Propriétés de jeux multi-agents

Arnaud da Costa da Costa Lopes Lopes

THÈSE DE DOCTORAT

DE L'ÉCOLE NORMALE SUPÉRIEURE DE CACHAN

Présentée par

Arnaud DA COSTA LOPES

pour obtenir le grade de

DOCTEUR DE L'ÉCOLE NORMALE SUPÉRIEURE DE CACHAN

Domaine :

INFORMATIQUE

Propriétés de jeux multi-agents

Thèse présentée et soutenue à Cachan le 20/09/2011 devant le jury composé de :

- Mme Pinchinat Sophie (rapporteur), professeur - IRISA, Rennes
- M. Zielonka Wieslaw (rapporteur), professeur - Université Paris Diderot - Paris 7, LIAFA
- M. Cassez Franck, chargé de recherches - Ecole Centrale de Nantes, IRCCYN
- M. Olivier Serre, chargé de recherches - Université Paris Diderot - Paris 7, LIAFA
- M. Brihaye Thomas - Université de Mons, Belgique
- M. Markey Nicolas (co-encadrant), chargé de recherches HDR - ENS Cachan, LSV
- M. Laroussinie Francois (directeur de thèse), professeur - Université Paris Diderot - Paris 7, LIAFA

Laboratoire Spécification et Vérification (LSV), ENS Cachan, CNRS & INRIA
61, avenue du Président Wilson, 94235 Cachan Cedex

Table des matières

Chapitre 1

Introduction

Vérification formelle Nos sociétés se reposent de plus en plus largement sur des programmes informatiques qui régissent le fonctionnement de nombreux appareils divers, comme des téléphones portables, ou des lave-vaisselle aux avions Rafale... Un défaut de certains de ces logiciels peut avoir des conséquences très graves : de tels logiciels sont d'ailleurs appelés logiciels *critiques*. Pour pallier ce genre de problèmes, la communauté informaticienne a cherché à mettre au point des techniques qui permettent de garantir qu'un programme est fiable (selon des critères qu'il est bien sûr nécessaire d'expliciter et de justifier). On parle alors de méthodes formelles appliquées à la vérification, ou plus simplement de vérification formelle. L'une des méthodes les plus répandues s'appelle le *model-checking* (ou "vérification automatique"), dont l'objectif est de vérifier algorithmiquement si un modèle donné satisfait une *spécification*. Le modèle est souvent une abstraction pertinente du système (car la taille du système réel est la plupart du temps trop grande pour être analysée automatiquement, on doit donc le simplifier), tandis que la spécification est souvent formulée au moyen d'un langage appelé *logique temporelle*.

Logiques temporelles et model-checking Les logiques temporelles (LTL, CTL) ont été introduites pour la spécification des systèmes réactifs il y aura bientôt trente ans [30, 13, 31]. Ces langages contiennent des modalités permettant d'énoncer le fait qu'un événement A apparaît après un événement B, qu'un événement A précède toujours un événement B, etc. Depuis lors, ces logiques ont été largement étudiées, en particulier au sein du paradigme du model-checking—*i.e.*, la vérification automatique d'une spécification en logique temporelle par un modèle. Cette approche de la vérification formelle a été largement étudiée et a permis l'émergence d'algorithmes puissants qui ont été implémentés, et qui se sont avérés utiles dans de nombreuses situations. Il existe plusieurs logiques temporelles (LTL [30], CTL [13, 31], CTL* [16] entre autres) qui se distinguent par le type de modèles sur lesquels sont interprétées les formules, par les opérateurs autorisés, *etc.*

Les logiques temporelles alternantes (ATL) Au cours de ces dix dernières années, une nouvelle gamme de logiques temporelles a été créée : celle des *logiques temporelles alternantes* (ATL et son extension ATL*) [3], qui sont des logiques permettant de spécifier des propriétés sur des *systèmes avec plusieurs joueurs* (que l'on modélise par des *structures de jeux concurrents* (CGS) [3]). Dans ces modèles, plusieurs protagonistes peuvent partiellement agir sur le comportement du système : son évolution dépend d'actions concurrentes simultanées des protagonistes (ou joueurs). Cette approche est particulièrement utile pour résoudre des problèmes de contrôlabilité.

Sur ces modèles, on peut non seulement souhaiter vérifier qu'un événement *peut* ou *va certainement* se produire (des propriétés qui sont facilement exprimables avec CTL ou LTL), mais aussi des propriétés plus élaborées comme "est-ce que certains joueurs sont à même de *contrôler* l'évolution du système afin de garantir une propriété donnée, et cela quel que soit le comportement des autres joueurs ?" Les logiques ATL et ATL* permettent précisément d'exprimer ce genre de propriétés : ainsi on peut par exemple exprimer que *le protagoniste A dispose d'une stratégie pour maintenir le système dans un ensemble d'états sans danger, quel que soit le comportement des autres protagonistes.* En effet, dans ces logiques, les quantificateurs classiques sur les exécutions (qui sont l'existentiel et l'universel, respectivement notés **E** et **A**) sont remplacés par un quantificateur de stratégies, noté $\langle\!\langle A \rangle\!\rangle$, où A est une coalition de joueurs. Ce quantificateur exprime l'existence d'une stratégie pour A garantissant une certaine propriété. Ainsi, dans ATL, l'exemple que l'on vient de donner, le fait que le joueur A a une stratégie pour rester dans un état sans danger, s'écrit

$$\langle\!\langle A \rangle\!\rangle \, \mathbf{G} \; \textit{état sans danger} \tag{1}$$

(où **G** désigne la modalité temporelle "toujours").

ATL et ATL* sont donc deux extensions naturelles des logiques temporelles classiques, qui s'appuient sur leurs sémantiques (notamment par l'usage des mêmes opérateurs temporels **X**, **U**, **F**, **G**, ...).

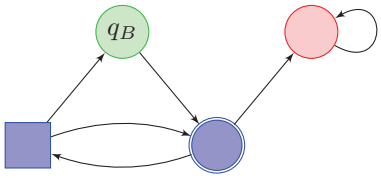

FIG. 1.1 – Exemple de modélisation de jeu à tours avec deux joueurs

Emboîter des quantificateurs de stratégies. Reprenons la formule (1) ci-dessus. Supposons que les "états sans danger" soient ceux depuis lesquels le joueur B dispose d'une stratégie pour atteindre son but (représenté par l'état q_B), et cela une infinité de fois. Considérons le système présenté dans la

Figure 1.1, où les états ronds représentent ceux où c'est au tour du protagoniste A de jouer (c'est-à-dire que le joueur A choisit seul la transition qui sera empruntée depuis ces états ; on dit de ces état qu'ils sont *contrôlés par* A) et les états carrés sont contrôlés par le joueur B. On voit facilement que ce jeu ne contient aucun "état sans danger" : après chaque passage en q_B, le joueur A peut décider d'amener le système dans l'état le plus à droite, depuis lequel q_B n'est pas accessible. On en déduit que le joueur A n'a pas de stratégie qui maintienne le système dans les états sans danger.

A présent, supposons que le joueur A ait déjà choisi d'emprunter systématiquement la transition de gauche, lorsque le système est dans l'état initial (doublement entouré). Alors sous cette stratégie, il suffit au joueur B de toujours aller en q_B quand le système est dans l'état carré pour remplir son objectif de passer par l'état q_B infiniment souvent. Le différence avec le cas précédent est qu'ici, le joueur B *profite* de la stratégie du joueur A pour remplir son objectif. Dans ce cas, depuis l'état initial, le joueur A dispose bien d'une stratégie pour maintenir le système dans des états sans danger.

Ces deux interprétations de la propriété énoncée peuvent faire sens, selon le contexte. Cependant, la sémantique originale de ATL ne permet pas d'envisager la seconde interprétation : les quantificateurs de stratégies dont l'on dispose dans ATL "effacent" les stratégies préalablement choisies. Tout en étant algorithmiquement très judicieuse, puisqu'elle rend le model-checking de ATL faisable en un temps polynomial, cette sémantique empêche ATL d'exprimer un certain nombre de propriétés intéressantes sur les jeux (en particulier sur les jeux à somme non nulle).

Dans [8], nous avons introduit une sémantique alternative pour ATL, dans laquelle les quantificateurs de stratégies *stockent* les stratégies dans un *contexte*. Ces stratégies interviennent alors dans l'évaluation de toutes les sous-formules, jusqu'à ce qu'elles soient explicitement retirées du contexte, ou bien remplacées par de nouvelles stratégies.

Nos contributions Une première contribution de cette thèse est qu'elle enrichit ATL et ATL* afin de pouvoir exprimer des propriétés inédites : nous avons modifié le quantificateur de stratégies de ATL pour qu'il considère toujours les stratégies introduites par les quantificateurs précédents lors de l'évaluation d'une formule. Ainsi, notre nouvelle modalité, que nous écrivons $\ll A \gg$, nous permet d'exprimer la propriété "A a une stratégie telle que :

1. Le joueur B a toujours une stratégie (compte tenu de celle de A) pour garantir Φ et
2. Le joueur C a toujours une stratégie (compte tenu de la *même* stratégie de A) pour garantir Ψ".

On écrit cela :

$$\ll A \gg \mathbf{G} \left(\ll B \gg \Phi \wedge \ll C \gg \Psi \right)$$

Remarquons que les tentatives naïves pour exprimer cette propriété par l'intermédiaire de ATL échouent : par exemple, dans la formule de ATL

$$\langle\langle A\rangle\rangle \mathbf{G}\,(\,\langle\langle B\rangle\rangle \Phi \wedge \langle\langle C\rangle\rangle \Psi),$$

les coalitions ne coopèrent plus, tandis que dans

$$\langle\langle A\rangle\rangle \mathbf{G}\,(\,\langle\langle A, B\rangle\rangle \Phi \wedge \langle\langle A, C\rangle\rangle \Psi),$$

la coalition A est autorisée à utiliser des stratégies différentes selon qu'elle joue avec B ou avec C.

Une autre contribution de ce travail est de paramétrer les quantificateurs de stratégies avec les *ressources* (en termes de mémoire) autorisées pour les stratégies : nous définissons le quantificateur $\langle\!\langle A\rangle\!\rangle_b$, où $b \in (\mathbb{N} \cup \{\infty\})$, qui restreint la quantification aux stratégies qui utilisent une mémoire de taille b (que l'on appelle une stratégie à mémoire b) pour le joueur A. La modalité classique $\langle\langle A\rangle\rangle$ considère toutes les stratégies (y compris les stratégies à mémoire infinie). Un résultat connu établit que les stratégies sans mémoire sont suffisantes pour garantir les propriétés de ATL, mais ce n'est pas le cas pour les formules de ATL*, et pas non plus pour nos extensions de ATL et ATL* avec contextes stratégiques. Ces définitions feront l'objet du deuxième chapitre de cette thèse.

Dans le troisième chapitre, nous présentons des définitions et des résultats classsiques sur les équivalences comportementales : équivalence de trace, bisimulation puis bisimulation alternante. Ces notions seront utilisées par la suite pour énoncer certains résultats. Et nous introduisons une notion originale de GL-équivalence, dont nous montrons qu'elle caractérise le fait, pour deux états, de vérifier exactement les mêmes formules de la logique GL [3]

Du point de vue de l'expressivité, nos résultats sont de deux formes : nous avons étudié précisément l'expressivité des nouvelles logiques que nous avons définies, en les comparant à celles des logiques classiques comme ATL, ATL*, le μ-calcul alternant (AMC) [3], la Game Logic (GL) [3], et également celles de nouveaux formalismes tels que Strategy Logic (SL) [11] et QD_μ [28, 29]. Alors que nos logiques n'englobent aucune de ces deux dernières, nous montrons qu'elles permettent d'exprimer la plupart des propriétés intéressantes, notamment celles qui concernent l'existence d'équilibres ou de stratégies dominantes. De plus, nous prouvons que ATL$^*_{sc,\infty}$ n'est pas plus expressive que ATL$_{sc,\infty}$: il s'agit d'un argument théorique qui témoigne de la puissance expressive des contextes stratégiques ; cela s'ajoute aux arguments plus pratiques que l'on vient d'exposer. Nous exposerons ces résultats dans le quatrième chapitre.

Nous nous penchons également sur les problèmes de model-checking de nos logiques. D'une part, nous proposons un algorithme en espace polynomial pour le model-checking de nos logiques dans le cas des stratégies sans mémoire, et nous l'étendons en un algorithme en espace exponentiel pour le cas plus général où la mémoire est bornée. D'autre part, dans le cas général, nous présentons un algorithme (non élémentaire) de model-checking pour ATL$^*_{sc,\infty}$, qui s'appuie sur la notion d'automates d'arbres alternants.

Notre algorithme passe par un codage original des stratégies dans l'arbre des exécutions de la structure de jeux concurrents sous-jacente. De cette manière, l'algorithme s'applique correctement sur toute la classe des structures de jeux concurrents, et cela sans restrictions sur les mémoires des stratégies, contrairement aux algorithmes déjà développés pour les autres extensions de ATL. Nous montrons enfin que ce problème de model-checking est non élémentaire. Ces résultats concernant les problèmes de model-checking constitueront le cinquième chapitre.

Travaux connexes. Ces trois dernières années, plusieurs travaux se sont focalisés sur des formes proches d'extension de ATL, qui ont apporté différentes solutions, et dont nous établissons ci-dessous une liste. De manière générale, ceci amène à des logiques très expressives, qui sont à même d'exprimer des propriétés sur des équilibres. En contrepartie, les problèmes de model-checking sont d'une grande complexité.

- IATL [1, 2] étend ATL avec des contextes stratégiques au moyen d'une définition semblable à la nôtre, mais elle oblige les joueurs à choisir une stratégie, qu'ils ne sont pas autorisés à remettre en cause par la suite. Un algorithme est proposé pour cette sous-classe de notre logique dans laquelle les stratégies stockées dans le contexte sont *irrévocables* et ne peuvent pas être remplacées. Cette logique est ensuite étudiée dans le cas sans mémoire (qui est une restriction stricte par rapport aux stratégies à mémoire bornée) dans [35], et un algorithme NP est proposé.
- *Strategy logic* [11, 12] (SL) étend LTL au moyen de quantifications du premier ordre sur les stratégies. Cela confère une forte expressivité : à titre d'exemple, la propriété décrite par la formule (1) s'écrirait

$$\exists \sigma_A.\ \forall \sigma'_B \left[\mathbf{G}\ \left(\exists \sigma_B.\ \left(\mathbf{G}\,\mathbf{F}\ q_B \right) (\sigma_A, \sigma_B) \right) \right] (\sigma_A, \sigma'_B)$$

  Dans [11, 12], cette logique a été étudiée uniquement dans le cas des structures de jeu à tours avec deux joueurs, et un algorithme de model-checking non élémentaire est donné. Il y a de plus une restriction syntaxique assez forte qui impose que les modalités temporelles ne s'appliquent qu'à des formules closes. L'algorithme que nous proposons dans cette thèse pourrait être modifié afin de s'appliquer à cette définition de Strategy Logic dans le cadre plus large des structures de jeux concurrents avec un nombre arbitraire de joueurs. C'est justement dans ce cadre qu'a été définie une version alternative de SL [23]. Cette définition généralise la précédente en ôtant toute restriction syntaxique. Cette étude contient un nouvel algorithme de model-checking dont nous reparlerons dans la section 5.6, ainsi qu'une preuve de l'indécidabilité du problème de satisfaisabilité.
- *Stochastic game logic* [5] est une extension de ATL semblable à la nôtre, mais dans le cadre des jeux stochastiques. Son problème de model-checking est indécidable dans le cas général, mais décidable lorsque les quantifications sur les stratégies se restreignent au cas des stratégies sans mémoire (randomisées ou déterministes).

– Q$D\mu$ [28] est une extension au second ordre du μ-calcul propositionel, à laquelle on a ajouté des modalités de décision. On y considère des stratégies comme des étiquetages de l'arbre d'exécutions de la structure de jeu par de nouvelles propositions atomiques. On peut ainsi parler explicitement de stratégies. Du point de vue de l'expressivité, les points fixes permettent une sémantique plus riche que les approches fondées sur CTL ou LTL. Dans ce cadre très général, le model-checking est également prouvé décidable, mais d'une complexité non élémentaire, sur une classe restreinte de CGS. Nous détaillerons ce point dans la section 5.5

Chapitre 2

Définitions

Cette partie a pour objet d'introduire les principales définitions dont nous aurons besoin dans la suite. Dans la première section, nous présentons les CGS [3], un modèle de jeu couramment utilisé, ainsi que plusieurs notions classiques de stratégies [3, 22]. Nous rappelons dans la deuxième section les définitions formelles d'un certain nombre de formalismes qui nous seront utiles. Le reste de ce chapitre est employé à l'introduction de nouvelles logiques temporelles. D'abord les logiques ATL^*_b permettant d'exprimer des propriétés sur des stratégies à mémoire bornée, puis à partir du paragraphe 2.4, des formalismes plus complexes qui traitent les stratégies comme des contextes. Nous avons distingué ces derniers en fonction du type de stratégies considérées.

2.1 Les modèles de la logique temporelle alternante

Nous définissons ici une structure permettant de modéliser des systèmes contrôlés par *plusieurs* protagonistes (que l'on appelera souvent *agents*, ou encore *joueurs*). Ce modèle est largement inspiré des *systèmes de transitions* dans lesquels on interprète principalement les logiques temporelles linéaires (par exemple LTL) et arborescentes (comme CTL).

Les trois définitions qui suivent proviennent de [3].

Définition 1. *Une* CGS *(de l'anglais Concurrent Game Structure)* $\mathcal{C}$ *est un* 7*-uple* $(\mathsf{Agt}, \mathsf{Loc}, \mathsf{AP}, \mathcal{M}, \mathsf{Lab}, \mathsf{Mov}, \mathsf{Edg})$ *tel que :*

- $\mathsf{Agt} = \{A_1, ..., A_k\}$ *est un ensemble fini de* joueurs *;*
- Loc *est un ensemble fini d'*états *;*
- AP *est un ensemble fini de* propositions atomiques *;*
- $\mathcal{M}$ *est un ensemble fini de mouvements ;*
- $\mathsf{Lab}\colon \mathsf{Loc} \to 2^{\mathsf{AP}}$ *est une fonction qui associe à chaque état l'ensemble des propositions atomiques qui sont vraies dans cet état ;*

– Mov: $\mathsf{Loc} \times \mathsf{Agt} \to \mathcal{P}(\mathcal{M}) \setminus \{\varnothing\}$ *est la fonction de choix. Etant donnés un joueur* A_i *et un état* s, $\mathsf{Mov}(s, A_i)$ *correspond à l'ensemble des mouvements que le joueur* A_i *peut jouer depuis* s ;
– Edg: $\mathsf{Loc} \times \mathcal{M}^k \to \mathsf{Loc}$, *où* $k = |\mathsf{Agt}|$, *est une fonction partielle qui définit la table des transitions. A chaque état et à chaque* k*-uple de mouvement des joueurs, elle associe l'état résultant.*

Si l'ensemble Agt est réduit à un singleton, c'est-à-dire s'il n'y a qu'un seul joueur, alors la structure $\mathcal{C}$ est appelée une *Structure de Kripke*.

Si chaque état de $\mathcal{C}$ est contrôlé par un seul joueur (pas forcément le même joueur pour chaque état), $\mathcal{C}$ est appelée une *Turn-Based Game Structure*, soit en français une structure de jeu à tours.

Une Structure de Kripke est donc un cas particulier de Turn-Based Game Structure avec un unique joueur.

Définition 2. *Soit* $\mathcal{C}$ *une* CGS, *un état* $s \in \mathsf{Loc}$, *une coalition* $A \subseteq \mathsf{Agt}$, *et une famille de mouvements autorisés pour cette coalition depuis cet état, c'est-à-dire que :*

$$m_A = (m_{A_i})_{A_i \in A} \in \prod_{A_i \in A} \mathsf{Mov}(s, A_i)$$

On note $\mathsf{Next}(s, A, m_A)$ *l'ensemble des successeurs de* s *lorsque chaque joueur* A_i *choisit le mouvement* m_{A_i}. *Formellement, on définit :*

$$\mathsf{Next}(s, A, m_A) = \{s' \in \mathsf{Loc} \mid \textit{il existe un vecteur de mouvements } m' = (m'_{A_i})_{A_i \in \mathsf{Agt}} \textit{ t.q. } \mathsf{Edg}(s, m') = s' \textit{ et } \forall A_i \in A, m'_{A_i} = m_{A_i}\}$$

Dans le cas particulier $A = \varnothing$, on a nécessairement $m_A = \varnothing$, et $\mathsf{Next}(s, A, m_A)$ sera alors noté plus simplement $\mathsf{Next}(s)$: il s'agit de l'ensemble des successeurs de s.

Une *exécution* de $\mathcal{C}$ est une suite infinie $\lambda = s_0 s_1 ...$ d'états qui vérifie que pour tout $k \in \mathbb{N}$, $s_{k+1} \in \mathsf{Next}(s_k)$. On utilisera les notations suivantes de préfixes et suffixes pour les exécutions :
– $\lambda[i, j]$ désigne la partie de λ comprise entre s_i et s_j
– $\lambda[i, \infty]$ désigne le suffixe de λ qui démarre à s_i
– $\lambda[i]$ désigne le i-ème état s_i de λ

Si $\lambda[0] = s$, on dit que λ est une s-exécution. On note $\mathsf{Exec}(s)$ l'ensemble des s-exécutions de $\mathcal{C}$.

2.1.1 Stratégies avec mémoire infinie et stratégies positionnelles

Une *stratégie* avec mémoire infinie pour un joueur $A_i \in \mathsf{Agt}$ est une application F qui associe un mouvement du joueur $A_i \in A$ à chaque préfixe d'exécution. On dit qu'une stratégie est *sans mémoire* si le mouvement associé ne dépend

que de l'état courant. Autrement dit, une *stratégie sans mémoire* (ou *stratégie positionnelle*) pour le joueur A_i est une application F qui associe à chaque état de Loc un mouvement autorisé de ce joueur A_i depuis cet état. On note respectivement $\mathsf{Strat}^{\infty}_{A_i}$ et $\mathsf{Strat}^{0}_{A_i}$ les ensembles des stratégies à mémoire infinie et des stratégies sans mémoire de $\mathcal{C}$ pour le joueur A_i.

Définition 3. *Formellement, on écrit :*

- $\mathsf{Strat}^{\infty}_{A_i} = \{F \in \mathcal{M}^{\bigcup_{j\in\mathbb{N}\setminus\{0\}} \mathsf{Loc}^j} \mid \forall j \in \mathbb{N}\setminus\{0\}, \forall \lambda \in \mathsf{Loc}^j, F(\lambda) \in \mathsf{Mov}(\lambda[j], A_i)\}$,
- $\mathsf{Strat}^{0}_{A_i} = \{F \in \mathcal{M}^{\mathsf{Loc}} \mid \forall q \in \mathsf{Loc}, F(q) \in \mathsf{Mov}(q, A_i)\}$.

Pour toute coalition $A \subseteq \mathsf{Agt}$, on note

$$\mathsf{Strat}^{0}_{A} = \prod_{A_i \in A} \mathsf{Strat}^{0}_{A_i} \quad \textit{et} \quad \mathsf{Strat}^{\infty}_{A} = \prod_{A_i \in A} \mathsf{Strat}^{\infty}_{A_i}$$

De plus, dans ces deux cas on notera $F(\lambda) = \big(F_{A_i}(\lambda)\big)_{A_i \in A}$.

Une stratégie (avec ou sans mémoire) F pour la coalition A induit un sous-ensemble de $\mathsf{Exec}(s)$, que l'on appelle *ensemble des outcomes de F depuis l'état s*, constitué des s-exécutions que les joueurs A_i de la coalition A contraignent lorsqu'ils jouent selon la stratégie F.

Définition 4.

- *Dans le cas des stratégies à mémoire infinie*
$$\lambda \in \mathsf{Out}(s, F) \textit{ ssi } \lambda[0] = s, \textit{ et pour tout } k \in \mathbb{N} \textit{ on a}$$
$$\lambda[k+1] \in \mathsf{Next}\Big(\lambda[k], A, F\big(\lambda[0,k]\big)\Big)$$
- *Dans le cas des stratégies sans mémoire*
$$\lambda \in \mathsf{Out}(s, F) \textit{ ssi } \lambda[0] = s, \textit{ et pour tout } k \in \mathbb{N} \textit{ on a}$$
$$\lambda[k+1] \in \mathsf{Next}\Big(\lambda[k], A, F\big(\lambda[k]\big)\Big).$$

Deux remarques :

- Dans le cas particulier $A = \varnothing$, on a $\mathsf{Strat}^{\infty}_{A} = \mathsf{Strat}^{0}_{A} = \{\varnothing\}$: l'unique stratégie pour la coalition vide est la stratégie vide. De plus,
$$\forall s \in \mathsf{Loc}, \mathsf{Out}(s, \varnothing) = \mathsf{Exec}(s)$$
- Dans l'autre cas particulier $A = \mathsf{Agt}$, une stratégie F, avec ou sans mémoire, pour la coalition A, ainsi qu'un état s, alors l'ensemble $\mathsf{Out}(s, F)$ est réduit à une s-exécution. Réciproquement, dans le cas où F est une stratégie à mémoire infinie, chaque s-exécution constitue l'ensemble des outcomes d'une stratégie globale (c'est-à-dire pour tous les joueurs).

Lorsque le contexte ne sera pas clair, on indiquera en indice la *CGS* que l'on souhaite considérer.

2.1.2 Les stratégies avec mémoire bornée

Dans cette partie, nous présentons deux définitions de stratégies avec mémoire finie, dont la première, plus naturelle, semble moins fertile, pour une raison que nous avons cherché à expliciter.

Première définition de la taille de la mémoire d'une stratégie, et limites de cette définition

Une première idée, intuitive mais naïve, connue sous le terme de *bounded recall*, consiste à définir les objets suivants :

Définition 5. *Soit un joueur $A_i \in \mathsf{Agt}$ et un entier naturel b. On note Strat_A^b l'ensemble des applications F qui vont de $\bigcup_{j=1,...,b+1} \mathsf{Loc}^j$ dans $\mathcal{M}$, et qui vérifient que :*

$$\forall j \in \{1, ..., b+1\}.\ \forall \lambda \in \mathsf{Loc}^j.\ F(\lambda) \in \mathsf{Mov}(\lambda[j-1], A_i)\}$$

Un élément F de Strat_A^b associe donc à chaque préfixe d'exécution λ de taille inférieure à $b+1$, un mouvement du joueur A_i disponible depuis le dernier état constituant λ.

Soient F une stratégie à mémoire b pour une coalition A, et un état s. F induit un sous-ensemble de $\mathsf{Exec}(s)$, appelé ensemble des *outcomes* de F depuis s, qui est constitué des s-exécutions que les joueurs $A_i \in A$ contraignent lorsqu'ils jouent selon la stratégie F. Une exécution λ est dans $\mathsf{Out}(s, F)$ ssi les deux conditions suivantes sont vraies :

1. $\lambda[0] = s$
2. pour tout entier k, on a $\lambda[k+1] \in \mathsf{Next}\Big(\lambda[k], A, F\big(\lambda[\mathsf{Max}(k-b, 0), k]\big)\Big)$

Remarques :

- Dans les cas $b = 0$ et $b = \infty$, on retrouve les notations déjà introduites.
- Dans le cas particulier où A est la coalition vide, pour toute taille de mémoire b, l'ensemble Strat_A^b est réduit au singleton $\{\varnothing\}$. L'unique stratégie pour la coalition $\varnothing$ est donc la stratégie vide, et on a :

$$\forall s \in \mathsf{Loc}, \mathsf{Out}(s, \varnothing) = \mathsf{Exec}(s)$$

Lorsque le contexte ne sera pas clair, on mettra les indices correspondant à une *CGS*.

La proposition suivante met en évidence les limites de cette définition de la mémoire bornée.

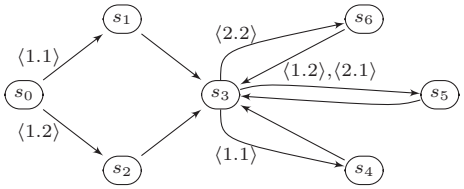

FIG. 2.1 – Les limites de l'approche

Proposition 1. *Sur la CGS de la Figure 2.1, aucune stratégie à mémoire bornée du joueur A_1 n'a toutes ses outcomes qui évitent s_6 si on est passé par s_1 (condition C_1), et qui évitent s_4 si on est passé par s_2 (condition C_2).*

Preuve : Soit $b \in \mathbb{N}$, et F une stratégie à mémoire b du joueur A_1. Supposons que b est impair, le cas où b est pair pouvant se traiter de la même façon. Considérons l'exécution finie $\alpha = s_3 s_5 s_3 s_5 ... s_3$ de longueur b. Supposons que $F(\alpha) = 1$ (si ce n'est pas le cas, alors $F(\alpha) = 2$, et on peut raisonner de façon similaire). Soit λ l'exécution infinie telle que :

- $\lambda[0] = s_0$,
- $\lambda[1] = s_2$,
- Si $m \geq 2$ est pair, alors $\lambda[m] = s_3$,
- $\lambda[b+2] = s_4$,
- Si $m \geq 3$ est impair et $m \neq b+2$, alors $\lambda[m] = s_5$

λ passe par s_2 et s_4, donc elle ne vérifie pas la condition C_2. De plus, c'est une outcome de F depuis s_0 puisque $F(\alpha) = 1$. Dans le cas où $F(\alpha) = 2$, on peut construire similairement une outcome qui ne vérifie pas la condition C_1. Il est donc impossible de satisfaire les deux conditions à la fois au moyen d'une stratégie à mémoire bornée.

□

Il n'y a pourtant qu'une seule information à retenir pour remplir simultanément les deux conditions : il suffirait de se rappeler si l'on est passé par s_1 ou par s_2 au début. C'est pourquoi nous présentons maintenant une autre définition de la mémoire bornée, qui permet de retenir une telle information avec une mémoire de 1.

Seconde définition de la taille de la mémoire d'une stratégie

Une autre idée plus féconde (qui est d'ailleurs la définition classique [22]) consiste à définir les objets suivants :

b représente ici un entier codé en binaire. On définit une stratégie à mémoire bornée comme une stratégie positionnelle sur une *CGS* et sur un ensemble de *cellules mémoire* : le mouvement choisi dépend de l'état de la *CGS*, mais aussi de la cellule mémoire courante. De plus, après chaque mouvement, le joueur peut actualiser sa mémoire en changeant de cellule. La taille de la mémoire est alors définie comme le nombre de cellules. On notera Cell l'ensemble de $b+1$ cellules mémoire $\{0, ..., b\}$.

Une stratégie F pour un joueur A_i *de mémoire* b est un triplet $(F^{\mathsf{mov}}, F^{\mathsf{cell}}, c)$, où :

- F^{mov} est une application de $\mathsf{Cell} \times \mathsf{Loc}$ vers $\mathcal{M}$ qui désigne un mouvement en fonction de la cellule mémoire courante, et de l'état courant de la *CGS*. Le mouvement doit être disponible, c'est-à-dire que $F^{\mathsf{mov}}(., q) \in \mathsf{Mov}(q, A_i)$.
- F^{cell} est une application de $\mathsf{Cell} \times \mathsf{Loc}$ vers Cell qui actualise la cellule mémoire.
- c est la cellule mémoire initiale de la stratégie.

Par souci de lisibilité, on écrira $F(q)$ plutôt que $F^{\mathsf{mov}}(c, q)$.

Nous étendons maintenant les notions de successeurs, d'exécutions et d'outcomes à ces nouvelles stratégies :

– l'ensemble $\mathsf{Next}\big(s, A_i, F(s)\big)$ contient les états successeurs possibles lorsque le joueur A_i joue selon la stratégie F depuis l'état s.
– étant donnés une stratégie $F = (F^{\mathsf{mov}}, F^{\mathsf{cell}}, c)$, une exécution ρ et un entier i, on définit $F^{\rho,i}$ comme la stratégie $(F^{\mathsf{mov}}, F^{\mathsf{cell}}, c_i)$, où c_i est défini récursivement par : $c_0 = c$ et $c_{j+1} = F^{\mathsf{cell}}(c_j, \rho[j])$. Ainsi, $F^{\rho,i}$ simule le comportement de F après le préfixe d'exécution $\rho[0, i]$.
– les outcomes $\mathsf{Out}(s, F)$ sont les exécutions $\rho = s_0 s_1 ...$ telles que $s_0 = s$ et pour tout entier i, $s_{i+1} \in \mathsf{Next}\big(s_i, A_i, F^{\rho,i}(s_i)\big)$.

On note $\mathsf{Strat}_{A_i}^{'b}$ l'ensemble des stratégies à mémoire de taille b pour le joueur A_i. Dans le cas d'une coalition $A \subseteq \mathsf{Agt}$, on note

$$\mathsf{Strat}_{A}^{'b} = \prod_{A_i \in A} \mathsf{Strat}_{A_i}^{'b}$$

Dans le cas d'une stratégie $F = (F_{A_i})_{A_i \in A}$ de mémoire b pour une coalition A, on notera $F(q)$ pour $\big(F_{A_i}(q)\big)_{A_i \in A}$.

Les *outcomes* de F depuis l'état s sont alors les exécutions $\rho = s_0 s_1 ...$ telles que $s_0 = s$ et pour tout entier i, $s_{i+1} \in \mathsf{Next}\big(s_i, A, F^{\rho,i}(s_i)\big)$.

Remarques :
– Lorsqu'il y aura ambiguïté sur la *CGS* considérée, on mettra celle-ci en indice.
– Dans le cas particulier où A est la coalition vide, les ensembles de stratégies $\mathsf{Strat}_{A}^{'b}$ sont toujours réduit au singleton $\{\varnothing\}$, quel que soit l'entier b. L'unique stratégie pour la coalition $\varnothing$ est donc la stratégie vide, et depuis tout état s, on a $\mathsf{Out}(s, \varnothing) = \mathsf{Exec}(s)$

2.2 Quelques logiques temporelles classiques

Nous présentons dans cette partie plusieurs logiques temporelles qui permettent d'énoncer des propriétés sur les *CGS*.

2.2.1 Logique du temps linéaire : LTL

La première d'entre elles est la logique LTL, sigle de l'anglais Linear-Time Temporal Logic. Elle fut à l'origine définie par Pnueli dans son article fondateur [30] qui a placé les logiques temporelles dans le champ de l'informatique théorique.

LTL s'appliquait alors sur des structures linéaires, c'est-à dire des suites infinies d'ensembles de propositions atomiques. Cette sémantique s'étend naturellement aux exécutions des structures de Kripke [33]. De la même manière, nous considérons ici des exécutions de CGS, qui sont les principaux modèles utilisés dans cette thèse.

Définition 6. *La syntaxe de LTL est définie par la grammaire suivante :*

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

où p décrit l'ensemble AP.

La sémantique de LTL est définie comme suit :
Soit λ une exécution de la structure $\mathcal{C}$.

$$\begin{array}{rcl}
(\mathcal{C},\lambda) \models p & \text{ssi} & p \in \mathsf{Lab}(\lambda[0]) \\
(\mathcal{C},\lambda) \models \neg\varphi_p & \text{ssi} & (\mathcal{C},\lambda) \not\models \varphi_p \\
(\mathcal{C},\lambda) \models \varphi_p \vee \psi_p & \text{ssi} & (\mathcal{C},\lambda) \models \varphi_p \text{ ou } (\mathcal{C},\lambda) \models \psi_p \\
(\mathcal{C},\lambda) \models \mathbf{X}\,\varphi_p & \text{ssi} & (\mathcal{C},\lambda[1,\infty]) \models \varphi_p \\
(\mathcal{C},\lambda) \models \varphi_p\,\mathbf{U}\,\psi_p & \text{ssi} & \exists i \geq 0.\ (\mathcal{C},\lambda[i,\infty]) \models \psi_p \text{ et} \\
 & & \forall 0 \leq j < i.\ (\mathcal{C},\lambda[j,\infty]) \models \varphi_p
\end{array}$$

Par convention, si q désigne un état de la structure $\mathcal{C}$:

$$(\mathcal{C},q) \models \varphi_p \Leftrightarrow \forall \lambda \in \mathsf{Exec}(q).\ (\mathcal{C},\lambda) \models \varphi_p$$

Nous utiliserons les abréviations habituelles suivantes :
– $\mathbf{F}\,\varphi$ sera l'abréviation de $\text{true}\,\mathbf{U}\,\varphi$ (c'est-à-dire "on aura un jour φ").
– $\mathbf{G}\,\varphi$ sera l'abréviation de $\neg\mathbf{F}\,\neg\varphi$ (c'est-à-dire "on a toujours φ").

Les opérateurs $\mathbf{X}$, $\mathbf{U}$, $\mathbf{F}$ et $\mathbf{G}$ constituent ce que l'on appelle les *opérateurs temporels*.

2.2.2 Logiques du temps arborescent : CTL et CTL*

Nous introduisons à présent une logique temporelle arborescente, CTL, pour Computation Tree Logic. Cette définition a été formulée par Emerson et Clarke dans [13]. Cette logique permet d'exprimer des propriétés dans lesquelles on quantifie existentiellement ou universellement sur les exécutions qui partent d'un état.

Définition 7. *La syntaxe de CTL est définie par la grammaire suivante :*

$$\begin{array}{rcl}
\mathsf{CTL} \ni \varphi_s, \psi_s & ::= & p \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \mathbf{A}\varphi_p, \mid \mathbf{E}\varphi_p \\
\mathsf{CTL}_p \ni \varphi_p & ::= & \varphi_s \mid \mathbf{X}\,\varphi_s \mid \varphi_s\,\mathbf{U}\,\psi_s
\end{array}$$

où p décrit l'ensemble AP.

La sémantique de CTL est définie comme suit :

Soit $q \in \mathsf{Loc}$

$$
\begin{array}{rcl}
(\mathcal{C}, q) \models p & \text{ssi} & p \in \mathsf{Lab}(q) \\
(\mathcal{C}, q) \models \neg\varphi_s & \text{ssi} & (\mathcal{C}, q) \not\models \varphi_s \\
(\mathcal{C}, q) \models \varphi_s \vee \psi_s & \text{ssi} & (\mathcal{C}, q) \models \varphi_s \text{ ou } (\mathcal{C}, q) \models \psi_s \\
(\mathcal{C}, q) \models \mathbf{A}\varphi_p & \text{ssi} & \forall \lambda \in \mathsf{Exec}(q).\ (\mathcal{C}, \lambda) \models \varphi_p \\
(\mathcal{C}, q) \models \mathbf{E}\varphi_p & \text{ssi} & \exists \lambda \in \mathsf{Exec}(q).\ (\mathcal{C}, \lambda) \models \varphi_p \\
(\mathcal{C}, \lambda) \models \varphi_s & \text{ssi} & (\mathcal{C}, \lambda[0]) \models \varphi_s \\
(\mathcal{C}, \lambda) \models \mathbf{X}\,\varphi_s & \text{ssi} & (\mathcal{C}, \lambda[1]) \models \varphi_s \\
(\mathcal{C}, \lambda) \models \varphi_s\,\mathbf{U}\,\psi_s & \text{ssi} & \exists i \geq 0.\ (\mathcal{C}, \lambda[i]) \models \psi_s \text{ et} \\
 & & \forall 0 \leq j < i.\ (\mathcal{C}, \lambda[j]) \models \varphi_s
\end{array}
$$

Nous allons à présent définir une extension de CTL que l'on nomme CTL*, introduite par Emerson et Halpern [16]. Les quantificateurs de chemin et les modalités temporelles peuvent être arbitrairement emboîtés, alors que dans CTL, chaque modalité temporelle devait être immédiatement précédée d'un quantificateur de chemin. Définie sur les CGS, cette logique étend à la fois CTL et LTL.

