBord^i_{arm} = 5$
			$\wedge imageStockéeBord^i_{arm}.pos \geqslant 0$
		$reqTrig$	$réponse^i_{arm}$= -1
			$\wedge positionUnitéBord^i_{arm} = 5$
			$\wedge imageStockéeBord^i_{arm}.pos \geqslant 0$
$décryptageOrdre^i$		$domPre$	$ordre^i_{arm}.crypt = 0$
		$domPost$	$ordre^i_{arm}.crypt = 1$
	$confidentialité$	$reqPre$	$imageStockéeSol^i_{arm}.pos$ = -1
			$\wedge réponse^i_{arm}.pos \geqslant 0$
		$reqTrig$	$imageStockéeSol^i_{arm}.pos$ = -1
			$\wedge réponse^i_{arm}.pos \geqslant 0$
$décryptageRéponse^i$		$domPre$	$réponse^i_{arm}.crypt = 1$
		$domPost$	$réponse^i_{arm}.crypt = 0$
	$exclusivité$	$reqPre$	$\wedge_i (tâcheEnAttente^i_{arm} = -1)$
			$\wedge \wedge_i (ordre^i_{arm} \geqslant 0)$
		$reqTrig$	$\wedge_i (tâcheEnAttente^i_{arm} = -1)$
			$\wedge \wedge_i (ordre^i_{arm} \geqslant 0)$

TABLE C.13 – Opérationnalisation pour les opérations de cryptage et de décryptage

C.6.3 Le modèle $\mathsf{Kh}_{C,A,G}$: introduction d'un troisième modèle de buts

Cette dernière variante de notre modèle prend en charge les buts pour une troisième mission, *GPS*, qui a également besoin d'une couverture terrestre régulière de la surface de la terre par images satellites. On illustre, grâce à ce modèle, la différence entre les propriétés de collaborations locale et globale.

Nous ne détaillons pas ici le modèle de buts pour le *GPS*, ni l'opérationnalisation de ses exigences et la construction de ses rôles induits : ils sont entièrement isomorphes à ceux pour l'*agence cartographique* dans le modèle $\mathsf{Kh}_{C,A}$. La prise en compte de *GPS* induit donc les rôles *segment embarqué GPS* et *segment sol GPS*. Le modèle de buts pour *armée* est celui du modèle $\mathsf{Kh}_{C,A}$.

La sémantique des rôles *segment embarqué GPS* et *segment sol GPS* est équivalente à la sémantique des rôles *segment embarqué carto* et *segment sol carto*.

On s'intéresse aux trois coalitions pour les segments embarqués, selon l'assignation représentée dans la Figure C.6.3 (dans la figure on a représenté exclusivement les segments embarqués) et à leurs interactions : le satellite S_{mem} participe aux coalitions pour chacun des trois rôles de segments embarqués. Chacune de ces coalitions contient par ailleurs un satellite *S1*, *S2* ou *S3*.

$X_{C,A}$	exigences		
	$\mathscr{L}$	embarqué	sol
	paramètres	$i \in \{1,2\}, act \in \{c, arm\}$	$i \in \{1,2\}, act \in \{c, arm\}$
	position	$positionUnitéBord^i_{act}$	$ordre^i_c$
		$tâcheEnAttente^i_{act}$	$besoin^i$
	image	$imageStockéeBord^i_{act}$	$imageStockéeSol^i_{act}$
		$réponse^i_c$	
	positionCryp		$ordre^i_{arm}$
	imageCryp	$réponse^i_{arm}$	
	booléen	$réponse^i_{arm}.crypt$	$ordre^i_{arm}.crypt$
	moyens		
	$\mathscr{L}$	embarqué	sol
	paramètres	$j \in \{1,2\}$	$i \in \{1,2\}, act \in \{c, transmetteur\}$
	position	$positionSatellite^j$	$signalSol^i_c$
		$positionSatellite^{S_{mem}}_j$	$besoinPerçu^i$
		$tâcheSatellite^j$	
		$enregistrementOrdre^{S_{mem}}_j$	
	image	$mémoireSatellite^j$	$mémoireSol^i_{act}$
		$mémoireSatellite^{S_{mem}}_j$	
		$signalSatellite^j$	
		$signalSatellite^{S_{mem}}_j$	
		$signalBord^i_{act}$	
	positionCryp		$signalSol^i_{arm}$
	imageCryp	$signalSatellite^j$	
	booléen	$signalSatellite^j.crypt$	$signalSol^i_{arm}.crypt$
pour toute $x \in X_{C,A}$, $x.old$ de même étiquette et même langage pour toute $x \in [x_i : \textbf{position}]$, $x.old$ de même étiquette et même langage			

TABLE C.14 – Tableau des variables utilisées dans le modèle Kh_{Secur}

$\{cryptage^i_x\}_{x \in \mathrm{Var(M)}}$	*préCond*	$\top$
	fenêtre	$x.crypt = 1$
$\{décryptage^i_x\}_{x \in VAR}$	*préCond*	$\top$
	fenêtre	$x.crypt = 0$

TABLE C.15 – Capacités de $crypteur^i$, pour $i \in \{1,2\}$

Comme on l'a observé dans les sections précédentes, chaque rôle de segment embarqué consiste en un dispositif de réponse à deux ordres. Par ailleurs, chacun des satellite *S1*, *S2* et *S3* est en mesure d'assurer la réponse à un ordre, respectivement pour les rôles *segment embarqué carto*, *segment embarqué armée* et *segment embarqué GPS*. Le satellite S_{mem} peut alors agir pour compléter deux des rôles au plus. En particulier, si la coalition pour *segment embarqué armée* assure son rôle, alors, une et une seule des deux coalitions pour *segment embarqué carto* et *segment embarqué GPS* est en mesure d'assurer son rôle. Le rôle *segment embarqué armée*

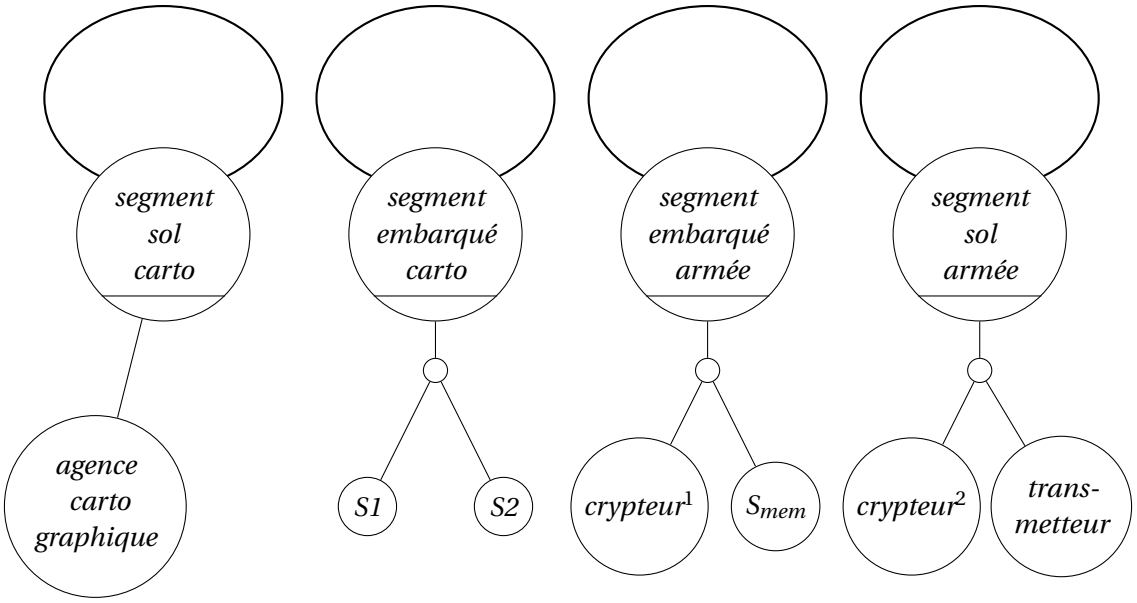

FIGURE C.6 – Assignation pour le modèle Kh_{Secur}

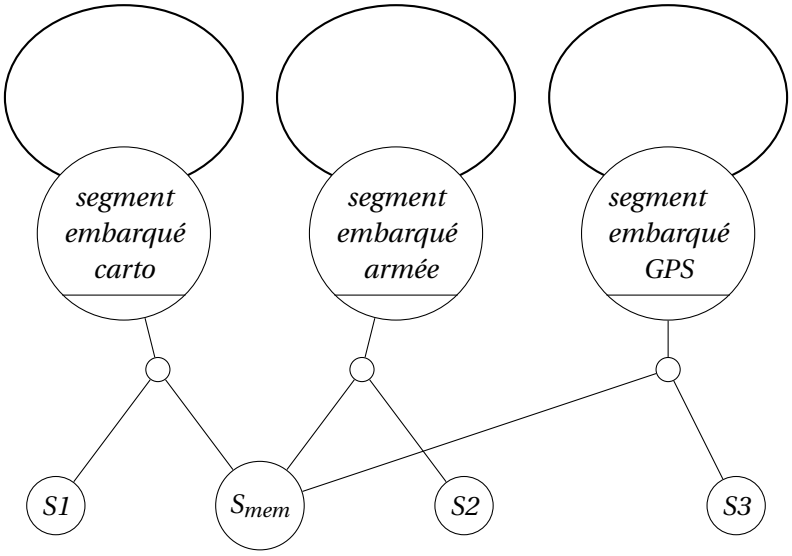

FIGURE C.7 – Assignation pour le modèle $\mathsf{Kh}_{C,A,G}$ (segments embarqués)