Définition 8. *La syntaxe de CTL* est définie par la grammaire suivante :*

$$
\begin{array}{rcl}
\mathit{CTL}^* \ni \varphi_s, \psi_s & ::= & p \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \mathbf{A}\varphi_p \mid \mathbf{E}\varphi_p \\
\mathit{CTL}^*_p \ni \varphi_p, \psi_p & ::= & \varphi_s \mid \neg\varphi_p \mid \varphi_p \vee \psi_p \mid \mathbf{X}\,\varphi_p \mid \varphi_p\,\mathbf{U}\,\psi_p
\end{array}
$$

où p décrit l'ensemble AP.

La sémantique de CTL* est définie comme suit :
Soit $q \in \mathsf{Loc}$.

$$
\begin{array}{rcl}
(\mathcal{C}, q) \models p & \text{ssi} & p \in \mathsf{Lab}(q) \\
(\mathcal{C}, q) \models \neg\varphi_s & \text{ssi} & (\mathcal{C}, q) \not\models \varphi_s \\
(\mathcal{C}, q) \models \varphi_s \vee \psi_s & \text{ssi} & (\mathcal{C}, q) \models \varphi_s \text{ ou } (\mathcal{C}, q) \models \psi_s \\
(\mathcal{C}, q) \models \mathbf{A}\varphi_p & \text{ssi} & \forall \lambda \in \mathsf{Exec}(q).\ (\mathcal{C}, \lambda) \models \varphi_p \\
(\mathcal{C}, q) \models \mathbf{E}\varphi_p & \text{ssi} & \exists \lambda \in \mathsf{Exec}(q).\ (\mathcal{C}, \lambda) \models \varphi_p \\
(\mathcal{C}, \lambda) \models \varphi_s & \text{ssi} & (\mathcal{C}, \lambda[0]) \models \varphi_s \\
(\mathcal{C}, \lambda) \models \neg\varphi_p & \text{ssi} & (\mathcal{C}, \lambda) \not\models \varphi_p \\
(\mathcal{C}, \lambda) \models \varphi_p \vee \psi_p & \text{ssi} & (\mathcal{C}, \lambda) \models \varphi_p \text{ ou } (\mathcal{C}, \lambda) \models \psi_p \\
(\mathcal{C}, \lambda) \models \mathbf{X}\,\varphi_p & \text{ssi} & (\mathcal{C}, \lambda[1, \infty]) \models \varphi_p \\
(\mathcal{C}, \lambda) \models \varphi_p\,\mathbf{U}\,\psi_p & \text{ssi} & \exists i \geq 0.\ (\mathcal{C}, \lambda[i, \infty]) \models \psi_p \text{ et} \\
 & & \forall 0 \leq j < i.\ (\mathcal{C}, \lambda[j, \infty]) \models \varphi_p
\end{array}
$$

2.2.3 Logiques du temps alternant : ATL et ATL*

Nous passons maintenant à une logique temporelle *alternante*, appelée ATL. Les logiques temporelles alternantes constituent un troisième type de logique (en

plus de celles du temps linéaire et du temps branchant), dans lequel on dispose de quantificateurs *sélectifs* de chemin, que l'on appelle des quantificateurs de stratégies.

Depuis un état s, on peut ainsi quantifier sur des sous-ensembles stricts de s-exécutions. Cette logique a été introduite par Alur, Henzinger et Kupferman dans [3], et elle est définie directement sur les CGS. La syntaxe de ATL est inspirée de celle de CTL.

Définition 9. *La syntaxe de ATL est définie par la grammaire suivante :*

$$\begin{aligned} \mathit{ATL} \ni \varphi_s, \psi_s &::= p \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \langle\!\langle A \rangle\!\rangle \varphi_p \\ \mathit{ATL}_p \ni \varphi_p &::= \varphi_s \mid \mathbf{X}\,\varphi_s \mid \varphi_s \,\mathbf{U}\, \psi_s, \mid \varphi_s \,\mathbf{W}\, \psi_s. \end{aligned}$$

où p décrit l'ensemble AP et A l'ensemble des parties de Agt.

La sémantique de ATL est définie comme suit :
Soient $A \subseteq \mathsf{Agt}$ et $q \in \mathsf{Loc}$

$$\begin{array}{rcl} (\mathcal{C}, q) \models p & \text{ssi} & p \in \mathsf{Lab}(q) \\ (\mathcal{C}, q) \models \neg\varphi_s & \text{ssi} & (\mathcal{C}, q) \not\models \varphi_s \\ (\mathcal{C}, q) \models \varphi_s \vee \psi_s & \text{ssi} & (\mathcal{C}, q) \models \varphi_s \text{ ou } (\mathcal{C}, q) \models \psi_s \\ (\mathcal{C}, q) \models \langle\!\langle A \rangle\!\rangle \varphi_p & \text{ssi} & \exists F \in \mathsf{Strat}_A^{\infty}.\ \forall \lambda \in \mathsf{Out}(q, F).\ (\mathcal{C}, \lambda) \models \varphi_p \\ (\mathcal{C}, \lambda) \models \varphi_s & \text{ssi} & (\mathcal{C}, \lambda[0]) \models \varphi_s \\ (\mathcal{C}, \lambda) \models \mathbf{X}\,\varphi_s & \text{ssi} & (\mathcal{C}, \lambda[1]) \models \varphi_s \\ (\mathcal{C}, \lambda) \models \varphi_s \,\mathbf{U}\, \psi_s & \text{ssi} & \exists i \geq 0.\ (\mathcal{C}, \lambda[i]) \models \psi_s \text{ et } \forall 0 \leq j < i.\ (\mathcal{C}, \lambda[j]) \models \varphi_s \\ (\mathcal{C}, \lambda) \models \varphi_s \,\mathbf{W}\, \psi_s & \text{ssi} & (\mathcal{C}, \lambda) \models \varphi_s \,\mathbf{U}\, \psi_s \text{ ou } \forall i \geq 0.\ (\mathcal{C}, \lambda[i]) \models \varphi_s \end{array}$$

Remarques :
- Les opérateurs $\langle\!\langle A \rangle\!\rangle$ sont appelés des *quantificateurs de stratégie*,
- Il est connu que dans ce cadre (celui de ATL), on peut se restreindre aux stratégies sans mémoire [3] lors des énumérations, c'est-à-dire que pour pour toute formule de chemin φ_p de ATL, on a :

$$(\mathcal{C}, q) \models \langle\!\langle A \rangle\!\rangle \varphi_p \Leftrightarrow \exists F \in \mathsf{Strat}_A^{0}.\ \forall \lambda \in \mathsf{Out}(q, F).\ (\mathcal{C}, \lambda) \models \varphi_p$$

- Dans le cadre de CTL, l'opérateur temporel $\mathbf{W}$ peut s'exprimer en fonction de l'opérateur $\mathbf{U}$, mais ce n'est pas vrai dans le cadre de ATL [20]. Ceci nous permet de justifier la différence entre nos définitions de ces deux logiques.

Nous introduisons maintenant la logique ATL*, qui est à ATL ce que CTL* est à CTL :

Définition 10. *La syntaxe de ATL* est définie par la grammaire suivante :*

$$\begin{aligned} \mathit{ATL}^* \ni \varphi_s, \psi_s &::= p \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \langle\!\langle A \rangle\!\rangle \varphi_p \\ \mathit{ATL}^*_p \ni \varphi_p, \psi_p &::= \varphi_s \mid \neg\varphi_p \mid \varphi_p \vee \psi_p \mid \mathbf{X}\,\varphi_p \mid \varphi_p \,\mathbf{U}\, \psi_p \end{aligned}$$

où p décrit l'ensemble AP et A l'ensemble des parties de Agt.

La sémantique de ATL^* est définie comme suit :
Soient $A \subseteq \mathsf{Agt}$ et $q \in \mathsf{Loc}$.

$$
\begin{array}{rcl}
(\mathcal{C}, q) \models p & \text{ssi} & p \in \mathsf{Lab}(q) \\
(\mathcal{C}, q) \models \neg\varphi_s & \text{ssi} & (\mathcal{C}, q) \not\models \varphi_s \\
(\mathcal{C}, q) \models \varphi_s \vee \psi_s & \text{ssi} & (\mathcal{C}, q) \models \varphi_s \text{ ou } (\mathcal{C}, q) \models \psi_s \\
(\mathcal{C}, q) \models \langle\!\langle A \rangle\!\rangle \varphi_p & \text{ssi} & \exists F \in \mathsf{Strat}_A^{\infty}.\ \forall \lambda \in \mathsf{Out}(q, F).\ (\mathcal{C}, \lambda) \models \varphi_p \\
(\mathcal{C}, \lambda) \models \varphi_s & \text{ssi} & (\mathcal{C}, \lambda[0]) \models \varphi_s \\
(\mathcal{C}, \lambda) \models \neg\varphi_p & \text{ssi} & (\mathcal{C}, \lambda) \not\models \varphi_p \\
(\mathcal{C}, \lambda) \models \varphi_p \vee \psi_p & \text{ssi} & (\mathcal{C}, \lambda) \models \varphi_p \text{ ou } (\mathcal{C}, \lambda) \models \psi_p \\
(\mathcal{C}, \lambda) \models \mathbf{X}\,\varphi_p & \text{ssi} & (\mathcal{C}, \lambda[1, \infty]) \models \varphi_p \\
(\mathcal{C}, \lambda) \models \varphi_p\,\mathbf{U}\,\psi_p & \text{ssi} & \exists i \geq 0.\ (\mathcal{C}, \lambda[i, \infty]) \models \psi_p \text{ et} \\
 & & \forall 0 \leq j < i.\ (\mathcal{C}, \lambda[j, \infty]) \models \varphi_p
\end{array}
$$

On peut voir CTL et CTL^* comme étant respectivement les fragments de ATL et $\mathsf{ATL}^\star$, dans lesquels les seuls quantificateurs de stratégie sont $\langle\!\langle \varnothing \rangle\!\rangle$ et $\langle\!\langle \mathsf{Agt} \rangle\!\rangle$. Il n'y a plus besoin de considérer de stratégies pour définir la sémantique dans ce cadre restreint ; on peut se contenter de la notion d'exécutions. La remarque précédente sur les ensembles de stratégies pour les coalitions $\varnothing$ et Agt met en évidence les faits suivants :

- $(\mathcal{C}, q) \models \langle\!\langle \varnothing \rangle\!\rangle \varphi_p$ ssi $\forall \lambda \in \mathsf{Exec}(q).\ (\mathcal{C}, \lambda) \models \varphi_p$,
- $(\mathcal{C}, q) \models \langle\!\langle \mathsf{Agt} \rangle\!\rangle \varphi_p$ ssi $\exists \lambda \in \mathsf{Exec}(q).\ (\mathcal{C}, \lambda) \models \varphi_p$.

Ceci explique que les quantificateurs $\langle\!\langle \varnothing \rangle\!\rangle$ et $\langle\!\langle \mathsf{Agt} \rangle\!\rangle$ soient respectivement notés **A** (pour all) et **E** (pour exists). Dans le cas particulier où la structure est une Structure de Kripke, les logiques ATL^* et CTL^* sont confondues.

2.2.4 La logique GL

La *Game Logic* a été introduite dans [3], notamment pour pouvoir exprimer le problème du *module-checking* [18]. Cette logique est une extension d'$\mathsf{ATL}^\star$ qui permet de travailler explicitement sur les arbres d'exécution induits par les stratégies : étant donné un tel arbre t, il est alors possible de quantifier sur les exécutions qui constituent t et de spécifier les propriétés temporelles habituelles. Par exemple, la formule $\exists A.\big((\exists P\,\mathbf{U}\,P') \wedge (\forall \mathbf{F}\,P'')\big)$ exprime qu'il existe une stratégie F_A pour A telle que l'arbre induit par F_A vérifie : (1) il existe une exécution satisfaisant $P\,\mathbf{U}\,P'$ et (2) toutes les exécutions vérifient $\mathbf{F}\,P''$.

Voici une définition formelle de GL :

Définition 11. *La syntaxe de GL est définie par la grammaire suivante :*

$$
\begin{array}{rcl}
\mathsf{GL} \ni \varphi_s, \psi_s & ::= & p \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \exists A \varphi_t \\
\mathsf{GL}_t \ni \varphi_t, \psi_t & ::= & \varphi_s \mid \neg\varphi_t \mid \varphi_t \vee \psi_t \mid \exists \varphi_p \\
\mathsf{GL}_p \ni \varphi_p, \psi_p & ::= & \varphi_t \mid \neg\varphi_p \mid \varphi_p \vee \psi_p \mid \mathbf{X}\,\varphi_p \mid \varphi_p\,\mathbf{U}\,\psi_p
\end{array}
$$

où p décrit AP et A les sous-ensembles de Agt.

La sémantique de GL est définie comme suit :

Soient $A \subseteq \mathsf{Agt}$, $q \in \mathsf{Loc}$, t un arbre d'exécution et λ une exécution, $t[0]$ désignant la racine de l'arbre t et $t(x)$ le sous-arbre complet de t ayant x pour racine.

$$
\begin{array}{rcl}
(\mathcal{C}, q) \models p & \text{ssi} & p \in \mathsf{Lab}(q), \\
(\mathcal{C}, q) \models \neg\varphi_s & \text{ssi} & (\mathcal{C}, q) \not\models \varphi_s \\
(\mathcal{C}, q) \models \varphi_s \vee \psi_s & \text{ssi} & (\mathcal{C}, q) \models \varphi_s \text{ ou } (\mathcal{C}, q) \models \psi_s \\
(\mathcal{C}, q) \models \exists A \varphi_t & \text{ssi} & \exists F \in \mathsf{Strat}_A^{\infty}.\ \big(\mathcal{C}, \mathsf{Out}(q, F)\big) \models \varphi_t \\
(\mathcal{C}, t) \models \varphi_s & \text{ssi} & (\mathcal{C}, t[0]) \models \varphi_s \\
(\mathcal{C}, t) \models \neg\varphi_t & \text{ssi} & (\mathcal{C}, t) \not\models \varphi_t \\
(\mathcal{C}, t) \models \varphi_t \vee \psi_t & \text{ssi} & (\mathcal{C}, t) \models \varphi_t \text{ ou } (\mathcal{C}, t) \models \psi_t \\
(\mathcal{C}, t) \models \exists \varphi_p & \text{ssi} & \exists \lambda \in t.\ (\mathcal{C}, t, \lambda) \models \varphi_p \\
(\mathcal{C}, t, \lambda) \models \varphi_t & \text{ssi} & (\mathcal{C}, t) \models \varphi_t \\
(\mathcal{C}, t, \lambda) \models \neg\varphi_p & \text{ssi} & (\mathcal{C}, t, \lambda) \not\models \varphi_p \\
(\mathcal{C}, t, \lambda) \models \varphi_p \vee \psi_p & \text{ssi} & (\mathcal{C}, t, \lambda) \models \varphi_p \text{ ou } (\mathcal{C}, t, \lambda) \models \psi_p \\
(\mathcal{C}, t, \lambda) \models \mathbf{X}\,\varphi_p & \text{ssi} & (\mathcal{C}, t(\lambda[1]), \lambda[1, \infty]) \models \varphi_p \\
(\mathcal{C}, t, \lambda) \models \varphi_p\,\mathbf{U}\,\psi_p & \text{ssi} & \exists i \geq 0.\ (\mathcal{C}, t(\lambda[i]), \lambda[i, \infty]) \models \psi_p \text{ et} \\
 & & \forall 0 \leq j < i.\ (\mathcal{C}, t(\lambda[j]), \lambda[j, \infty]) \models \varphi_p
\end{array}
$$

Considérons maintenant la spécification : "il existe une stratégie pour le joueur A tel qu'un certain comportement de la coalition des autres joueurs entraîne que l'on vérifie toujours la proposition p, et qu'un autre comportement de cette même coalition implique que l'on ait toujours la proposition q."

Cela s'écrit en GL de la manière suivante :

$$\exists A.\big(\exists \mathbf{G}\, p \wedge \exists \mathbf{G}\, q\big)$$

2.2.5 La logique SL

On donne la définition formelle de cette logique car nous présenterons dans la section 4.7.2 une traduction impliquant ce formalisme.

Strategy Logic (que l'on abrège en SL) a été définie une première fois dans [11] comme une extension de LTL avec des quantifications du premier ordre sur les stratégies. Par exemple, le fait qu'un joueur A a une stratégie qui permette de garantir φ s'écrit alors

$$\exists \sigma_A.\ \forall \sigma_B.\ \varphi(\sigma_A, \sigma_B)$$

où les arguments donnés à φ servent à indiquer l'exécution sur laquelle φ est évaluée. Nous noterons SL_{CHP} cette première version de Strategy Logic, CHP étant le sigle formé par les noms de ses créateurs.

Cette logique a par la suite été redéfinie dans un second article [23], avec trois modifications principales :

– on a cette fois défini la logique sur les CGS en général, avec nombre de joueurs quelconque. Cette définition des CGS suppose de plus que la fonction de transition est complète, c'est-à-dire que chaque joueur dispose de tous les mouvements depuis chaque état
– on s'est défait d'une restriction sémantique qui obligeait les modalités temporelles à n'opérer que sur des formules *closes* (c'est-à-dire sans variables libres)
– la quantification sur une stratégie et son attribution à un joueur constituent deux opérations distinctes dans cette sémantique

La première modification implique que depuis chaque état, tous les joueurs ont le même ensemble de mouvements disponibles. Ainsi l'ensemble de stratégies est le même pour chaque joueur ; cette notion de stratégie est donc différente, et est dite *agent-independant*, car une stratégie n'est plus associée à un joueur particulier lors de sa définition.

La deuxième modification change considérablement les choses, puisqu'il est maintenant possible de maintenir une stratégie le long d'une exécution.

La troisième modification entraîne que, les stratégies pouvant être communes à plusieurs joueurs par la première modification, on peut maintenant spécifier explicitement dans une formule que des joueurs partagent une même stratégie.

Le fait qu'un joueur A a une stratégie qui permette de garantir φ s'écrit dans cette nouvelle version :

$$\langle\!\langle \sigma \rangle\!\rangle \,.\, [\![\sigma']\!] \,.\, (A, \sigma).\ (B, \sigma').\ \varphi$$

Dans la suite, la simple notation SL désignera la seconde version de Strategy Logic.

Nous donnons à présent une définition formelle de la logique SL :

Var désigne un ensemble de variables.

Définition 12. *La syntaxe de SL est définie par la grammaire suivante :*

$$\mathsf{SL} ::= \quad p \mid \neg\varphi \mid \varphi \vee \psi \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\psi \mid \varphi\,\mathbf{R}\,\psi \mid \langle\!\langle x \rangle\!\rangle\,\varphi \mid [\![x]\!]\,\varphi \mid (A_i, x)\varphi$$

où p décrit AP, A_i les joueurs de Agt et x l'ensemble de variables Var.

Remarque : nous avons rapporté telle quelle la syntaxe qui figure dans [23]. Celle-ci utilise la modalité "release" $\mathbf{R}$, qui est le dual de $\mathbf{U}$. Par habitude, nous utilisons plutôt la modalité $\mathbf{W}$. Cela n'a pas une grande importance, car ces deux modalités sont proches : elles sont liées par les formules suivantes.

$$\varphi\,\mathbf{W}\,\psi = \psi\,\mathbf{R}\,(\psi \vee \varphi)$$
$$\varphi\,\mathbf{R}\,\psi = \psi\,\mathbf{W}\,(\psi \wedge \varphi)$$

Un *assignement* est une fonction partielle

$$\chi \colon \mathsf{Agt} \cup \mathsf{Var} \to \bigcup_{A_i \in \mathsf{Agt}} \mathsf{Strat}^{\infty}_{A_i}$$

Un assignement est dit *complet* si $\mathsf{Agt} \subseteq \mathsf{dom}(\chi)$.

Soit χ un assignement, A_i un joueur, x une variable et F une stratégie du joueur A_i. Alors $\chi[A_i \mapsto F]$ et $\chi[x \mapsto F]$ représentent respectivement les nouveaux assignements définis sur $\mathsf{dom}(\chi) \cup \{A_i\}$ et $\mathsf{dom}(\chi) \cup \{x\}$ qui valent F en A_i et en x, et qui sont égaux à χ sur le reste de leur domaine.

Si λ est une exécution infinie et k est un entier, on notera $\chi^{\lambda,k}$ l'assignement qui vaut $\chi(A_i)^{\lambda,k}$ pour tout joueur A_i de $\mathsf{dom}(\chi)$. La sémantique de SL est définie comme suit : Soient $a \in \mathsf{Agt}, q \in \mathsf{Loc}$ et un assignement χ

$$
\begin{array}{rcl}
(\mathcal{C}, \chi, q) \models p & \text{ssi} & p \in \mathsf{Lab}(q) \\
(\mathcal{C}, \chi, q) \models \neg\varphi & \text{ssi} & (\mathcal{C}, \chi, q) \not\models \varphi \\
(\mathcal{C}, \chi, q) \models \varphi \vee \psi & \text{ssi} & (\mathcal{C}, \chi, q) \models \varphi \text{ ou } (\mathcal{C}, \chi, q) \models \psi \\
(\mathcal{C}, \chi, q) \models \langle\!\langle x \rangle\!\rangle \varphi & \text{ssi} & \exists F \in \bigcup_{A_i \in \mathsf{Agt}} \mathsf{Strat}^{\infty}_{A_i}.\ (\mathcal{C}, \chi[x \mapsto F], q) \models \varphi \\
(\mathcal{C}, \chi, q) \models [\![x]\!] \varphi & \text{ssi} & \forall F \in \bigcup_{A_i \in \mathsf{Agt}} \mathsf{Strat}^{\infty}_{A_i}.\ (\mathcal{C}, \chi[x \mapsto F], q) \models \varphi \\
(\mathcal{C}, \chi, q) \models (A_i, x)\varphi & \text{ssi} & (\mathcal{C}, \chi[A_i \mapsto \chi(x)], q) \models \varphi
\end{array}
$$

Enfin, si l'assignement χ est complet, en notant λ l'outcome de la stratégie globale $\chi(\mathsf{Agt})$ qui part de q :

$$
\begin{array}{rcl}
(\mathcal{C}, \chi, q) \models \varphi & \text{ssi} & (\mathcal{C}, \chi, \lambda, 0) \models \varphi \\
(\mathcal{C}, \chi, \lambda, k) \models \varphi & \text{ssi} & (\mathcal{C}, \chi^{\lambda,k}, \lambda[k]) \models \varphi
\end{array}
$$

Nous introduisons maintenant une notion relative à la syntaxe d'une formule de SL. Si dans une telle formule φ, tous les joueurs se voient attribuer une variable (*via* l'opérateur $(\cdot,\cdot)$) et toutes les variables sont quantifiées (par l'une des modalités $\langle\!\langle \cdot \rangle\!\rangle$ et $[\![\cdot]\!]$), alors cette formule est dite *close*.

Par la suite, si φ est une formule close, alors $(\mathcal{C}, \chi, q) \models \varphi$ pourra être simplement noté $(\mathcal{C}, q) \models \varphi$.

Par exemple, pour dire que les joueurs A_1 et A_2 ont des stratégies (appelées respectivement x et y) qui garantissent qu'un état d'échec n'est jamais atteint, et cela indépendamment du comportement du joueur A_3, on écrit dans SL :

$$\langle\!\langle x \rangle\!\rangle \langle\!\langle y \rangle\!\rangle [\![z]\!] (A_1, x)(A_2, y)(A_3, z)(\mathbf{G}\,\text{échec})$$

De plus, dans le cas où cette formule est évaluée dans une CGS qui a son ensemble de joueurs limité à A_1, A_2 et A_3, alors cette formule est une formule close.

Dans les sections suivantes de ce chapitre, nous définissons les nouvelles logiques que nous avons introduites au cours de notre travail.

2.3 Stratégies à mémoire bornée - Les logiques ATL^*_b

Ici nous allons présenter de nouveaux formalismes pour spécifier la quantité de mémoire que peuvent utiliser les stratégies invoquées par les quantificateurs $\langle\!\langle A \rangle\!\rangle$ de ATL.

b désigne un entier codé en binaire.

Définition 13. *La syntaxe de ATL^*_b est définie par la grammaire suivante :*

$$\begin{array}{rcl} \mathsf{ATL}^*_b \ni \varphi_s, \psi_s & ::= & p \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \langle\!\langle A \rangle\!\rangle_b \, \varphi_p \\ \mathsf{ATL}^*_{b,p} \ni \varphi_p, \psi_p & ::= & \varphi_s \mid \neg\varphi_p \mid \varphi_p \vee \psi_p \mid \mathbf{X}\,\varphi_p \mid \varphi_p \,\mathbf{U}\, \psi_p \end{array}$$

où p décrit l'ensemble AP et A l'ensemble des parties de Agt.

La sémantique de ATL^*_b est définie comme suit :
Soient $A \subseteq \mathsf{Agt}$ et $q \in \mathsf{Loc}$.

$$\begin{array}{rcll} (\mathcal{C}, q) \models p & \text{ssi} & p \in \mathsf{Lab}(q) \\ (\mathcal{C}, q) \models \neg\varphi_s & \text{ssi} & (\mathcal{C}, q) \not\models \varphi_s \\ (\mathcal{C}, q) \models \varphi_s \vee \psi_s & \text{ssi} & (\mathcal{C}, q) \models \varphi_s \text{ ou } (\mathcal{C}, q) \models \psi_s \\ (\mathcal{C}, q) \models \langle\!\langle A \rangle\!\rangle_b \, \varphi_p & \text{ssi} & \exists F \in \mathsf{Strat}^b_A.\ \forall \lambda \in \mathsf{Out}(q, F).\ (\mathcal{C}, \lambda) \models \varphi_p \\ (\mathcal{C}, \lambda) \models \varphi_s & \text{ssi} & (\mathcal{C}, \lambda[0]) \models \varphi_s \\ (\mathcal{C}, \lambda) \models \neg\varphi_p & \text{ssi} & (\mathcal{C}, \lambda) \not\models \varphi_p \\ (\mathcal{C}, \lambda) \models \varphi_p \vee \psi_p & \text{ssi} & (\mathcal{C}, \lambda) \models \varphi_p \text{ ou } (\mathcal{C}, \lambda) \models \psi_p \\ (\mathcal{C}, \lambda) \models \mathbf{X}\,\varphi_p & \text{ssi} & (\mathcal{C}, \lambda[1, \infty]) \models \varphi_p \\ (\mathcal{C}, \lambda) \models \varphi_p \,\mathbf{U}\, \psi_p & \text{ssi} & \exists i \geq 0.\ (\mathcal{C}, \lambda[i, \infty]) \models \psi_p \\ & & \text{et } \forall 0 \leq j < i.\ (\mathcal{C}, \lambda[j, \infty]) \models \varphi_p \end{array}$$

2.4 Emboîter et oublier des stratégies

Dans toutes les logiques introduites jusqu'à présent (sauf SL), les stratégies induites par un quantificateur de stratégie sont utilisées ponctuellement, c'est-à-dire qu'elles ne sont plus prises en compte dès que l'on a affaire à un autre quantificateur de stratégie.

Ainsi, par exemple, si l'on veut exprimer le fait, pour un ascenseur, que le contrôleur dispose d'une stratégie pour que chaque utilisateur dispose toujours d'une stratégie lui permettant d'atteindre chaque étage, on le ferait dans ATL par la formule suivante :

$$\langle\!\langle \text{contrôleur} \rangle\!\rangle \, \mathbf{G} \, (\bigwedge_{i,j} \langle\!\langle \text{utilisateur}_j \rangle\!\rangle \, \mathbf{F} \, \text{étage}_i)$$

Le problème ici est que si l'on se réfère à la sémantique d'ATL, la recherche d'une stratégie pour un utilisateur ne se fait pas dans le contexte restreint

par la stratégie du contrôleur : cette dernière a été oubliée dès la première requantification.

C'est pourquoi nous avons décidé d'introduire une logique qui n'utilise plus ponctuellement des stratégies, mais qui les assimile à des contextes que l'on peut choisir de préserver ou d'oublier tout au long d'une formule. Nous allons dans un premier temps exploiter cette idée en ne considérant que des stratégies sans mémoire.

Pour cela, nous allons définir une opération d'emboîtement de stratégies, notée $\circ$.

Définition 14. *Soient $\mathcal{C}$ une* CGS*, deux coalitions A et B, et deux stratégies F et F' pour A et B respectivement.*

On définit la nouvelle stratégie $F' \circ F$ pour la coalition $A \cup B$, telle que pour tout joueur A_i de $A \cup B$:

$$(F' \circ F)_{|A_i} = \begin{cases} F'_{|A_i} & \text{si } A_i \in B \\ F_{|A_i} & \text{sinon.} \end{cases}$$

La loi $\circ$ n'est pas commutative, mais elle est associative, et l'on pourra donc omettre les parenthèses. Son élément neutre est la stratégie vide de la coalition $\varnothing$.

Remarquons que si F et F' sont de mémoire b, alors $F \circ F'$ est aussi de mémoire b.

Nous allons à présent définir une opération d'oubli de stratégie, qui correspond en fait à une restriction, et que l'on note avec une barre verticale : $|$

Définition 15. *Soient $\mathcal{C}$ une* CGS*, deux coalitions A et C telles que $C \subseteq A$, et une stratégie F pour A.*

On définit la stratégie $F|_C$ de la coalition C, en oubliant les stratégies qui ne concernent pas les joueurs de C. Ainsi, pour tout joueur A_i de C, on pose :

$$(F|_C)_{|A_i} = F_{|A_i}$$

En particulier, quelle que soit la stratégie F, on a toujours $F|_\varnothing = \varnothing$. De plus, si F est de mémoire b, alors $F|_C$ l'est aussi.

2.5 Contextes de stratégies sans mémoire

2.5.1 La logique $\mathsf{ATL}_{sc,0}$

On définit d'abord une logique $\mathsf{ATL}_{sc,0}$ inspirée de ATL, c'est-à-dire où chaque opérateur temporel est immédiatement précédé d'un quantificateur de stratégie. De plus, dans cette logique, on ne s'autorise pas à oublier des stratégies, mais juste à les emboîter par un nouveau quantificateur de stratégie noté $\llangle \cdot \rrangle_0$.

Définition 16. *La syntaxe de $\mathsf{ATL}_{sc,0}$ est définie par la grammaire suivante :*

$$\begin{array}{rcl} \mathsf{ATL}_{sc,0} \ni \varphi_s, \psi_s & ::= & p \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \ll A \gg_0 \varphi_p \\ \mathsf{ATL}_{sc,0,p} \ni \varphi_p & ::= & \varphi_s \mid \mathbf{X}\,\varphi_s \mid \varphi_s\,\mathbf{U}\,\psi_s \mid \varphi_s\,\mathbf{W}\,\psi_s \end{array}$$

où p décrit l'ensemble AP et A l'ensemble des parties de Agt.

La sémantique de $\mathsf{ATL}_{sc,0}$ est définie comme suit :
Soient $A, B \subseteq \mathsf{Agt}$, $q \in \mathsf{Loc}$ et $F \in \mathsf{Strat}^0_A$

$$\begin{array}{rcll} (\mathcal{C}, q) \models_F p & \text{ssi} & p \in \mathsf{Lab}(q) \\ (\mathcal{C}, q) \models_F \neg\varphi_s & \text{ssi} & (\mathcal{C}, q) \not\models_F \varphi_s \\ (\mathcal{C}, q) \models_F \varphi_s \vee \psi_s & \text{ssi} & (\mathcal{C}, q) \models_F \varphi_s \text{ ou } (\mathcal{C}, q) \models_F \psi_s \\ (\mathcal{C}, q) \models_F \ll B \gg_0 \varphi_p & \text{ssi} & \exists F' \in \mathsf{Strat}^0_B.\ \forall \lambda \in \mathsf{Out}(q, F' \circ F). \\ & & (\mathcal{C}, \lambda) \models_{F' \circ F} \varphi_p \\ (\mathcal{C}, \lambda) \models_F \varphi_s & \text{ssi} & (\mathcal{C}, \lambda[0]) \models_F \varphi_s \\ (\mathcal{C}, \lambda) \models_F \mathbf{X}\,\varphi_s & \text{ssi} & (\mathcal{C}, \lambda[1]) \models_F \varphi_s \\ (\mathcal{C}, \lambda) \models_F \varphi_s\,\mathbf{U}\,\psi_s & \text{ssi} & \exists i \geq 0.\ (\mathcal{C}, \lambda[i]) \models_F \psi_s \\ & & \text{et } \forall 0 \leq j < i.\ (\mathcal{C}, \lambda[j]) \models_F \varphi_s \\ (\mathcal{C}, \lambda) \models_F \varphi_s\,\mathbf{W}\,\psi_s & \text{ssi} & (\mathcal{C}, \lambda) \models_F \varphi_s\,\mathbf{U}\,\psi_s \text{ ou } \forall i \geq 0.\ (\mathcal{C}, \lambda[i]) \models_F \psi_s \end{array}$$

Remarques :
– Le symbole $\models_\varnothing$ sera plus simplement noté $\models$
– $(\mathcal{C}, q) \models_F \ll \varnothing \gg_0 \varphi_p$ signifie $\forall \lambda \in \mathsf{Out}(q, F), (\mathcal{C}, \lambda) \models_F \varphi_p$

Imaginons que l'on cherche à formuler la spécification suivante : "Le joueur A dispose d'une stratégie telle que, *sous ce contexte*, les autres joueurs puissent toujours atteindre un état qui vérifie la proposition atomique p".
Nous pouvons exprimer cela dans $\mathsf{ATL}_{sc,0}$, par la formule :

$$\ll A \gg_0 \mathbf{G}\ \ll \mathsf{Agt} \setminus A \gg_0 \mathbf{F}\,p$$

Cela est dû au fait que la prise en compte des seules stratégies positionnelles est reconnue comme étant suffisante pour ce type d'objectifs [3].

Voici maintenant une proposition qui nous sera utile plus tard :

Proposition 2. *Pour toute formule d'état φ_s de $\mathsf{ATL}_{sc,0}$,*

$$(\mathcal{C}, q) \models_F \neg \ll B \gg_0 \neg\varphi_s$$

signifie que pour toute stratégie positionnelle F' de la coalition B,

$$(\mathcal{C}, q) \models_{F' \circ F} \varphi_s$$

2.5.2 La logique $\mathsf{ATL}^*_{sc,0}$

On étend $\mathsf{ATL}_{sc,0}$ en $\mathsf{ATL}^*_{sc,0}$, d'une part en généralisant les formules de chemin, et d'autre part en s'autorisant à oublier des stratégies, ceci par un nouveau quantificateur de stratégie noté $\rangle\cdot\langle\cdot$.

Définition 17. *La syntaxe de $\mathsf{ATL}^*_{sc,0}$ est définie par la grammaire suivante :*

$$\begin{array}{rcl} \mathsf{ATL}^*_{sc,0} \ni \varphi_s, \psi_s & ::= & p \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \ll A \gg_0 \varphi_p \mid \rangle A\langle \varphi_p \\ \mathsf{ATL}^*_{sc,0,p} \ni \varphi_p, \psi_p & ::= & \varphi_s \mid \neg\varphi_p \mid \varphi_p \vee \psi_p \mid \mathbf{X}\,\varphi_p \mid \varphi_p\,\mathbf{U}\,\psi_p \end{array}$$

où p décrit AP et A les sous-ensembles de Agt.

La sémantique de $\mathsf{ATL}^*_{sc,0}$ est définie comme suit :
Soient $A, B \subseteq \mathsf{Agt}, q \in \mathsf{Loc}$ et $F \in \mathsf{Strat}^0_A$.

$$\begin{array}{rcll} (\mathcal{C}, q) \models_F p & \text{ssi} & p \in \mathsf{Lab}(q) \\ (\mathcal{C}, q) \models_F \neg\varphi_s & \text{ssi} & (\mathcal{C}, q) \not\models_F \varphi_s \\ (\mathcal{C}, q) \models_F \varphi_s \vee \psi_s & \text{ssi} & (\mathcal{C}, q) \models_F \varphi_s \text{ ou } (\mathcal{C}, q) \models_F \psi_s \\ (\mathcal{C}, q) \models_F \ll B \gg_0 \varphi_p & \text{ssi} & \exists F' \in \mathsf{Strat}^0_B.\ \forall \lambda \in \mathsf{Out}(q, F' \circ F). \\ & & (\mathcal{C}, \lambda) \models_{F' \circ F} \varphi_p \\ (\mathcal{C}, q) \models_F \rangle B\langle \varphi_p & \text{ssi} & \forall \lambda \in \mathsf{Out}(q, F|_{A \setminus B}).\ (\mathcal{C}, \lambda) \models_{F|_{A\setminus B}} \varphi_p \\ (\mathcal{C}, \lambda) \models_F \varphi_s & \text{ssi} & (\mathcal{C}, \lambda[0]) \models_F \varphi_s \\ (\mathcal{C}, \lambda) \models_F \neg\varphi_p & \text{ssi} & (\mathcal{C}, \lambda) \not\models_F \varphi_p \\ (\mathcal{C}, \lambda) \models_F \varphi_p \vee \psi_p & \text{ssi} & (\mathcal{C}, \lambda) \models_F \varphi_p \text{ ou } \lambda \models_F \psi_p \\ (\mathcal{C}, \lambda) \models_F \mathbf{X}\,\varphi_p & \text{ssi} & (\mathcal{C}, \lambda[1, \infty]) \models_F \varphi_p \\ (\mathcal{C}, \lambda) \models_F \varphi_p\,\mathbf{U}\,\psi_p & \text{ssi} & \exists i \geq 0.\ (\mathcal{C}, \lambda[i, \infty]) \models_F \psi_p \\ & & \text{et } \forall 0 \leq j < i.\ (\mathcal{C}, \lambda[j, \infty]) \models_F \varphi_p \end{array}$$

Nous faisons remarquer la propostition suivante :

$$(\mathcal{C}, q) \models_F \rangle\mathsf{Agt}\langle \varphi_p \Leftrightarrow \forall \lambda \in \mathsf{Exec}(q).\ (\mathcal{C}, \lambda) \models \varphi_p$$

En s'interdisant de recourir à l'opérateur $\rangle\cdot\langle$, on définit la logique $\mathsf{ATL}^*_{sc,0} \setminus \rangle\cdot\langle$.

Définition 18. *Les formules de chemin de $\mathsf{ATL}^*_{sc,0} \setminus \rangle\cdot\langle$ sont définies par la même grammaire que celles de $\mathsf{ATL}^*_{sc,0}$, et les formules d'état sont définies par la grammaire suivante :*

$$\varphi_s, \psi_s \quad ::= \quad p \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \ll A \gg_0 \varphi_p$$

où A décrit les sous-ensembles de Agt, et φ_p les formules de chemin.

2.6 Contextes de stratégies à mémoire infinie

Il s'agit à présent d'étendre les principes d'emboîtement et d'oubli des stratégies au cas où celles-ci sont à mémoire infinie. Pour cela, nous allons simplement introduire les mêmes idées que dans la section précédente, mais en considérant cette fois des stratégies avec mémoire infinie.