collabore localement avec *segment embarqué carto* et *segment embarqué GPS*, mais ne peut pas collaborer globalement avec cet ensemble de rôles :

Proposition C.5.

– *Le modèle* $Kh_{C,A,G}$ *satisfait*

$$\mathsf{Col}_L(\{\textit{segment embarqué armée}\}, \{\textit{segment embarqué carto}, \textit{segment embarqué GPS}\})$$

– *Le modèle $Kh_{C,A,G}$ ne satisfait pas*

$\mathsf{Col}_G(\{$*segment embarqué armée*$\}, \{$*segment embarqué carto, segment embarqué GPS*$\})$

C.7 Preuves de correction des opérationnalisations

Cette section fournit l'écriture de l'opérationnalisation des exigences pour le modèle $\mathsf{Kh}_{C,A}$, telle qu'elle a été prouvée correcte par NuSMV. Cette vérification est donc valable pour la sémantique LTL, qui n'utilise que des exécutions infinies. On se convainc facilement cependant que l'enchaînement des opérations spécifiant les buts pour l'*agence cartographique* garantit la survie du système. En particulier, les spécifications pour les exigences *planification* et *rapidité*$_c$ suffisent à garantir cette survie. Une exécution qui respecte l'ensemble des spécifications de $\mathsf{Kh}_{C,A}$ est donc nécessairement infinie et l'emploi de NuSMV est adéquat pour vérifier la correction de l'opérationalisation.

C.7.1 Déclarations et spécifications des variables

Les déclarations de variables sont les mêmes pour chacune des vérifications. On les donne une fois pour toutes. Pour simplifier les notations, dans ces spécifications on désigne la position d'une variable d'image directement par le nom de cette variable d'image. Par ailleurs, les noms choisis sont de la forme *ident*·*i*·*act* où $i \in \{1,2\}$ et *act* identifie la mission à laquelle la variable se rapporte : *a* pour l'*armée* et *c* pour l'*agence cartographique*. On ne donne ici les déclarations que pour les variables relatives à l'armée.

Les spécifications de ces variables encodent les axiomes pour les variables *x.old,* ainsi que les propriétés du domaine qui décrivent l'état initial du système et le mouvement des satellites :

```
MODULE main
VAR
iss1a :  {-1, 0, 1,2,3,4,5}; -- imageStockéeSol
iss2a :  {-1, 0, 1,2,3,4,5};
oiss1a :  {-1, 0, 1,2,3,4,5}; -- old(imageStockéeSol)
oiss2a :  {-1, 0, 1,2,3,4,5};
o1a : {-1, 0, 1,2,3,4,5}; -- ordre
o2a : {-1, 0, 1,2,3,4,5};
oo1a : {-1, 0, 1,2,3,4,5}; -- old(ordre)
oo2a : {-1, 0, 1,2,3,4,5};
tea1a :{-1, 0, 1,2,3,4,5}; -- tâcheEnAttente
tea2a : {-1, 0, 1,2,3,4,5};
otea1a :{-1, 0, 1,2,3,4,5}; -- old(tâcheEnAttente)
otea2a : {-1, 0, 1,2,3,4,5};
isb1a : {-1, 0, 1,2,3,4,5}; -- imageStockéeBord
isb2a : {-1, 0, 1,2,3,4,5};
oisb1a : {-1, 0, 1,2,3,4,5}; -- old(imageStockéeBord)
oisb2a : {-1, 0, 1,2,3,4,5};
pub1a : {-1, 0, 1,2,3,4,5}; -- positionUnitéBord
pub2a : {-1, 0, 1,2,3,4,5};
r1a : {-1, 0, 1,2,3,4,5}; -- réponse
r2a : {-1, 0, 1,2,3,4,5};
or1a : {-1, 0, 1,2,3,4,5};  -- old(réponse)
or2a : {-1, 0, 1,2,3,4,5};
b1a : {-1, 0, 1,2,3,4,5};  -- besoin
b2a : {-1, 0, 1,2,3,4,5};
```

```
ob1a : {-1, 0, 1,2,3,4,5};  -- old(besoin)
ob2a : {-1, 0, 1,2,3,4,5};

ASSIGN
init(iss1a) :=  0; -- imageStockéeSol
init(iss2a) :=  0;
init(oiss1a) :=  -1; -- old(imageStockéeSol)
init(oiss2a) :=  -1;
init(o1a) := -1; -- ordre
init(o2a) := -1;
init(b1a) := -1; -- besoin
init(b2a) := -1;
init(ob1a) := -1; -- old(besoin)
init(ob2a) := -1;
init(oo1a) := -1; -- ordre
init(oo2a) := -1;
init(tea1a) := -1; -- tâcheBord
init(tea2a) := -1;
init(otea1a) := -1; -- tâcheBord
init(otea2a) := -1;
init(isb1a) := -1; -- imageStockéeBord
init(isb2a) := -1;
init(pub1a) := -1; -- positionUnitéBord
init(pub2a) := -1;
init(r1a) := -1; -- réponse
init(r2a) := -1;

next(oiss1a) := iss1a;
next(oiss2a) := iss2a;
next(oo1a) := o1a;
next(oo2a) := o2a;
next(oisb1a) := isb1a;
next(oisb2a) := isb2a;
next(or1a) := r1a;
next(or2a) := r2a;
next(otea1a) := tea1a;
next(otea2a) := tea2a;

next(pub1a) := (pub1a + 1) mod 6;
next(pub2a) := (pub2a + 1) mod 6;
```

C.7.2 *planification*

```
LTLSPEC

(
G ((o1c =-1 & X (o1c >=0)) <-> (o2c =-1 & X (o2c >=0))) -- élaboration ordre
& G (iss1c >=0  -> ( X (iss1c = -1) <-> (o1c =-1 & X (o1c >= 0))))
& G (iss2c >=0  -> ( X (iss2c = -1) <-> (o2c =-1 & X (o2c >= 0))))
& G ((iss1c >= 0 & iss2c >= 0 &  o2c =-1) -> (o1c =-1 -> X (o1c >=0)))
& G ((iss2c >= 0 & iss1c >= 0 &  o1c =-1) -> (o2c =-1 -> X (o2c >=0)))

& G ((iss1c =-1 & X (iss1c >= 0)) <-> (o1c >= 0 & X (o1c = -1))) -- réceptionPhoto
& G ((iss2c =-1 & X (iss2c >= 0)) <-> (o2c >= 0 & X (o2c = -1)))
)

->(G
```

```
(((iss1c =-1 xnor iss2c = -1) & X (iss1c >=0 & iss2c >= 0))
 -> ( X ((o1c = -1 & X (o1c >= 0))&(o2c = -1 & X (o2c >= 0)))
|   X X ((o1c = -1 & X (o1c >= 0))&(o2c = -1 & X (o2c >= 0)))
 | X X X ((o1c = -1 & X (o1c >= 0))& (o2c = -1 & X (o2c >= 0))))))
```