Définition 19. *Soit $F \in \mathsf{Strat}^{\infty}_{A_i}$, une exécution ρ et un entier $i \geq 0$. On définit la stratégie $F^{\rho,i}$ qui simule le comportement de F après le préfixe $\rho[0,i]$, en posant :*
$F^{\rho,i}(\lambda) = F(\rho[0,i] \cdot \lambda)$

Nous avons déjà utilisé cette notation $F^{\rho,i}$ à la section 2.1.2, où F désignait alors une stratégie avec mémoire bornée. Dans les deux cas, $F^{\rho,i}$ simule le comportement de F après le préfixe $\rho[0,i]$; c'est pourquoi nous avons choisi d'utiliser une notation commune pour ces deux notions, dont les définitions formelles sont pourtant différentes.

2.6.1 La logique $\mathsf{ATL}_{sc,\infty}$

On définit une logique $\mathsf{ATL}_{sc,\infty}$ où chaque opérateur temporel est immédiatement précédé d'un quantificateur de stratégie. De plus, dans cette logique, on ne s'autorise pas encore à oublier des stratégies, mais juste à les emboîter. L'unique différence avec la logique $\mathsf{ATL}_{sc,0}$ définie précédemment est que les stratégies considérées sont à mémoire infinie.

Définition 20. *La syntaxe de $\mathsf{ATL}_{sc,\infty}$ est définie par la grammaire suivante :*

$$\begin{array}{rcl} \mathsf{ATL}_{sc,\infty} \ni \varphi_s, \psi_s & ::= & p \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \ll A \gg \varphi_p \\ \mathsf{ATL}_{sc,\infty,p} \ni \varphi_p & ::= & \varphi_s \mid \mathbf{X}\,\varphi_s \mid \varphi_s\,\mathbf{U}\,\psi_s \mid \varphi_s\,\mathbf{W}\,\psi_s \end{array}$$

où p décrit l'ensemble AP et A l'ensemble des parties de Agt.

La sémantique de $\mathsf{ATL}_{sc,\infty}$ est définie comme suit :
Soient $A, B \subseteq \mathsf{Agt}$, $q \in \mathsf{Loc}$ et $F \in \mathsf{Strat}^{\infty}_{A}$

$$\begin{array}{rcll} (\mathcal{C},q) \models_F p & \text{ssi} & p \in \mathsf{Lab}(q) \\ (\mathcal{C},q) \models_F \neg\varphi_s & \text{ssi} & (\mathcal{C},q) \not\models_F \varphi_s \\ (\mathcal{C},q) \models_F \varphi_s \vee \psi_s & \text{ssi} & (\mathcal{C},q) \models_F \varphi_s \text{ ou } (\mathcal{C},q) \models_F \psi_s \\ (\mathcal{C},q) \models_F \ll B \gg \varphi_p & \text{ssi} & \exists F' \in \mathsf{Strat}^{\infty}_{B}.\ \forall \lambda \in \mathsf{Out}(q, F' \circ F). \\ & & (\mathcal{C},\lambda) \models_{F' \circ F} \varphi_p \\ (\mathcal{C},\lambda) \models_F \varphi_s & \text{ssi} & (\mathcal{C},\lambda[0]) \models_F \varphi_s \\ (\mathcal{C},\lambda) \models_F \mathbf{X}\,\varphi_s & \text{ssi} & (\mathcal{C},\lambda[1]) \models_{F^{\lambda,1}} \varphi_s \\ (\mathcal{C},\lambda) \models_F \varphi_s\,\mathbf{U}\,\psi_s & \text{ssi} & \exists i \geq 0.\ (\mathcal{C},\lambda[i]) \models_{F^{\lambda,i}} \psi_s \\ & & \text{et } \forall 0 \leq j < i.\ (\mathcal{C},\lambda[j]) \models_{F^{\lambda,j}} \varphi_s \\ (\mathcal{C},\lambda) \models_F \varphi_s\,\mathbf{W}\,\psi_s & \text{ssi} & \lambda \models_F \varphi_s\,\mathbf{U}\,\psi_s \text{ ou } \forall i \geq 0.\ (\mathcal{C},\lambda[i]) \models_{F^{\lambda,i}} \psi_s \end{array}$$

Remarques :

– Les opérateurs $\ll\varnothing\gg_0$ et $\ll\varnothing\gg$ sont équivalents ; en revanche, si $A \neq \varnothing$, les opérateurs $\ll A\gg_0$ sont différents des opérateurs $\ll A\gg$.
– Si φ est une formule de $\mathsf{ATL}_{sc,\infty}$, alors

$$(\mathcal{C}, q) \models_F \neg \ll B \gg \neg\varphi \text{ signifie } \forall F' \in \mathsf{Strat}_B^\infty, (\mathcal{C}, q) \models_{F' \circ F} \varphi$$

Considérons la spécification que l'on avait introduite à titre d'exemple dans le paragraphe 2.2.4 sur GL : "il existe une stratégie pour le joueur A tel qu'un certain comportement de la coalition des autres joueurs entraîne que l'on vérifie toujours la proposition p, et qu'un autre comportement de cette même coalition implique que l'on ait toujours la proposition q."

Nous l'écrivons cette fois par l'intermédiaire de la logique $\mathsf{ATL}_{sc,\infty}$:

$$\varphi = \ll A \gg \big(\ll \overline{A} \gg \mathbf{G}\, p \wedge \ll \overline{A} \gg \mathbf{G}\, q \big)$$

Remarquons que cette formule est différente si on l'écrit avec $\mathsf{ATL}_{sc,0}$, bien que les objectifs considérés ($\mathbf{G}\, p$ et $\mathbf{G}\, q$) paraissent assez simples pour ne pas requérir de mémoire. En effet, la formule $\varphi' = \ll A\gg_0 \big(\ll\overline{A}\gg_0 \mathbf{G}\, p \wedge \ll\overline{A}\gg_0 \mathbf{G}\, q\big)$, obtenue en remplaçant dans φ les quantificateurs $\ll \cdot \gg$ par $\ll \cdot \gg_0$, n'est pas équivalente à φ.

Pour montrer cela, nous donnons avec la Figure 2.2 un modèle qui vérifie φ mais pas φ' (il est par ailleurs évident que tout modèle vérifiant φ' vérifierait aussi φ). Cette CGS contient deux joueurs A et B, et les états rectangulaires (resp. ronds) sont contrôlés par A (resp. B).

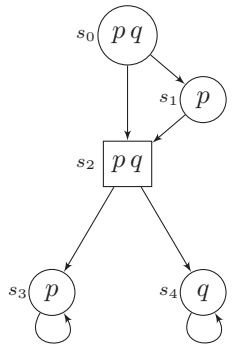

FIG. 2.2 – Une *CGS* qui vérifie φ, mais pas φ'

La stratégie qui consiste pour A à aller en s_3 si on est passé par s_1, et en s_4 sinon, permet au joueur B (c'est-à-dire à la coalition $\overline{A}$) d'obtenir à la fois $\mathbf{G}\, p$ et $\mathbf{G}\, q$ (selon qu'il passe par s_1 ou va directement en s_2 depuis s_0). Ceci montre que φ est vraie ; en revanche, φ' n'est pas vérifiée ici car les deux stratégies sans mémoire du joueur A font boucler *toutes les outcomes* dans s_3, resp. dans s_4, et empêchent ainsi au joueur B d'obtenir $\mathbf{G}\, q$, resp. $\mathbf{G}\, p$.

2.6.2 La logique $\mathsf{ATL}^*_{sc,\infty}$

On étend $\mathsf{ATL}_{sc,\infty}$ en $\mathsf{ATL}^*_{sc,\infty}$ en s'autorisant à oublier des stratégies, cela par l'opération que l'on vient de définir. De plus, on autorise des objectifs de LTL.

Définition 21. *La syntaxe de $\mathsf{ATL}^*_{sc,\infty}$ est définie par la grammaire suivante :*

$$\begin{array}{rcl} \mathsf{ATL}^*_{sc,\infty} \ni \varphi_s, \psi_s & ::= & p \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \ll A \gg \varphi_p \mid \rangle A \langle \varphi_p \\ \mathsf{ATL}^*_{sc,\infty,p} \ni \varphi_p, \psi_p & ::= & \varphi_s \mid \neg\varphi_p \mid \varphi_p \vee \psi_p \mid \mathbf{X}\,\varphi_p \mid \varphi_p \,\mathbf{U}\, \psi_p \end{array}$$

où p décrit AP et A les sous-ensembles de Agt.

La sémantique de $\mathsf{ATL}^*_{sc,\infty}$ est définie comme suit :
Soient $A, B \subseteq \mathsf{Agt}, q \in \mathsf{Loc}$ et $F \in \mathsf{Strat}^\infty_A$,

$$\begin{array}{rcll} (\mathcal{C}, q) \models_F p & \text{ssi} & p \in \mathsf{Lab}(q) \\ (\mathcal{C}, q) \models_F \neg\varphi_s & \text{ssi} & (\mathcal{C}, q) \not\models_F \varphi_s \\ (\mathcal{C}, q) \models_F \varphi_s \vee \psi_s & \text{ssi} & (\mathcal{C}, q) \models_F \varphi_s \text{ ou } (\mathcal{C}, q) \models_F \psi_s \\ (\mathcal{C}, q) \models_F \ll B \gg \varphi_p & \text{ssi} & \exists F' \in \mathsf{Strat}^\infty_B.\ \forall \lambda \in \mathsf{Out}(q, F' \circ F). \\ & & (\mathcal{C}, \lambda) \models_{F' \circ F} \varphi_p \\ (\mathcal{C}, q) \models_F \rangle B \langle \varphi_p & \text{ssi} & \forall \lambda \in \mathsf{Out}(q, F|_{A \setminus B}).\ (\mathcal{C}, \lambda) \models_{F|_{A\setminus B}} \varphi_p \\ (\mathcal{C}, \lambda) \models_F \varphi_s & \text{ssi} & (\mathcal{C}, \lambda[0]) \models_F \varphi_s \\ (\mathcal{C}, \lambda) \models_F \neg\varphi_p & \text{ssi} & (\mathcal{C}, \lambda) \not\models_F \varphi_p \\ (\mathcal{C}, \lambda) \models_F \varphi_p \vee \psi_p & \text{ssi} & (\mathcal{C}, \lambda) \models_F \varphi_p \text{ ou } (\mathcal{C}, \lambda) \models_F \psi_p \\ (\mathcal{C}, \lambda) \models_F \mathbf{X}\,\varphi_p & \text{ssi} & (\mathcal{C}, \lambda[1, \infty]) \models_{F^{\lambda,1}} \varphi_p \\ (\mathcal{C}, \lambda) \models_F \varphi_p \,\mathbf{U}\, \psi_p & \text{ssi} & \exists i \geq 0.\ (\mathcal{C}, \lambda[i, \infty]) \models_{F^{\lambda,i}} \psi_p \\ & & \text{et } \forall 0 \leq j < i.\ (\mathcal{C}, \lambda[j, \infty]) \models_{F^{\lambda,j}} \varphi_p \end{array}$$

On définit à présent la logique $\mathsf{ATL}^*_{sc,\infty} \setminus \rangle \cdot \langle$.

Définition 22. *Les formules d'état de $\mathsf{ATL}^*_{sc,\infty} \setminus \rangle \cdot \langle$ sont construites inductivement sur la grammaire suivante :*

$$\varphi_s, \psi_s \quad ::= \quad p \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \ll A \gg \varphi_p$$

où les formules d'état représentées par φ_p sont les mêmes que dans $\mathsf{ATL}^*_{sc,\infty}$, et A décrit les sous-ensembles de Agt.

La sémantique de $\mathsf{ATL}^*_{sc,\infty} \setminus \rangle \cdot \langle$ est héritée celle de $\mathsf{ATL}^*_{sc,\infty}$.

2.7 Contextes de stratégies à mémoire bornée

Nous allons définir des nouvelles logiques sur le modèle des précédentes, en sauvegardant toujours les principes d'emboîtement et d'oubli de stratégies, cette fois en s'appuyant sur notre définition de stratégies à mémoire bornée.

Nous rappelons la définition suivante, que l'on avait introduite dans la section 2.1.2 :

Définition 23. *Soit $F = (F^{\mathsf{mov}}, F^{\mathsf{cell}}, c)$ une stratégie à mémoire bornée pour une coalition A quelconque, et un état q. Pour toute exécution infinie ρ et tout entier i, on définit :*

$$F^{\rho,i}(q) = F^{\mathsf{mov}}(c', q),$$

où c' est le $i+1$-ème terme de la suite finie d'entiers définie récursivement par :

- *$c_0 = c$*
- *pour tout $j \leq i$,*

$$c_{j+1} = F^{\mathsf{cell}}(c_j, \rho[j])$$

2.7.1 La logique $\mathsf{ATL}^*_{sc,b}$

Soit $b \in \mathbb{N} \cup \{\infty\}$. On définit la logique $\mathsf{ATL}^*_{sc,b}$.

Définition 24. *La syntaxe de $\mathsf{ATL}^*_{sc,b}$ est définie par la grammaire suivante :*

$$\begin{array}{rcl} \mathsf{ATL}^*_{sc,b} \ni \varphi_s, \psi_s & ::= & p \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \ll A \gg_b \varphi_p \mid \rangle\!\rangle A \langle\!\langle \varphi_p \\ \mathsf{ATL}^*_{sc,b,p} \ni \varphi_p, \psi_p & ::= & \varphi_s \mid \neg\varphi_p \mid \varphi_p \vee \psi_p \mid \mathbf{X}\,\varphi_p \mid \varphi_p\,\mathbf{U}\,\psi_p \end{array}$$

où p décrit AP et A les sous-ensembles de Agt.

La sémantique de $\mathsf{ATL}^*_{sc,b}$ est définie comme suit :

Soient $A, B \subseteq \mathsf{Agt}, q \in \mathsf{Loc}$ et $F \in \mathsf{Strat}^{\prime b}_A$,

$$\begin{array}{rcll} (\mathcal{C}, q) \models_F p & \text{ssi} & p \in \mathsf{Lab}(q) \\ (\mathcal{C}, q) \models_F \neg\varphi_s & \text{ssi} & (\mathcal{C}, q) \not\models_F \varphi_s \\ (\mathcal{C}, q) \models_F \varphi_s \vee \psi_s & \text{ssi} & (\mathcal{C}, q) \models_F \varphi_s \text{ ou } (\mathcal{C}, q) \models_F \psi_s \\ (\mathcal{C}, q) \models_F \ll B \gg_b \varphi_p & \text{ssi} & \exists F' \in \mathsf{Strat}^{\prime b}_B.\ \forall \lambda \in \mathsf{Out}(q, F' \circ F). \\ & & (\mathcal{C}, \lambda) \models_{F' \circ F} \varphi_p \\ (\mathcal{C}, q) \models_F \rangle\!\rangle B \langle\!\langle \varphi_p & \text{ssi} & \forall \lambda \in \mathsf{Out}(q, F|_{A \setminus B}).\ (\mathcal{C}, \lambda) \models_{F|_{A\setminus B}} \varphi_p \\ (\mathcal{C}, \lambda) \models_F \varphi_s & \text{ssi} & (\mathcal{C}, \lambda[0]) \models_F \varphi_s \\ (\mathcal{C}, \lambda) \models_F \neg\varphi_p & \text{ssi} & (\mathcal{C}, \lambda) \not\models_F \varphi_p \\ (\mathcal{C}, \lambda) \models_F \varphi_p \vee \psi_p & \text{ssi} & (\mathcal{C}, \lambda) \models_F \varphi_p \text{ ou } (\mathcal{C}, \lambda) \models_F \psi_p \\ (\mathcal{C}, \lambda) \models_F \mathbf{X}\,\varphi_p & \text{ssi} & (\mathcal{C}, \lambda[1, \infty]) \models_{F^{\lambda,1}} \varphi_p \\ (\mathcal{C}, \lambda) \models_F \varphi_p\,\mathbf{U}\,\psi_p & \text{ssi} & \exists i \geq 0.\ (\mathcal{C}, \lambda[i, \infty]) \models_{F^{\lambda,i}} \psi_p \\ & & \text{et } \forall 0 \leq j < i.\ (\mathcal{C}, \lambda[j, \infty]) \models_{F^{\lambda,j}} \varphi_p \end{array}$$

Le symbole $\models_{\varnothing}$ sera plus simplement noté $\models$

Cette syntaxe peut s'avérer utile pour exprimer des spécifications comme : "Dans un ascenseur, il existe une stratégie pour le contrôleur, de taille 1000, telle que l'utilisateur puisse atteindre tous les étages, au moyen d'une stratégie (différente selon l'étage) de mémoire 5."

$$\ll \mathit{contrôleur} \gg_{1000} \mathbf{G} \left(\bigwedge_i \ll \mathit{utilisateur} \gg_5 \mathbf{F}\ \mathit{étage}_i \right)$$

Cette différence de taille de mémoire est bien sûr justifiée par le fait que l'utilisateur est supposé être un humain, et le contrôleur une machine.

On pourrait naturellement définir une logique englobant toutes les autres, qui contiendrait tous les quantificateurs de stratégie $\ll \cdot \gg_b$, pour tout entier $b \in \mathbb{N}$, ainsi que le quantificateur $\ll \cdot \gg$.

Les introductions de ces logiques soulèvent de nouvelles questions, se rapportant en particulier à leur problème de model-checking, aux limites de leur expressivité et à la comparaison sur ces plans avec d'autres formalismes, que nous étudierons dans la suite.

Chapitre 3

Notions d'équivalences de structures

Ce chapitre est consacré à quatre notions d'équivalences comportementales, les trois premières étant bien connues et présentées en détails, et dont la dernière est une nouveauté. Leurs définitions sont données par ordre croissant de complexité.

Nous aurions souhaité caractériser l'équivalence comportementale associée aux contextes stratégiques avec mémoire infinie, mais nous n'avons pas résolu cette question. La notion de GL-équivalence apporte une solution très partielle à ce problème, dans la mesure où les arbres d'exécution associés à des stratégies (que l'on quantifie explicitement dans GL) s'apparentent en fait à des contextes stratégiques, dans lesquels on peut ensuite spécifier certaines formules, qui sont simplement celles de $\mathsf{CTL}^\star$. Cependant, nous verrons dans le chapitre suivant que si cette nouvelle relation s'approche plus du but que la bisimulation alternante (cf. remarque à la suite du corollaire 1), elle ne l'atteint pas (voir la section 4.5.1).

Les deux premières sections s'appuient sur [6].

3.1 Trace-équivalence

Définition 25. *Soit $\lambda = s_0 s_1 s_2 \ldots$ une exécution d'une structure de Kripke. On définit la* trace *de λ, que l'on note $\mathsf{trac}(\lambda)$, comme étant la suite de 2^{AP} : $Lab(s_0)Lab(s_1)Lab(s_2)\ldots$.*

On remarque que pour tout entier $i \geq 0$, $\mathsf{trac}(\lambda[i,\infty]) = \mathsf{trac}(\lambda)[i,\infty]$. Etant donné un état s de $\mathcal{K}$, on notera

$$\mathsf{Trac}(s) = \{\mathsf{trac}(\lambda) \mid \ \lambda \in \mathsf{Exec}(s)\}$$

Nous donnons une définition inductive de $[\![\varphi]\!]$, où φ est une formule de LTL.

Définition 26. *Nous définissons le sous-ensemble* $[\![\varphi]\!] \subseteq (2^{AP})^\omega$. π *représente ici une suite infinie d'éléments de* 2^{AP}.

- *Si* $\varphi = a \in AP$, *alors* $[\![\varphi]\!] = \{\pi \mid a \in \pi[0]\}$
- *Si* $\varphi = \neg\psi$, *alors* $[\![\varphi]\!] = (2^{AP})^\omega \setminus [\![\psi]\!]$
- *Si* $\varphi = \psi_1 \vee \psi_2$, *alors* $[\![\varphi]\!] = [\![\varphi_1]\!] \cup [\![\varphi_2]\!]$
- *Si* $\varphi = \mathbf{X}\,\psi$, *alors* $[\![\varphi]\!] = \{\pi \mid \pi[1,\infty] \in [\![\psi]\!]\}$
- *Si* $\varphi = \psi_1\,\mathbf{U}\,\psi_2$, *alors* $[\![\varphi]\!] = \{\pi \mid \exists i \geq 0, \pi[i,\infty] \in [\![\varphi_2]\!] et\ \forall 0 \leq j < i.\ \pi[j,\infty] \in [\![\varphi_1]\!]\}$

Proposition 3. $(\mathcal{K},\lambda) \models \varphi$ *ssi* $\mathsf{trac}_\mathcal{K}(\lambda) \in [\![\varphi]\!]$. *Nous avons comme corollaire :* $(\mathcal{K},s) \models \varphi$ *ssi* $\mathsf{Trac}_\mathcal{K}(s) \subseteq [\![\varphi]\!]$

Démonstration. Nous prouvons cela par une induction structurelle sur les formules de LTL. Pour le cas de base, on considère $\varphi = a$, où a est une proposition atomique de AP. Alors $\mathsf{trac}_\mathcal{K}(\lambda) \in [\![\varphi]\!]$ signifie que $a \in \mathsf{Lab}(\lambda[0])$, donc que $(\mathcal{K},\lambda) \models \varphi$.

1. $\varphi = \neg\psi$:

$$\begin{array}{lcl} (\mathcal{K},\lambda) \models \varphi & \Leftrightarrow & (\mathcal{K},\lambda) \not\models \psi \\ & \Leftrightarrow & \mathsf{trac}_\mathcal{K}(\lambda) \notin [\![\psi]\!] \text{ (hypothèse d'induction)} \\ & \Leftrightarrow & \mathsf{trac}_\mathcal{K}(\lambda) \in [\![\varphi]\!] \end{array}$$

2. $\varphi = \psi_1 \vee \psi_2$:

$$\begin{array}{lcl} (\mathcal{K},\lambda) \models \varphi & \Leftrightarrow & (\mathcal{K},s) \models \psi_1 \text{ ou } (\mathcal{K},s) \models \psi_2 \\ & \Leftrightarrow & \mathsf{trac}_\mathcal{K}(\lambda) \in [\![\psi_1]\!] \text{ ou } \mathsf{trac}_\mathcal{K}(\lambda) \in [\![\psi_2]\!] \text{ (HI)} \\ & \Leftrightarrow & \mathsf{trac}_\mathcal{K}(\lambda) \in [\![\varphi]\!] \end{array}$$

3. $\varphi = \mathbf{X}\,\psi$:

$$\begin{array}{lcl} (\mathcal{K},\lambda) \models \varphi & \Leftrightarrow & (\mathcal{K},\lambda[1,\infty]) \models \psi \\ & \Leftrightarrow & \mathsf{trac}_\mathcal{K}(\lambda[1,\infty]) \in [\![\psi]\!] \text{ (hypothèse d'induction)} \\ & \Leftrightarrow & \mathsf{trac}_\mathcal{K}(\lambda)[1,\infty] \in [\![\psi]\!] \text{ (remarque)} \\ & \Leftrightarrow & \mathsf{trac}_\mathcal{K}(\lambda) \in [\![\varphi]\!] \end{array}$$

4. $\varphi = \psi_1\,\mathbf{U}\,\psi_2$: Supposons que $(\mathcal{K},\lambda) \models \varphi$. Alors, par définition du $\mathbf{U}$, il existe $i \geq 0$ tel que $(\mathcal{K},\lambda[i,\infty]) \models \psi_2$ et pour tout $0 \leq j < i$, $(\mathcal{K},\lambda[j,\infty]) \models \psi_1$. Par hypothèse d'induction, $\mathsf{trac}_\mathcal{K}(\lambda[i,\infty]) \in [\![\psi_2]\!]$ et pour tout $0 \leq j < i$, $\mathsf{trac}_\mathcal{K}(\lambda[j,\infty]) \in [\![\psi_1]\!]$. Par la remarque, cela signifie que $\mathsf{trac}_\mathcal{K}(\lambda)[i,\infty] \in [\![\psi_2]\!]$ et pour tout $0 \leq j < i$, $\mathsf{trac}_\mathcal{K}(\lambda)[j,\infty] \in [\![\psi_1]\!]$. C'est-à-dire que $\mathsf{trac}_\mathcal{K}(\lambda) \in [\![\varphi]\!]$.

□

Voici une définition formelle de l'équivalence de trace.

Définition 27. *On dit de deux structures de Kripke* $\mathcal{K} = (AP, S, s_0, R, Lab)$ *et* $\mathcal{K}' = (AP, S', s'_0, R', Lab')$*, qu'elles sont* trace-équivalentes *ssi* $\mathsf{Trac}_{\mathcal{K}}(s_0) = \mathsf{Trac}_{\mathcal{K}'}(s'_0)$.

Nous présentons ci-dessous deux structures trace-équivalentes.

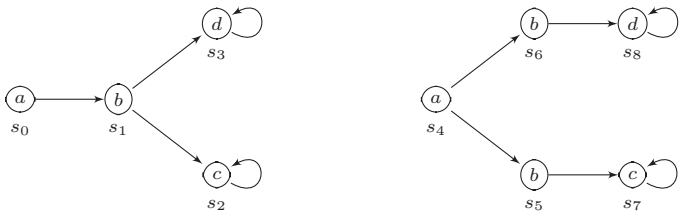

FIG. 3.1 – Deux structures trace-équivalentes

Les ensembles de traces sont les mêmes depuis s_0 et s_4.

Théorème 1. *Deux structures de Kripke* $\mathcal{K}$ *et* $\mathcal{K}'$ *sont trace-équivalentes si et seulement si elles satisfont les mêmes formules de* LTL*, c'est-à-dire :*
pour toute formule φ *de* LTL*,* $(\mathcal{K}, s_0) \models \varphi$ *ssi* $(\mathcal{K}', s'_0) \models \varphi$.

Ce théorème est un corollaire immédiat du résultat suivant :

Lemme 1. *Soit deux structures de Kripke* $\mathcal{K}$ *et* $\mathcal{K}'$*. Les deux énoncés suivants sont équivalents :*

1. $\mathsf{Trac}_{\mathcal{K}}(s_0) \subseteq \mathsf{Trac}_{\mathcal{K}'}(s'_0)$
2. *pour toute formule* φ *de* LTL*,* $(\mathcal{K}', s'_0) \models \varphi$ *implique* $(\mathcal{K}, s_0) \models \varphi$

Démonstration. Nous traitons d'abord le sens direct de l'équivalence. Supposons que $\mathsf{Trac}_{\mathcal{K}}(s_0) \subseteq \mathsf{Trac}_{\mathcal{K}'}(s'_0)$, et soit φ une formule de LTL telle que $(\mathcal{K}', s'_0) \models \varphi$. Par définition de la sémantique de LTL, ça signifie que toutes les exécutions de la structure $\mathcal{K}'$ qui partent de s'_0 vérifient la formule φ. D'après la proposition 3, cela implique $\mathsf{Trac}_{\mathcal{K}'}(s'_0) \subseteq [\![\varphi]\!]$. Puisque $\mathsf{Trac}_{\mathcal{K}}(s_0) \subseteq \mathsf{Trac}_{\mathcal{K}'}(s'_0)$, on en déduit $\mathsf{Trac}_{\mathcal{K}}(s_0) \subseteq [\![\varphi]\!]$, ce qui prouve $(\mathcal{K}, s_0) \models \varphi$ encore d'après la proposition 3.

Passons au sens réciproque. On va raisonner par contraposée : supposons que $\mathsf{Trac}_{\mathcal{K}}(s_0) \not\subseteq \mathsf{Trac}_{\mathcal{K}'}(s'_0)$. Soit alors $\pi = a_0 a_1 a_2 ...$ une trace de $\mathsf{Trac}_{\mathcal{K}}(s_0)$ qui n'est pas dans $\mathsf{Trac}_{\mathcal{K}'}(s'_0)$. Soit n un entier tel que $a_0 a_1 a_2 ... a_n$ ne soit le préfixe d'aucune trace de $\mathsf{Trac}_{\mathcal{K}'}(s'_0)$. Notons φ la formule $\neg \bigwedge_{i=0}^{n} \mathbf{X}^i a_i$. On a alors $(\mathcal{K}', s'_0) \models \varphi$ et $(\mathcal{K}, s_0) \models \neg\varphi$, ce qui contredit l'hypothèse 2. □

3.2 Bisimulation

Définition 28. *Etant données deux structures de Kripke* $\mathcal{K} = (AP, S, s_0, R, Lab)$ *et* $\mathcal{K}' = (AP, S', s'_0, R', Lab')$*, une relation binaire non vide* $Z \subseteq S \times S'$ *est une*

bisimulation *entre* $\mathcal{K}$ *et* $\mathcal{K}'$ *ssi pour tous* $s \in S$ *et* $s' \in S'$, $Z(s, s')$ *implique les trois conditions suivantes :*

- $\mathsf{Lab}(s) = \mathsf{Lab}'(s')$;
- *pour tout* $t \in S$ *tel que* $R(s, t)$, *il existe* $t' \in S'$ *tel que* $R'(s', t')$ *et* $Z(t, t')$;
- *pour tout* $t' \in S'$ *tel que* $R(s', t')$, *il existe* $t \in S$ *tel que* $R(s, t)$ *et* $Z(t, t')$;

S'il existe une bisimulation Z entre $\mathcal{K}$ et $\mathcal{K}'$ telle que $(s_0, s'_0) \in Z$, alors on écrit $\mathcal{K} \sim \mathcal{K}'$, et on dit que les structures $\mathcal{K}$ et $\mathcal{K}'$ sont *bisimilaires*.

Nous présentons dans la Figure 3.2 ci-dessous deux structures bisimilaires. Elles sont liées par la bisimulation Z suivante :

$$Z = \{(s_0, s_4), (s_1, s_5), (s_1, s_6), (s_2, s_7), (s_2, s_8), (s_3, s_9), (s_3, s_{10})\}$$

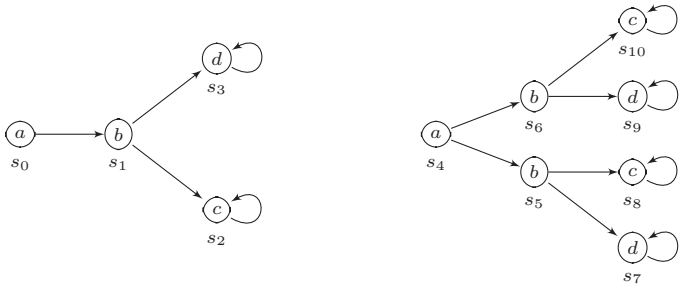

FIG. 3.2 – Deux structures bisimilaires

Définition 29. *Une formule* Φ *sur des structures de Kripke est* invariante par bisimulation *si la condition suivante est vraie :*

pour toutes structures $\mathcal{K}$ *et* $\mathcal{K}'$ *telles que* $\mathcal{K} \sim \mathcal{K}'$ *:*

$$(\mathcal{K}, s_0) \models \Phi \Leftrightarrow (\mathcal{K}', s'_0) \models \Phi$$

Définition 30. *Soit* Z *une relation entre les états de* $\mathcal{K}$ *et de* $\mathcal{K}'$.

1. *Deux exécutions* finies $\lambda = s_0 s_1 s_2 ... s_n$ *dans* $\mathcal{K}$, *et* $\lambda' = s'_0 s'_1 s'_2 ... s'_{n'}$ *dans* $\mathcal{K}'$, correspondent par Z *l'une à l'autre (ou* Z*-correspondent) ssi* $n = n'$, *et pour tout* $0 \leq i \leq n$, $Z(s_i, s'_i)$.
2. *Deux exécutions* infinies $\lambda = s_0 s_1 s_2 ...$ *dans* $\mathcal{K}$ *et* $\lambda' = s'_0 s'_1 s'_2 ...$ *dans* $\mathcal{K}'$ correspondent par Z *l'une à l'autre (ou* Z-correspondent*) ssi pour tout* $i \geq 0$, $Z(s_i, s'_i)$.

Propriété de la bisimulation :

Théorème 2. *Toutes les formules de* CTL^* *sont invariantes par bisimulation.*

On prouve cela au moyen de deux lemmes.

Lemme 2. *Soient deux structures de Kripke $\mathcal{K}$ et $\mathcal{K}'$ telles que $\mathcal{K} \sim \mathcal{K}'$. Appelons Z une bisimulation qui les lie. Alors toute exécution dans $\mathcal{K}$ qui part de s_0 a une exécution Z-correspondante dans $\mathcal{K}'$ qui part de s'_0. Réciproquement, toute exécution dans $\mathcal{K}'$ qui part de s'_0 a une exécution Z-correspondante dans $\mathcal{K}$ qui part de s_0.*

Démonstration. Nous commençons par montrer ce résultat sur des exécutions finies. Soit $\lambda = s_0 s_1 ...$ une exécution finie dans $\mathcal{K}$ qui part de s_0. Par induction sur la longueur de λ, nous contruisons une exécution correspondante $\lambda' = s'_0 s'_1 ...$ dans $\mathcal{K}'$ qui part de s'_0. Pour le cas initial $|\lambda| = 1$, on choisit bien sûr $\lambda' = s'$; les deux exécutions correspondent bien puisque $(s_0, s'_0) \in Z$.

Pour prouver l'hérédité, soit maintenant une exécution λ de longueur $n+1$. Considérons le préfixe $\lambda[0, n-1]$. D'après l'hypothèse d'induction, il existe une exécution ρ' qui Z-correspond à ce préfixe, et qui a la même longueur. Pour établir notre résultat, il reste à choisir un s'_n qui complète ρ' en une exécution λ' Z-correspondant à λ. L'hypothèse d'induction garantit $Z(s_{n-1}, s'_{n-1})$, de plus on a $R(s_{n-1}, s_n)$. Par définition de la bisimulation, il existe donc un état s'_n tel que $R(s'_{n-1}, s'_n)$ et $Z(s_n, s'_n)$. L'exécution $\lambda' = \rho' s'_n$, de longueur n, correspond alors bien à λ.

Soit λ une exécution de longueur *infinie*. En répétant une infinité de fois cette étape inductive, nous pouvons construire une exécution infinie λ' telle que λ et λ' Z-correspondent l'une à l'autre, comme annoncé.

La partie réciproque de cette preuve, où l'on construit λ à partir de λ', est analogue. □

Lemme 3. *Etant donnée une bisimulation Z entre deux structures $\mathcal{K}$ et $\mathcal{K}'$, soient $(s, s') \in Z$, et λ et λ' deux exécutions correspondantes qui partent respectivement de s et s'. Alors :*

- *Pour toute formule d'état φ de CTL^*, $(\mathcal{K}, s) \models \varphi$ ssi $(\mathcal{K}', s') \models \varphi$.*
- *Pour toute formule de chemin φ de CTL^*_p, $(\mathcal{K}, \lambda) \models \varphi$ ssi $(\mathcal{K}', \lambda') \models \varphi$.*

Démonstration. Nous allons prouver ce résultat par induction sur la structure des formules de CTL^*. Pour le cas de base, on se donne $\varphi = a$, où a est une proposition atomique de AP. Puisque $Z(s, s')$, on sait que $\mathsf{Lab}(s) = \mathsf{Lab}'(s')$, par définition de la bisimulation. Par conséquent, $(\mathcal{K}, s) \models a$ ssi $(\mathcal{K}', s') \models a$. Pour les différentes étapes de l'induction, considérons les cas suivants. Dans les 4 premiers cas, φ est une formule d'état, dans la suite c'est une formule de chemin.

1. $\varphi = \neg\psi$:

$$\begin{aligned} (\mathcal{K}, s) \models \varphi \quad &\Leftrightarrow \quad (\mathcal{K}, s) \not\models \psi \\ &\Leftrightarrow \quad (\mathcal{K}', s') \not\models \psi \text{ (hypothèse d'induction)} \\ &\Leftrightarrow \quad (\mathcal{K}', s') \models \varphi \end{aligned}$$

2. $\varphi = \psi_1 \wedge \psi_2$:

$$\begin{array}{lll} (\mathcal{K}, s) \models \varphi & \Leftrightarrow & (\mathcal{K}, s) \models \psi_1 \text{ et } (\mathcal{K}, s) \models \psi_2 \\ & \Leftrightarrow & (\mathcal{K}', s') \models \psi_1 \text{ et } (\mathcal{K}', s') \models \psi_2 \text{ (HI)} \\ & \Leftrightarrow & (\mathcal{K}', s') \models \varphi \end{array}$$

3. $\varphi = \mathbf{E}\psi$:
   Supposons que $(\mathcal{K}, s) \models \mathbf{E}\psi$. Alors il existe une exécution λ, qui part de s et qui satisfait la formule de chemin ψ. D'après le lemme précédent, il existe une exécution λ' partant de s' qui lui correspond. Par hypothèse d'induction, $(\mathcal{K}, \lambda) \models \psi \Leftrightarrow (\mathcal{K}', \lambda') \models \psi$. Par conséquent, ψ est également vraie le long de λ', et donc $(\mathcal{K}', s') \models \mathbf{E}\psi$, comme voulu. La preuve de l'implication $(\mathcal{K}', s') \models \varphi \Rightarrow (\mathcal{K}, s) \models \varphi$ est similaire.
4. $\varphi = \psi_s$ (où ψ_s est une formule d'état) :

$$\begin{array}{lll} (\mathcal{K}, \lambda) \models \varphi & \Leftrightarrow & (\mathcal{K}, s) \models \psi_s \\ & \Leftrightarrow & (\mathcal{K}', s') \models \psi_s \text{ (hypothèse d'induction)} \\ & \Leftrightarrow & (\mathcal{K}', \lambda') \models \varphi \end{array}$$

5. $\varphi = \neg\psi$: comme dans le cas 1.
6. $\varphi = \psi_1 \wedge \psi_2$: comme dans le cas 2.
7. $\varphi = \mathbf{X}\,\psi$: Supposons que $(\mathcal{K}, \lambda) \models \mathbf{X}\,\psi$. Par définition de l'opérateur temporel $\mathbf{X}$, $(\mathcal{K}, \lambda[1, \infty]) \models \psi$. Puisque λ et λ' correspondent l'un à l'autre, il est évident que leurs suffixes $\lambda[1, \infty]$ et $\lambda'[1, \infty]$ correspondent aussi. D'après l'hypothèse d'induction, $(\mathcal{K}, \lambda[1, \infty]) \models \psi \Leftrightarrow (\mathcal{K}', \lambda'[1, \infty]) \models \psi$. Ainsi, $(\mathcal{K}', \lambda'[1, \infty]) \models \psi$, et il s'ensuit que $(\mathcal{K}', \lambda') \models \varphi$. La preuve de l'autre sens est analogue.
8. $\varphi = \psi_1\,\mathbf{U}\,\psi_2$: Supposons que $(\mathcal{K}, \lambda) \models \psi_1\,\mathbf{U}\,\psi_2$. Par définition de $\mathbf{U}$, il existe $i \geq 0$ tel que $(\mathcal{K}, \lambda[i, \infty]) \models \psi_1$ et pour tout $0 \leq j < i$, $(\mathcal{K}, \lambda[j, \infty]) \models \psi_2$. Comme λ et λ' sont deux exécutions correspondantes, leurs suffixes $\lambda[k, \infty]$ et $\lambda'[k, \infty]$, pour tout $0 \leq k \leq i$, le sont aussi. D'après l'hypothèse d'induction, $(\mathcal{K}, \lambda[k, \infty]) \models \psi_1 \Leftrightarrow (\mathcal{K}', \lambda'[k, \infty]) \models \psi_1$ lorsque $k < i$, et $(\mathcal{K}, \lambda[k, \infty]) \models \psi_2 \Leftrightarrow (\mathcal{K}', \lambda'[k, \infty]) \models \psi_2$ lorsque $k = i$. On en déduit que pour tout $0 \leq j < i$, $(\mathcal{K}', \lambda'[j, \infty]) \models \psi_1$, et $(\mathcal{K}', \lambda'[i, \infty]) \models \psi_2$. Donc $(\mathcal{K}', \lambda') \models \varphi$. La preuve de l'autre sens est analogue.

□

La preuve du théorème 1 est une conséquence immédiate de la définition d'invariance par bisimulation et des lemmes précédents.

Théorème 3. *Deux états qui vérifient les mêmes formules de CTL sont bisimilaires.*

Démonstration. Soit $\mathcal{K}$ et $\mathcal{K}'$ deux structures de Kripke, et s et s' deux états qui vérifient les mêmes formules de CTL, c'est-à-dire : $\forall \varphi \in \mathsf{CTL}, (\mathcal{K}, s) \models \varphi$ ssi $(\mathcal{K}', s') \models \varphi$. On va montrer que la relation binaire "vérifier les mêmes formules de CTL " est elle-même une bisimulation. Soit Z cette relation binaire. Deux états liés par Z, disons s et s', sont étiquetés de la même façon, puisque l'on a, pour n'importe quelle étiquette $a \in \mathsf{AP}$, $(\mathcal{K}, s) \models a$ ssi $(\mathcal{K}', s') \models a$. Nous allons prouver la deuxième condition par contraposition. Soit donc un successeur t de s tel qu'il n'existe aucun successeur t'_i de s' qui vérifie $Z(t, t'_i)$. Par définition de Z, pour chaque i, on peut donc trouver une formule φ_i de CTL telle que : $t \models \varphi_i$ et $t'_i \not\models \varphi_i$, quitte à remplacer φ_i par sa négation. Il n'y a qu'un nombre fini de telles formules φ_i, car le nombre de successeurs de s' est lui-même-fini (c'est une conséquence du fait que nos structures ont toujours un nombre fini d'états). Soit maintenant φ la formule $\mathbf{EX}\,(\varphi_1 \wedge \varphi_2 \wedge ... \wedge \varphi_k)$; φ est une formule de CTL. On a $s \models \varphi$ et $t \not\models \varphi$, donc $(s, t) \notin Z$. On peut prouver la troisième condition par le même raisonnement. □

3.3 Bisimulation alternante

Cette notion fut introduite dans [4].

Définition 31. *Soient deux CGS* $\mathcal{C} = (\mathsf{Agt}, \mathsf{Loc}, \mathsf{AP}, \mathcal{M}, \mathsf{Lab}, \mathsf{Mov}, \mathsf{Edg})$ *et* $\mathcal{C}' = (\mathsf{Agt}, \mathsf{Loc}', \mathsf{AP}, \mathcal{M}', \mathsf{Lab}', \mathsf{Mov}', \mathsf{Edg}')$. *Pour toute coalition* $A \subseteq \mathsf{Agt}$, *une relation binaire non vide* $Z \subseteq \mathsf{Loc} \times \mathsf{Loc}'$ *est une* A-bisimulation *entre* $\mathcal{C}$ *et* $\mathcal{C}'$ *ssi pour tous états* $s \in \mathsf{Loc}$ *et* $s' \in \mathsf{Loc}'$, $Z(s, s')$ *implique les trois conditions suivantes :*

- $\mathsf{Lab}(s) = \mathsf{Lab}'(s')$;
- *pour tout vecteur de mouvements* $m \in \mathsf{Mov}(s, A)$, *il existe un vecteur de mouvements* $m' \in \mathsf{Mov}(s', A)$ *tel que quel que soit l'état* t' *de* $\mathsf{Next}(s', A, m')$, *il existe un état* $t \in \mathsf{Next}(s, A, m)$ *qui vérifie* $Z(t, t')$;
- *pour tout vecteur de mouvements* $m' \in \mathsf{Mov}(s', A)$, *il existe un vecteur de mouvements* $m \in \mathsf{Mov}(s, A)$ *tel que quel que soit l'état* t *de* $\mathsf{Next}(s, A, m)$, *il existe un état* $t' \in \mathsf{Next}(s', A, m')$ *qui vérifie* $Z(t, t')$;

Si il existe une A-bisimulation Z entre deux états s et s' qui appartiennent respectivement aux CGS $\mathcal{C}$ et $\mathcal{C}'$, alors on écrit $(\mathcal{C}, s) \equiv_A (\mathcal{C}', s')$, et on dit que les états s et s' sont *A-bisimilaires*. De plus, si λ et λ' sont deux exécutions (finies ou non) telles que pour tout entier i, $Z(\lambda[i], \lambda'[i])$, on notera $(\mathcal{C}, \lambda) \equiv_A (\mathcal{C}', \lambda')$.

Remarquons que dans les cas $A = \varnothing$ et $A = \mathsf{Agt}$, la notion de A-bisimulation recoupe celle de la simple bisimulation.

Intuitivement, dire de deux états s et s' qu'ils sont A-bisimilaires signifie qu'à tout mouvement m de la coalition A depuis s correspond un mouvement m' de cette même coalition depuis s', dont tous les successeurs sont A-simulés par un successeur de m ; de plus, à tout mouvement m' depuis s' correspond un mouvement m depuis s dont tous les successeurs sont A-simulés par un successeur de m'.

Cette intuition est à l'origine d'une interprétation en termes de jeu de la notion de bisimulation alternante. Considérons un jeu à deux joueurs, dont les positions possibles sont les paires d'états $(q, q') \in \mathsf{Loc} \times \mathsf{Loc}'$. La position de départ est (s, s'). Le jeu oppose un antagoniste à un protagoniste, et consiste en une suite de tours. Chaque tour se décompose en quatre étapes, et se déroule de la manière suivante. On suppose que la position courante est (q, q').

1. L'antagoniste choisit l'état q ou q', puis choisit un mouvement m_1 pour la coalition A depuis cet état ;
2. Le protagoniste choisit un mouvement m_2 pour A depuis l'autre état (celui que l'antagoniste n'a pas choisi) ;
3. L'antagoniste choisit un état q_2 parmi les successeurs de cet état, compatible avec le mouvement m_2 ;
4. Le protagoniste choisit un état q_1 parmi les successeurs de l'état que l'antagoniste avait choisit au début du tour, compatible avec m_1, et tel que q_1 soit étiqueté comme q_2.

Si l'on atteint une position depuis laquelle le protagoniste est incapable de désigner un état q_1 convenable, il est déclaré perdant. L'autre possibilité est que le jeu se poursuive infiniment ; dans ce cas le protagoniste gagne. Le protagoniste gagne si $(\mathcal{C}, s) \equiv_A (\mathcal{C}', s')$, et perd sinon.

Lemme 4. *Soient deux CGS $\mathcal{C}$ et $\mathcal{C}'$, et $A \subseteq \mathsf{Agt}$ une coalition. Si Z est une A-bisimulation entre $\mathcal{C}$ et $\mathcal{C}'$, alors pour toute paire d'états telle que $Z(s, s')$:*

- *pour toute stratégie F_A de la coalition A dans $\mathcal{C}$, il existe une stratégie F'_A de A dans $\mathcal{C}'$ telle que pour toute exécution $\lambda' \in \mathsf{Out}_{\mathcal{C}'}(s', F'_A)$, il existe une exécution $\lambda \in \mathsf{Out}_{\mathcal{C}}(s, F_A)$ telle que $(\mathcal{C}, \lambda) \equiv_A (\mathcal{C}', \lambda')$;*
- *Réciproquement, pour toute stratégie F'_A de A dans $\mathcal{C}'$, il existe une stratégie F_A de A dans $\mathcal{C}$ telle que pour toute exécution $\lambda \in \mathsf{Out}_{\mathcal{C}}(s, F_A)$, il existe une exécution $\lambda' \in \mathsf{Out}_{\mathcal{C}'}(s', F'_A)$ telle que $(\mathcal{C}, \lambda) \equiv_A (\mathcal{C}', \lambda')$.*

Démonstration. On construit $F'_A(\pi')$ par récurrence sur la longueur du préfixe d'exécution π', tel que pour tout état q' de $\mathsf{Next}_{\mathcal{C}'}\big(s', A, F'_A(\pi')\big)$, il existe un préfixe π de $\mathsf{Out}_{\mathcal{C}}(s, F_A)$ tel que $(\mathcal{C}, \pi) \equiv_A (\mathcal{C}', \pi' q')$.

Si $\pi' = s'$, comme $Z(s, s')$, par définition de la A-bisimulation on peut trouver un mouvement m' tel que que quel que soit l'état t' de $\mathsf{Next}_{\mathcal{C}'}(s', A, m')$, il existe un état $t \in \mathsf{Next}\big(s, A, F_A(s)\big)$ qui vérifie $Z(t, t')$. On prend $F'_A(s') = m'$.

Si $|\pi'| = n$. On suppose construit $F'_A(\pi'[0, n-2])$, tel pour tout état q' de $\mathsf{Next}_{\mathcal{C}'}\big(\pi'[n-2], A, F'_A(\pi'[0, n-2])\big)$, il existe un préfixe π de $\mathsf{Out}_{\mathcal{C}}(s, F_A)$ tel que $(\mathcal{C}, \pi) \equiv_A (\mathcal{C}', \pi'[0, n-2]q')$. Soit donc $\pi \in \mathsf{Out}_{\mathcal{C}}(s, A, F_A)$ tel que $(\mathcal{C}, \pi) \equiv_A (\mathcal{C}', \pi')$. En particulier $Z(\pi[n-1], \pi'[n-1])$, donc par définition de la A-bisimulation on peut trouver un mouvement m'' tel que que quel que soit l'état u' de $\mathsf{Next}(\pi'[n-1], A, m'')$, il existe un état $u \in \mathsf{Next}\big(\pi[n-1], A, F_A(\pi)\big)$ qui vérifie $Z(u, u')$. On prend $F'_A(\pi') = m''$.

On peut procéder de même pour montrer l'autre sens. □

Définition 32. *Soit A une coalition fixée. On appelera A-$\mathsf{ATL}^\star$ (resp. A-$\mathsf{ATL}^\star_p$) le fragment de $\mathsf{ATL}^\star$ (resp. de $\mathsf{ATL}^\star_p$) où les quantificateurs de stratégie portent*

tous sur la coalition A. Cela correspond à la syntaxe suivante :

$$A\text{-}\mathsf{ATL}^* \ni \varphi_s, \psi_s \quad ::= \quad p \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \langle\!\langle A \rangle\!\rangle \varphi_p$$
$$A\text{-}\mathsf{ATL}^*_p \ni \varphi_p, \psi_p \quad ::= \quad \varphi_s \mid \neg\varphi_p \mid \varphi_p \vee \psi_p \mid \mathbf{X}\,\varphi_p \mid \varphi_p\,\mathbf{U}\,\psi_p$$

où p décrit l'ensemble AP.

Théorème 4. *Soient $\mathcal{C}$ et $\mathcal{C}'$ deux CGS avec deux états s et s' tels que $(\mathcal{C}, s) \equiv_A (\mathcal{C}', s')$, ainsi que deux exécutions λ et λ' qui partent respectivement de s et s' telles que $(\mathcal{C}, \lambda) \equiv_A (\mathcal{C}', \lambda')$. Alors :*

- *pour toute formule d'état φ de $A\text{-}\mathsf{ATL}^*$, $(\mathcal{C}, s) \models \varphi$ ssi $(\mathcal{C}', s') \models \varphi$;*
- *pour toute formule de chemin φ de $A\text{-}\mathsf{ATL}^*_p$, $(\mathcal{C}, \lambda) \models \varphi$ ssi $(\mathcal{C}', \lambda') \models \varphi$.*

Démonstration. Nous procédons par induction structurelle sur les formules de ATL^* et ATL^*_p. Nous développons ici le cas où φ est une formule d'état de $A\text{-}\mathsf{ATL}^\star$ de la forme $\langle\!\langle A \rangle\!\rangle \psi$. On suppose que $(\mathcal{C}, s) \equiv_A (\mathcal{C}', s')$ et $(\mathcal{C}, s) \models \varphi$.

Alors, par la sémantique de $\mathsf{ATL}^\star$, il existe une stratégie F_A pour la coalition A dans $\mathcal{C}$ dont toutes les outcomes $\lambda \in \mathsf{Out}_{\mathcal{C}}(s, F_A)$ satisfont $(\mathcal{C}, \lambda) \models \psi$. Prenons une stratégie F'_A de la même coalition dans $\mathcal{C}'$ telle que pour toute exécution λ' de $\mathsf{Out}_{\mathcal{C}'}(s', F'_A)$, il existe une exécution λ dans $\mathsf{Out}_{\mathcal{C}}(s, F_A)$ qui vérifie $(\mathcal{C}, \lambda) \equiv_A (\mathcal{C}', \lambda')$. Une telle stratégie F'_A existe d'après le lemme. Maintenant, puisque ψ est satisfaite par toutes les exécutions de $\mathsf{Out}_{\mathcal{C}}(s, F_A)$, il ne reste plus qu'appliquer l'hypothèse d'induction à ψ (qui est une formule de chemin de $A\text{-}\mathsf{ATL}^*_p$) pour obtenir le résultat escompté. On peut procéder de la même manière pour prouver l'autre implication. □

Sens réciproque :

Théorème 5. *Soit A une coalition. Si s et s' sont deux états respectifs de deux CGS $\mathcal{C}$ et $\mathcal{C}'$, tels que les formules de $A\text{-}\mathsf{ATL}$ vraies en s sont les mêmes que celles vraies en s', alors $(\mathcal{C}, s) \equiv_A (\mathcal{C}', s')$.*

Démonstration. Nous allons prouver que la relation binaire "vérifie les mêmes formules de $A\text{-}\mathsf{ATL}$ " est une A-bisimulation. Appelons Z cette relation. On se donne deux états s et s' qui satisfont les mêmes formules de ATL (donc tels que $Z(s, s')$). Il reste à justifier que Z remplit les trois conditions qui définissent une A-bisimulation.

1. $\mathsf{Lab}(s) = \mathsf{Lab}(s')$. Les états s et s' sont étiquetés par les mêmes propositions atomiques, puisqu'ils vérifient les mêmes formules de $A\text{-}\mathsf{ATL}$, dont font partie toutes les formules atomiques.
2. La deuxième condition se montre par l'absurde. Supposons qu'il existe depuis s un vecteur de mouvements m pour la coalition A, tel que tout vecteur de mouvements m' depuis s' pour A a un successeur $t'_{m'}$ qui n'est lié par Z à aucun successeur de m. On note t_i les successeurs de m. Par définition de Z, il existe pour chaque m' et pour chaque i une formule $\varphi_{m',i}$ de $A\text{-}\mathsf{ATL}$ telle que :

$$(\mathcal{C}', t'_{m'}) \models \varphi_{m',i} \qquad \text{et} \qquad (\mathcal{C}, t_i) \not\models \varphi_{m',i}$$

Ainsi, $(\mathcal{C}', t'_{m'}) \models \bigwedge_i \varphi_{m',i}$. Puisque il existe un tel $t'_{m'}$ pour chaque m', on en déduit qu'il est impossible pour A d'éviter la négation des disjonctions de ces formules depuis s', c'est-à-dire que :

$$(\mathcal{C}', s') \not\models \langle\langle A \rangle\rangle \mathbf{X} \neg \bigvee_{m'} \bigwedge_i \varphi_{m',i}.$$

Du côté de s, chaque successeur t_i du mouvement m satisfait $\neg\varphi_{m',i}$, pour tout m'. Cela prouve bien que

$$(\mathcal{C}, s) \models \langle\langle A \rangle\rangle \mathbf{X} \neg \bigvee_{m'} \bigwedge_i \varphi_{m',i}$$

On a exhibé une formule de A-ATL qui est vraie en s mais fausse en s', ce qui contredit l'hypothèse selon laquelle $Z(s, s')$. Remarquons que la disjonction et la conjonction qui portent respectivement sur m' et i sont toutes deux finies, car les nombres de mouvements et d'états (et par conséquent le branchement) d'une structure sont supposés finis.

3. La troisième condition se montre comme la deuxième.

□

Si Z est une A-bisimulation entre deux CGS $\mathcal{C}$ et $\mathcal{C}'$ depuis les états s et s' pour toutes les coalitions $A \subseteq \mathsf{Agt}$, on dit que Z est une bisimulation alternante, et on écrit $(\mathcal{C}, s) \equiv (\mathcal{C}, s)$.

Voici un exemple de structures alternativement bisimilaires :

Elles sont liées par la bisimulation alternante Z suivante :

$$Z = \{(s_0, s_4), (s_1, s_5), (s_2, s_6)\}$$

3.4 GL-équivalence

A notre connaissance, cette définition est une contribution originale.

Définition 33. *Soient deux CGS* $\mathcal{C} = (\mathsf{Agt}, \mathsf{Loc}, \mathsf{AP}, \mathcal{M}, \mathsf{Lab}, \mathsf{Mov}, \mathsf{Edg})$ *et* $\mathcal{C}' = (\mathsf{Agt}, \mathsf{Loc}', \mathsf{AP}, \mathcal{M}', \mathsf{Lab}', \mathsf{Mov}', \mathsf{Edg}')$. *Une relation binaire non vide* $Z \subseteq \mathsf{Loc} \times \mathsf{Loc}'$ *est une* GL-équivalence *de* $\mathcal{C}$ *vers* $\mathcal{C}'$ *ssi pour tous états* $s \in \mathsf{Loc}$ *et* $s' \in \mathsf{Loc}'$, $Z(s, s')$ *implique les trois conditions suivantes :*

- $\mathsf{Lab}(s) = \mathsf{Lab}'(s')$;
- *pour toute coalition* $A \subseteq \mathsf{Agt}$, *pour tout vecteur de mouvements* $m \in \mathsf{Mov}(s, A)$, *il existe un vecteur de mouvements* $m' \in \mathsf{Mov}(s', A)$ *tel que*

1. *quel que soit l'état* t' *de* $\mathsf{Next}(s', A, m')$*, il existe un état* t *parmi* $\mathsf{Next}(s, A, m)$ *qui vérifie* $Z(t, t')$ *;*
2. *quel que soit l'état* t *de* $\mathsf{Next}(s, A, m)$*, il existe un état* t' *parmi* $\mathsf{Next}(s', A, m')$ *qui vérifie* $Z(t, t')$*.*

– *pour toute coalition* $A \subseteq \mathsf{Agt}$*, pour tout vecteur de mouvements* $m' \in \mathsf{Mov}(s', A)$*, il existe un vecteur de mouvements* $m \in \mathsf{Mov}(s, A)$ *tel que*

1. *quel que soit l'état* t' *de* $\mathsf{Next}(s', A, m')$*, il existe un état* t *parmi* $\mathsf{Next}(s, A, m)$ *qui vérifie* $Z(t, t')$ *;*
2. *quel que soit l'état* t *de* $\mathsf{Next}(s, A, m)$*, il existe un état* t' *parmi* $\mathsf{Next}(s', A, m')$ *qui vérifie* $Z(t, t')$*.*

Nous avons également besoin de la définition suivante :

Définition 34. *Soit* Z *une relation binaire entre les nœuds de deux arbres* t *et* t'*.*

On écrit que t Z-correspond *à* t' *lorsque* Z *lie leur racine, et forme une bisimulation entre ces deux arbres, vus comme des structures de Kripke* infinies*.*

Lemme 5. *Soient deux CGS* $\mathcal{C}$ *et* $\mathcal{C}'$*. Si* Z *est une* GL*-équivalence entre* $\mathcal{C}$ *et* $\mathcal{C}'$*, alors pour toute paire d'états telle que* $Z(s, s')$ *et pour toute coalition* A *:*

– *pour toute stratégie* F_A *de la coalition* A *dans* $\mathcal{C}$*, il existe une stratégie* F'_A *de* A *dans* $\mathcal{C}'$ *telle que les arbres d'outcomes* $\mathsf{Out}_{\mathcal{C}'}(s', F'_A)$ *et* $\mathsf{Out}_{\mathcal{C}}(s, F_A)$ Z*-correspondent ;*
– *pour toute stratégie* F'_A *de la coalition* A *dans* $\mathcal{C}'$*, il existe une stratégie* F_A *de* A *dans* $\mathcal{C}$ *telle que* $\mathsf{Out}_{\mathcal{C}'}(s', F'_A)$ *et* $\mathsf{Out}_{\mathcal{C}}(s, F_A)$ Z*-correspondent.*

Démonstration. Supposons $Z(s, s')$. Soit F_A une stratégie pour A. On définit la stratégie $F'_A(\pi')$ par récurrence sur $|\pi'|$, et telle que les préfixes d'outcomes de F_A et F'_A de longueur $|\pi'| + 1$ soient bisimilaires par Z.

Si $\pi' = s'$, on considère le mouvement $F_A(s)$; comme $Z(s, s')$, il existe un mouvement m' tel que

1. quel que soit l'état t' de $\mathsf{Next}(s', A, m')$, il existe un état t parmi l'ensemble de successeurs $\mathsf{Next}\big(s, A, F_A(s)\big)$ qui vérifie $Z(t, t')$;
2. quel que soit l'état t de $\mathsf{Next}\big(s, A, F_A(s)\big)$, il existe un état t' parmi l'ensemble de successeurs $\mathsf{Next}(s', A, m')$ qui vérifie $Z(t, t')$.

On pose $F'_A(s') = m'$.

Si $|\pi'| = n$. On suppose construit $F'_A(\pi'[0, n-2])$ tel que les préfixes d'outcomes de F_A et F'_A de longueur n sont bisimilaires par Z. Il existe donc un préfixe d'outcome $\pi \in \mathsf{Out}(s, F_A)$ de longueur n qui correspond à π'. Comme $Z(\pi[n-1], \pi'[n-1])$, il existe un mouvement m' tel que

1. quel que soit l'état t' de $\mathsf{Next}(\pi'[n-1], A, m')$, il existe un état $t \in \mathsf{Next}\big(\pi[n-1], A, F_A(\pi)\big)$ qui vérifie $Z(t, t')$;
2. quel que soit l'état t de $\mathsf{Next}\big(\pi[n-1], A, F_A(\pi)\big)$, il existe un état $t' \in \mathsf{Next}(\pi'[n-1], A, m')$ qui vérifie $Z(t, t')$.

On pose $F'_A(\pi') = m'$.

On peut procéder de même pour montrer l'autre sens. □

Lemme 6. *Soient t et t' deux arbres Z-correspondants, avec deux nœuds respectifs x et x' tels que $Z(x, x')$.*

Alors les sous-arbres complets $t(x)$ et $t'(x')$ Z-correspondent aussi.

Démonstration. On doit montrer que Z forme une bisimulation entre $t(x)$ et $t'(x')$ et qu'elle lie leur racine. La deuxième affirmation est vraie par l'hypothèse $Z(x, x')$, et la première résulte facilement du fait que les sous-arbres sont complets. □

Théorème 6. *Soient $\mathcal{C}$ et $\mathcal{C}'$ deux CGS avec deux états s et s' tels que $(\mathcal{C}, s)$ et $(\mathcal{C}', s')$ soient GL-équivalents par une relation binaire Z, deux exécutions λ et λ' qui partent respectivement de s et s' telles que $(\mathcal{C}, \lambda)$ et $(\mathcal{C}', \lambda')$ correspondent par Z, et deux arbres t et t' qui ont respectivement s et s' comme racines et sont tels que $(\mathcal{C}, t)$ et $(\mathcal{C}, t')$ Z-correspondent. Alors :*

- *pour toute formule d'état φ de GL, $(\mathcal{C}, s) \models \varphi$ ssi $(\mathcal{C}', s') \models \varphi$;*
- *pour toute formule de chemin φ de GL_p, $(\mathcal{C}, t, \lambda) \models \varphi$ ssi $(\mathcal{C}', t', \lambda') \models \varphi$;*
- *pour toute formule d'arbre φ de GL_t, $(\mathcal{C}, t) \models \varphi$ ssi $(\mathcal{C}', t') \models \varphi$;*

Démonstration. On procède une nouvelle fois par induction structurelle. Nous ne développons que certains cas.

1. $\varphi = \exists A \varphi_t$, où A représente une coalition ; φ est une formule d'état.
   Si $(\mathcal{C}, s) \models \varphi$, alors il existe une stratégie F pour pour A dans $\mathcal{C}$ telle que $\big(\mathcal{C}, \mathsf{Out}(s, F)\big) \models \varphi_t$. D'après le lemme 6, puisque s et s' sont liés par Z, on peut trouver une stratégie F' dans $\mathcal{C}'$, pour la même coalition, telle que Z est une bisimulation entre $\mathsf{Out}_{\mathcal{C}'}(s', F'_A)$ et $\mathsf{Out}_{\mathcal{C}}(s, F_A)$, vues comme des structures.
   Par hypothèse d'induction, on obtient $\big(\mathcal{C}', \mathsf{Out}(s', F')\big) \models \varphi_t$, et la stratégie F' permet d'affirmer $(\mathcal{C}', s') \models \varphi$.
   L'autre partie du lemme sert à prouver l'autre implication.
2. $\varphi = \exists \varphi_p$; φ est une formule d'arbre.
   Supposons que $(\mathcal{C}, t) \models \varphi$, et notons λ l'exécution de l'arbre t qui vérifie φ_p. Nous avons demandé dans l'hypothèse que les arbres t et t' soient bisimilaires par Z, ce qui par le lemme sur la bisimulation, garantit l'existence d'une exécution λ' de l'arbre t' qui Z-correspond à λ.
   D'où par HI, $(\mathcal{C}', t', \lambda') \models \varphi_p$, et finalement $(\mathcal{C}', t') \models \varphi$.
   L'autre sens s'établit de manière analogue.
3. $\varphi = \mathbf{X}\, \varphi_p$; φ est ici une formule de chemin.
   $(\mathcal{C}, t, \lambda) \models \varphi$ signifie que $(\mathcal{C}, t(\lambda[1]), \lambda[1, \infty]) \models \varphi_p$. Par hypothèse λ Z-correspond à λ', donc $\lambda[1, \infty]$ Z-correspond à $\lambda'[1, \infty]$, et en particulier Z lie $\lambda[1]$ et $\lambda'[1]$. De plus t Z-correspond à t', et d'après le lemme, les sous-arbres $t(\lambda[1])$ et $t'(\lambda'[1])$ sont aussi Z-correspondants.
   En appliquant HI, on obtient $(\mathcal{C}', t'(\lambda'[1]), \lambda'[1, \infty]) \models \varphi_p$, c'est-à-dire que $(\mathcal{C}', t', \lambda') \models \varphi$. L'autre sens s'établit de même.

4. $\varphi = \varphi_p \,\mathbf{U}\, \psi_p$;

□

Théorème 7. *La relation binaire "vérifie exactement les mêmes formules de* GL *" forme une* GL*-équivalence entre les états de deux CGS* $\mathcal{C}$ *et* $\mathcal{C}'$*.*

Démonstration. Notons Z la relation binaire "vérifie exactement les mêmes formules de GL ". Soient s et s' deux états tels que $Z(s, s')$. On montre comme d'habitude que Z vérifie la première partie de la définition de GL-équivalence. Nous développons ici la deuxième partie :

Supposons en vue d'une contradiction que Z ne vérifie pas la deuxième condition, c'est-à-dire que :

il existe un vecteur de mouvements m pour une coalition A depuis l'état s, tel que tout vecteur de mouvements m' pour A depuis s' vérifie l'une des assertions suivantes :

1. il existe un état $t'_{m'}$ de $\mathsf{Next}(s', A, m')$, qui n'est lié par Z à aucun état $t \in \mathsf{Next}(s, A, m)$;
2. il existe un état $t_{m'}$ de $\mathsf{Next}(s, A, m)$, qui n'est lié par Z à aucun état $t' \in \mathsf{Next}(s', A, m')$.

Appelons t_i les états de $\mathsf{Next}(s, A, m)$, et t'_j ceux de $\mathsf{Next}(s', A, m')$. Par définition de Z, quel que soit le mouvement m', l'une des assertions est vraie :

1. il existe $t'_{m'}$ tel que pour tout i, il existe une formule $\varphi_{m',i}$ telle que :

$$(\mathcal{C}', t'_{m'}) \models \varphi_{m',i} \qquad \text{et} \qquad (\mathcal{C}, t_i) \not\models \varphi_{m',i}$$

2. il existe $t_{m'}$ tel que pour tout j, il existe une formule $\varphi_{m',j}$ telle que :

$$(\mathcal{C}, t_{m'}) \models \varphi_{m',j} \qquad \text{et} \qquad (\mathcal{C}', t'_j) \not\models \varphi_{m',j}$$

Soit un mouvement m' pour A depuis s'.

1. Si la première assertion est vraie, alors m' a un successeur $t'_{m'}$ tel que $(\mathcal{C}', t'_{m'}) \models \bigwedge_i \varphi_{m',i}$. C'est-à-dire que depuis s', toute stratégie qui démarre par le mouvement m' a son arbre d'outcome qui vérifie :

$$\exists \mathbf{X} \bigwedge_i \varphi_{m',i}$$

Donc toute stratégie depuis s' a son arbre d'outcomes qui vérifie :

$$\bigvee_{m'} \exists \mathbf{X} \bigwedge_i \varphi_{m',i}$$

C'est-à-dire que, dans ce premier cas :

$$(\mathcal{C}, s) \not\models \exists A \neg \Big(\bigvee_{m'} \exists \mathbf{X} \bigwedge_i \varphi_{m',i} \Big)$$

Du côté de s, chaque successeur t_i de m vérifie $\neg\varphi_{m',i}$. Donc la stratégie qui commence par le mouvement m a un arbre d'outcome qui vérifie :

$$\forall \mathbf{X} \bigvee_i \neg\varphi_{m',i}$$

Comme c'est vrai pour chaque m', l'arbre d'outcome de cette stratégie vérifie même :

$$\bigwedge_{m'} \forall \mathbf{X} \bigvee_i \neg\varphi_{m',i}$$

Donc, pour ce cas :

$$(\mathcal{C}, s) \models \exists A\Big(\bigwedge_{m'} \forall \mathbf{X} \bigvee_i \neg\varphi_{m',i}\Big)$$

2. Si la seconde assertion est vraie, alors chaque successeur t'_j de m' satisfait $\neg\varphi_{m',j}$, donc toute stratégie qui démarre par le mouvement m' a son arbre d'outcome qui vérifie :

   $$\forall \mathbf{X} \bigvee_j \neg\varphi_{m',j}$$

   Tous les arbres d'outcomes d'une stratégie depuis s' vérifient :

   $$\bigvee_{m'} \forall \mathbf{X} \bigvee_j \neg\varphi_{m',j}$$

   C'est pourquoi, dans ce second cas :

   $$(\mathcal{C}, s') \not\models \exists A\neg\Big(\bigvee_{m'} \forall \mathbf{X} \bigvee_j \neg\varphi_{m',j}\Big)$$

   Du côté de s, m a un successeur $t_{m'}$ tel que $(\mathcal{C}, t_{m'}) \models \bigwedge_j \varphi_{m',j}$. Ainsi depuis s, la stratégie qui démarre avec le mouvement m a un arbre d'outcome qui vérifie :

   $$\exists \mathbf{X} \bigwedge_j \varphi_{m',j}$$

   et même, puisque c'est vrai pour chaque m' :

   $$\bigwedge_{m'} \exists \mathbf{X} \bigwedge_j \varphi_{m',j}$$

   Finalement dans ce cas,

   $$(\mathcal{C}, s) \models \exists A\Big(\bigwedge_{m'} \exists \mathbf{X} \bigwedge_j \varphi_{m',j}\Big)$$

Ainsi, dans les deux cas, nous sommes en mesure d'exhiber une formule qui est vraie en s mais pas en s', ce qui contredit l'hypothèse $Z(s, s')$. Une nouvelle fois, les disjonctions et conjonctions des formules que l'on a considérées sont finies, car elles portent sur les mouvements ou les états de la structure, qui sont par définition en nombre fini. □

Si l'on compare la définition de la GL-équivalence avec celle de la bisimulation alternante, on voit que cette dernière est moins forte : une GL-équivalence est un cas particulier de bisimulation alternante. Cela est rassurant puisqu'on sait par ailleurs que la logique ATL* peut être vue comme un fragment de GL. Nous verrons dans le corollaire 1 de la section 4.5 que la réciproque n'est pas vraie ; autrement dit, la GL-équivalence est une relation strictement plus forte que la bisimulation alternante.

3.5 Conclusion

Nous n'avons pas trouvé de caractérisation semblable de la relation "satisfaire exactement les mêmes formules de $\mathsf{ATL}_{sc,\infty}$". Nous pouvons toutefois affirmer que cette caractérisation serait strictement plus contraignante que celle de la GL-équivalence qui figure ci-dessus (car $\mathsf{ATL}_{sc,\infty}$ est strictement plus expressive que GL, voir section 4.5.1).

Chapitre 4

Expressivité

Cette partie est constituée des résultats d'expressivité comparée que nous avons établis. Après avoir donné deux critères de comparaison (le pouvoir d'expression et le pouvoir de distinction), nous les utilisons pour confronter nos logiques avec les logiques existantes que nous avions rappelées dans le deuxième chapitre (CTL, ATL, GL, SL, ...).

Nous avons regroupé dans la section 4.7.3 quelques formules remarquables de $\mathsf{ATL}^*_{sc,\infty}$ exprimant certaines propriétés standards de théorie des jeux [17], comme l'existence d'un équilibre de Nash. Nous établissons à la fin de la section 4.7.4 qu'$\mathsf{ATL}_{sc,\infty}$ et $\mathsf{ATL}^*_{sc,\infty}$ ont le même pouvoir d'expression. Nous terminons par une comparaison rapide avec d'autres formalismes intéressants, puis résumons les principaux résultats de la section à l'aide d'une figure.

4.1 Deux critères formels pour comparer l'expressivité de deux logiques

Définition 35. *Soient $\mathcal{L}$ et $\mathcal{L}'$ deux logiques. Nous présentons les définitions de deux relations d'ordre $\leq_{ex}$ et $\leq_{dp}$ qui permettent de comparer les expressivités :*

- *Deux formules d'état φ et φ' sont* équivalentes *lorsque pour toute* CGS $\mathcal{C}$ *et tout état q :*
$$(\mathcal{C}, q) \models \varphi \Leftrightarrow (\mathcal{C}, q) \models \varphi'.$$
- *Deux formules de chemin φ et φ' sont* équivalentes *lorsque pour toute* CGS $\mathcal{C}$*, tout état q et toute exécution $\lambda \in \mathsf{Exec}(q)$:*
$$(\mathcal{C}, \lambda) \models \varphi \Leftrightarrow (\mathcal{C}, \lambda) \models \varphi'.$$

Nous écrirons $\mathcal{L} \leq_{ex} \mathcal{L}'$ pour signifier que toute formule de $\mathcal{L}$ est équivalente à une formule de $\mathcal{L}'$. On dit alors que $\mathcal{L}'$ est plus *expressive* que $\mathcal{L}$.

Définition 36.
- *Deux* CGS $\mathcal{C}$ *et* $\mathcal{C}'$ *sont* indistingables vis-à-vis d'une logique $\mathcal{L}$ *ssi elles satisfont le même ensemble de formules de $\mathcal{L}$.*
- *On écrit que $\mathcal{L} \leq_{dp} \mathcal{L}'$ ssi deux* CGS $\mathcal{C}$ *et $\mathcal{C}'$ indistingables vis-à-vis de $\mathcal{L}'$ le sont aussi vis-à-vis de $\mathcal{L}$.*

Remarques :
- On définit également les relations $\equiv_{ex}$, $\equiv_{dp}$, $<_{ex}$ et $<_{dp}$ de la manière habituelle.
- $\leq_{ex}$ compare ce que l'on appelle le *pouvoir d'expression* de deux logiques.
- $\leq_{dp}$ compare ce que l'on appelle le *pouvoir de distinction* de deux logiques.
- Notons que si $\mathcal{L} \leq_{ex} \mathcal{L}'$, alors $\mathcal{L} \leq_{dp} \mathcal{L}'$, autrement dit la relation $\leq_{ex}$ est plus fine que $\leq_{dp}$.
- Ces relations ne constituant pas un ordre total, nous utiliserons parfois la notation $\neg(x \leq_{dp} y)$.

4.2 Expressivité de la logique $\mathsf{ATL}_{sc,0}$

On va montrer que $\neg(\mathsf{ATL}_{sc,0} \leq_{dp} \mathsf{ATL}^*)$, c'est-à-dire que la logique $\mathsf{ATL}_{sc,0}$ permet de distinguer des structures qui sont indistingables par $\mathsf{ATL}^\star$.

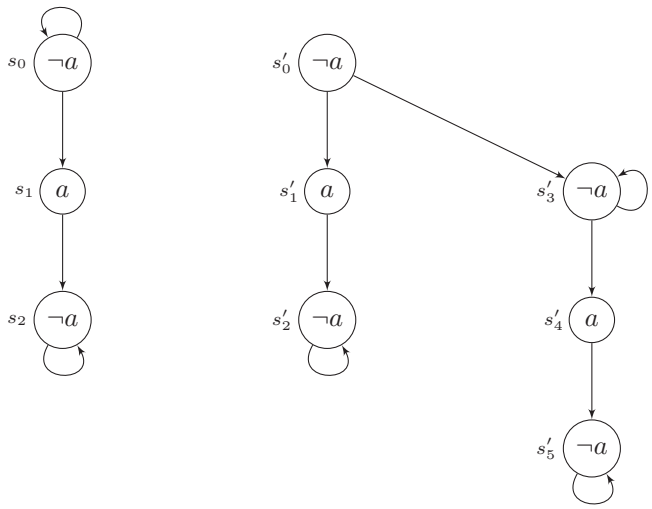

FIG. 4.1 – $\mathcal{C}_1$ et $\mathcal{C}_2$, deux structures de Kripke indistingables par $\mathsf{ATL}^\star$

Lemme 7. *On prouve d'abord que les deux structures de Kripke représentées sur la Figure 4.1, avec pour états initiaux respectifs s_0 et s'_0, sont indistingables par ATL^*.*

Preuve : La relation $R = \{(s_0, s'_0), (s_0, s'_3), (s_1, s'_1), (s_1, s'_4), (s_2, s'_2), (s_2, s'_5)\}$ est une bisimulation qui lie les deux états initiaux. Donc les deux structures considérées sont bisimilaires, et par conséquent indistingables par CTL^*. Les formules de ATL^* étant les mêmes que celles de CTL^* sur les structures de Kripke, on en déduit que ces modèles sont indistingables par ATL^*. □

Il s'agit maintenant d'exhiber une formule de $\mathsf{ATL}_{sc,0}$ qui permette de distinguer ces deux structures.

Lemme 8. *On a* $s'_0 \models \langle\!\langle Agt \rangle\!\rangle_0 \mathbf{X} \langle\!\langle \varnothing \rangle\!\rangle_0 \mathbf{X}\, a$ *mais* $s_0 \not\models \langle\!\langle Agt \rangle\!\rangle_0 \mathbf{X} \langle\!\langle \varnothing \rangle\!\rangle_0 \mathbf{X}\, a$

Preuve : Remarquons que l'opérateur $\langle\!\langle \mathsf{Agt} \rangle\!\rangle_0$, que l'on pourrait noter $\langle\!\langle \mathbf{E} \rangle\!\rangle_0$, a pour effet de transformer la structure de Kripke en l'un de ses fragments déterministes (ce qui correspond à une exécution sans mémoire). Dans chaque état, on supprime toutes les flèches sauf une, et on obtient alors une structure linéaire. Dans $\mathcal{C}_2$, la stratégie pour le joueur consiste à transformer le modèle comme sur la Figure 4.2. Cette structure linéaire vérifie bien la formule de LTL $\mathbf{X}\,\mathbf{X}\, a$.