C.7.3 *adéquation*$_{arm}$

```
LTLSPEC

(
G ((o1a =-1 & X (o1a >=0)) <-> (b1a >= 0 & ob1a =-1 & o1a = -1))  -- transmissionBesoin
& G ((o1a =-1 & X (o1a >=0)) -> X (o1a = ob1a & iss1a = -1 ))
& G ((o2a =-1 & X (o2a >=0)) <-> (b2a >= 0 & ob2a =-1 & o2a = -1))
& G ((o2a =-1 & X (o2a >=0)) -> X (o2a = ob2a & iss2a = -1 ))

& G ((tea1a = -1 & X (tea1a >= 0)) <-> (o1a >= 0  & oo1a = -1)) -- réceptionOrdre
& G ((tea1a = -1 & X (tea1a >= 0)) -> X (tea1a = oo1a & r1a = -1))
& G ((tea2a = -1 & X (tea2a >= 0)) <-> (o2a >= 0  & oo2a = -1))
& G ((tea2a = -1 & X (tea2a >= 0)) -> X (tea2a = oo2a & r2a = -1))

& G ((tea1a >= 0 & X (tea1a >= 0)) <-> (tea1a >= 0 & -- maintienOrdreAttente
(isb1a = -1 | pub1a < 5)))
& G ((tea1a >= 0 & X (tea1a >= 0)) -> X (otea1a = tea1a))
& G ((tea2a >= 0 & X (tea2a >= 0)) <-> (tea2a >= 0 &
(isb2a = -1 | pub2a < 5)))
& G ((tea2a >= 0 & X (tea2a >= 0)) -> X (otea2a = tea2a))

& G ((isb1a = -1 & X (isb1a >=0 )) -> (isb1a = -1 & tea1a >= 0 -- prisePhoto
& tea1a = pub1a))
& G ((isb1a = -1 & X (isb1a >=0 )) -> X (isb1a = otea1a))
& G ((isb2a = -1 & X (isb2a >=0 )) <-> (isb2a = -1 & tea2a >= 0
& tea2a = pub2a))
& G ((isb2a = -1 & X (isb2a >=0 )) -> X (isb2a = otea2a))

& G ((isb1a >= 0 & X (isb1a >= 0)) -> X (oisb1a = isb1a)) -- maintienMémoireBord
& G ((isb2a >= 0 & X (isb2a >= 0)) -> X (oisb2a = isb2a))

& G ((r1a =-1 & X (r1a >= 0 )) -> X (r1a = oisb1a & isb1a = -1 & tea1a = -1)) --envoiPhoto
& G ((r2a =-1 & X (r2a >= 0 )) -> X (r2a = oisb2a & isb2a = -1 & tea2a = -1))

& G ((iss1a =-1 & X (iss1a >= 0 )) <-> (r1a >= 0 & or1a =-1 )) -- réceptionPhoto
& G ((iss1a =-1 & X (iss1a >= 0 )) -> X (iss1a = or1a))
& G ((iss2a =-1 & X (iss2a >= 0 )) <-> (r2a >= 0 & or2a =-1 ))
& G ((iss2a =-1 & X (iss2a >= 0 )) -> X (iss2a = or2a))

& G ((o1a >= 0 & X (o1a  >= 0)) <-> (o1a >= 0 & !( r1a >= 0 & or1a =-1 ))) -- maintienOrdre
& G ((o1a >= 0 & X (o1a  >= 0)) -> X (o1a = oo1a ))
& G ((o2a >= 0 & X (o2a  >= 0)) <-> (o2a >= 0 & !( r2a >= 0 & or2a =-1 )))
& G ((o2a >= 0 & X (o2a  >= 0)) -> X (o2a = oo2a ))
 )

->
```

```
(G
((ob1a = -1  &  b1a >= 0  & o1a = -1  )
 -> (X ((o1a = ob1a) &  ((G(iss1a =-1))
 | ((iss1a =-1   & X ((o1a = oo1a) U (iss1a = oo1a)))))))
& G
((ob2a = -1  &  b2a >= 0  & o2a = -1  )
 -> (X ((o2a = ob2a) &  ((G(iss2a =-1))
 | ((iss2a =-1   & X ((o2a = oo2a) U (iss2a = oo2a)))))))
)
```

C.7.4 *adéquation*$_c$

```
LTLSPEC

(
G ((o1c =-1 & X (o1c >=0)) -> X (iss1c =-1)) -- élaborationOrdre
& G ((o2c =-1 & X (o2c >=0)) -> X (iss2c =-1))

& G ((tea1c >= 0 & X (tea1c >= 0)) <-> (tea1c >= 0 & -- maintienOrdreAttente
(isb1c = -1 | pub1c < 5)))
& G ((tea1c >= 0 & X (tea1c >= 0)) -> X (otea1c = tea1c))
& G ((tea2c >= 0 & X (tea2c >= 0)) <-> (tea2c >= 0 &
(isb2c = -1 | pub2c < 5)))
& G ((tea2c >= 0 & X (tea2c >= 0)) -> X (otea2c = tea2c))

& G ((isb1c = -1 & X (isb1c >=0 )) -> (isb1c = -1 & tea1c >= 0 -- prisePhoto
& tea1c = pub1c))
& G ((isb1c = -1 & X (isb1c >=0 )) -> X (isb1c = otea1c))
& G ((isb2c = -1 & X (isb2c >=0 )) <-> (isb2c = -1 & tea2c >= 0
& tea2c = pub2c))
& G ((isb2c = -1 & X (isb2c >=0 )) -> X (isb2c = otea2c))

& G ((isb1c >= 0 & X (isb1c >= 0)) -> X (oisb1c = isb1c)) -- maintienMémoireBord
& G ((isb2c >= 0 & X (isb2c >= 0)) -> X (oisb2c = isb2c))

& G ((r1c =-1 & X (r1c >= 0 )) -> X (r1c = oisb1c & isb1c = -1 & tea1c = -1)) --envoiPhoto
& G ((r2c =-1 & X (r2c >= 0 )) -> X (r2c = oisb2c & isb2c = -1 & tea2c = -1))

& G ((iss1c =-1 & X (iss1c >= 0 )) <-> (r1c >= 0 & or1c =-1 )) -- réceptionPhoto
& G ((iss1c =-1 & X (iss1c >= 0 )) -> X (iss1c = or1c))
& G ((iss2c =-1 & X (iss2c >= 0 )) <-> (r2c >= 0 & or2c =-1 ))
& G ((iss2c =-1 & X (iss2c >= 0 )) -> X (iss2c = or2c))

& G ((o1c >= 0 & X (o1c  >= 0)) <-> (o1c >= 0 & !( r1c >= 0 & or1c =-1 ))) -- maintienOrdre
& G ((o1c >= 0 & X (o1c  >= 0)) -> X (o1c = oo1c ))
& G ((o2c >= 0 & X (o2c  >= 0)) <-> (o2c >= 0 & !( r2c >= 0 & or2c =-1 )))
& G ((o2c >= 0 & X (o2c  >= 0)) -> X (o2c = oo2c ))
 )

->

(G
((ob1c = -1  &  b1c >= 0  & o1c = -1  )
 -> (X ((o1c = ob1c) &  ((G(iss1c =-1))
 | ((iss1c =-1   & X ((o1c = oo1c) U (iss1c = oo1c)))))))
& G
((ob2c = -1  &  b2c >= 0  & o2c = -1  )
```

```
 -> (X ((o2c = ob2c) &  ((G(iss2c =-1))
 | ((iss2c =-1   & X ((o2c = oo2c) U (iss2c = oo2c))))))))
)
```