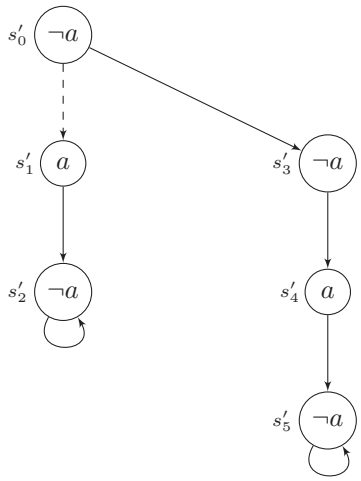

FIG. 4.2 – La stratégie gagnante du joueur dans la CGS $\mathcal{C}_2$

Dans $\mathcal{C}_1$, les deux seules stratégies possibles transforment la structure comme présenté sur la Figure 4.3. Aucune de ces structures linéaires ne vérifie $\mathbf{X}\,\mathbf{X}\, a$, donc la formule $\langle\!\langle \mathsf{Agt} \rangle\!\rangle_0\ \mathbf{X}\ \langle\!\langle \varnothing \rangle\!\rangle_0\ \mathbf{X}\, a$ est fausse en s_0.

□

4.3 Expressivité de la logique $\mathsf{ATL}^*_{sc,0} \setminus \rangle \cdot \langle$

Lemme 9. $CTL^* \leq_{ex} ATL^*_{sc,0} \setminus \rangle \cdot \langle$, *avec une traduction en temps linéaire.*

Preuve : On prolonge naturellement l'application de traduction σ qui permet d'exprimer CTL dans $\mathsf{ATL}_{sc,0}$ en une application qui va de CTL^*_p vers les formules de chemin de $\mathsf{ATL}^*_{sc,0} \setminus \rangle \cdot \langle$. Elle est définie exactement de la même manière, sauf pour les deux étapes inductives qui concernent les modalités temporelles :

- Si $\varphi = \mathbf{X}\,\psi$, alors $\sigma(\varphi) = \mathbf{X}\,\sigma(\psi)$,
- Si $\varphi = \psi_1\,\mathbf{U}\,\psi_2$, alors $\sigma(\varphi) = \sigma(\psi_1)\,\mathbf{U}\,\sigma(\psi_2)$,

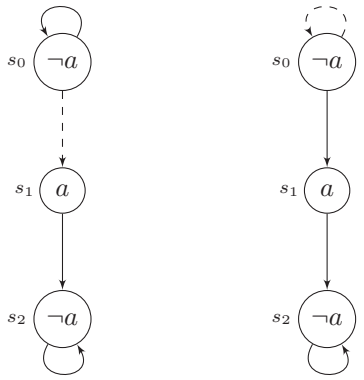

FIG. 4.3 – Les deux stratégies à mémoire 0 du joueur dans la CGS $\mathcal{C}_1$

La différence réside dans le fait que ψ, ψ_1 et ψ_2 désignent cette fois des formules de chemin.

Il s'agit toujours de prouver que toute formule $\varphi \in \mathsf{CTL}^\star_p$ est équivalente à $\sigma(\varphi)$, c'est-à-dire que pour toute CGS $\mathcal{C}$, tout état s et toute exécution λ partant de s,

$$(\mathcal{C}, \lambda) \models \varphi \text{ ssi } (\mathcal{C}, \lambda) \models \sigma(\varphi)$$

Nous allons maintenant développer les cas des modalités temporelles :

– $\varphi = \mathbf{X}\,\psi$:

$$\begin{array}{lll} (\mathcal{C}, \lambda) \models \mathbf{X}\,\psi & \Leftrightarrow & (\mathcal{C}, \lambda[1, \infty]) \models \psi \\ & \Leftrightarrow & (\mathcal{C}, \lambda[1, \infty]) \models \sigma(\psi) \text{ (hypothèse d'induction)} \\ & \Leftrightarrow & (\mathcal{C}, \lambda) \models \sigma(\varphi) \end{array}$$

– Le cas $\varphi = \psi_1\,\mathbf{U}\,\psi_2$ se traite facilement de manière analogue, en appliquant l'hypothèse d'induction sur ψ_2 au chemin $\lambda[i, \infty]$ et celle sur ψ_1 à tous les $\lambda[j, \infty]$ tels que $0 \leq j < i$.

La restriction de ce résultat aux formules d'états de CTL^* termine la preuve. □

4.4 Expressivité de la logique $\mathsf{ATL}^*_{sc,0}$

Lemme 10. *$ATL \leq_{ex} ATL^*_{sc,0}$, avec une traduction en temps linéaire.*

Preuve : On traduit les formules de chemin de ATL en des formules de chemin de $\mathsf{ATL}^*_{sc,0}$, par l'intermédiaire de la fonction σ définie par :

– Si $\varphi = a \in \mathsf{AP}$, alors $\sigma(\varphi) = a$,

– Si $\varphi = \neg\psi$, alors $\sigma(\varphi) = \neg\sigma(\psi)$,
– Si $\varphi = \psi_1 \vee \psi_2$, alors $\sigma(\varphi) = \sigma(\psi_1) \vee \sigma(\psi_2)$,
– Si $\varphi = \langle\!\langle B \rangle\!\rangle \psi$, alors $\sigma(\varphi) = \cdot\rangle\mathsf{Agt}\langle\cdot \ \prec B \succ_0 \ \sigma(\psi)$,
– Si $\varphi = \mathbf{X}\,\psi$, alors $\sigma(\varphi) = \mathbf{X}\,\sigma(\psi)$,
– Si $\varphi = \psi_1 \,\mathbf{U}\, \psi_2$, alors $\sigma(\varphi) = \sigma(\psi_1) \,\mathbf{U}\, \sigma(\psi_2)$,
– Si $\varphi = \psi_1 \,\mathbf{W}\, \psi_2$, alors $\sigma(\varphi) = \big(\sigma(\psi_1) \,\mathbf{U}\, \sigma(\psi_2)\big) \vee \mathbf{G}\,\sigma(\psi_1)$.

On va prouver que toute formule φ de ATL_p est équivalente à $\sigma(\varphi)$.

Pour toute *CGS* $\mathcal{C}$, toute formule φ de ATL_p, toute exécution λ et tout contexte stratégique sans mémoire F :

$$(\mathcal{C}, \lambda) \models \varphi \text{ ssi } (\mathcal{C}, \lambda) \models_F \sigma(\varphi)$$

Nous développons le cas $\varphi = \langle\!\langle B \rangle\!\rangle \psi$.

Soit une exécution λ et un contexte stratégique positionnel F.

$$\begin{array}{lcl}
(\mathcal{C}, \lambda) \models_F \sigma(\varphi) & \Leftrightarrow & (\mathcal{C}, \lambda[0]) \models_F \ \cdot\rangle\mathsf{Agt}\langle\cdot \ \prec B \succ_0 \ \sigma(\psi) \\
& \Leftrightarrow & \forall \lambda' \in \mathsf{Exec}(\lambda[0]), (\mathcal{C}, \lambda') \models_\varnothing \ \prec B \succ_0 \ \sigma(\psi) \\
& \Leftrightarrow & (\mathcal{C}, \lambda[0]) \models_\varnothing \ \prec B \succ_0 \ \sigma(\psi)
\end{array}$$

Cette dernière expression signifie qu'il existe une stratégie F' sans mémoire pour la coalition B telle que toutes les outcomes ρ de la stratégie $F' \circ \varnothing$ qui partent de $\lambda[0]$ satisfassent $\sigma(\psi)$, sous le contexte stratégique $F' \circ \varnothing$, c'est-à-dire F'.

L'hypothèse d'induction permet d'établir l'équivalence avec

$$\exists G \in \mathsf{Strat}_B^0 \ . \ \forall \rho \in \mathsf{Out}(\lambda[0], G), (\mathcal{C}, \rho) \models \psi$$

Nous montrons le sens direct : prenons $G = F'$, et supposons qu'il existe un ρ de $\mathsf{Out}(\lambda[0], G)$ ne vérifiant pas ψ, alors par HI appliquée au contexte F', $(\mathcal{C}, \rho) \not\models_{F'} \sigma(\psi)$, ce qui contredit l'hypothèse.

Le sens réciproque se montre de la même manière.

Or ψ est une formule de chemin de ATL_p, c'est-à-dire qu'elle vaut $\mathbf{X}\,\psi_s$, $\psi_s \,\mathbf{U}\, \varphi_s$ ou bien $\psi_s \,\mathbf{W}\, \varphi_s$, où φ_s et ψ_s désignent des formules d'états de ATL. Dans ces trois cas l'existence d'une stratégie gagnante garantit l'existence d'une stratégie *positionnelle* gagnante.

Donc c'est équivalent à

$$\exists G \in \mathsf{Strat}_B \ . \ \forall \rho \in \mathsf{Out}(\lambda[0], G), (\mathcal{C}, \rho) \models \psi.$$

En restreignant ce résultat aux formules de ATL et à la stratégie $\varnothing$, on prouve le résultat annoncé. □

4.5 Expressivité de la logique $\mathsf{ATL}_{sc,\infty}$

On va montrer la proposition

Proposition 4. $\neg(\mathsf{ATL}_{sc,\infty} \leq_{dp} \mathsf{ATL}^*)$. *Autrement dit, la logique* $\mathsf{ATL}_{sc,\infty}$ *permet de distinguer des structures qui sont indistingables par* ATL^*.

Pour cela, nous allons prouver les deux lemmes suivants.

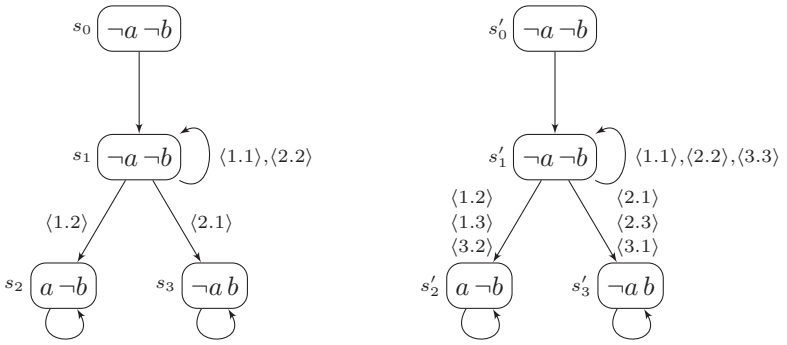

FIG. 4.4 – $\mathcal{C}_1$ et $\mathcal{C}_2$, deux *CGS* indistingables par ATL^*

Lemme 11. *Les deux* CGS *représentées sur la Fig. 5., avec pour états initiaux respectifs* s_0 *et* s'_0*, sont alternativement bisimilaires.*

Preuve : La relation $Z = \{(s_0, s'_0), (s_1, s'_1), (s_2, s'_2), (s_3, s'_3)\}$ est une bisimulation alternante. Chaque paire a ses deux états qui portent les mêmes étiquettes.

Les joueurs sont appelés A_1 et A_2.

L'application σ qui à toute exécution (ou préfixe d'exécution) λ de $\mathcal{C}_1$ fait correspondre l'exécution (ou préfixe d'exécution) λ' de $\mathcal{C}_2$ obtenue en mettant des $'$ à chaque état constituant λ, est manifestement une bijection entre les ensembles des exécutions (ou préfixe d'exécution) des deux structures.

Soit $i \in \{1, 2\}$. On définit les applications :

$$\nu_i : \begin{cases} \mathsf{Mov}_{\mathcal{C}}(s_1, A_j) & \to & \mathsf{Mov}'_{\mathcal{C}'}(s'_1, A_j) \\ 1 & \mapsto & 1 \\ 2 & \mapsto & 2 \end{cases}$$

$$\nu'_i : \begin{cases} \mathsf{Mov}'_{\mathcal{C}'}(s'_1, A_j) & \to & \mathsf{Mov}_{\mathcal{C}}(s_1, A_j) \\ 1 & \mapsto & 1 \\ 2 & \mapsto & 2 \\ 3 & \mapsto & 1 \end{cases}$$

Alors pour chacun des deux joueurs $A_i \in \mathsf{Agt}$, les deux affirmations suivantes sont vérifiées :

– pour tout mouvement m' de $\mathsf{Mov}'_{\mathcal{C}'}(s'_1, A_i)$,

$$\mathsf{Next}(s'_1, A_i, m') \subseteq \sigma\Big(\mathsf{Next}\big(s_1, A_i, \nu'_{A_i}(m')\big)\Big)$$

– pour tout mouvement m de $\mathsf{Mov}_{\mathcal{C}}(s_1, A)$,

$$\mathsf{Next}(s_1, A_i, m) \subseteq \sigma^{-1}\Big(\mathsf{Next}\big(s'_1, A_i, \nu_{A_i}(m)\big)\Big)$$

Ceci établit que Z est une A_i-bisimulation (pour les deux joueurs A_i). Pour conclure est une bisimulation alternante, il suffit de remarquer que Z est également une bisimulation.

□

Il s'agit maintenant d'exhiber une formule de $\mathsf{ATL}_{sc,\infty}$ qui permette de distinguer ces deux structures.

Lemme 12. *Considérons la formule* $\varphi = \ll A_1 \gg \mathbf{X}\,(\ll A_2 \gg \mathbf{X}\,a \wedge \ll A_2 \gg \mathbf{X}\,b)$. *Sur la Figure 4.4, on a :*

1. $(\mathcal{C}_2, s'_0) \models \varphi$
2. $(\mathcal{C}_1, s_0) \not\models \varphi$

Preuve : Dans $\mathcal{C}_2$, une stratégie f_{A_1} telle que $f_{A_1}(s'_0 s'_1) = 3$ nous met dans un contexte qui vérifie que des stratégies f_{A_2} et f'_{A_2} telles que $f_{A_2}(s'_0 s'_1) = 2$ et $f'_{A_2}(s'_0 s'_1) = 1$ prouvent respectivement $\ll A_2 \gg \mathbf{X}\,a$ et $\ll A_2 \gg \mathbf{X}\,b$.

Dans $\mathcal{C}_1$, une stratégie f_{A_1} telle que $f_{A_1}(s'_0 s'_1) = 1$ nous met dans un contexte tel qu'il sera impossible à A_2 d'obtenir b au coup suivant. De même, une stratégie f_{A_1} telle que $f_{A_1}(s'_0 s'_1) = 2$ nous met dans un contexte tel qu'il sera impossible à A_2 d'obtenir a au coup suivant. □

Ceci termine la preuve de la proposition 4.

Par ailleurs, on voit facilement que les mêmes arguments permettraient d'établir que la formule suivante de GL distingue $\mathcal{C}_1$ et $\mathcal{C}_2$:

$$\exists A_1(\exists \mathbf{X}\,\mathbf{X}\,a \wedge \exists \mathbf{X}\,\mathbf{X}\,b)$$

On en déduit le corollaire suivant :

Corollaire 1. *La logique GL permet de distinguer des CGS indistingables par ATL^*.*

De cela, on déduit que la Figure 4.4 présente un couple de CGS qui sont alternativement bisimilaires, mais pas GL-équivalentes.

4.5.1 Comparaison avec GL

Nous avons qu'aussi bien GL que $\mathsf{ATL}_{sc,\infty}$ permettent de distinguer des CGS alternativement bisimilaires. Nous comparons maintenant ces deux logiques entre elles, et nous allons montrer que $\neg(\mathsf{ATL}_{sc,\infty} \leq_{dp} \mathsf{GL})$, c'est-à-dire que la logique $\mathsf{ATL}_{sc,\infty}$ permet de distinguer des structures qui sont indistingables par GL.

Lemme 13. *On prouve d'abord que les deux* CGS *représentées sur la Figure 4.5, avec pour états initiaux respectifs s_0 et s'_0, sont indistingables par GL.*

Preuve : La relation binaire $R = \{(s_0, s'_0); (s_1, s'_1); (s_2, s'_2); (s_3, s'_3)\}$ est une GL-équivalence.

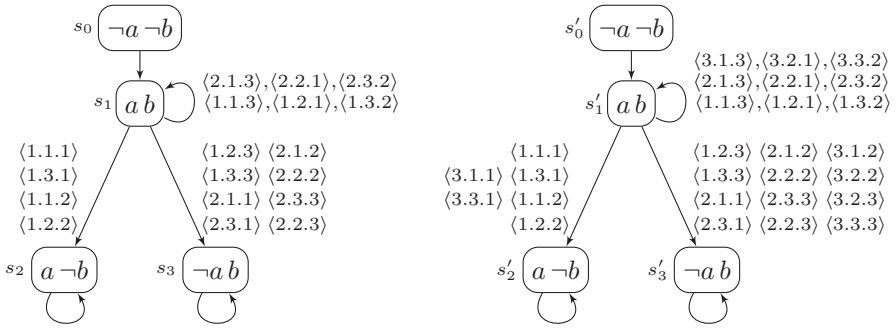

FIG. 4.5 – $\mathcal{C}_1$ et $\mathcal{C}_2$, deux *CGS* indistingables par GL

Soient les applications :

$$\nu_1 : \begin{cases} \mathsf{Mov}_{\mathcal{C}}(s_1, A_1) & \rightarrow & \mathsf{Mov}'_{\mathcal{C}'}(s'_1, A_1) \\ 1 & \mapsto & 1 \\ 2 & \mapsto & 2 \end{cases}$$

$$\nu'_1 : \begin{cases} \mathsf{Mov}'_{\mathcal{C}'}(s'_1, A_1) & \rightarrow & \mathsf{Mov}_{\mathcal{C}}(s_1, A_1) \\ 1 & \mapsto & 1 \\ 2 & \mapsto & 2 \\ 3 & \mapsto & 1 \end{cases}$$

Et pour $j \in \{2, 3\}$, on définit les applications :

$$\nu_j : \begin{cases} \mathsf{Mov}_{\mathcal{C}}(s_1, A_j) & \rightarrow & \mathsf{Mov}'_{\mathcal{C}'}(s'_1, A_j) \\ 1 & \mapsto & 1 \\ 2 & \mapsto & 2 \\ 3 & \mapsto & 3 \end{cases}$$

$$\nu'_j : \begin{cases} \mathsf{Mov}'_{\mathcal{C}'}(s'_1, A_j) & \rightarrow & \mathsf{Mov}_{\mathcal{C}}(s_1, A_j) \\ 1 & \mapsto & 1 \\ 2 & \mapsto & 2 \\ 3 & \mapsto & 3 \end{cases}$$

En faisant le produit de ces applications, on étend ces définitions en une définition de ν_A pour chaque coalition $A \subseteq \mathsf{Agt}$. Ainsi, quelle que soit la coalition A, on a :

- $\forall m' \in \mathsf{Mov}'_{\mathcal{C}'}(s'_1, A), \mathsf{Next}(s'_1, A, m') = \sigma\Big(\mathsf{Next}\big(s_1, A, \nu'_A(m')\big)\Big)$
- $\forall m \in \mathsf{Mov}_{\mathcal{C}}(s_1, A), \mathsf{Next}(s_1, A, m) = \sigma^{-1}\Big(\mathsf{Next}\big(s'_1, A, \nu_A(m)\big)\Big)$

□

Il s'agit maintenant d'exhiber une formule de $\mathsf{ATL}_{sc,\infty}$ qui permette de distinguer ces deux structures.

Lemme 14. *Sur la Figure 4.5, on a :*

1. $(\mathcal{C}_1, s_0) \not\models \ll A_1 \gg \mathbf{X}\,(\ll A_2 \gg \mathbf{X}\,b \wedge \ll A_3 \gg \mathbf{X}\,a)$;
2. $(\mathcal{C}_2, s'_0) \models \ll A_1 \gg \mathbf{X}\,(\ll A_2 \gg \mathbf{X}\,b \wedge \ll A_3 \gg \mathbf{X}\,a)$

Démonstration. Considérons la CGS $\mathcal{C}_1$. Depuis l'état s_1, si le joueur A_1 joue son mouvement 1, alors le joueur A_2 ne peut pas éviter l'état s_2 ; il ne peut pas garantir la propriété $\mathbf{X}\, b$. Maintenant, si A_1 joue 2 depuis s_1, alors A_3 ne peut éviter s_3, et n'a donc pas de stratégie dont l'arbre des outcomes a toutes ses branches qui satisfont $\mathbf{X}\, a$. Cela montre bien que depuis s_0, le joueur A_1 ne dispose pas d'un contexte stratégique qui permet $\mathbf{X}\,(\langle\!\langle A_2 \rangle\!\rangle \mathbf{X}\, b \wedge \langle\!\langle A_3 \rangle\!\rangle \mathbf{X}\, a)$.

Depuis l'état s'_1 de l'autre CGS, A_1 dispose d'un nouveau mouvement 3, plus permissif, donne l'opportunité au joueur A_2 d'assurer $\mathbf{X}\, b$, via son mouvement 2, aussi bien que celle pour A_3 d'atteindre un objectif $\mathbf{X}\, a$ en jouant 1. □

4.5.2 Comparaison avec CTL*

L'usage de contextes stratégiques dans ATL permet d'atteindre l'expressivité de CTL* :

Lemme 15. *$CTL^* \leq_{ex} ATL_{sc,\infty}$, avec une traduction en temps linéaire.*

On définit l'application de traduction σ depuis CTL^*_p vers $\mathsf{ATL}_{sc,\infty}$:
- Si $\varphi = a \in \mathsf{AP}$, alors $\sigma(\varphi) = a$
- Si $\varphi = \neg\psi$, alors $\sigma(\varphi) = \neg\sigma(\psi)$
- Si $\varphi = \psi_1 \vee \psi_2$, alors $\sigma(\varphi) = \sigma(\psi_1) \vee \sigma(\psi_2)$
- Si $\varphi = \mathbf{X}\,\psi$, alors $\sigma(\varphi) = \langle\!\langle \varnothing \rangle\!\rangle \mathbf{X}\, \sigma(\varphi)$
- Si $\varphi = \psi_1 \mathbf{U}\, \psi_2$, alors $\sigma(\varphi) = \langle\!\langle \varnothing \rangle\!\rangle \sigma(\psi_1) \mathbf{U}\, \sigma(\psi_2)$
- Si $\varphi = \mathbf{E}\psi$, alors $\sigma(\varphi) = \langle\!\langle \mathsf{Agt} \rangle\!\rangle \sigma(\psi)$
- Si $\varphi = \mathbf{A}\psi$, alors $\sigma(\varphi) = \neg \langle\!\langle \mathsf{Agt} \rangle\!\rangle \neg\sigma(\psi)$

Voici deux exemples qui illustrent cette transformation :
- $\mathbf{EG}\,(P \vee \mathbf{X}\, P)$ est équivalente à $\langle\!\langle \mathsf{Agt} \rangle\!\rangle \mathbf{G}\,(P \vee \langle\!\langle \varnothing \rangle\!\rangle \mathbf{X}\, P)$;
- $\mathbf{E}(\overset{\infty}{\mathbf{F}}\, P \Rightarrow \overset{\infty}{\mathbf{F}}\, P')$ est équivalente à

$$\langle\!\langle \mathsf{Agt} \rangle\!\rangle \mathbf{F}\,(\langle\!\langle \varnothing \rangle\!\rangle \mathbf{G}\, \neg P) \vee \langle\!\langle \mathsf{Agt} \rangle\!\rangle \mathbf{G}\,(\langle\!\langle \varnothing \rangle\!\rangle \mathbf{F}\, P')$$

Preuve : Nous allons montrer le lemme intermédiaire suivant :

Lemme 16. *Soit $\mathcal{C}$ une CGS. On montre par induction structurelle sur CTL^*_p que pour toute formule φ de CTL^*_p, pour toute stratégie F de la coalition globale Agt et tout état s, si l'on note $\lambda_{s,F}$ l'unique élément de $\mathsf{Out}(s, F)$, on a :*

$$(\mathcal{C}, \lambda_{s,F}) \models \varphi \;\; ssi \; (\mathcal{C}, s) \models_F \sigma(\varphi)$$

Preuve :

1. $\varphi = \mathbf{X}\,\psi$:
   Soit F une stratégie pour tous les joueurs, et un état s.

$$
\begin{array}{lll}
(\mathcal{C}, s) \models_F \sigma(\mathbf{X}\,\psi) & \Leftrightarrow & (\mathcal{C}, s) \models_F \ll\varnothing\gg \mathbf{X}\,\sigma(\varphi) \\
& \Leftrightarrow & \forall \lambda \in \mathsf{Out}(s, F).\ (\mathcal{C}, \lambda) \models_F \mathbf{X}\,\sigma(\psi) \\
& \Leftrightarrow & (\mathcal{C}, \lambda_{s,F}) \models_F \mathbf{X}\,\sigma(\psi), \\
& & \text{car l'ensemble } \mathsf{Out}(s, F) \text{ est réduit à } \{\lambda_{s,F}\} \\
& \Leftrightarrow & (\mathcal{C}, \lambda_{s,F}[1]) \models_{F^{\lambda,1}} \sigma(\psi), \\
& \Leftrightarrow & (\mathcal{C}, \lambda_{\lambda_{s,F}[1], F^{\lambda,1}}) \models \psi \text{ (par HI)} \\
& \Leftrightarrow & (\mathcal{C}, \lambda_{s,F}[1, \infty]) \models \psi \\
& & \text{puisque la seule outcome de } F^{\lambda,1} \text{ depuis} \\
& & \lambda_{s,F}[1] \text{ est } \lambda_{s,F}[1, \infty] \\
& \Leftrightarrow & (\mathcal{C}, \lambda_{s,F}) \models \mathbf{X}\,\psi
\end{array}
$$

2. Le cas $\varphi = \psi_1\,\mathbf{U}\,\psi_2$ peut se démontrer par un raisonnement semblable.
3. Si $\varphi = \mathbf{E}\psi$: soit une stratégie F de Agt et s un état. Alors φ se traduit par $\ll\mathsf{Agt}\gg \sigma(\psi)$.
   Cette formule est vraie en s ssi il existe une stratégie F' pour Agt telle que tous les outcomes de $\mathsf{Out}(s, F' \circ F)$ satisfassent $\sigma(\psi)$, sous le contexte stratégique $F' \circ F$.
   Or $\sigma(\psi)$ est une formule d'état de $\mathsf{ATL}_{sc,\infty}$, donc c'est plus simplement s qui doit vérifier $\sigma(\psi)$ sous le contexte $F' \circ F$, qui est évidemment égal à F'.
   En appliquant l'hypothèse d'induction à chaque stratégie F' depuis l'état s, on obtient l'équivalence avec l'existence d'une stratégie F' telle que $\lambda_{s,F'} \models \psi$.
   Or dans toute *CGS* $\mathcal{C}$, si l'on fixe un état s, l'application entre $\mathsf{Strat}^{\infty}_{\mathsf{Agt}}$ et $\mathsf{Exec}(s)$ qui à F' associe $\lambda_{s,F'}$ est bijective ; l'existence d'une stratégie est donc équivalente à l'existence d'une s-exécution, ce qui implique que tout ceci est enfin équivalent à $s \models \mathbf{E}\psi$, ou bien à $\lambda_{s,F} \models \mathbf{E}\psi$.
4. On peut traiter le dernier cas $\varphi = \mathbf{A}\psi$ avec les mêmes arguments.

□

On peut maintenant montrer que pour toute formule φ de CTL^* et tout état s,

$$(\mathcal{C}, s) \models \varphi \text{ ssi } (\mathcal{C}, s) \models \sigma(\varphi).$$

Supposons que $\varphi = \mathbf{E}\psi$, avec ψ une formule de chemin de CTL^*_p.

Soit s un état. Alors $(\mathcal{C}, s) \models \sigma(\varphi)$ signifie qu'il existe une stratégie F pour la coalition globale Agt telle que $\forall \lambda \in \mathsf{Out}(s, F), (\mathcal{C}, \lambda) \models_F \sigma(\psi)$. Comme $\sigma(\psi)$ est une formule d'état de $\mathsf{ATL}_{sc,\infty}$, c'est équivalent à l'existence d'une stratégie F pour Agt telle que $(\mathcal{C}, s) \models_F \sigma(\psi)$.

En appliquant le résultat du lemme ci-dessus à la formule de chemin ψ depuis l'état s et à toutes les stratégies F de Agt, on obtient l'équivalence avec l'existence de F pour Agt telle que $\mathcal{C}, \lambda_{s,F}) \models \psi$.

Or il existe une stratégie pour Agt ssi il existe une s-exécution, ce qui implique que tout ceci est enfin équivalent à $(\mathcal{C}, s) \models \mathbf{E}\psi$.

On a prouvé que les formules φ et $\sigma(\varphi)$ sont équivalentes sur les CGS. On voit bien que la traduction se fait en temps linéaire. □

4.6 Expressivité de la logique $\mathsf{ATL}^*_{sc,\infty} \setminus \rangle \cdot \langle$

Lemme 17. *Lorsqu'on se limite à* 1 *joueur,* $\mathsf{ATL}^*_{sc,\infty} \setminus \rangle \cdot \langle \leq_{ex} \mathsf{CTL}^*$

Preuve : Nous définissons par induction deux applications de traduction $\sigma_\varnothing$ et σ_{Agt}, qui vont des formules de chemin de $\mathsf{ATL}^*_{sc,\infty} \setminus \rangle \cdot \langle$, vers CTL^*_p. Remarquons que la première transforme toute formule de chemin de $\mathsf{ATL}^*_{sc,\infty} \setminus \rangle \cdot \langle$ en une formule d'*état* de CTL^*.

$$
\begin{array}{rcl}
\sigma_\varnothing(a) = a & \text{et} & \sigma_{\mathsf{Agt}}(a) = a \\
\sigma_\varnothing(\neg\psi) = \neg\sigma_\varnothing(\psi) & \text{et} & \sigma_{\mathsf{Agt}}(\neg\psi) = \neg\sigma_{\mathsf{Agt}}(\psi) \\
\sigma_\varnothing(\psi \vee \varphi) = \sigma_\varnothing(\psi) \vee \sigma_\varnothing(\varphi) & \text{et} & \sigma_{\mathsf{Agt}}(\psi \vee \varphi) = \sigma_{\mathsf{Agt}}(\psi) \vee \sigma_{\mathsf{Agt}}(\varphi) \\
\sigma_\varnothing(\mathbf{X}\,\psi) = \mathbf{X}\,\sigma_\varnothing(\psi) & \text{et} & \sigma_{\mathsf{Agt}}(\mathbf{X}\,\psi) = \mathbf{X}\,\sigma_{\mathsf{Agt}}(\psi) \\
\sigma_\varnothing(\psi\,\mathbf{U}\,\varphi) = \sigma_\varnothing(\psi)\,\mathbf{U}\,\sigma_\varnothing(\varphi) & \text{et} & \sigma_{\mathsf{Agt}}(\psi\,\mathbf{U}\,\varphi) = \sigma_{\mathsf{Agt}}(\psi)\,\mathbf{U}\,\sigma_{\mathsf{Agt}}(\varphi) \\
\sigma_\varnothing(\langle\!\langle \mathsf{Agt} \rangle\!\rangle\ \psi) = \mathbf{E}\sigma_{\mathsf{Agt}}(\psi) & \text{et} & \sigma_{\mathsf{Agt}}(\langle\!\langle \mathsf{Agt} \rangle\!\rangle\ \psi) = \mathbf{E}\sigma_{\mathsf{Agt}}(\psi) \\
\sigma_\varnothing(\langle\!\langle \varnothing \rangle\!\rangle\ \psi) = \mathbf{A}\sigma_\varnothing(\psi) & \text{et} & \sigma_{\mathsf{Agt}}(\langle\!\langle \varnothing \rangle\!\rangle\ \psi) = \sigma_{\mathsf{Agt}}(\psi)
\end{array}
$$

Soit $\mathcal{C}$ une CGS à 1 joueur. On va montrer que pour toute formule φ de $\mathsf{ATL}^*_{sc,\infty} \setminus \rangle \cdot \langle$, pour tout contexte stratégique F d'une coalition quelconque A (soit $\varnothing$ soit Agt puisqu'on suppose qu'il n'y a qu'un joueur), et pour toute outcome λ de F depuis un état s fixé :

$$(\mathcal{C}, \lambda) \models \sigma_A(\varphi) \text{ ssi } (\mathcal{C}, \lambda) \models_F \varphi.$$

1. Traitons d'abord le cas $\varphi = \psi_1\,\mathbf{U}\,\psi_2$.
   Soient F une stratégie pour une coalition A et λ l'une de ses outcomes depuis s. Alors $(\mathcal{C}, \lambda) \models \sigma_A(\varphi)$ est équivalent à $(\mathcal{C}, \lambda) \models \sigma_A(\psi_1)\,\mathbf{U}\,\sigma_A(\psi_2)$, c'est-à-dire qu'il existe i tel que $(\mathcal{C}, \lambda[i, \infty]) \models \sigma_A(\psi_2)$, et pour tout $j < i$, $(\mathcal{C}, \lambda[j, \infty]) \models \sigma_A(\psi_1)$.
   Appliquons maintenant HI, pour tout les k compris entre 0 et i, à la stratégie $F^{\lambda,k}$ et au chemin $\lambda[k, \infty]$ (qui est bien une outcome de $F^{\lambda,k}$ depuis $\lambda[k]$). Si $k = i$, on applique l'hypothèse d'induction portant sur φ_2, et celle sur φ_1 dans les autres cas.
   On a alors $(\mathcal{C}, \lambda[i, \infty]) \models \sigma_A(\varphi_1\,\mathbf{U}\,\psi_2)$ est équivalent à $(\mathcal{C}, \lambda) \models_F \psi_1\,\mathbf{U}\,\psi_2$.

2. $\varphi = \ll \mathsf{Agt} \gg \psi$: on fixe $A \subseteq \mathsf{Agt}, F \in \mathsf{Strat}^{\infty}_{A}, s \in \mathsf{Loc}$ et $\lambda \in \mathsf{Out}(s, F)$.
   $(\mathcal{C}, \lambda) \models \sigma_A(\varphi)$ signifie qu'il existe une exécution λ' qui part de s et qui satisfait $\sigma_{\mathsf{Agt}}(\psi)$.
   Par la bijection déjà utilisée, c'est équivalent à l'existence d'une stratégie F' pour Agt, telle que toutes les outcomes λ' depuis s (il n'y en a en fait qu'une seule) vérifient $\sigma_{\mathsf{Agt}}(\psi)$.
   Par hypothèse d'induction, c'est équivalent à

$$\exists F' \in \mathsf{Strat}^{\infty}_{\mathsf{Agt}} \text{ telle que } \forall \lambda' \in \mathsf{Out}(s, F'), (\mathcal{C}, \lambda') \models_{F'} \psi.$$

   Comme $F' = F' \circ F$, c'est équivalent à

$$\exists F' \in \mathsf{Strat}^{\infty}_{\mathsf{Agt}} \text{ telle que } \forall \lambda' \in \mathsf{Out}(s, F' \circ F), (\mathcal{C}, \lambda') \models_{F' \circ F} \psi,$$

   c'est-à-dire à $(\mathcal{C}, s) \models_F \ll \mathsf{Agt} \gg \psi$, et enfin à $(\mathcal{C}, \lambda) \models_F \ll \mathsf{Agt} \gg \psi$.
3. Si $\varphi = \ll \varnothing \gg \psi$.
   On distingue cette fois les cas $A = \varnothing$ et $A = \mathsf{Agt}$.
   Si $A = \mathsf{Agt}$, alors $(\mathcal{C}, \lambda) \models \sigma_A(\varphi)$ signifie que $(\mathcal{C}, \lambda) \models \sigma_{\mathsf{Agt}}(\psi)$, ce qui est équivalent par HI à : $(\mathcal{C}, \lambda) \models_F \psi$. On voit directement que ceci est équivalent à $(\mathcal{C}, \lambda) \models_F \ll \varnothing \gg \psi$, car F est un contexte stratégique global.
   Maintenant si $A = \varnothing$, $(\mathcal{C}, \lambda) \models \sigma_{\varnothing}(\varphi)$ signifie $\forall \lambda' \in \mathsf{Exec}(s), (\mathcal{C}, \lambda') \models \sigma_{\varnothing}(\psi)$. Par HI, c'est équivalent à $\forall \lambda' \in \mathsf{Exec}(s), (\mathcal{C}, \lambda') \models_F \psi$, c'est-à-dire à $(\mathcal{C}, \lambda) \models_F \ll \varnothing \gg \psi$, puisque F est le contexte stratégique vide.

En appliquant ceci dans le cas où $\varphi \in \mathsf{ATL}^{*}_{sc,\infty} \setminus \langle\cdot\rangle \cdot \langle\cdot$, on prouve que les formules φ et $\sigma_{\varnothing}(\varphi)$ sont équivalentes sur les CGS avec 1 joueur. Cette traduction se fait évidemment en temps linéaire. □

4.7 Expressivité de la logique $\mathsf{ATL}^{*}_{sc,\infty}$

4.7.1 Comparaison avec GL

Lemme 18. *$GL \leq_{ex} ATL^{*}_{sc,\infty}$, avec une traduction en temps linéaire.*

Preuve : On définit par induction l'application $\sigma \colon \mathsf{GL}_p \to \mathsf{ATL}^{*}_{sc,\infty,p}$ telle que :
- Si $\varphi = a \in \mathsf{AP}$, alors $\sigma(\varphi) = a$
- Si $\varphi = \neg\psi$, alors $\sigma(\varphi) = \neg\sigma(\psi)$
- Si $\varphi = \psi_1 \vee \psi_2$, alors $\sigma(\varphi) = \sigma(\psi_1) \vee \sigma(\psi_2)$
- Si $\varphi = \mathbf{X}\,\psi$, alors $\sigma(\varphi) = \mathbf{X}\,\sigma(\varphi)$
- Si $\varphi = \psi_1 \,\mathbf{U}\, \psi_2$, alors $\sigma(\varphi) = \sigma(\psi_1) \,\mathbf{U}\, \sigma(\psi_2)$
- Si $\varphi = \exists\psi$, alors $\sigma(\varphi) = \neg \ll \varnothing \gg \neg\sigma(\psi)$
- Si $\varphi = \exists A\psi$, alors $\sigma(\varphi) = \langle\cdot\mathsf{Agt}\cdot\rangle \ll A \gg \sigma(\psi)$

Soit $\mathcal{C}$ une CGS. On montre par induction sur les formules de chemin de GL_p que, quel que soit le contexte stratégique F de domaine A quelconque, et quelle que soit l'outcome λ de F depuis un état q :

$$(\mathcal{C}, \mathsf{Out}(q, F), \lambda) \models \varphi \text{ ssi } (\mathcal{C}, \lambda) \models_F \sigma(\varphi)$$

1. Si $\varphi = \mathbf{X}\,\psi$. Etant donnés un contexte stratégique F et une outcome λ, $(\mathcal{C}, \lambda) \models_F \sigma(\varphi)$ est équivalent à $(\mathcal{C}, \lambda[1,\infty]) \models_{F^{\lambda,1}} \sigma(\psi)$.
   Appliquons à présent HI au contexte stratégique décalé $F^{\lambda,1}$ et à l'exécution $\lambda[1,\infty]$, qui est bien une outcome de $F^{\lambda,1}$ depuis $\lambda[1]$. En remarquant que :
   $$\mathsf{Out}(\lambda[1], F^{\lambda,1}) = \mathsf{Out}(q, F)(\lambda[1]),$$
   $(\mathcal{C}, \lambda[1,\infty]) \models_{F^{\lambda,1}} \sigma(\psi)$ est équivalent à $(\mathcal{C}, \mathsf{Out}(q,F)(\lambda[1]), \lambda[1,\infty]) \models \psi$.
   Cela signifie exactement $(\mathcal{C}, \mathsf{Out}(q,F), \lambda) \models \mathbf{X}\,\psi$.
2. Si $\varphi = \psi_1\,\mathbf{U}\,\psi_2$, alors un raisonnement similaire donne le résultat.
3. Si $\varphi = \exists\psi$. Soient $A \subseteq \mathsf{Agt}, F \in \mathsf{Strat}^\infty_A, q \in \mathsf{Loc}$ et $\lambda \in \mathsf{Out}(q,F)$. Alors on a la suite d'équivalences :

$$\begin{array}{lll}
(\mathcal{C}, \lambda) \models_F \sigma(\varphi) & \Leftrightarrow & (\mathcal{C}, \lambda) \models_F \neg \llparenthesis \varnothing \rrparenthesis \neg\sigma(\psi) \\
& \Leftrightarrow & \forall \lambda \in \mathsf{Out}(s,F).\ (\mathcal{C}, \lambda) \models_F \mathbf{X}\,\sigma(\psi) \\
& \Leftrightarrow & (\mathcal{C}, q) \not\models_F \llparenthesis \varnothing \rrparenthesis \neg\sigma(\psi) \\
& \Leftrightarrow & \neg\big(\forall \lambda' \in \mathsf{Out}(q,F), (\mathcal{C}, \lambda') \not\models_F \sigma(\psi)\big) \\
& \Leftrightarrow & \exists \lambda' \in \mathsf{Out}(q,F), (\mathcal{C}, \lambda') \models_F \sigma(\psi) \\
& & \text{par HI appliquée à chaque } \lambda' \\
& \Leftrightarrow & \exists \lambda' \in \mathsf{Out}(q,F), (\mathcal{C}, \mathsf{Out}(q,F), \lambda') \models \psi \\
& \Leftrightarrow & \big(\mathcal{C}, \mathsf{Out}(q,F)\big) \models \exists\psi \\
& \Leftrightarrow & (\mathcal{C}, \mathsf{Out}(q,F), \lambda) \models \exists\psi
\end{array}$$

4. Si $\varphi = \exists B\psi$. Soient $A \subseteq \mathsf{Agt}, F \in \mathsf{Strat}^\infty_A, q \in \mathsf{Loc}$ et $\lambda \in \mathsf{Out}(q,F)$. On raisonne par équivalence :

$$
\begin{array}{lll}
(\mathcal{C},\lambda) \models_F \sigma(\varphi) & \Leftrightarrow & (\mathcal{C},\lambda) \models_F \rangle\mathsf{Agt}\langle\cdot \ll B \gg \sigma(\psi) \\
& \Leftrightarrow & (\mathcal{C},q) \models_F \rangle\mathsf{Agt}\langle\cdot \ll B \gg \sigma(\psi) \\
& \Leftrightarrow & (\mathcal{C},q) \models \ll B \gg \sigma(\psi) \\
& \Leftrightarrow & \exists F' \in \mathsf{Strat}^{\infty}_{B}.\ \forall \lambda' \in \mathsf{Out}(q,F'). \\
& & (\mathcal{C},\lambda') \models_{F'} \sigma(\psi) \\
& & \text{par HI appliquée à chaque } \lambda' \\
& \Leftrightarrow & \exists F' \in \mathsf{Strat}^{\infty}_{B}.\ \forall \lambda' \in \mathsf{Out}(q,F'). \\
& & (\mathcal{C},\mathsf{Out}(q,F'),\lambda') \models \psi \\
& & \text{puisque } \psi \text{ est une formule d'arbre} \\
& \Leftrightarrow & \exists F' \in \mathsf{Strat}^{\infty}_{B}, \big(\mathcal{C},\mathsf{Out}(q,F')\big) \models \psi \\
& \Leftrightarrow & (\mathcal{C},q) \models \exists B\psi \\
& & \text{puisque } \exists B\psi \text{ est une formule d'état} \\
& \Leftrightarrow & (\mathcal{C},\mathsf{Out}(q,F),\lambda) \models \exists B\psi
\end{array}
$$

En appliquant ce résultat à la stratégie $\varnothing$ pour la coalition $\varnothing$, dans le cas où φ est une formule d'état, on a prouvé que les formules φ et $\sigma(\varphi)$ sont équivalentes sur les CGS. On voit bien que la traduction se fait en temps linéaire. □

Nous avons le corollaire suivant :

Théorème 8. $\mathsf{ATL}^{*}_{sc,\infty} >_{ex} \mathsf{GL}$

En effet, on a établi dans la section 4.5.1 que $\neg(\mathsf{ATL}_{sc,\infty} \leq_{dp} \mathsf{GL})$. Or ceci implique directement que $\neg(\mathsf{ATL}_{sc,\infty} \leq_{ex} \mathsf{GL})$, et a fortiori que $\neg(\mathsf{ATL}^{*}_{sc,\infty} \equiv_{ex} \mathsf{GL})$

4.7.2 Comparaison avec SL

Théorème 9. $\mathsf{ATL}^{*}_{sc,\infty} \setminus \rangle \cdot \langle \leq_{ex} \mathsf{SL}$, *avec une traduction en temps linéaire.*

Démonstration. Soit une coalition A. On définit par induction une application $\sigma_A \colon \mathsf{ATL}^{*}_{sc,p,\infty} \setminus \rangle \cdot \langle \to \mathsf{SL}$ telle que :

- Si $\varphi = p \in \mathsf{AP}$, alors $\sigma_A(\varphi) = p$
- Si $\varphi = \neg\psi$, alors $\sigma_A(\varphi) = \neg\sigma_A(\psi)$
- Si $\varphi = \psi_1 \vee \psi_2$, alors $\sigma_A(\varphi) = \sigma_A(\psi_1) \vee \sigma_A(\psi_2)$
- Si $\varphi = \ll B \gg \psi$, alors $\sigma_A(\varphi) = \langle\!\langle x \rangle\!\rangle\, (B,x)\, [\![y]\!]\, (\overline{A \cup B}, y)\, \sigma_{A\cup B}(\psi)$
- Si $\varphi = \mathbf{X}\,\psi$, alors $\sigma_A(\varphi) = \mathbf{X}\,\sigma_A(\varphi)$
- Si $\varphi = \psi_1\,\mathbf{U}\,\psi_2$, alors $\sigma_A(\varphi) = \sigma_A(\psi_1)\,\mathbf{U}\,\sigma_A(\psi_2)$

Si F est un contexte stratégique, on dira qu'un assignement complet χ est *fidèle à* F si pour tout les joueurs A_i de $\mathsf{dom}(F)$, $\chi(A_i) = F_{|A_i}$.

Soit $\mathcal{C}$ une CGS. Soit φ une formule de chemin de $\mathsf{ATL}^{*}_{sc,p,\infty} \setminus \rangle \cdot \langle$.

On montre par induction sur φ que, quel que soit le contexte stratégique F de domaine A, et quelle que soit la q-outcome λ de F, si χ est un assignement complet fidèle à F (et tel que la q-outcome de $\chi(\mathsf{Agt})$ est λ), on a :

$$(\mathcal{C}, \lambda) \models_F \varphi \text{ ssi } (\mathcal{C}, \chi, \lambda, 0) \models \sigma_A(\varphi)$$

Nous précisons uniquement le cas $\varphi = \ll B \gg \psi$. Pour se convaincre de l'équivalence dans ce cas, il suffit de remarquer que la donnée d'une outcome de $\mathsf{Out}(q, F_B \circ F)$, où F_B est une stratégie de la coalition B, est équivalente à la donnée d'une stratégie pour la coalition $\overline{A \cup B}$; les quantifications universelles sur les stratégies de $\overline{A \cup B}$ et sur $\mathsf{Out}(q, F_B \circ F)$ sont donc équivalentes.

L'application de ce résultat dans le cas où F est le contexte stratégique vide et φ une formule d'état quelconque de $\mathsf{ATL}^*_{sc,\infty} \setminus \{\cdot\rangle \cdot \langle\cdot\}$, donne que depuis tout état q :

$$(\mathcal{C}, q) \models \varphi \text{ ssi } (\mathcal{C}, q) \models \sigma_\varnothing(\varphi)$$

□

4.7.3 Quelques formules remarquables de $\mathsf{ATL}^*_{sc,\infty}$

Nous commençons par introduire un nouveau quantificateur de stratégies afin de pouvoir spécifier que "pour toute stratégie d'une coalition A, *toutes* les outcomes correspondantes satisfont la formule φ" :

$$[\![A]\!]\,\varphi \stackrel{\text{def}}{=} \neg \ll A \gg \neg \ll \varnothing \gg \varphi.$$

qui se traduit par :

$$(\mathcal{C}, \ell) \models_F [\![A]\!]\,\varphi \quad \text{ssi} \quad \forall F_A \in \mathsf{Strat}(A).\ \forall \rho \in \mathsf{Out}(\ell, F_A \circ F).\ (\mathcal{C}, \rho) \models_{F_A \circ F} \varphi.$$

Cette modalité n'est pas le dual de $\ll A \gg$, parce que nous souhaitons une quantification universelle sur les outcomes. Elle est par conséquent différente du $[\![A]\!]$ de [3] qui spécifie l'existence de *contre*-stratégies garantissant φ pour $\mathsf{Agt} \setminus A$.

Exemples de propriétés s'appuyant sur des contextes stratégiques

Les modalités $\ll A \gg$ nous permettent d'exprimer de nombreuses propriétés concernant les stratégies, bien connues en théorie des jeux (voir par exemple [17]). En particulier, nos logiques peuvent exprimer les différents exemples de propriétés qui ont motivé les introductions de SL, de $\mathrm{Q}D_\mu$ et de IATL :

– Nous pouvons exprimer le fait que *"Le joueur* 1 *peut garantir* Φ_1 *dès lors que le joueur* 2 *joue pour garantir* Φ_2*"* par la formule suivante :

$$\Psi = \ll A_1 \gg [\![A_2]\!] \Big(\big(\langle\!\cdot\!\rangle A_1 \langle\!\cdot\!\rangle \ll \varnothing \gg \Phi_2 \big) \Rightarrow \Phi_1 \Big). \tag{4.1}$$

En interprétant Ψ avec la sémantique d'$\mathsf{ATL}^*_{sc,\infty}$, nous obtenons :

$$(\mathcal{C},\ell) \models \Psi \quad \text{ssi} \quad \exists F_1.\ \forall F_2.(\forall \rho_2 \in \mathsf{Out}(\ell,\{F_2\}).\ (\mathcal{C},\rho_2) \models_{F_2} \Phi_2) \ \Rightarrow \\ (\forall \rho \in \mathsf{Out}(\ell,\{F_1,F_2\}).\ (\mathcal{C},\rho) \models_{\{F_1,F_2\}} \Phi_1) \quad (4.2)$$

L'usage de la modalité $\rangle A_1 \langle$ nous permet de spécifier que Φ_1 doit être garantie par le joueur 1 uniquement lorsque le joueur 2 joue selon une *vraie* stratégie qui garantit (quoi que fasse le joueur 1) Φ_2.

– Etant donnés deux joueurs A_1 et A_2 ayant pour objectifs respectifs Φ_1 et Φ_2, on dit de deux stratégies F_1 et F_2 qu'elles constituent un *équilibre de Nash* si il n'y a pas de "meilleure" stratégie F_1' pour A_1 (par rapport à l'objectif Φ_1) lorsque le joueur 2 joue selon F_2, et réciproquement pour le joueur 2. Cette propriété caractérise des paires de stratégies. La formule suivante est vraie dans les états ℓ ssi il existe un équilibre de Nash depuis ℓ :

$$\lessdot A_1, A_2 \gtrdot \Big((\lessdot A_1 \gtrdot \Phi_1) \Rightarrow \Phi_1 \ \wedge \ (\lessdot A_2 \gtrdot \Phi_2) \Rightarrow \Phi_2 \Big)$$

– le *winning secure equilibrium* [10] est une forme particulière d'équilibre de Nash, où les deux joueurs peuvent coopérer pour satisfaire un objectif $\Phi_1 \wedge \Phi_2$, et si le joueur 1 (resp. 2) abandonne la coopération pour garantir $\neg\Phi_2$ (resp. $\neg\Phi_1$) alors le joueur 2 (resp. 1) peut obtenir $\neg\Phi_1$ (resp. $\neg\Phi_2$). L'existence d'un tel équilibre peut être formulée comme suit (Φ_1 et Φ_2 sont supposées être des formules de LTL) :

$$\lessdot A_1, A_2 \gtrdot \Big(\Phi_1 \wedge \Phi_2 \ \wedge \ [\![A_1]\!] (\neg\Phi_2 \Rightarrow \neg\Phi_1) \ \wedge \ [\![A_2]\!] (\neg\Phi_1 \Rightarrow \neg\Phi_2) \Big).$$

En effet cette formule exprime que les joueurs 1 et 2 peuvent coopérer afin de garantir leurs objectifs respectifs, mais qu'il n'est possible à aucun joueur de garantir son propre objectif tout en empêchant l'autre joueur d'atteindre le sien.

– On peut également exprimer la notion de stratégie *dominante*, *i.e.*, le fait qu'une stratégie soit meilleure que toutes les autres pour un objectif donné. Un état ℓ satisfait la formule suivante ssi il existe une stratégie dominante du joueur A_1 depuis ℓ pour l'objectif Φ :

$$\lessdot A_1 \gtrdot \Big(\lessdot \mathsf{Agt} \setminus A_1 \gtrdot (\neg\Phi \Rightarrow \neg \lessdot A_1 \gtrdot \Phi) \Big)$$

En effet, si l'adversaire a une contre-stratégie pour garantir $\neg\Phi$ lorsque A_1 joue selon sa stratégie dominante, alors A_1 n'avait aucune stratégie gagnante contre cette contre-stratégie.

Borner la mémoire de l'adversaire

Notre définition de stratégie à mémoire bornée ne donne pas de restriction aux possibles contre-stratégies des adversaires. Pourtant, il pourrait dans certains cas être intéressant de chercher des stratégies qui ne gagnent que contre

des adversaires *déterministes* qui ont une mémoire bornée. Cela peut être exprimé dans notre logique en dualisant le quantificateur de stratégie à mémoire bornée. En effet, si l'on pose :

$$[\![A]\!]_b\, \varphi \stackrel{\text{def}}{=} \neg \ll A \gg \neg \ll \varnothing \gg_b \varphi,$$

alors la formule

$$\ll A \gg\ [\![\mathsf{Agt} \setminus A]\!]_b\, \Phi$$

spécifie que la coalition A a une stratégie garantissant Φ si la coalition adverse a une mémoire bornée par b.

Expressivité de l'opérateur $\rangle A \langle$

Jusqu'ici, nous avons illustré l'utilité de la modalité $\rangle A \langle$ en exprimant la modalité classique d'$\mathsf{ATL}^\star$ $\langle\!\langle A \rangle\!\rangle$ par $\rangle \mathsf{Agt} \langle \ll A \gg$: on oublie d'abord le contexte stratégique courant, puis on quantifie sur l'existence d'une stratégie pour A : l'utilisation de l'opérateur $\rangle \cdot \langle$ est justifiée par le fait que la stratégie doit être réelle, autrement dit correcte pour n'importe quel choix des autres joueurs. En réalité, cette modalité n'apporte pas d'expressivité à $\mathsf{ATL}^*_{sc,\infty}$:

Théorème 10. *Pour toute formule Φ d'$\mathsf{ATL}^*_{sc,\infty}$, il existe une formule Ψ qui ne contient aucun opérateur $\rangle - \langle$, et telle que $\Phi \equiv \Psi$.*

Démonstration. Etant donnés un ensemble de joueurs $C \subseteq \mathsf{Agt}$ et $\Phi \in \mathsf{ATL}^\star_{sc,\infty}$, nous définissons récursivement une formule $\overline{\Phi}^C$ comme suit :

$$\overline{\ll A \gg \Phi}^C \stackrel{\text{def}}{=} \ll A \gg\ [\![C \setminus A]\!]\, \overline{\Phi}^{C \setminus A} \qquad \overline{\rangle A \langle\, \Phi}^C \stackrel{\text{def}}{=} \overline{\Phi}^{C \cup A}$$

$$\overline{\Phi\, \mathbf{U}\, \Psi}^C \stackrel{\text{def}}{=} \overline{\Phi}^C\, \mathbf{U}\, \overline{\Psi}^C \qquad \overline{\mathbf{X}\, \Phi}^C \stackrel{\text{def}}{=} \mathbf{X}\, \overline{\Phi}^C$$

$$\overline{\Phi \wedge \Psi}^C \stackrel{\text{def}}{=} \overline{\Phi}^C \wedge \overline{\Psi}^C \qquad \overline{\neg \Phi}^C \stackrel{\text{def}}{=} \neg \overline{\Phi}^C$$

$$\overline{P}^C \stackrel{\text{def}}{=} P$$

Dans la suite de cette section, si F est un contexte stratégique pour une coalition A, on utilisera la notation $\mathsf{dom}(F)$ pour désigner le domaine de ce contexte, c'est-à-dire ici A.

De plus, on abrègera $F_{|\mathsf{Agt} \setminus C}$ en $F_{\setminus C}$.

Lemme 19. *Pour tout contexte stratégique F, toute coalition $C \subseteq \mathsf{dom}(F)$, toute formule $\Phi \in \mathsf{ATL}^\star_{sc,\infty}$ et toute formule de chemin Φ_p, on a :*

$$(\mathcal{C}, q) \models_{F_{\setminus C}} \Phi \quad \Leftrightarrow \quad (\mathcal{C}, q) \models_F \overline{\Phi}^C$$
$$(\mathcal{C}, \rho) \models_{F_{\setminus C}} \Phi_p \quad \Leftrightarrow \quad (\mathcal{C}, \rho) \models_F \overline{\Phi_p}^C$$

Démonstration. Nous le prouvons par induction structurelle sur les formules. Dans cette preuve nous utiliserons C' comme abréviation de la coalition $C \setminus A$. De plus, pour une coalition B fixée, on notera F_B pour indiquer qu'il s'agit d'une stratégie de B.

– $\Psi \stackrel{\text{def}}{=} \ll A \gg \varphi$.

Nous avons les équivalences suivantes : $(\mathcal{C}, q) \models_F \ll A \gg [\!\!\restriction C'] \overline{\varphi}^{C'}$ signifie par définition

$$\exists F_A.\ \forall F_{C'}.\ \forall \rho \in \mathsf{Out}(q, F_{C'} \circ F_A \circ F).\ (\mathcal{C}, \rho) \models_{F_{C'} \circ F_A \circ F} \overline{\varphi}^{C'},$$

L'hypothèse d'induction entraîne

$$\exists F_A.\ \forall F_{C'}.\ \forall \rho \in \mathsf{Out}(q, F_{C'} \circ F_A \circ F).\ (\mathcal{C}, \rho) \models_{(F_{C'} \circ F_A \circ F)_{\setminus (C')}} \varphi,$$

ce qui est équivalent à, puisque $(F_{C'} \circ F_A \circ F)_{\setminus C'} = F_A \circ (F_{\setminus C})$:

$$\exists F_A.\ \forall F_{C'}.\ \forall \rho \in \mathsf{Out}(q, F_{C'} \circ F_A \circ F).\ (\mathcal{C}, \rho) \models_{F_A \circ (F_{\setminus C})} \varphi,$$

ou bien, comme on a :

$$\bigcup_{F_{C'} \in \mathsf{Strat}(C')} \mathsf{Out}(q, F_{C'} \circ F_A \circ F) = \mathsf{Out}\big(q, F_A \circ \big(F_{\setminus C}\big)\big)$$

$$\exists F_A.\ \forall \rho \in \mathsf{Out}(q, F_A \circ \big(F_{\setminus C}\big)).\ (\mathcal{C}, \rho) \models_{F_A \circ (F_{\setminus C})} \varphi,$$

cela nous amène à $(\mathcal{C}, q) \models_{F_{\diagdown C}} \ll A \gg \varphi$, qui est le résultat annoncé.

– $\Psi \stackrel{\text{def}}{=} \langle\!\cdot A \cdot\!\rangle \varphi$. D'une part, d'après la sémantique d'$\mathsf{ATL}^{\star}_{sc,\infty}$, nous avons :

$$(\mathcal{C}, q) \models_{F_{\setminus C}} \langle\!\cdot A \cdot\!\rangle \varphi \quad \text{ssi} \quad (\mathcal{C}, q) \models_{F_{\setminus (C \cup A)}} \varphi.$$

D'autre part, l'hypothèse d'induction dit que :

$$(\mathcal{C}, q) \models_{F_{\setminus (C \cup A)}} \varphi \quad \text{ssi} \quad (\mathcal{C}, q) \models_F \overline{\varphi}^{C \cup A}.$$

En assemblant les deux équivalences, nous obtenons le résultat annoncé.

– $\Psi \stackrel{\text{def}}{=} \varphi \,\mathbf{U}\, \psi$. Par définition d'$\mathsf{ATL}^{\star}_{sc,\infty}$, $(\mathcal{C}, \rho) \models_{F_{\diagdown C}} \Psi$ si et seulement si :

$$\exists i.\ (\mathcal{C}, \rho[i, \infty]) \models_{(F_{\setminus C})^{\rho, i}} \psi \text{ et } \forall 0 \leq j < i.\ (\mathcal{C}, \rho[j, \infty]) \models_{(F_{\setminus C})^{\rho, j}} \varphi.$$

D'après l'hypothèse d'induction, la formule précédente est équivalente à la suivante :

$$\exists i.\ (\mathcal{C}, \rho[i, \infty]) \models_{F^{\rho, i}} \overline{\psi}^{C} \text{ et } \forall 0 \leq j < i.\ (\mathcal{C}, \rho[j, \infty]) \models_{F^{\rho, j}} \overline{\varphi}^{C},$$

ce qui signifie que $(\mathcal{C}, \rho) \models_F \overline{\varphi}^{C} \,\mathbf{U}\, \overline{\psi}^{C}$. Nous avons démontré le résultat annoncé.

– Les cas restants sont faciles.

□

En prenant $\Psi = \overline{\Phi}^{\varnothing}$, on achève la démonstration du théorème.

□

4.7.4 Expressivités comparées de $\mathsf{ATL}_{sc,\infty}$ et $\mathsf{ATL}^{*}_{sc,\infty} \setminus \rangle \cdot \langle \cdot$

Nous allons à présent montrer que les contextes stratégiques permettent à $\mathsf{ATL}_{sc,\infty}$ d'atteindre le niveau d'expressivité de $\mathsf{ATL}^{*}_{sc,\infty} \setminus \rangle \cdot \langle \cdot$. On peut déjà le remarquer dans cet exemple simple : considérons la formule de $\mathsf{CTL}^{\star}$ $\varphi = \mathsf{EG}\,\mathbf{F}\,a$ (qui exprime que a apparaît infiniment souvent le long d'une exécution) ; il est bien connu dans la littérature que φ ne peut pas s'exprimer dans CTL [16]. Pourtant φ est clairement équivalente (sur les structures de Kripke, qui sont des CGS à un seul joueur) à la formule $\ll A \gg \mathbf{G} \ \ll \varnothing \gg \mathbf{F}\,a$, qui est dans $\mathsf{ATL}_{sc,\infty}$: en effet, dès qu'un contexte stratégique est fixé pour l'unique joueur A, l'usage du quantificateur $\ll\varnothing\gg$ ne modifie pas le choix de l'exécution sous-jacente, où $\mathbf{F}\,a$ est interprétée. Il est possible de faire valoir cette approche sur toutes les formules de $\mathsf{ATL}^{*}_{sc,\infty} \setminus \rangle \cdot \langle \cdot$; l'idée est de :

1. d'abord utiliser systématiquement des contextes stratégiques *globaux* (en complétant chaque contexte à l'aide de quantificateurs de stratégies portant sur les joueurs manquants), ce qui garantit que l'on peut sans conséquences insérer la modalité $\ll\varnothing\gg$ devant chaque opérateur temporel, puis
2. s'assurer que pour chaque quantificateur de stratégie $\ll A\gg$, la coalition A ne peut pas profiter des stratégies que l'on a ajoutées. Pour cela, nous remplaçons $\ll A \gg \varphi$ par la sous-formule $\ll A \gg \neg \ll \mathsf{Agt} \setminus (A \cup B) \gg \neg\varphi$, où B constitue la partie du contexte sur laquelle A peut s'appuyer pour choisir sa stratégie (c'est-à-dire que A ne "connaît" que les stratégies utilisées par les joueurs de B). Nous faisons suivre une description formelle de ce mécanisme.

Voici une traduction depuis $\mathsf{ATL}^{*}_{sc,\infty} \setminus \rangle \cdot \langle \cdot$ vers $\mathsf{ATL}_{sc,\infty}$. Etant données une formule Φ de $\mathsf{ATL}^{*}_{sc,\infty} \setminus \rangle \cdot \langle \cdot$ ainsi qu'une coalition B, on définit inductivement $\widehat{\Phi}^{[B]}$ comme suit :

$$\widehat{P}^{[B]} \stackrel{\text{def}}{=} P \qquad \widehat{\varphi \wedge \psi}^{[B]} \stackrel{\text{def}}{=} \widehat{\varphi}^{[B]} \wedge \widehat{\psi}^{[B]}$$

$$\widehat{\mathbf{X}\,\varphi}^{[B]} \stackrel{\text{def}}{=} \ll \varnothing \gg \mathbf{X}\,\widehat{\varphi}^{[B]} \qquad \widehat{\neg\varphi}^{[B]} \stackrel{\text{def}}{=} \neg\widehat{\varphi}^{[B]}$$

$$\widehat{\varphi\,\mathbf{U}\,\psi}^{[B]} \stackrel{\text{def}}{=} \ll \varnothing \gg (\widehat{\varphi}^{[B]}\,\mathbf{U}\,\widehat{\psi}^{[B]})$$

$$\widehat{\ll A \gg \varphi}^{[B]} \stackrel{\text{def}}{=} \ll A \gg \neg \ll \mathsf{Agt} \setminus (A \cup B) \gg \neg\widehat{\varphi}^{[A\cup B]}$$

Il est clair que $\widehat{\Phi}^{[B]}$ est une formule de $\mathsf{ATL}_{sc,\infty}$. Nous allons d'abord prouver le lemme suivant, qui établit que la valeur de vérité attribuée à une formule d'état $\widehat{\varphi}^{[A]}$, interprétée sous un contexte stratégique H, dépend uniquement de $H_{|A}$:

Lemme 20. *Pour toute formule d'état φ, tous contextes stratégiques F, G et G' tels que $\mathsf{dom}(F) \cap \mathsf{dom}(G') = \varnothing$, $\mathsf{dom}(G) \subseteq \mathsf{dom}(G')$ et $G'_{|\mathsf{dom}(G)} = G$, on a :*

$$(\mathcal{C}, q) \models_{G\circ F} \widehat{\varphi}^{[\mathsf{dom}(F)]} \ \Leftrightarrow \ (\mathcal{C}, q) \models_{G'\circ F} \widehat{\varphi}^{[\mathsf{dom}(F)]}$$

Démonstration. La preuve se fait par induction structurelle sur les formules. Les cas des propositions atomiques et des opérateurs booléens se traitent facilement.

Prenons $\varphi \stackrel{\text{def}}{=} \ll A \gg \psi$. Alors $(\mathcal{C}, q) \models_{G\circ F} \widehat{\ll A \gg \psi}^{[\mathsf{dom}(F)]}$ se traduit par $(\mathcal{C}, q) \models_{G\circ F} \ll A \gg \neg \ll \mathsf{Agt} \setminus \big(A \cup \mathsf{dom}(F)\big) \gg \neg\widehat{\psi}^{[\mathsf{dom}(F)\cup A]}$. Par conséquent il existe $F_A \in \mathsf{Strat}(A)$ telle que pour tout $F' \in \mathsf{Strat}\Big(\mathsf{Agt} \setminus \big(A \cup \mathsf{dom}(F)\big)\Big)$, on a $(\mathcal{C}, \pi) \models_{F'\circ F_A \circ G\circ F} \widehat{\psi}^{[\mathsf{dom}(F)\cup A]}$, où π représente la seule exécution de $\mathsf{Out}(q, F' \circ F_A \circ G \circ F)$.

On voit bien que $(\mathcal{C}, \pi) \models_{F'\circ F_A\circ G'\circ F} \widehat{\psi}^{[\mathsf{dom}(F)\cup A]}$ puisque $F' \circ F_A$ écrase les stratégies G et G' du contexte. On peut prouver l'implication réciproque en suivant les mêmes étapes.

□

Le lemme suivant permet d'établir un lien entre la valeur de vérité de φ et celle de $\widehat{\varphi}^{[B]}$:

Lemme 21. *Soit $\mathcal{C}$ une CGS, q un de ses états, et F un contexte stratégique. Alors pour toute formule φ dans $\mathsf{ATL}^*_{sc,\infty} \setminus \rangle\cdot\langle$, pour tout contexte stratégique G tel que $\mathsf{dom}(G) = \mathsf{Agt} \setminus \mathsf{dom}(F)$, et pour toute outcome infinie $\pi \in \mathsf{Out}(q, G\circ F)$:*

$$(\mathcal{C}, \pi) \models_F \varphi \;\Leftrightarrow\; (\mathcal{C}, \pi) \models_{G\circ F} \widehat{\varphi}^{[\mathsf{dom}(F)]}.$$

De plus, si φ est une formule d'état, ce résultat reste vrai pour tout contexte stratégique G dont le domaine est disjoint de $\mathsf{dom}(F)$.

Démonstration. La preuve est une induction sur la structure de φ.

1. Si $\varphi = \mathbf{X}\,\psi$: pour tout G t.q. $\mathsf{dom}(G) = \mathsf{Agt} \setminus \mathsf{dom}(F)$, en notant π l'outcome résultante depuis q, nous avons la suite d'équivalences :

$$\begin{aligned}
(\mathcal{C}, \pi) \models_F \varphi \quad &\Leftrightarrow \quad (\mathcal{C}, \pi[1,\infty]) \models_{F^{\pi,1}} \psi \\
&\Leftrightarrow \quad (\mathcal{C}, \pi[1,\infty]) \models_{(G\circ F)^{\pi,1}} \widehat{\psi}^{[\mathsf{dom}(F)]} \qquad \text{(par HI)} \\
&\Leftrightarrow \quad (\mathcal{C}, \pi) \models_{G\circ F} \mathbf{X}\,\widehat{\psi}^{[\mathsf{dom}(F)]} \\
&\Leftrightarrow \quad (\mathcal{C}, \pi) \models_{G\circ F} \ll \varnothing \gg \mathbf{X}\,\widehat{\psi}^{[\mathsf{dom}(F)]} \\
&\qquad\qquad \text{(parce que } \mathsf{Out}(q, G\circ F) = \{\pi\}\text{)}
\end{aligned}$$

2. Si $\varphi = \psi_1\,\mathbf{U}\,\psi_2$: ce cas peut se traiter comme le cas précédent.
3. Si $\varphi = \ll A \gg \psi$: nous allons prouver la seconde affirmation, qui est plus générale. Soit G un contexte stratégique tel que $\mathsf{dom}(G) \cap \mathsf{dom}(F) = \varnothing$, et π une outcome de $G \circ F$ depuis q. Dans cette suite d'équivalences, la coalition $\mathsf{dom}(F) \cup A$ sera notée D.

$$
\begin{aligned}
(\mathcal{C},q) \models_F \varphi \quad &\Leftrightarrow \quad (\mathcal{C},q) \models_F \ll A \gg \psi \\
&\Leftrightarrow \quad \exists F_A \in \mathsf{Strat}(A).\ \forall \pi' \in \mathsf{Out}(q, F_A \circ F).\ (\mathcal{C},\pi') \models_{F_A \circ F} \psi \\
&\Leftrightarrow \quad \exists F_A \in \mathsf{Strat}(A).\ \forall F' \in \mathsf{Strat}\Big(\mathsf{Agt} \setminus (D)\Big). \\
&\qquad\qquad \forall \pi' \in \mathsf{Out}(q, F' \circ F_A \circ F).\ (\mathcal{C},\pi') \models_{F_A \circ F} \psi \\
&\Leftrightarrow \quad \exists F_A \in \mathsf{Strat}(A).\ \forall F' \in \mathsf{Strat}\Big(\mathsf{Agt} \setminus (D)\Big). \\
&\qquad\qquad \forall \pi' \in \mathsf{Out}(q, F' \circ F_A \circ F).\ (\mathcal{C},\pi') \models_{F' \circ F_A \circ F} \widehat{\psi}^{[D]} \\
&\qquad\qquad\qquad \text{(par hypothèse d'induction)} \\
&\Leftrightarrow \quad \exists F_A \in \mathsf{Strat}(A).\forall F' \in \mathsf{Strat}\Big(\mathsf{Agt} \setminus (D)\Big). \\
&\qquad \forall \pi' \in \mathsf{Out}(q, F' \circ F_A \circ G \circ F). \\
&\qquad\qquad (\mathcal{C},\pi') \models_{F' \circ F_A \circ G \circ F} \widehat{\psi}^{[D]} \\
&\qquad\qquad (F' \circ F_A \circ G \circ F = F' \circ F_A \circ F, \text{ car } F' \text{ écrase } G) \\
&\Leftrightarrow \quad \exists F_A \in \mathsf{Strat}(A).\forall F' \in \mathsf{Strat}\Big(\mathsf{Agt} \setminus (D)\Big). \\
&\qquad \exists \pi' \in \mathsf{Out}(q, F' \circ F_A \circ G \circ F). \\
&\qquad\qquad (\mathcal{C},\pi') \models_{F' \circ F_A \circ G \circ F} \widehat{\psi}^{[D]} \\
&\qquad\qquad \text{(puisque } |\mathsf{Out}(q, F' \circ F_A \circ G \circ F)| = 1) \\
&\Leftrightarrow \quad \exists F_A \in \mathsf{Strat}(A). \\
&\qquad\qquad (\mathcal{C},q) \models_{F_A \circ G \circ F} \neg \ll \mathsf{Agt} \setminus (D) \gg \neg \widehat{\psi}^{[D]} \\
&\Leftrightarrow \quad (\mathcal{C},q) \models_{G \circ F} \ll A \gg \neg \ll \mathsf{Agt} \setminus (D) \gg \neg \widehat{\psi}^{[D]} \\
&\Leftrightarrow \quad (\mathcal{C},q) \models_{G \circ F} \widehat{\varphi}^{[\mathsf{dom}(F)]}
\end{aligned}
$$

□

Corollaire 2. *Soit φ une formule de $\mathsf{ATL}^*_{sc,\infty} \setminus \rangle \cdot \langle$, et A une coalition d'un ensemble de joueurs Agt fixé. Alors φ est équivalente à $\widehat{\varphi}^{[A]}$ sous n'importe quel contexte stratégique F de domaine A, et pour toute CGS construite sur Agt, autrement dit pour toute CGS $\mathcal{C}$ dont l'ensemble de joueurs est Agt, quelle que soit l'exécution ρ, nous avons :*

$$(\mathcal{C},\rho) \models_F \varphi \qquad ssi \qquad (\mathcal{C},\rho) \models_F \widehat{\varphi}^{[\mathsf{dom}(F)]}.$$

Comme notre transformation ne dépend pas de la CGS impliquée, on en déduit :

Théorème 11. *Etant donné un ensemble de joueurs Agt, toute formule φ de $\mathsf{ATL}^*_{sc,\infty} \setminus \rangle \cdot \langle$ peut être transformée en une formule équivalente $\widehat{\varphi}$ (sous le contexte vide) de $\mathsf{ATL}_{sc,\infty}$, et cela pour n'importe quelle CGS construite sur Agt.*

Enfin, on déduit des Théorèmes 10 et 11 que :

Corollaire 3. *Toute formule de $\mathsf{ATL}^{*}_{sc,\infty}$ peut se traduire, en un temps linéaire, en une formule équivalente de $\mathsf{ATL}_{sc,\infty}$.*

4.7.5 Comparaison avec quelques autres formalismes

Comparaison avec AMC

Le μ-calcul alternant a une expressivité qui est incomparable avec celle de nos extensions de ATL, ce qui signifie qu'aucune n'est plus expressive que l'autre :

Lemme 22. *$\mathsf{ATL}_{sc,\infty} \not\leq_{ex} \mathsf{AMC}$ et $\mathsf{AMC} \not\leq_{ex} \mathsf{ATL}^{\star}_{sc,\infty}$.*

Démonstration. La CGS $\mathcal{C}_2$ (qui est représentée dans la Figure 4.4) est alternativement bisimilaire à $\mathcal{C}_1$, par conséquent elles satisfont les mêmes formules de AMC. Cependant elles sont distinguées par $\mathsf{ATL}_{sc,\infty}$, d'où le premier résultat. Lorsque l'on considère des CGS à un joueur (c'est-à-dire des Structures de Kripke), alors notre logique $\mathsf{ATL}^{\star}_{sc,\infty}$ est équivalente à $\mathsf{CTL}^{\star}$, qui est elle-même strictement moins expressive que le μ-calcul classique. □

Comparaison avec $\text{Q}D_\mu$ [28]

La logique $\text{Q}D_\mu$ [28] est une extension du *μ-calcul décisionnel* par la possibilité de quantifier sur les stratégies. Dans ce cadre, on conçoit les stratégies comme des étiquetages de l'arbre d'exécution infini du système, au moyen de nouvelles propositions atomiques. Ceci permet d'envisager les stratégies comme des contextes. Il n'y a pas quantificateurs sur les stratégies à mémoire bornée dans $\text{Q}D_\mu$.

Cependant nous remarquons que $\text{Q}D_\mu$ est capable d'exprimer le fait qu'une stratégie est sans mémoire (en supposant qu'on puissse accéder aux noms des états de la CGS) : on y parvient en spécifiant que la stratégie étiquette toujours le même ensemble de successeurs depuis chaque copie d'un même état dans l'arbre d'exécution.

Nous poursuivrons cette comparaison, en nous intéressant cette fois à la complexité, au début de la section 5.5.

Comparaison avec IATL [1]

IATL est une autre extension de ATL avec des contextes, d'une manière proche de celle que nous avons présentée avec notre logique $\mathsf{ATL}_{sc,\infty}$, mais dans laquelle on n'a pas la possibilité de remplacer une stratégie par une autre : une fois qu'un joueur s'est vu associer une stratégie, il ne peut alors plus en envisager une autre. Il est facile à partir de cela de conclure que $\mathsf{ATL}_{sc,\infty}$ est strictement plus expressive que IATL.

Figure récapitulative

La Figure 4.6 illustre les principaux résultats d'expressivité que l'on a présentés dans cette section.