C.7.5 $rapidité_{arm}$

```
LTLSPEC

(
G ((o1a =-1 & X (o1a >=0)) <-> (b1a >= 0 & ob1a =-1 & o1a = -1))  -- transmissionBesoin
& G ((o2a =-1 & X (o2a >=0)) <-> (b2a >= 0 & ob2a =-1 & o2a = -1))

& G ((tea1a = -1 & X (tea1a >= 0)) <-> (o1a >= 0  & oo1a = -1)) -- réceptionOrdre
& G ((tea1a = -1 & X (tea1a >= 0)) -> X ( r1a = -1))
& G ((tea2a = -1 & X (tea2a >= 0)) <-> (o2a >= 0  & oo2a = -1))
& G ((tea2a = -1 & X (tea2a >= 0)) -> X ( r2a = -1))

& G ((tea1a >= 0 & X (tea1a >= 0)) <-> (tea1a >= 0 & -- maintienOrdreAttente
(isb1a = -1 | pub1a < 5)))
& G ((tea1a >= 0 & X (tea1a >= 0)) -> X (otea1a = tea1a))
& G ((tea2a >= 0 & X (tea2a >= 0)) <-> (tea2a >= 0 &
(isb2a = -1 | pub2a < 5)))
& G ((tea2a >= 0 & X (tea2a >= 0)) -> X (otea2a = tea2a))

& G ((isb1a = -1 & X (isb1a >=0 )) <-> (isb1a = -1 & tea1a >= 0 -- prisePhoto
& tea1a = pub1a))
& G ((isb2a = -1 & X (isb2a >=0 )) <-> (isb2a = -1 & tea2a >= 0
& tea2a = pub2a))

& G ((isb1a >= 0 & X (isb1a >= 0)) <-> (isb1a >= 0 & -- maintienMémoireBord
pub1a < 5))
& G ((isb2a >= 0 & X (isb2a >= 0)) <-> (isb2a >= 0 &
pub2a < 5))

& G ((r1a = -1 & X (r1a >= 0 )) <-> (r1a = -1 & isb1a >= 0 -- envoiPhoto
& pub1a = 5))
& G ((r2a = -1 & X (r2a >= 0 )) <-> (r2a = -1 & isb2a >= 0
& pub2a = 5))

& G ((iss1a =-1 & X (iss1a >= 0 )) <-> (r1a >= 0 & or1a =-1 )) -- réceptionPhoto
& G ((iss2a =-1 & X (iss2a >= 0 )) <-> (r2a >= 0 & or2a =-1 ))
)

->

(G(
((b1a = -1 )   & X (b1a >= 0 ))
  -> ( (iss1a =-1  ) & X  (iss1a >=0 ))
| X ( (iss1a =-1 ) & X (iss1a >=0 ))
| X X ( (iss1a =-1 ) & X (iss1a >=0 ))
| X X X ( (iss1a =-1 ) & X (iss1a >=0 ))
| X X X X( (iss1a =-1 ) & X (iss1a >=0 ))
| X X X X X( (iss1a =-1 ) & X (iss1a >=0 ))
| X X X X X X ( (iss1a =-1 ) & X (iss1a >=0 ))
| X X X X X X X( (iss1a =-1 ) & X (iss1a >=0 ))
| X X X X X X X X( (iss1a =-1 ) & X (iss1a >=0 ))
| X X X X X X X X X( (iss1a =-1 ) & X (iss1a >=0 ))
```

```
| X X X X X X X X X X( (iss1a =-1 ) & X (iss1a >=0 ))
| X X X X X X X X X X( (iss1a =-1 ) & X (iss1a >=0 ))
| X X X X X X X X X X X( (iss1a =-1 ) & X (iss1a >=0 ))
| X X X X X X X X X X X X( (iss1a =-1 ) & X (iss1a >=0 ))
| X X X X X X X X X X X X X( (iss1a =-1 ) & X (iss1a >=0 ))
| X X X X X X X X X X X X X X( (iss1a =-1 ) & X (iss1a >=0 ))
| X X X X X X X X X X X X X X X( (iss1a =-1 ) & X (iss1a >=0 ))
 )
& G(
((b2a = -1 )   & X (b2a >= 0 ))
  -> ( (iss2a =-1  ) & X  (iss2a >=0 ))
| X ( (iss2a =-1 ) & X (iss2a >=0 ))
| X X ( (iss2a =-1 ) & X (iss2a >=0 ))
| X X X ( (iss2a =-1 ) & X (iss2a >=0 ))
| X X X X( (iss2a =-1 ) & X (iss2a >=0 ))
| X X X X X( (iss2a =-1 ) & X (iss2a >=0 ))
| X X X X X X ( (iss2a =-1 ) & X (iss2a >=0 ))
| X X X X X X X( (iss2a =-1 ) & X (iss2a >=0 ))
| X X X X X X X X( (iss2a =-1 ) & X (iss2a >=0 ))
| X X X X X X X X X( (iss2a =-1 ) & X (iss2a >=0 ))
| X X X X X X X X X X( (iss2a =-1 ) & X (iss2a >=0 ))
| X X X X X X X X X X X( (iss2a =-1 ) & X (iss2a >=0 ))
| X X X X X X X X X X X X( (iss2a =-1 ) & X (iss2a >=0 ))
| X X X X X X X X X X X X X( (iss2a =-1 ) & X (iss2a >=0 ))
| X X X X X X X X X X X X X X( (iss2a =-1 ) & X (iss2a >=0 ))
  )
)
```