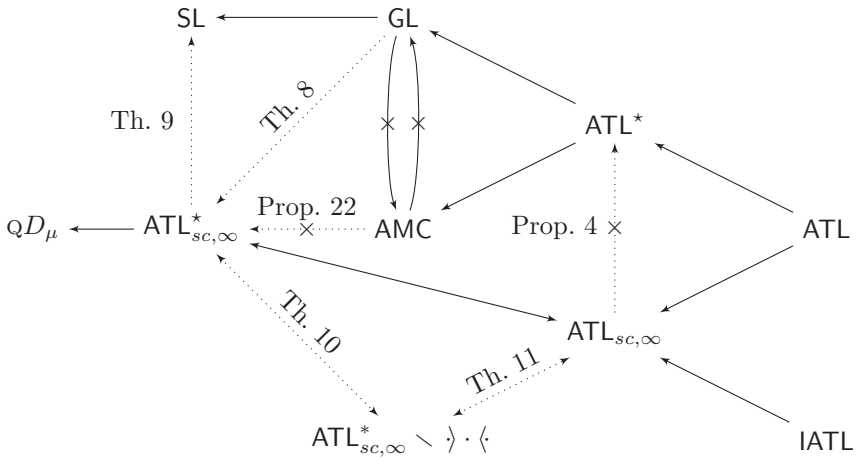

FIG. 4.6 – Expressivités comparées de $\mathsf{ATL}_{sc,\infty}$ et $\mathsf{ATL}^{\star}_{sc,\infty}$ avec les logiques classiques

Chapitre 5

Complexité du model-checking

La dernière partie de ce travail est consacrée aux problèmes de model-checking des différentes logiques que nous avons étudiées. Après un bref rappel de la notion de model-checking, nous montrons dans la section 5.2 que le model-checking de $\mathsf{ATL}_{sc,0}$ est PSPACE-dur, puis nous décrivons dans la section 5.3 un algorithme PSPACE de model-checking pour $\mathsf{ATL}^*_{sc,0}$. Nous en déduisons que ces deux logiques ont leur problème de model-checking qui est PSPACE-complet.

Nous abordons dans la section 5.4 le model-checking des logiques $\mathsf{ATL}^*_{sc,b}$, et en fournissons un algorithme EXPSPACE. La dernière section a trait aux logiques avec mémoire infinie : nous présentons un algorithme non élémentaire qui repose sur la construction d'automates d'arbre [22], après avoir rapidement rappelé leur définition. Ce travail s'achève sur une preuve de dureté non élémentaire du model-checking de $\mathsf{ATL}^*_{sc,\infty}$.

5.1 Le problème de model-checking

Définition 37. *Soit $\mathcal{L}$ une logique. Le problème de* model-checking sur les CGS *de $\mathcal{L}$, que l'on notera MC $\mathcal{L}$, est le problème de décision suivant :*

Etant donnés une CGS *$\mathcal{C}$, un état $q \in \mathsf{Loc}$ et une formule φ de $\mathcal{L}$, a-t-on $(\mathcal{C}, q) \models \varphi$?*

Dans le cas des logiques temporelles classiques, comme LTL ou CTL, on vérifie une formule sur une structure de Kripke plutôt que sur une CGS.

Théorème 12. *Les problèmes de model-checking suivants sont bien connus dans la littérature :*

- *MC LTL est PSPACE-complet [33]*
- *MC CTL est PTIME-complet [14]*
- *MC CTL^* est PSPACE-complet [14]*
- *MC ATL est PTIME-complet [3]*

– *MC ATL* est 2EXPTIME-complet [3]*

Les définitions des classes de complexité peuvent se retrouver dans [27].

Une conséquence du Théorème 11 est que toute formule d'état φ de ATL* peut être traduite dans ATL$_{sc,\infty}$ par la formule équivalente $\widehat{\varphi}^{\varnothing}$ en un temps polynomial. Ceci, au vu du Théorème 12, nous permet déjà de déduire que :

Corollaire 4. *Le model-checking de ATL$_{sc,\infty}$ est 2*EXPTIME*-dur.*

5.2 Dureté du model-checking de ATL$_{sc,0}$

On définit maintenant un problème classique de satisfaction d'une formule booléenne avec quantificateurs. Il s'agit du problème QBF – SAT (de l'anglais Quantified Boolean Formula Satisfiability).

Soient $x_1, ..., x_n$ des variables propositionnelles, et φ une formule propositionnelle sous forme normale conjonctive (on dira sous forme CNF), qui porte sur les variables $x_1, ..., x_n$.

Considérons la formule :

$$\theta = \exists x_1 \forall x_2 \exists x_3 ... Q_n x_n \varphi(x_1, ..., x_n),$$

où Q_n est un $\forall$ lorsque n est pair et un $\exists$ sinon.

Cette formule constitue une instance quelconque de QBF – SAT. On notera $\{0, 1\}$ le domaine décrit par les variables propositionnelles, 0 et 1 représentant respectivement "vrai" et "faux".

De plus, si $\varphi(x_1, x_2)$ est une formule booléenne quantifiée, avec x_1 et x_2 comme variables libres, on utilisera la notation suivante :

$$\Big(\{0, 1\}, x_1 \leftarrow 0, x_2 \leftarrow 1\Big) \models \varphi(x_1, x_2)$$

pour signifier que φ est vérifiée si l'on assigne les valeurs 0 à x_1 et 1 à x_2.

Définition 38. *Le problème standard* QBF – SAT*, ou QSAT correspond au problème de décision suivant :*

$$\textit{a-t-on } \{0, 1\} \models \theta \textit{ ?}$$

Le symbole $\models$ est ici celui de la logique du premier ordre.

Théorème 13. *Le problème* QBF – SAT *est* PSPACE*-dur [27].*

Cela va nous permettre d'établir le résultat suivant :

Théorème 14. *Le model-checking de ATL$_{sc,0}$ est* PSPACE*-dur.*

Preuve : Nous allons effectuer une réduction du problème QBF – SAT à MC ATL$_{sc,0}$.

Soit

$$\varphi = \bigwedge_{j=1,...,J} C_j$$

une formule de la logique propositionnelle sous forme CNF.

On associe à cette instance de $\mathsf{QBF-SAT}$ la $CGS\ \mathcal{C} = (\mathsf{Agt}, \mathsf{Loc}, \mathsf{AP}, \mathcal{M}, \mathsf{Lab}, \mathsf{Mov}, \mathsf{Edg})$ telle que :

1. $\mathsf{Agt} = \{A_1, ..., A_n\}$, où n est le nombre de variables propositionnelles distinctes qui constitue la formule φ ;
2. L'ensemble d'états Loc est une réunion de trois sous-ensembles d'états : à chaque *variable propositionnelle* x_i, on associe un *état* x_i, un état $\neg x_i$ et un état w_i. De plus, on ajoute un état final w_{n+1} :
$$\mathsf{Loc} = \{x_i \mid \ 1 \leq i \leq n\} \cup \{\neg x_i \mid \ 1 \leq i \leq n\} \cup \{w_i \mid \ 1 \leq i \leq n+1\} ;$$
3. L'ensemble des propositions atomiques AP est constitué d'une proposition pour chaque clause C_j ; pour simplifier les notations, on appelera également ces propositions C_j.
$$\mathsf{AP} = \{C_j \mid \ 1 \leq j \leq J\} ;$$
4. On définit deux mouvements différents : $\mathcal{M} = \{0, 1\}$;
5. On étiquette les états x_i par les propositions atomiques qui correspondent aux clauses dans lesquelles la variable x_i apparaît sous forme positive (c'est-à-dire $C_j = x_i \wedge \bigwedge ...$). De même, pour tout j, les états $\neg x_i$ que l'on étiquette par C_j sont ceux qui apparaissent sous forme négative dans la clause C_j (si $C_j = \neg x_i \wedge \bigwedge ...$). Les états w_i, quant à eux, ne sont pas étiquetés.
$$\mathsf{Lab}(x_i) = \{C_j \mid \ x_i \text{ apparaît dans la clause } C_j\},$$
$$\mathsf{Lab}(\neg x_i) = \{C_j \mid \ \neg x_i \text{ apparaît dans la clause } C_j\},$$
$$\mathsf{Lab}(w_i) = \varnothing.$$
6. La CGS est une structure *turn-based*, où les états w_i sont contrôlés par le joueur A_i lorsque $i \neq n+1$. Aucun autre état n'est contrôlé :
$$\forall 1 \leq i, k \leq n, \mathsf{Mov}(x_i, A_k) = \mathsf{Mov}(\neg x_i, A_k) = \{1\},$$
$$\forall 1 \leq i \leq n+1, \forall 1 \leq k \leq n, \mathsf{Mov}(w_i, A_k) = \begin{cases} \{1, 2\} & \text{si } i = k \\ \{1\} & \text{sinon} \end{cases}$$
7. Edg relie chaque x_i à w_{i+1}, chaque $\neg x_i$ à w_{i+1}, chaque w_i à la fois à $\neg x_i$ par le mouvement 1 et à x_i par le mouvement 2, et enfin w_{n+1} à lui-même (voir la Fig. 4.).

Voici une figure ; les vecteurs de mouvements ont été simplifiés pour ne pas alourdir le dessin. Nous n'avons pas fait figurer de propositions atomiques pour la même raison.

On définit par récurrence une suite finie $(\varphi_k)_{k \in \{0,..,n\}}$ de formules booléennes quantifiées telle que :

- $\varphi_0 = \varphi$,
- pour tout $k \in \{0, .., n-1\}$, $\varphi_{k+1} = \begin{cases} \forall x_{n-k}.\ \varphi_k & \text{si } n-k \text{ est pair} \\ \exists x_{n-k}.\ \varphi_k & \text{sinon} \end{cases}$

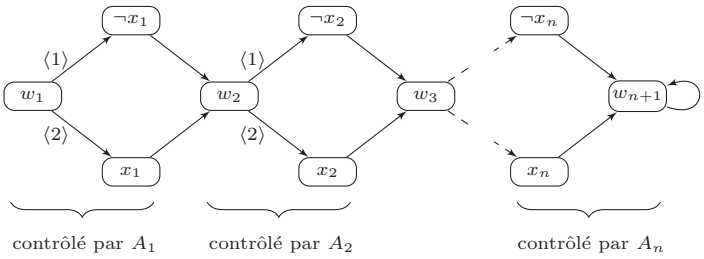

FIG. 5.1 – Une représentation simplifiée de la CGS $\mathcal{C}$

En particulier, on a $\varphi_n = \theta$.

A présent, nous allons définir par récurrence sur l'ensemble $\{\varphi_k \mid 0 \leq k \leq n\}$ des formules que l'on vient de construire, une application ν à valeurs dans les formules d'état de $\mathsf{ATL}_{sc,0}$:

- $\nu(\varphi_0) = \bigwedge_{j=1,...,J} \ll \varnothing \gg_0 \mathbf{F}\, C_j$,
- pour tout $k \in \{0, .., n-1\}$,

$$\nu(\varphi_{k+1}) = \begin{cases} \neg \ll A_{n-k} \gg_0 \neg \nu(\varphi_k) & \text{si } n-k \text{ est pair ,} \\ \ll A_{n-k} \gg_0 \nu(\varphi_k) & \text{sinon} \end{cases}$$

La CGS $\mathcal{C}$ et $\nu(\theta)$ sont constructibles avec un espace logarithmique en fonction de la taille de l'instance θ de $\mathsf{QBF} - \mathsf{SAT}$. Il nous reste à établir que cette instance du model-checking de $\mathsf{ATL}_{sc,0}$, avec état initial en w_1, donne une réponse positive ssi l'instance de $\mathsf{QBF} - \mathsf{SAT}$ est vraie, c'est-à-dire que :

$$(\mathcal{C}, w_1) \models \nu(\theta) \text{ ssi } \{0,1\} \models \theta.$$

Dans ce but, nous allons montrer un résultat plus général. Pour tout entier k compris entre 0 et n et pour toute stratégie positionnelle F de la coalition $\{A_1, ..., A_{n-k}\}$, on a :

$$\Big(\{0,1\}, x_1 \leftarrow F_{|A_1}(w_1), ..., x_{n-k} \leftarrow F_{|A_{n-k}}(w_{n-k})\Big) \models \varphi_k$$

$$\Leftrightarrow$$

$$(\mathcal{C}, w_1) \models_F \nu(\varphi_k),$$

On prouve cela par récurrence sur k :

1. Supposons $k = 0$. Soit F un contexte stratégique positionnel pour la coalition globale. Il s'agit de montrer que :

$$\Big(\{0,1\}, x_1 \leftarrow F_{|A_1}(w_1), ..., x_n \leftarrow F_{|A_n}(w_n)\Big) \models \bigwedge_{j=1,...,J} C_j$$

$$\Leftrightarrow$$

$$(\mathcal{C}, w_1) \models_F \bigwedge_{j=1,\dots,J} \lessdot \varnothing \gtrdot_0 \mathbf{F}\, C_j$$

Remarquons que l'ensemble $\mathsf{Out}(w_1, F)$ est un singleton, puisque tous les joueurs ont une stratégie fixée dans le contexte F. Nous noterons λ l'unique chemin de cet ensemble. Voici une description exhaustive de λ :

$$\lambda[i] = \begin{cases} w_{\frac{i}{2}+1} & \text{si } 0 \leq i \leq 2n-1 \text{ et } i \text{ est pair,} \\ \neg x_{\frac{i+1}{2}} & \text{si } 0 \leq i \leq 2n-1,\ i \text{ est impair et } F_{|A_{\frac{i+1}{2}}}(w_{\frac{i+1}{2}}) = 0, \\ x_{\frac{i+1}{2}} & \text{si } 0 \leq i \leq 2n-1,\ i \text{ est impair et } F_{|A_{\frac{i+1}{2}}}(w_{\frac{i+1}{2}}) = 1, \\ w_{n+1} & \text{si } i \geq 2n \end{cases}$$

Considérons maintenant une clause C_j fixée, pour un $j \in \{1, .., J\}$. On va montrer que :

$$\Big(\{0,1\}, x_1 \leftarrow F_{|A_1}(w_1), ..., x_n \leftarrow F_{|A_n}(w_n)\Big) \models C_j$$

$$\Leftrightarrow$$

$$(\mathcal{C}, \lambda) \models_F \mathbf{F}\, C_j$$

On effectue un raisonnement par équivalence :

$$\begin{array}{lll} (\mathcal{C}, \lambda) \models_F \mathbf{F}\, C_j & \Leftrightarrow & \exists i \geq 0, (\mathcal{C}, \lambda[i, \infty]) \models_F C_j \\ & \Leftrightarrow & \exists i \geq 0, C_j \in \mathsf{Lab}(\lambda[i]) \\ & \Leftrightarrow & \exists i \geq 0.\ \lambda[i] \text{ apparaît dans la clause } C_j \\ & \Leftrightarrow & \exists i \geq 0, \exists l \in \{1, ..., L\}, \lambda[i] = \epsilon_l x_l \\ & & \text{(on pose } C_j = \bigvee_{l=1,\dots,L} \epsilon_l x_l, \text{ avec } \epsilon_l = \neg \text{ ou rien)} \\ & \Leftrightarrow & \exists i \geq 0, \exists l \in \{1, ..., L\}, \lambda[i] = x_l \\ & & \text{en supposant que } \epsilon_l = \varnothing, \text{ le cas } \epsilon_l = \neg \\ & & \text{se traitant de la même façon} \\ & \Leftrightarrow & \exists l \in \{1, ..., L\}, F_{A_l}(w_l) = 1 \\ & & \text{d'après ce que l'on sait sur } \lambda \\ & \Leftrightarrow & \exists l \in \{1, ..., L\}, \\ & & \qquad \Big(\{0,1\}, x_l \leftarrow F_{|A_l}(w_l)\Big) \models x_l \\ & \Leftrightarrow & \Big(\{0,1\}, x_1 \leftarrow F_{|A_1}(w_1), ..., \\ & & \qquad x_n \leftarrow F_{|A_n}(w_n)\Big) \models C_j \end{array}$$

2. Supposons maintenant que la propriété est vérifiée au rang $k-1 \in \{0, .., n-1\}$. Pour établir l'hérédité, nous allons distinguer deux cas selon que $n-k$

est pair ou non. Supposons $n-k$ impair. On est alors dans le cas où $\varphi_k = \forall x_{n-k-1}.\ \varphi_{k-1}$. Soit F un contexte stratégique positionnel pour la coalition $\{A_1, ..., A_{n-k}\}$. On doit montrer :

$$\Big(\{0,1\}, x_1 \leftarrow F_{|A_1}(w_1), ..., x_{n-k} \leftarrow F_{|A_{n-k}}(w_{n-k})\Big) \models \varphi_k$$

$$\Leftrightarrow$$

$$(\mathcal{C}, w_1) \models_F \nu(\varphi_k).$$

On procède une nouvelle fois par équivalence :

$$(\mathcal{C}, w_1) \models_F \nu(\varphi_k)$$

$$\Leftrightarrow$$

$$(\mathcal{C}, w_1) \models_F \neg \lessdot A_{n-k+1} \gtrdot_0 \neg\nu(\varphi_{k-1})$$

$$\Leftrightarrow$$

$$\forall f \in \mathsf{Strat}^0_{A_{n-k+1}}.\ (\mathcal{C}, w_1) \models_{f \circ F} \nu(\varphi_{k-1})$$
car $\nu(\varphi_{k-1})$ est une formule d'état de $\mathsf{ATL}_{sc,0}$

$$\Leftrightarrow$$

$$\forall f \in \mathsf{Strat}^0_{A_{n-k+1}}.$$
$$\Big(\{0,1\}, x_1 \leftarrow F_{|A_1}(w_1), ..., x_{n-k+1} \leftarrow f(w_{n-k+1})\Big) \models \varphi_{k-1}$$
par hypothèse de récurrence

$$\Leftrightarrow$$

$$\forall x \in \{0,1\}.\ \Big(\{0,1\}, x_1 \leftarrow F_{|A_1}(w_1), ..., x_{n-k+1} \leftarrow x\Big) \models \varphi_{k-1}$$
car $\mathsf{Strat}^0_{A_{n-k+1}} = \{f_1, f_2\}$, avec $f_1(w_{n-k-1}) = 0$ et $f_2(w_{n-k-1}) = 1$

$$\Leftrightarrow$$

$$\Big(\{0,1\}, x_1 \leftarrow F_{|A_1}(w_1), ..., x_{n-k-1} \leftarrow F_{|A_{n-k-1}}(w_{n-k-1})\Big) \models \varphi_k$$

Le cas où $n-k$ est impair s'appuie sur le même raisonnement.

La récurrence est achevée, et cette propriété appliquée au rang n, c'est-à-dire au cas où F est le contexte vide, constitue la réduction annoncée entre les deux problèmes. □

On va voir que ce problème est en fait PSPACE-complet : ce sera une conséquence du résultat suivant concernant $\mathsf{ATL}^*_{sc,0}$

5.3 Complexité du problème MC $\mathbf{ATL}^*_{sc,0}$

Nous nous attaquons maintenant au problème MC $\mathsf{ATL}^*_{sc,0}$. Pour cela, l'idée essentielle est de remarquer que les issues d'une stratégie sans mémoire correspondent exactement aux exécutions d'une *CGS* particulière (c'est ce qui motive la définition et le lemme suivants), puis de recourir au model-checking de LTL. La difficulté principale est alors la préservation des contextes imposés par les stratégies ; elle est résolue en rajoutant un nombre suffisant de propositions atomiques.

Soient $\mathcal{C} = (\mathsf{Agt}, \mathsf{Loc}, \mathsf{AP}, \mathcal{M}, \mathsf{Lab}, \mathsf{Mov}, \mathsf{Edg})$ une *CGS*, et F un contexte stratégique positionnel pour une coalition $A \subseteq \mathsf{Agt}$. Comme d'habitude, k représente le nombre total de joueurs de la structure.

Définition 39. *On définit* $(\mathcal{C}, F) = (\mathsf{Agt}, \mathsf{Loc}, \mathsf{AP}, \mathcal{M}, \mathsf{Lab}, \mathsf{Mov}', \mathsf{Edg}')$*, la sous-structure de* $\mathcal{C}$ induite *par le contexte stratégique sans mémoire* F*, de la manière suivante :*

1. *pour tout joueur* $A_i \in \mathsf{Agt}$ *et tout état* q*,*

$$\mathsf{Mov}'(q, A_i) = \begin{cases} \{F_{|A_i}(q)\} & \text{si } A_i \in A \\ \mathsf{Mov}(q, A_i) & \text{sinon} \end{cases}$$

2. *La nouvelle fonction de transition* Edg' *est simplement la restriction de* Edg *aux mouvements licites, c'est-à-dire que pour tout état* q *et pour tout vecteur complet de mouvements* m *:*

$$\mathsf{Edg}'(q, m) = \mathsf{Edg}(q, m)$$

De plus, $\mathsf{Edg}'(q, m)$ *n'est défini que si*

$$m \in \prod_{A_i \in \mathsf{Agt}} \mathsf{Mov}'(s, A_i)$$

Remarques :
- Dans le cas particulier où F est le contexte stratégique vide, on a $\mathsf{Mov}' = \mathsf{Mov}$ et $(\mathcal{C}, F) = \mathcal{C}$.
- Intuitivement, cette opération consiste à élaguer la structure en supprimant, pour les joueurs concernés par le contexte stratégique F, tous leurs mouvements qui n'apparaissent pas dans F.
- Lorsque l'on effectue cette opération, on peut couper des transitions et il peut même arriver que l'on isole des états.

Lemme 23. *Pour tout état* s *d'une* CGS $\mathcal{C}$*, et pour tout contexte stratégique positionnel* F *d'une coalition quelconque* $A \subseteq \mathsf{Agt}$*, il est vrai que :*

$$\mathsf{Out}_{\mathcal{C}}(s, F) = \mathsf{Exec}_{(\mathcal{C},F)}(s)$$

Preuve : On se donne une structure $\mathcal{C} = (\mathsf{Agt}, \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \mathsf{Mov}, \mathsf{Edg})$, un état s, une coalition A, et F un contexte stratégique sans mémoire pour cette coalition.

Pour tout état q de Loc, on a :

$$\mathsf{Next}_{\mathcal{C}}\big(q, A, F(q)\big) = \mathsf{Next}_{(\mathcal{C},F)}(q)$$

En effet, les vecteurs de mouvements qui mènent d'un état à un autre dans $(\mathcal{C}, F)$ sont exactement ceux qui relient les mêmes états dans $\mathcal{C}$, et qui vérifient en plus que chacune de leur composante sur un joueur A_i de A vaut $F_{|A_i}(q)$.

A présent, on raisonne par équivalence :

$$\begin{array}{lcl}
\lambda \in \mathsf{Out}_{\mathcal{C}}(s, F) & \Leftrightarrow & \lambda[0] = s \text{ et } \forall k \in \mathbb{N}.\ \lambda[k+1] \in \mathsf{Next}_{\mathcal{C}}\big(\lambda[k], A, F(\lambda[k])\big) \\
& \Leftrightarrow & \lambda[0] = s \text{ et } \forall k \in \mathbb{N}.\ \lambda[k+1] \in \mathsf{Next}_{(\mathcal{C},F)}(\lambda[k]) \\
& \Leftrightarrow & \lambda \in \mathsf{Exec}_{(\mathcal{C},F)}(s)
\end{array}$$

□

On définit le *rang de quantification* d'une formule φ vue comme un arbre, que l'on note $rq(\varphi)$, comme étant le nombre maximum de quantificateurs de stratégie contenus dans une branche.

Théorème 15. *Le model-checking de* $\mathsf{ATL}^*_{sc,0}$ *est dans* PSPACE.

Preuve : On définit par induction sur $\mathsf{ATL}^*_{sc,0,p}$ (qui inclut les formules d'état et de chemin de $\mathsf{ATL}^*_{sc,0}$) l'application σ à valeurs dans LTL (avec toutefois plus de propositions atomiques) telle que :

- Si $\varphi_p = a \in \mathsf{AP}$, alors $\sigma(\varphi_p) = a$,
- Si $\varphi_p = \neg\psi$, alors $\sigma(\varphi_p) = \neg\sigma(\psi)$,
- Si $\varphi_p = \psi_1 \vee \psi_2$, alors $\sigma(\varphi_p) = \sigma(\psi_1) \vee \sigma(\psi_2)$,
- Si $\varphi_p = \langle\!\langle B \rangle\!\rangle_0\ \psi$, alors $\sigma(\varphi_p) = a_{\varphi_p}$,
- Si $\varphi_p = [\![B]\!]\, \psi$, alors $\sigma(\varphi_p) = a_{\varphi_p}$,
- Si $\varphi_p = \mathbf{X}\,\psi$, alors $\sigma(\varphi_p) = \mathbf{X}\,\sigma(\psi)$,
- Si $\varphi_p = \psi_1\,\mathbf{U}\,\psi_2$, alors $\sigma(\varphi_p) = \sigma(\psi_1)\,\mathbf{U}\,\sigma(\psi_2)$.

On va montrer que l'algorithme 1 retourne OUI si et seulement si :

$$\forall \lambda \in \mathsf{Out}_{\mathcal{C}}(s_0, F), (\mathcal{C}, \lambda) \models_F \theta_p$$

On désigne par $\mathcal{M}_{f,(\mathcal{C},F,\theta_p)}$ la structure contenue dans la variable $\mathcal{M}$ en fin d'exécution de l'algorithme 1 sur l'entrée $(\mathcal{C}, F, s_0, \theta_p)$. Cette notation est justifiée par le fait que la valeur de s_0 n'importe pas. Ici, θ_p désigne une formule de $\mathsf{ATL}^*_{sc,0,p}$.

Il est clair que pour toute entrée $(\mathcal{C}, F, s_0, \theta_p)$ de l'algorithme 1, les structures $(\mathcal{C}, F)$ et $\mathcal{M}_{f,(\mathcal{C},F,\theta_p)}$ ont les mêmes états et les mêmes transitions (elles ne diffèrent que par leurs étiquetages), donc pour tout état s de Loc,

$$\mathsf{Exec}_{(\mathcal{C},F)}(s) = \mathsf{Exec}_{\mathcal{M}_{f,(\mathcal{C},F,\theta_p)}}(s)$$

Soit $\mathcal{C}$ une *CGS* fixée. On va montrer par induction sur la structure de $\mathsf{ATL}^*_{sc,0,p}$ que pour toute formule $\varphi_p \in \mathsf{ATL}^*_{sc,0,p}$, pour tout contexte stratégique

Algorithm 1 MC $\mathsf{ATL}^*_{sc,0}$

Require: une *CGS* $\mathcal{C}$, un contexte stratégique sans mémoire F pour une coalition A, un état s_0 et une formule θ_p de $\mathsf{ATL}^*_{sc,0,p}$
Ensure: OUI ssi $\forall \lambda \in \mathsf{Out}_{\mathcal{C}}(s_0, F), (\mathcal{C}, \lambda) \models_F \theta_p$

```
M := (C, F)
Φ1 := { ≪B≫0 ψ ∈ Sub(θp) | rq(≪B≫0 ψ) = rq(θp) }
Φ2 := { ⟩B⟨ ψ ∈ Sub(θp) | rq(⟩B⟨ ψ) = rq(θp) }
for ≪B≫0 ψ ∈ Φ1 do
  AP_M := AP_M ∪ {a_{≪B≫0 ψ}}
  for s ∈ Loc do
    for F' ∈ Strat^0_B do
      if MC ATL*_{sc,0}(C, F' ∘ F, s, ψ) then
        Lab_M(s) := Lab_M(s) ∪ {a_{≪B≫0 ψ}}
      end if
    end for
  end for
end for
for ⟩B⟨ ψ ∈ Φ2 do
  AP_M := AP_M ∪ {a_{⟩B⟨ ψ}}
  for s ∈ Loc do
    if MC ATL*_{sc,0}(C, F|_{A\B}, s, ψ) then
      Lab_M(s) := Lab_M(s) ∪ {a_{⟩B⟨ ψ}}
    end if
  end for
end for
return MC LTL (M, s0, σ(θp))
```

positionnel F d'une coalition A et pour toute exécution λ de la structure $(\mathcal{C}, F)$ depuis un état s quelconque, on a :

$$(\mathcal{C}, \lambda) \models_F \varphi_p \Leftrightarrow (\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)}, \lambda) \models \sigma(\varphi_p)$$

Nous ne développerons pas ici les cas faciles où $\varphi_p = \neg\psi$ et $\varphi_p = \psi_1 \wedge \psi_2$.

1. $\varphi_p = a \in \mathsf{AP}$;
   Il s'agit de montrer que :

   $$a \in \mathsf{Lab}_{\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)}}(s) \Leftrightarrow a \in \mathsf{Lab}_{\mathcal{C}}(s)$$

   Cela est vrai, puisque la transformation de $\mathcal{C}$ en $\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)}$ ne modifie pas les étiquetages par des propositions atomiques de $\mathcal{C}$.
   Dans chacun des cas suivants, on commencera par se fixer une coalition A, un contexte stratégique positionnel F pour cette coalition, ainsi qu'une exécution qui part d'un état s de la structure $(\mathcal{C}, F)$.
2. $\varphi_p = \ll B \gg_0 \psi$;
   On a la suite d'équivalences :

   $$\begin{array}{lll} (\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)}, \lambda) \models \sigma(\varphi_p) & \Leftrightarrow & (\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)}, \lambda) \models a_{\varphi_p} \\ & \Leftrightarrow & a_{\varphi_p} \in \mathsf{Lab}_{\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)}}(s) \end{array}$$

   Or cet étiquetage n'est possible que si l'on a pu trouver une stratégie positionnelle F' pour la coalition B, telle que l'algorithme 1 retourne OUI sur l'entrée $(\mathcal{C}, F' \circ F, s, \psi)$, c'est-à-dire que :

   $$\begin{array}{lll} a_{\varphi_p} \in \mathsf{Lab}_{\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)}}(s) & \Leftrightarrow & \exists F' \in \mathsf{Strat}^0_B. \\ & & \qquad \text{MC } \mathsf{LTL}\big(\mathcal{M}_{f,(\mathcal{C},F'\circ F,\psi)}, s, \sigma(\psi)\big) \\ & \Leftrightarrow & \exists F' \in \mathsf{Strat}^0_B.\ \forall \lambda' \in \mathsf{Exec}_{(\mathcal{C},F'\circ F)}(s). \\ & & \qquad (\mathcal{M}_{f,(\mathcal{C},F'\circ F,\psi)}, \lambda') \models \sigma(\psi) \\ & \Leftrightarrow & \exists F' \in \mathsf{Strat}^0_B.\ \forall \lambda' \in \mathsf{Exec}_{(\mathcal{C},F'\circ F)}(s). \\ & & \qquad (\mathcal{C}, \lambda') \models_{F'\circ F} \psi \\ & & \text{(par HI)} \\ & \Leftrightarrow & (\mathcal{C}, s) \models_F \ll B \gg_0 \psi \\ & & \text{d'après le lemme 23} \\ & \Leftrightarrow & (\mathcal{C}, \lambda) \models_F \varphi_p \end{array}$$

3. $\varphi_p = \rangle B \langle \psi$;
   On raisonne une nouvelle fois par équivalence :

$$
\begin{array}{lll}
(\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)},\lambda) \models \sigma(\varphi_p) & \Leftrightarrow & a_{\rangle B \langle\, \psi} \in \mathsf{Lab}_{\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)}}(s) \\
& \Leftrightarrow & \text{MC } \mathsf{LTL}\big(\mathcal{M}_{f,(\mathcal{C},F|_{A\setminus B},\psi)}, s, \sigma(\psi)\big) \\
& \Leftrightarrow & \forall \lambda' \in \mathsf{Exec}_{(\mathcal{C},F|_{A\setminus B})}(s). \\
& & \qquad (\mathcal{M}_{f,(\mathcal{C},F|_{A\setminus B},\psi)}, \lambda') \models \sigma(\psi) \\
& \Leftrightarrow & \forall \lambda' \in \mathsf{Exec}_{(\mathcal{C},F|_{A\setminus B})}(s). \\
& & \qquad\qquad (\mathcal{C}, \lambda') \models_{F|_{A\setminus B}} \psi \\
& & \text{par hypothèse d'induction} \\
& \Leftrightarrow & (\mathcal{C}, s) \models_F \rangle B \langle\, \psi \\
& & \text{par le lemme 23} \\
& \Leftrightarrow & (\mathcal{C}, \lambda) \models_F \varphi_p
\end{array}
$$

4. $\varphi_p = \mathbf{X}\,\psi$;
   Alors $(\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)},\lambda) \models \sigma(\varphi_p)$ signifie que $(\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)},\lambda[1,\infty]) \models \sigma(\psi)$. Comme les structures $\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)}$ et $\mathcal{M}_{f,(\mathcal{C},F,\psi)}$ sont identiques, c'est équivalent à $(\mathcal{M}_{f,(\mathcal{C},F,\psi)},\lambda[1,\infty]) \models \sigma(\psi)$, ou, en appliquant l'hypothèse d'induction, à $(\mathcal{C},\lambda[1,\infty]) \models_F \psi$. Cette dernière expression s'écrit aussi : $(\mathcal{C},\lambda) \models_F \varphi_p$.
5. $\varphi_p = \psi_1\,\mathbf{U}\,\psi_2$;
   On a la suite d'équivalences :
   $(\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)},\lambda) \models \sigma(\varphi_p)$ ssi
   $(\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)},\lambda) \models \sigma(\psi_1)\,\mathbf{U}\,\sigma(\psi_2)$ ssi
   $\exists i.\ (\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)},\lambda[i,\infty]) \models \sigma(\psi_2)$ et $\forall j < i.\ (\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)},\lambda[j,\infty]) \models \sigma(\psi_1)$ ssi, puisque $\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)} = \mathcal{M}_{f,(\mathcal{C},F,\psi_1)} = \mathcal{M}_{f,(\mathcal{C},F,\psi_2)}$
   $\exists i.\ (\mathcal{M}_{f,(\mathcal{C},F,\psi_2)},\lambda[i,\infty]) \models \sigma(\psi_2)$ et $\forall j < i.\ (\mathcal{M}_{f,(\mathcal{C},F,\psi_1)},\lambda[j,\infty]) \models \sigma(\psi_1)$ ssi, par HI
   $\exists i.\ (\mathcal{C},\lambda[i,\infty]) \models_F \psi_2$ et $\forall j < i.\ (\mathcal{C},\lambda[j,\infty]) \models_F \psi_1$ ssi
   $(\mathcal{C},\lambda) \models_F \varphi_p$

On sait que l'algorithme 1 retourne OUI sur l'entrée $(\mathcal{C}, F, s_0, \varphi_p)$ ssi :

$$(\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)}, s_0) \models \sigma(\varphi_p),$$

au sens de LTL.

La propriété que l'on vient d'établir, le lemme 23 et le fait que $(\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)})$ et $(\mathcal{C}, F)$ aient les mêmes exécutions permettent de prouver que l'algorithme 1 retourne OUI sur l'entrée $(\mathcal{C}, F, s_0, \varphi_p)$ ssi

$$\forall \lambda \in \mathsf{Out}_{\mathcal{C}}(s_0, F), (\mathcal{C}, \lambda) \models_F \varphi_p,$$

comme on l'avait annoncé.

Cet algorithme est bien un algorithme de model-checking pour $\mathsf{ATL}^*_{sc,0}$: étant donnés une *CGS* $\mathcal{C}$, un état s_0 de $\mathcal{C}$ et une formule φ de $\mathsf{ATL}^*_{sc,0}$, si l'on veut savoir si $(\mathcal{C}, s_0) \models \varphi$, il suffit de lancer l'algorithme 1 sur l'entrée $(\mathcal{C}, \varnothing, s_0, \varphi)$. En effet, il retournera OUI ssi $\forall \lambda \in \mathsf{Out}_{\mathcal{C}}(s_0, \varnothing), (\mathcal{C}, \lambda) \models \varphi$, c'est-à-dire ssi $(\mathcal{C}, s_0) \models \varphi$ lorsque φ est une fornule d'état de $\mathsf{ATL}^*_{sc,0}$.

Nous allons à présent montrer que l'algorithme 1 est dans PSPACE. Soit une *CGS* $\mathcal{C}$. On va montrer par récurrence sur le rang de quantification que pour toute formule $\varphi_p \in \mathsf{ATL}^*_{sc,0,p}$, on a :

Pour tout contexte stratégique positionnel F d'une coalition A et tout état état s_0, l'algorithme 1 utilise un espace polynomial en fonction de la taille de l'entrée $(\mathcal{C}, F, s_0, \varphi_p)$:

Si $rq(\varphi_p) = 0$, alors l'algorithme 1 effectue seulement les affectations initiales qui se font facilement et l'opération MC LTL $((\mathcal{C}, F), s_0, \varphi_p)$, dont on sait qu'elle utilise un espace polynomial en fonction de son entrée.

Si $rq(\varphi_p) = n+1$. La taille de $(\mathcal{C}, F)$ est bornée par celle de $\mathcal{C}$. On ajoute un nombre de propositions atomiques borné par le nombre de sous-formules de φ_p à $(\mathcal{C}, F)$ pour obtenir la structure $\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)}$. Donc le contenu de la variable $\mathcal{M}$ est toujours de longueur polynomiale en fonction de celle de l'entrée.

Les opérations d'assignations sont très faciles : restreindre une structure selon une stratégie, parcourir un arbre de formule et ajouter un élément à un ensemble donné, sont des actions pouvant aisément être effectuées dans PSPACE.

Les énumérations portent toutes sur des objets qui ont une taille polynomiales en fonction de celle de l'entrée : sous-formules de φ_p, stratégies sans mémoire d'une structure avec autant d'états que $\mathcal{C}$, et états de $\mathcal{C}$. On peut donc les énumérer sans problème dans PSPACE.

Enfin, les tests peuvent être effectués dans PSPACE : par HR, les test MC $\mathsf{ATL}^*_{sc,0}(\mathcal{C}, F' \circ F, s, \psi)$ et MC $\mathsf{ATL}^*_{sc,0}(\mathcal{C}, F|_{A \setminus B}, s, \psi)$ prennent un espace polynomial en fonction de la taille de leur entrée, qui est elle-même plus petite que $(\mathcal{C}, F, s_0, \varphi_p)$. Enfin, on sait que le test MC LTL $(\mathcal{M}, s_0, \mathbf{A}\sigma(\varphi_p))$ prend un espace polynomial en fonction de la taille de son entrée, qui est elle-même polynomiale en fonction de celle de $(\mathcal{C}, F, s_0, \varphi_p)$. □

Nous obtenons comme corollaire de ces théorèmes que :

Corollaire 5. *Les problèmes :*
- *MC* $\mathsf{ATL}_{sc,0}$
- *MC* $\mathsf{ATL}^*_{sc,0}$
- *MC* $\mathsf{ATL}^*_{sc,0} \setminus \rangle \cdot \langle$

sont PSPACE*-complets.*

5.4 Model-checking des logiques $\mathsf{ATL}^*_{sc,b}$

On va maintenant s'attaquer au problème MC $\mathsf{ATL}^*_{sc,b}$. Pour cela, l'idée essentielle est de modéliser la mémoire de chaque joueur par une *CGS* à $b+1$ éléments entièrement contrôlée par ce joueur, et qui possède toutes les transitions. On va ensuite le faire jouer à la fois sur la *CGS* donnée en entrée, et celle

modélisant sa mémoire, puis recourir au model-checking de LTL. La méthode pour préserver des contextes imposés par les stratégies a déjà été employée dans le model-checking de la logique $\mathsf{ATL}^*_{sc,0}$.