C.7.6 *rapidité*$_c$

```
LTLSPEC

(
G ((tea1c = -1 & X (tea1c >= 0)) <-> (o1c >= 0  & oo1c = -1)) -- réceptionOrdre
& G ((tea1c = -1 & X (tea1c >= 0)) -> X ( r1c = -1))
& G ((tea2c = -1 & X (tea2c >= 0)) <-> (o2c >= 0  & oo2c = -1))
& G ((tea2c = -1 & X (tea2c >= 0)) -> X ( r2c = -1))

& G ((tea1c >= 0 & X (tea1c >= 0)) <-> (tea1c >= 0 & -- maintienOrdreAttente
(isb1c = -1 | pub1c < 5)))
& G ((tea1c >= 0 & X (tea1c >= 0)) -> X (otea1c = tea1c))
& G ((tea2c >= 0 & X (tea2c >= 0)) <-> (tea2c >= 0 &
(isb2c = -1 | pub2c < 5)))
& G ((tea2c >= 0 & X (tea2c >= 0)) -> X (otea2c = tea2c))

& G ((isb1c = -1 & X (isb1c >=0 )) <-> (isb1c = -1 & tea1c >= 0 -- prisePhoto
& tea1c = pub1c))
& G ((isb2c = -1 & X (isb2c >=0 )) <-> (isb2c = -1 & tea2c >= 0
& tea2c = pub2c))

& G ((isb1c >= 0 & X (isb1c >= 0)) <-> (isb1c >= 0 & -- maintienMémoireBord
pub1c < 5))
& G ((isb2c >= 0 & X (isb2c >= 0)) <-> (isb2c >= 0 &
pub2c < 5))
```

```
& G ((r1c = -1 & X (r1c >= 0 )) <-> (r1c = -1 & isb1c >= 0 -- envoiPhoto
& pub1c = 5))
& G ((r2c = -1 & X (r2c >= 0 )) <-> (r2c = -1 & isb2c >= 0
& pub2c = 5))

& G ((iss1c =-1 & X (iss1c >= 0 )) <-> (r1c >= 0 & or1c =-1 )) -- réceptionPhoto
& G ((iss2c =-1 & X (iss2c >= 0 )) <-> (r2c >= 0 & or2c =-1 ))
)

->

(G(
((b1c = -1 )   & X (b1c >= 0 ))
  -> ( (iss1c =-1  ) & X  (iss1c >=0 ))
| X ( (iss1c =-1 ) & X (iss1c >=0 ))
| X X ( (iss1c =-1 ) & X (iss1c >=0 ))
| X X X ( (iss1c =-1 ) & X (iss1c >=0 ))
| X X X X( (iss1c =-1 ) & X (iss1c >=0 ))
| X X X X X( (iss1c =-1 ) & X (iss1c >=0 ))
| X X X X X X ( (iss1c =-1 ) & X (iss1c >=0 ))
| X X X X X X X( (iss1c =-1 ) & X (iss1c >=0 ))
| X X X X X X X X( (iss1c =-1 ) & X (iss1c >=0 ))
| X X X X X X X X X( (iss1c =-1 ) & X (iss1c >=0 ))
| X X X X X X X X X X( (iss1c =-1 ) & X (iss1c >=0 ))
| X X X X X X X X X X X( (iss1c =-1 ) & X (iss1c >=0 ))
| X X X X X X X X X X X X( (iss1c =-1 ) & X (iss1c >=0 ))
| X X X X X X X X X X X X X( (iss1c =-1 ) & X (iss1c >=0 ))
| X X X X X X X X X X X X X X( (iss1c =-1 ) & X (iss1c >=0 ))
 )
& G(
((b2c = -1 )   & X (b2c >= 0 ))
  -> ( (iss2c =-1  ) & X  (iss2c >=0 ))
| X ( (iss2c =-1 ) & X (iss2c >=0 ))
| X X ( (iss2c =-1 ) & X (iss2c >=0 ))
| X X X ( (iss2c =-1 ) & X (iss2c >=0 ))
| X X X X( (iss2c =-1 ) & X (iss2c >=0 ))
| X X X X X( (iss2c =-1 ) & X (iss2c >=0 ))
| X X X X X X ( (iss2c =-1 ) & X (iss2c >=0 ))
| X X X X X X X( (iss2c =-1 ) & X (iss2c >=0 ))
| X X X X X X X X( (iss2c =-1 ) & X (iss2c >=0 ))
| X X X X X X X X X( (iss2c =-1 ) & X (iss2c >=0 ))
| X X X X X X X X X X( (iss2c =-1 ) & X (iss2c >=0 ))
| X X X X X X X X X X X( (iss2c =-1 ) & X (iss2c >=0 ))
| X X X X X X X X X X X X( (iss2c =-1 ) & X (iss2c >=0 ))
| X X X X X X X X X X X X X( (iss2c =-1 ) & X (iss2c >=0 ))
| X X X X X X X X X X X X X X( (iss2c =-1 ) & X (iss2c >=0 ))
  )
)
```

C.7.7 $volume_{arm}$

```
LTLSPEC
```

```
G ((o1a =-1 & X (o1a >=0)) -> (o2a =-1  & X (o1a >=0)) -- transmissionBesoin
& G ((o1a =-1 & X (o1a >=0)) -> (o2a =-1  & X (o1a >=0)))
)
-> (G ((o1a = -1& X (o1a >=0)) <->  (o1a = -1& X (o1a >=0))))
```

C.7.8 *volume$_c$*

```
LTLSPEC

G ((o1c =-1 & X (o1c >=0)) -> (o2c =-1  & X (o1c >=0)) -- élaborationOrdre
& G ((o1c =-1 & X (o1c >=0)) -> (o2c =-1  & X (o1c >=0)))
)
-> (G ((o1c = -1& X (o1c >=0)) <->  (o1c = -1& X (o1c >=0))))
```

C.7.9 *permanence*

```
LTLSPEC

(G ((issa1a =-1 & X (issa1a >=0)) -> (o1a = -1))  -- réceptionPhoto
& G ((issa2a =-1 & X (issa2a >=0)) -> (o2a = -1))
& G ((o1a =-1) -> ((b1a =-1) <-> (X (o1a =-1)))) -- maintienEtatAttente
& G ((o2a =-1) -> ((b2a =-1) <-> (X (o2a =-1)))
)

->

(
G(
(issa1a =-1 & X (issa1a >=0))
->((G (o1a = -1) | ((o1a =-1) U (b1a >=0))))))
&G(
(issa2a =-1 & X (issa2a >=0))
->((G (o2a = -1) | ((o2a =-1) U (b2a >=0))))))
)
```