Définition 40. *Soient deux CGS* $\mathcal{C}_1 = (\mathsf{Agt}, \mathsf{Loc}_1, \mathsf{AP}_1, \mathcal{M}_1, \mathsf{Lab}_1, \mathsf{Mov}_1, \mathsf{Edg}_1)$ *et* $\mathcal{C}_2 = (\mathsf{Agt}, \mathsf{Loc}_2, \mathsf{AP}_2, \mathcal{M}_2, \mathsf{Lab}_2, \mathsf{Mov}_2, \mathsf{Edg}_2)$*. On notera* k *le nombre total de joueurs, et* $A_1, ..., A_k$ *chacun des joueurs, qui sont supposés communs aux deux structures.*

On définit de la manière suivante le produit des deux structures $\mathcal{C}_1$ *et* $\mathcal{C}_2$*, désigné par* $\mathcal{C}_1 \times \mathcal{C}_2 = (\mathsf{Agt}, \mathsf{Loc}, \mathsf{AP}, \mathcal{M}, \mathsf{Lab}, \mathsf{Mov}, \mathsf{Edg})$ *:*

1. *Le nouvel ensemble d'états* Loc *est le produit des ensembles d'états* Loc_1 *et* Loc_2 *:*
$$\mathsf{Loc} = \mathsf{Loc}_1 \times \mathsf{Loc}_2$$
2. *L'ensemble de propositions atomiques* AP *est la réunion des ensembles* AP_1 *et* AP_2 *:*
$$\mathsf{AP} = \mathsf{AP}_1 \cup \mathsf{AP}_2$$
3. *Tout mouvement de la structure produit est le produit d'un mouvement de* $\mathcal{M}_1$ *et d' un mouvement de* $\mathcal{M}_2$ *:*
$$\mathcal{M} = \mathcal{M}_1 \times \mathcal{M}_2$$
4. *Chaque état* (q_1, q_2) *de la structure produit est étiqueté par les étiquettes de* q_1 *et ceux de* q_2 *:*
$$\mathsf{Lab}(q_1, q_2) = \mathsf{Lab}_1(q_1) \cup \mathsf{Lab}_2(q_2)$$
5. *Pour tout état* $q = (q_1, q_2)$ *de* Loc *et pour tout joueur* A_i*, les mouvements de* A_i *depuis* q *sont les couples composés d'un mouvement de* A_i *depuis* q_1 *(c'est-à-dire dans la structure* $\mathcal{C}_1$*) et d'un mouvement de* A_i *depuis* q_2 *(dans la structure* $\mathcal{C}_2$*) :*
$$\mathsf{Mov}(q, A_i) = \mathsf{Mov}_1(q_1, A_i) \times \mathsf{Mov}_2(q_2, A_i)$$
6. *Supposons qu'un vecteur de mouvements global* m_1 *relie l'état* q_1 *à* s_1 *dans la structure* $\mathcal{C}_1$*, et que* m_2 *relie* q_2 *à* s_2 *dans* $\mathcal{C}_2$*. Alors on définit la nouvelle fonction de transition* Edg *de sorte que le mouvement* $m = (m_1, m_2)$ *de la structure-produit amène de l'état* (q_1, q_2) *vers* (s_1, s_2) *:*
$$\mathsf{Edg}\Big((q_1, q_2), \big((m_{1|A_1}, m_{2|A_1}), ..., (m_{1|A_k}, m_{2|A_k})\big)\Big) =$$
$$\Big(\mathsf{Edg}_1(q_1, m_{1|A_1}, ..., m_{1|A_k}), \mathsf{Edg}_2(q_2, m_{2|A_1}, ..., m_{2|A_k})\Big)$$

On généralise trivialement cette définition pour parvenir à la notion de produit d'une famille finie. On convient qu'un produit indexé par une famille vide de structures, est la structure qui ne contient qu'un état, non étiqueté et relié à lui-même.

Remarque :
- Pour pouvoir effectuer le produit de deux structures, il est nécessaire que celles-ci fassent intervenir les mêmes joueurs. Intuitivement, faire jouer un joueur sur le produit de deux structures, c'est le faire jouer sur les deux structures à la fois.
- Si q désigne un état de la structure-produit $\mathcal{C}_1 \times \mathcal{C}_2$, on notera $q_{|\mathcal{C}_1}$ et $q_{|\mathcal{C}_2}$ ses projetés respectifs sur les états de la structure $\mathcal{C}_1$ et ceux de la structure $\mathcal{C}_2$.
- De même, si m désigne un mouvement du joueur A_i sur la structure-produit $\mathcal{C}_1 \times \mathcal{C}_2$, on notera $m_{|\mathcal{C}_1}$ et $m_{|\mathcal{C}_2}$ ses projetés respectifs sur les mouvements de la structure $\mathcal{C}_1$ et ceux de la structure $\mathcal{C}_2$.
- Enfin, si m désigne un vecteur de mouvements d'une coalition quelconque A sur la structure-produit $\mathcal{C}_1 \times \mathcal{C}_2$, on notera :

$$m_{|\mathcal{C}_1} = \Big((m_{|A_i})_{|\mathcal{C}_1}\Big)_{A_i \in A} \qquad \text{et} \qquad m_{|\mathcal{C}_2} = \Big((m_{|A_i})_{|\mathcal{C}_2}\Big)_{A_i \in A}$$

Le lemme qui suit a pour objet de caractériser le Next d'une structure-produit.

Lemme 24. *Dans une structure-produit $\mathcal{C}_1 \times \mathcal{C}_2$, si q est un état et m un vecteur de mouvement d'une coalition A quelconque, on a :*

$$\mathsf{Next}_{\mathcal{C}_1 \times \mathcal{C}_2}(q, A, m) = \mathsf{Next}_{\mathcal{C}_1}(q_{|\mathcal{C}_1}, A, m_{|\mathcal{C}_1}) \times \mathsf{Next}_{\mathcal{C}_2}(q_{|\mathcal{C}_2}, A, m_{|\mathcal{C}_2})$$

En particulier, si m est un vecteur de mouvements global :

$$\mathsf{Edg}_{\mathcal{C}_1 \times \mathcal{C}_2}(q, m) = \mathsf{Edg}_{\mathcal{C}_1}(q_{|\mathcal{C}_1}, m_{|\mathcal{C}_1}) \times \mathsf{Edg}_{\mathcal{C}_2}(q_{|\mathcal{C}_2}, m_{|\mathcal{C}_2})$$

Nous allons maintenant modéliser la cellule mémoire d'un joueur $A_i \in \mathsf{Agt}$. Pour cela, nous introduisons une structure de Kripke $\mathcal{M}_{A_i,b}$, complète avec $b+1$ états, et qui est entièrement contrôlée par le joueur A_i.

Définition 41. *Soit $\mathcal{C} = (\mathsf{Agt}, \mathsf{Loc}, \mathsf{AP}, \mathcal{M}, \mathsf{Lab}, \mathsf{Mov}, \mathsf{Edg})$ une CGS fixée. Pour tout joueur A_i et tout entier b, on définit la structure $\mathcal{M}_{A_i,b}$, dont les composantes $(\mathsf{Agt}, \mathsf{Loc}', \mathsf{AP}', \mathcal{M}', \mathsf{Lab}', \mathsf{Mov}', \mathsf{Edg}')$ sont telles que :*

1. *La structure contient $b+1$ états. Pour simplifier les notations, ceux-ci seront simplement notés de 0 à b :*

$$\mathsf{Loc}' = \{0, ..., b\}$$

2. *Il n'y a pas de propositions atomiques : $\mathsf{AP}' = \varnothing$; la fonction d'étiquetage est donc la fonction vide.*
3. *On se donne $b+1$ mouvements désignés par les entiers $0, ..., b$, ainsi qu'un mouvement noté m' :*

$$\mathcal{M}' = \{0, ..., b\} \cup \{m'\}$$

4. *Le joueur A_i contrôle tous les états de $\mathcal{M}_{A_i,b}$. Pour tout état s de Loc' et tout joueur P de Agt :*

$$\mathsf{Mov}'(s,P) = \begin{cases} \{0,...,b\} & si\ P = A_i \\ \{m'\} & sinon \end{cases}$$

5. *Pour tout état s et tout vecteur global de mouvements m, on pose :*

$$\mathsf{Edg}'(s,m) = m_{|A_i}$$

Remarques :
- Soit un joueur A_j, un entier naturel b, un état q de la structure $\mathcal{M}_{A_j,b}$ et un vecteur de mouvement m pour une coalition fixée A. On a alors $\mathsf{Next}_{\mathcal{M}_{A_j,b}}(q,A,m) = \begin{cases} \{m_{|A_j}\} & \text{si } A_j \in A \\ \mathsf{Loc}_{\mathcal{M}_{A_j,b}} & \text{sinon} \end{cases}$
- ici tous les joueurs jouent avec la même quantité de mémoire mais on pourrait facilement autoriser des valeurs différentes pour les joueurs.

Dans la suite, on notera :

$$\mathcal{C}_b = \Big(\prod_{A_i \in \mathsf{Agt}} \mathcal{M}_{A_i,b} \Big) \times \mathcal{C}$$

Soit m un vecteur de mouvement de $\mathcal{C}_b$ pour une coalition A quelconque. Si les joueurs $A_i \in A$ et A_j sont distincts, alors $(m_{|A_j})_{|\mathcal{M}_{A_i,b}} = m'$. Le mouvement $(m_{|A_i})_{|\mathcal{M}_{A_i,b}}$ est donc le seul intéressant. Dans la suite, pour tout joueur A_i de la coalition A, on notera :

$$(m_{|A_i})_{|\mathcal{M}_{A_i,b}} = m_{A_i}$$

Lemme 25. *Soit $\mathcal{C}$ une CGS et b un entier naturel fixés.*

Pour tout état q de la structure $\mathcal{C}_b$ et pour tout vecteur de mouvements m d'une coalition quelconque $A \subseteq \mathsf{Agt}$, on a $q' \in \mathsf{Next}_{\mathcal{C}_b}(q,A,m)$ ssi :
- $q'_{|\mathcal{C}} \in \mathsf{Next}_{\mathcal{C}}(q_{|\mathcal{C}}, A, m_{|\mathcal{C}})$
- *pour tout joueur A_j de A, $q'_{|\mathcal{M}_{A_j,b}} = m_{A_j}$*

Preuve : D'après le lemme précédent, pour tout état q de la structure $\mathcal{C}_b$, et pour tout vecteur de mouvements m d'une coalition quelconque $A \subseteq \mathsf{Agt}$:

$$\mathsf{Next}_{\mathcal{C}_b}(q,A,m) = \Big(\prod_{A_j \in \mathsf{Agt}} \mathsf{Next}_{\mathcal{M}_{A_j,b}}(q_{|\mathcal{M}_{A_j,b}}, A, m_{|\mathcal{M}_{A_j,b}}) \Big) \times \mathsf{Next}_{\mathcal{C}}(q_{|\mathcal{C}}, A, m_{|\mathcal{C}})$$

On conclut en utilisant ce que l'on sait de $\mathsf{Next}_{\mathcal{M}_{A_j,b}}$ (voir la remarque). □

Définition 42. *Soient $\mathcal{C}$ une* CGS, *A_i un joueur, et b un entier naturel.*

A tout contexte stratégique $F = (F^{\mathsf{mov}}, F^{\mathsf{cell}}, c) \in \mathsf{Strat}'^{b}_{A_i,\mathcal{C}}$ (voir la définition page 10), on fait correspondre un contexte stratégique positionnel *$\alpha_{A_i,b}(F) \in \mathsf{Strat}^0_{A_i,\mathcal{C}_b}$ de la manière suivante :*

A chaque état q de la structure $\mathcal{C}_b$, on fait correspondre le mouvement $\alpha_{A_i,b}(F)(q)$ de $\mathcal{C}_b$ dont :

1. *la composante sur* $\mathcal{C}$ *est* $F^{\mathsf{mov}}(q_{|\mathcal{M}_{A_i,b}}, q_{|\mathcal{C}})$
2. *la composante sur* $\mathcal{M}_{A_i,b}$ *est* $F^{\mathsf{cell}}(q_{|\mathcal{M}_{A_i,b}}, q_{|\mathcal{C}})$
3. *toutes les autres composantes valent* m'

On étend cette application α aux coalitions : si on se donne une coalition $A \subseteq \mathsf{Agt}$ et un entier naturel b, on fait correspondre à tout contexte stratégique $F \in \mathsf{Strat}'^{b}_{A,\mathcal{C}}$ de la coalition A, un contexte stratégique positionnel pour la même coalition :

$$\alpha_{A,b}(F) = \big(\alpha_{A_i,b}\big(F_{|A_i}\big)\big)_{A_i \in A}$$

De plus, à chaque état s et à chaque contexte stratégique $F = (F^{\mathsf{mov}}, F^{\mathsf{cell}}, c)$ de mémoire b pour une coalition A fixée, on fait correspondre l'état $\alpha(F, s)$ de $\mathcal{C}_b$ dont :

– la composante sur $\mathcal{C}$ est s,
– pour tout joueur A_i de A, la composante sur $\mathcal{M}_{A_i,b}$ est $c_{|A_i}$,
– toutes les autres composantes valent 0.

Si λ désigne une exécution de $\mathcal{C}$ et k un entier, on note $c_{\lambda,k}$ la cellule mémoire initiale du contexte stratégique $F^{\lambda,k}$. Cette suite peut se définir effectivement par la récurrence : $c_{\lambda,0} = c$ et $\forall k.\ c_{\lambda,k+1} = F^{\mathsf{cell}}(c_{\lambda,k}, \lambda[k])$.

En particulier, quel que soit l'état s', on a :

$$F^{\mathsf{mov}}(c_{\lambda,k}, s') = F^{\lambda,k}(s')$$

Nous sommes maintenant en mesure de caractériser l'ensemble d'exécutions $\mathsf{Out}_{\mathcal{C}_b}\big(q, \alpha_{A,b}(F)\big)$:

Lemme 26. *Une exécution* λ *de* $\mathcal{C}_b$*, qui part de l'état* $\alpha(F, s)$*, est dans l'ensemble d'outcomes* $\mathsf{Out}_{\mathcal{C}_b}\big(\alpha(F, s), \alpha_{A,b}(F)\big)$ *ssi :*

1. $\lambda_{|\mathcal{C}}$ *est dans* $\mathsf{Out}_{\mathcal{C}}(s, F)$*, et*
2. *pour tout joueur* A_i *de la coalition* A*, et tout entier* k *:*

$$\lambda_{|\mathcal{M}_{A_i,b}}[k] = (c_{\lambda_{|\mathcal{C}},k})_{|A_i}$$

Preuve : Soit λ une exécution de $\mathsf{Exec}_{\mathcal{C}_b}\big(\alpha(F, s)\big)$.

Nous allons d'abord montrer par récurrence sur k que pour tout joueur A_i de A :

$$\alpha_{A,b}(F)(\lambda[k])_{A_i} = (c_{\lambda_{|\mathcal{C}},k+1})_{|A_i}$$

Fixons un joueur $A_i \in A$.

Nous traitons en premier lieu l'hérédité :

$$\begin{array}{rcl} \alpha_{A,b}(F)(\lambda[k])_{A_i} & = & F^{\mathsf{cell}}(\lambda[k]_{|\mathcal{M}_{A_i,b}}, \lambda[k]_{|\mathcal{C}}) \\ & = & F^{\mathsf{cell}}((c_{\lambda_{|\mathcal{C}},k})_{|A_i}, \lambda_{|\mathcal{C}}[k]) \\ & = & (c_{\lambda_{|\mathcal{C}},k+1})_{|A_i} \end{array}$$

Le cas particulier de l'initialisation se montre par les mêmes arguments, sans utiliser l'hypothèse de récurrence, mais le fait que $\lambda[0]$ est en fait $\alpha(F,s)$, et on a défini $\alpha(F,s)$ de sorte que $\alpha(F,s)_{|\mathcal{M}_{A_i,b}} = c_{|A_i}$ pour tout joueur A_i de A.

Pour montrer le sens réciproque, supposons que :

$$\lambda_{|\mathcal{C}} \in \mathsf{Out}_{\mathcal{C}}(s,F) \text{ et } \forall A_i \in A.\ \forall k.\ \lambda_{|\mathcal{M}_{A_i,b}}[k] = (c_{\lambda_{|\mathcal{C}},k})_{|A_i}$$

Nous allons montrer que $\lambda \in \mathsf{Out}_{\mathcal{C}_b}\big(\alpha(F,s), \alpha_{A,b}(F)\big)$. Il est évident que $\lambda[0] = \alpha(F,s)$. Soit $k \in \mathbb{N}$. Il reste à établir que :

$$\lambda[k+1] \in \mathsf{Next}_{\mathcal{C}_b}\big(\lambda[k], A, \alpha_{A,b}(F)(\lambda[k])\big)$$

Par le lemme 25, il suffit de prouver que :

1. $\lambda[k+1]_{|\mathcal{C}} \in \mathsf{Next}_{\mathcal{C}}\Big(\lambda[k]_{|\mathcal{C}}, A, \alpha_{A,b}(F)(\lambda[k])_{|\mathcal{C}}\Big)$

   On sait déjà que $\lambda[k+1]_{|\mathcal{C}} \in \mathsf{Next}_{\mathcal{C}}\Big(\lambda[k]_{|\mathcal{C}}, A, F^{\lambda_{|\mathcal{C}},k}(\lambda_{|\mathcal{C}}[k])\Big)$, puisque $\lambda_{|\mathcal{C}}$ est par hypothèse une outcome du contexte stratégique à mémoire bornée F.

   On peut conclure en remarquant que :

$$\begin{aligned}
\alpha_{A,b}(F)(\lambda[k])_{|\mathcal{C}} &= \Big(F^{\mathsf{mov}}_{|A_i}(\lambda[k]_{|\mathcal{M}_{A_i,b}}, \lambda[k]_{|\mathcal{C}})\Big)_{A_i \in A} \\
&= \Big(F^{\mathsf{mov}}_{|A_i}((c_{\lambda_{|\mathcal{C}},k})_{|A_i}, \lambda[k]_{|\mathcal{C}})\Big)_{A_i \in A} \\
&= \Big(F^{\lambda_{|\mathcal{C}},k}_{|A_i}(\lambda[k]_{|\mathcal{C}})\Big)_{A_i \in A} \\
&= F^{\lambda_{|\mathcal{C}},k}(\lambda[k]_{|\mathcal{C}})
\end{aligned}$$

2. pour tout joueur A_j de A,

$$\lambda[k+1]_{|\mathcal{M}_{A_j,b}} = \alpha_{A,b}(F)(\lambda[k])_{A_j}$$

   Pour tout joueur A_j de la coalition A, on a :

$$\begin{aligned}
\alpha_{A,b}(F)(\lambda[k])_{A_j} &= F^{\mathsf{cell}}(\lambda[k]_{|\mathcal{M}_{A_j,b}}, \lambda[k]_{|\mathcal{C}}) \\
&= F^{\mathsf{cell}}((c_{\lambda_{|\mathcal{C}},k})_{|A_j}, \lambda[k]) \\
&= (c_{\lambda_{|\mathcal{C}},k+1})_{|A_i}
\end{aligned}$$

Passons maintenant à l'autre sens. Supposons à présent que λ est dans $\mathsf{Out}_{\mathcal{C}_b}\big(\alpha(F,s), \alpha_{A,b}(F)\big)$.

Alors pour tout joueur A_i, une récurrence sur k établit que :

$$\lambda[k]_{|\mathcal{M}_{A_i,b}} = (c_{\lambda_{|\mathcal{C}},k})_{|A_i}$$

Soit k et A_i. $\lambda[k+1] \in \mathsf{Next}_{\mathcal{C}_b}\big(\lambda[k], A, \alpha_{A,b}(F)(\lambda[k])\big)$, donc :

1.

$$\begin{aligned}
\lambda[k+1]_{|\mathcal{M}_{A_i,b}} &= \alpha_{A,b}(F)(\lambda[k])_{A_i} \\
&= F^{\mathsf{cell}}(\lambda[k]_{|\mathcal{M}_{A_i,b}}, \lambda[k]_{|\mathcal{C}}) \\
&= F^{\mathsf{cell}}((c_{\lambda_{|\mathcal{C}},k})_{|A_i}, \lambda[k]_{|\mathcal{C}}) \\
&= (c_{\lambda_{|\mathcal{C}},k+1})_{|A_i}
\end{aligned}$$

2. $\lambda[k+1]_{|\mathcal{C}} \in \mathsf{Next}_{\mathcal{C}}\big(\lambda[k]_{|\mathcal{C}}, A, \alpha_{A,b}(F)(\lambda[k])_{|\mathcal{C}}\big)$. Or

$$\begin{aligned}
\alpha_{A,b}(F)(\lambda[k])_{|\mathcal{C}} &= \Big(F^{\mathsf{mov}}_{|A_i}(\lambda[k]_{|\mathcal{M}_{A_i,b}}, \lambda[k]_{|\mathcal{C}})\Big)_{A_i \in A} \\
&= \Big(F^{\mathsf{mov}}_{|A_i}((c_{\lambda_{|\mathcal{C}},k})_{|A_i}, \lambda[k]_{|\mathcal{C}})\Big)_{A_i \in A} \\
&= \Big(F^{\lambda_{|\mathcal{C}},k}_{|A_i}(\lambda[k]_{|\mathcal{C}})\Big)_{A_i \in A} \\
&= F^{\lambda_{|\mathcal{C}},k}(\lambda[k]_{|\mathcal{C}})
\end{aligned}$$

□

Or on sait par le lemme 23 que :

$$\mathsf{Exec}_{\left(\mathcal{C}_b, \alpha_{A,b}(F)\right)}\big(\alpha(F,s)\big) = \mathsf{Out}_{\mathcal{C}_b}\big(\alpha(F,s), \alpha_{A,b}(F)\big)$$

De cela, et du lemme 26 que nous venons d'établir, on déduit que :

Corollaire 6. *Pour tout état s et tout contexte stratégique F à mémoire b d'une coalition A :*

$$\mathsf{Out}_{\mathcal{C}}(s,F) = \mathsf{Exec}_{\left(\mathcal{C}_b, \alpha_{A,b}(F)\right)}\big(\alpha(F,s)\big)_{|\mathcal{C}}$$

Soit un entier b fixé. Nous présentons l'algorithme 2 de model-checking pour la logique $\mathsf{ATL}^*_{sc,b}$.

On définit par induction sur $\mathsf{ATL}^*_{sc,b,p}$ l'application σ à valeurs dans LTL (avec toutefois plus de propositions atomiques) telle que :

- Si $\varphi_p = a \in \mathsf{AP}$, alors $\sigma(\varphi_p) = a$,
- Si $\varphi_p = \neg\psi$, alors $\sigma(\varphi_p) = \neg\sigma(\psi)$,
- Si $\varphi_p = \psi_1 \vee \psi_2$, alors $\sigma(\varphi_p) = \sigma(\psi_1) \vee \sigma(\psi_2)$,
- Si $\varphi_p = \ll B \gg_b \psi$, alors $\sigma(\varphi_p) = a_{\varphi_p}$,
- Si $\varphi_p = \langle\!\cdot B \cdot\rangle\, \psi$, alors $\sigma(\varphi_p) = a_{\varphi_p}$,
- Si $\varphi_p = \mathbf{X}\,\psi$, alors $\sigma(\varphi_p) = \mathbf{X}\,\sigma(\psi)$,
- Si $\varphi_p = \psi_1\,\mathbf{U}\,\psi_2$, alors $\sigma(\varphi_p) = \sigma(\psi_1)\,\mathbf{U}\,\sigma(\psi_2)$.

Algorithm 2 MC $\mathsf{ATL}^*_{sc,b}$

Require: une *CGS* $\mathcal{C}$, un contexte stratégique F à mémoire b pour une coalition A, un état s_0 et une formule θ_p de $\mathsf{ATL}^*_{sc,b,p}$
Ensure: OUI ssi $\forall \lambda \in \mathsf{Out}_{\mathcal{C}}(s_0, F), (\mathcal{C}, \lambda) \models_F \theta_p$
  $\mathcal{M} := \big(\mathcal{C}_b, \alpha_{A,b}(F)\big)$
  $\Phi_1 := \{\psi \in Sub(\theta_p) \mid rq(\psi) = rq(\theta_p)$ et $\exists B \subseteq \mathsf{Agt}, \exists \psi' \in \mathsf{ATL}^*_{sc,b}, \psi = \langle\!\langle B \rangle\!\rangle_b \psi'\}$
  $\Phi_2 := \{\psi \in Sub(\theta_p) \mid rq(\psi) = rq(\theta_p)$ et $\exists B \subseteq \mathsf{Agt}, \exists \psi' \in \mathsf{ATL}^*_{sc,b}, \psi = [\![B]\!] \psi'\}$
  for $\psi \in \Phi_1 \cup \Phi_2$ **do**
    $\mathsf{AP}_{\mathcal{M}} := \mathsf{AP}_{\mathcal{M}} \cup \{a_\psi\}$
    for $s \in \mathsf{Loc}_{\mathcal{M}}$ **do**
      if $\psi \in \Phi_1$ **then**
        for $F' \in \mathsf{Strat}^{'b}_{B,\mathcal{C}}$ **do**
          if MC $\mathsf{ATL}^*_{sc,b}(\mathcal{C}, F' \circ F, s_{|\mathcal{C}}, \psi')$ **then**
            $\mathsf{Lab}_{\mathcal{M}}(s) := \mathsf{Lab}_{\mathcal{M}}(s) \cup \{a_\psi\}$
          end if
        end for
      else
        if MC $\mathsf{ATL}^*_{sc,b}(\mathcal{C}, F|_{A \setminus B}, s_{|\mathcal{C}}, \psi')$ **then**
          $\mathsf{Lab}_{\mathcal{M}}(s) := \mathsf{Lab}_{\mathcal{M}}(s) \cup \{a_\psi\}$
        end if
      end if
    end for
  end for
  return MC LTL $(\mathcal{M}, \alpha(F, s_0), \sigma(\theta_p))$

On va montrer que l'algorithme 2 fait bien ce que l'on affirme. On désigne par $\mathcal{M}_{f,(\mathcal{C},F,\theta_p)}$ la structure contenue dans la variable $\mathcal{M}$ en fin d'exécution de l'algorithme 2 sur l'entrée $(\mathcal{C}, F, s_0, \theta_p)$. Cette notation est justifiée par le fait que la valeur de s_0 n'importe pas.

Il est clair que pour toute entrée $(\mathcal{C}, F, s_0, \theta_p)$ de l'algorithme 2, les structures $\big(\mathcal{C}_b, \alpha_{A,b}(F)\big)$ et $\mathcal{M}_{f,(\mathcal{C},F,\theta_p)}$ ont les mêmes états et les mêmes transitions (de fait, elles ne diffèrent que par leurs propositions atomiques et leurs étiquettes). Donc depuis tout état q de la structure $\mathcal{C}_b$, on a :

$$\mathsf{Exec}_{\mathcal{M}_{f,(\mathcal{C},F,\theta_p)}}(q) = \mathsf{Exec}_{\big(\mathcal{C}_b,\alpha_{A,b}(F)\big)}(q)$$

Soit une *CGS* $\mathcal{C}$. Nous allons montrer par induction structurelle sur les formules φ_p de la logique $\mathsf{ATL}^*_{sc,b,p}$, que pour tout contexte stratégique F à mémoire b d'une coalition A quelconque et pour toute exécution λ de $\big(\mathcal{C}_b, \alpha_{A,b}(F)\big)$ qui part d'un état s, on a :

$$(\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)}, \lambda) \models \sigma(\varphi_p) \Leftrightarrow (\mathcal{C}, \lambda_{|\mathcal{C}}) \models_F \varphi_p$$

1. $\varphi_p = \ll\! B \gg_b \psi$;
   On se donne un contexte stratégique F de mémoire b pour une coalition A et une exécution λ de la structure $\big(\mathcal{C}_b, \alpha_{A,b}(F)\big)$ qui part de s. On a la suite d'équivalences :

$$\begin{array}{lcl}(\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)}, \lambda) \models \sigma(\varphi_p) & \Leftrightarrow & (\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)}, \lambda) \models a_{\varphi_p} \\ & \Leftrightarrow & a_{\ll B \gg_b \psi} \in \mathsf{Lab}_{\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)}}(s)\end{array}$$

   Or cet étiquetage n'est possible que si l'on a pu trouver une stratégie F' à mémoire b pour la coalition B, telle que l'algorithme 2 retourne OUI sur l'entrée $(\mathcal{C}, F' \circ F, s_{|\mathcal{C}}, \psi)$, c'est-à-dire que :

$$\begin{array}{lcl}a_{\varphi_p} \in \mathsf{Lab}_{\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)}}(s) & \Leftrightarrow & \exists F' \in \mathsf{Strat}^{'b}_{B,\mathcal{C}}. \\ & & \quad \text{MC } \mathsf{LTL}\big(\mathcal{M}_{f,(\mathcal{C},F'\circ F,\psi)}, \alpha(F' \circ F, s_{|\mathcal{C}}), \sigma(\psi)\big) \\ & \Leftrightarrow & \exists F' \in \mathsf{Strat}^{'b}_{B,\mathcal{C}}. \\ & & \quad \forall \lambda' \in \mathsf{Exec}_{\big(\mathcal{C}_b,\alpha_{A,b}(F'\circ F)\big)}\big(\alpha(F' \circ F, s_{|\mathcal{C}})\big). \\ & & \qquad (\mathcal{M}_{f,(\mathcal{C},F'\circ F,\psi)}, \lambda') \models \sigma(\psi) \\ & \Leftrightarrow & \exists F' \in \mathsf{Strat}^{'b}_{B,\mathcal{C}}. \\ & & \quad \forall \lambda' \in \mathsf{Exec}_{\big(\mathcal{C}_b,\alpha_{A,b}(F'\circ F)\big)}\big(\alpha(F' \circ F, s_{|\mathcal{C}})\big). \\ & & \qquad (\mathcal{C}, \lambda'_{|\mathcal{C}}) \models_{F'\circ F} \psi\end{array}$$

$$
\begin{array}{lll}
 & & \text{(par HI)} \\
\Leftrightarrow & & \exists F' \in \mathsf{Strat}^{'b}_{B,\mathcal{C}}.\ \forall \lambda' \in \mathsf{Out}_{\mathcal{C}}(s_{|\mathcal{C}}, F' \circ F). \\
 & & \qquad (\mathcal{C}, \lambda') \models_{F' \circ F} \psi \\
 & & \text{d'après le corollaire précédent} \\
\Leftrightarrow & & (\mathcal{C}, \lambda_{|\mathcal{C}}) \models_F \varphi_p
\end{array}
$$

2. $\varphi_p = \mathbf{X}\,\psi$;
   Soient un contexte stratégique F de mémoire b pour une coalition A quelconque, et une exécution λ de la structure $\big(\mathcal{C}_b, \alpha_{A,b}(F)\big)$, partant de s. Alors l'énoncé $(\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)}, \lambda) \models \sigma(\varphi_p)$ signifie

   $$(\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)}, \lambda[1,\infty]) \models \sigma(\psi)$$

   Les structures $\mathcal{M}_{f,(\mathcal{C},F,\varphi_p)}$ et $\mathcal{M}_{f,(\mathcal{C},F^{\lambda,1},\psi)}$ étant identiques, on peut réécrire cela $(\mathcal{M}_{f,(\mathcal{C},F^{\lambda,1},\psi)}, \lambda[1,\infty]) \models \sigma(\psi)$, et c'est équivalent d'après l'hypothèse d'induction à $(\mathcal{C}, \lambda[1,\infty]_{|\mathcal{C}}) \models_{F^{\lambda,1}} \psi$, c'est-à-dire à

   $$(\mathcal{C}, \lambda_{|\mathcal{C}}) \models_F \varphi_p$$

On sait que l'algorithme 2 retourne OUI sur l'entrée $(\mathcal{C}, F, s_0, \theta_p)$ ssi, au sens de LTL :

$$\Big(\big(\mathcal{M}_{f,(\mathcal{C},F,\theta_p)}\big), \alpha(F, s_0)\Big) \models \sigma(\theta_p)$$

La propriété que l'on vient d'établir, le corollaire 6 et le fait que $(\mathcal{M}_{f,(\mathcal{C},F,\theta_p)})$ et $\big(\mathcal{C}_b, \alpha_{A,b}(F)\big)$ aient les mêmes exécutions permettent de prouver que l'algorithme 2 retourne OUI sur l'entrée $(\mathcal{C}, F, s_0, \theta_p)$ ssi

$$\forall \lambda \in \mathsf{Out}_{\mathcal{C}}(s_0, F), (\mathcal{C}, \lambda) \models_F \theta_p$$

comme on l'avait annoncé.

Cet algorithme est bien un algorithme de model-checking pour $\mathsf{ATL}^*_{sc,b}$: étant donnés une *CGS* $\mathcal{C}$, un état s_0 de $\mathcal{C}$ et une formule θ de $\mathsf{ATL}^*_{sc,b}$, si l'on veut savoir si $(\mathcal{C}, s_0) \models \theta$, il suffit de lancer l'algorithme 2 sur l'entrée $(\mathcal{C}, \varnothing, s_0, \theta)$. En effet, il retournera OUI ssi $\forall \lambda \in \mathsf{Out}_{\mathcal{C}}(s_0, \varnothing), (\mathcal{C}, \lambda) \models \theta$, c'est-à-dire ssi $(\mathcal{C}, s_0) \models \theta$ puisque θ est une formule d'état. Cet algorithme est visiblement dans EXPSPACE, toutefois si l'on fixe le nombre de joueur, il devient PSPACE.

Les résultats obtenus ne nous permettent pas de conclure sur la complexité précise de ce problème.

5.5 Model-checking des logiques $\mathsf{ATL}_{sc,\infty}$, $\mathsf{ATL}^*_{sc,\infty}$ et $\mathsf{ATL}^*_{sc,\infty} \setminus \cdot\rangle \cdot \langle\cdot$

Le résultat principal de cette section est le suivant :

Théorème 16. *Le problème de model-checking des formules de $\mathsf{ATL}_{sc,\infty}$ qui contiennent au plus k quantificateurs de stratégies emboîtés peut se faire en temps $(k+1)$*EXPTIME*. La* complexité en programme *(qui est la complexité du model-checking avec une formule fixée de $\mathsf{ATL}_{sc,\infty}$) est dans* EXPTIME*.*

Un corollaire immédiat, au vu des théorèmes de traduction, est que :

Corollaire 7. *Les problèmes de model-checking des logiques $\mathsf{ATL}_{sc,\infty}$, $\mathsf{ATL}^*_{sc,\infty}$, $\rangle\cdot\langle$ et $\mathsf{ATL}^*_{sc,\infty}$ sont résolubles par un algorithme non élémentaire.*

La preuve s'appuie principalement sur la construction d'un automate d'arbres alternant, à partir d'une formule et d'une CGS. Des approches similaires ont été développées pour les problèmes de model-checking des logiques SL_{CHP} [12] et $\text{Q}D\mu$ [28], mais elles sont définies sur des sous-classes de CGSs : tout d'abord, l'étude de SL_{CHP} est restreinte aux structures de jeux à tours.

Nous allons à présent développer notre analyse de $\text{Q}D\mu$. L'algorithme présenté pour cette logique s'applique à des CGS qui vérifient une condition que nous appellerons *condition @*, et qui demande que depuis tout état s et pour toute coalition A :

$$\mathsf{Next}(s, A, m) = \bigcap_{A_i \in A} \mathsf{Next}(s, A_i, m_{|A_i})$$

Cette condition @ est indiquée explicitement dans [29], mais elle est nécessaire également dans [28] : en effet, dans ces deux articles, la notion de stratégie utilisée est différente de la définition classique (que nous avons utilisée dans ce manuscrit). Or cette nouvelle définition ne recoupe la définition standard d'une stratégie que si l'on travaille dans des CGS qui vérifient la condition @. Bien sûr la notion d'outcomes repose sur celle de stratégie, et l'on peut donc faire la même remarque pour ces objets.

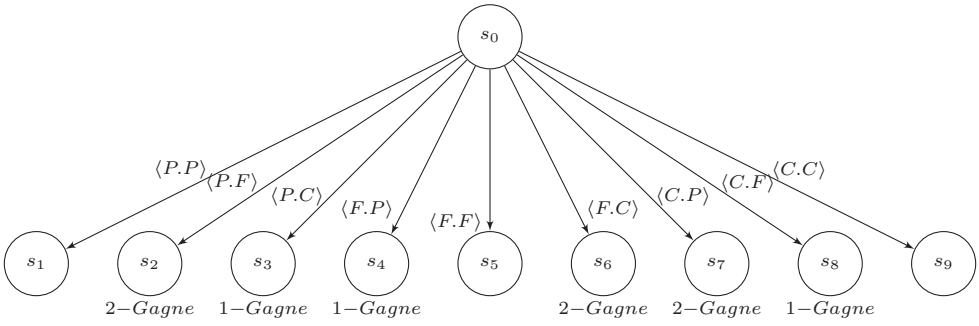

FIG. 5.2 – Le jeu Pierre-Feuille-Ciseaux présenté dans [28]

Dans [28, 29], la notion de stratégie pour une coalition A repose sur un étiquetage par des propositions atomiques Q_{A_i} (pour $A_i \in A$) des états de la CGS accessibles lorsque A_i joue selon la stratégie fixée.

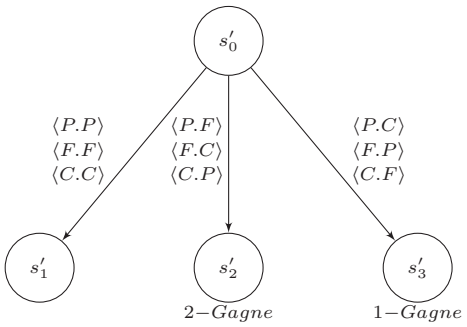

FIG. 5.3 – Le jeu Pierre-Feuille-Ciseaux tel qu'il est présenté ordinairement

Dans l'exemple de la Figure 5.3, la stratégie où A_1 joue "P" et A_2 joue "C" conduit à étiqueter s'_1, s'_2 et s'_3 par Q_{A_1} et Q_{A_2} car tous ces états sont des successeurs possibles de s'_0 losque A_1 (resp. A_2) joue "P" (resp. "C"). Ainsi les outcomes engendrées par cette stratégie contiennent les trois exécutions $s'_0s'_1, s'_0s'_2$ et $s'_0s'_3$. Ceci est évidemment faux : la seule outcome pour cette stratégie est $s'_0s'_3$.

Le problème vient du fait que la condition @ n'est pas vérifiée par cette CGS : $\mathsf{Next}(s'_0, \{A_1, A_2\}, < P, C >) = \{s'_3\}$ mais $\mathsf{Next}(s'_0, A_1, P) \cap \mathsf{Next}(s'_0, A_2, C) = \{s'_1, s'_2, s'_3\}$.

Cependant, on pourrait remarquer que toute CGS est bisimilaire à une autre CGS qui, elle, vérifie la condition @. Pour obtenir cette nouvelle structure, une manière systématique de procéder serait d'attribuer à chaque vecteur de mouvement une transition différente, en multipliant les états d'arrivée. C'est ainsi que la CGS de la figure 5.3 se transformerait en la CGS de la Figure 5.2. Par conséquent, pour une logique qui est invariante par bisimulation, comme ATL^* par exemple, on peut effectuer indifféremment un algorithme de model-checking sur la première ou sur la deuxième figure, l'algorithme retournant nécessairement la même réponse dans les deux cas, puisque les ensembles de formules vraies coïncident.

Mais cette technique n'est pas transposable à une procédure de *model-checking sur CGS* pour ATL_{sc} car cette logique n'est pas invariable par bisimulation, comme nous l'avons établi avec la Proposition 4 dans la section 4.5. C'est pourquoi nous ne pouvons utiliser l'algorithme de [28] pour résoudre notre problème.

L'idée importante de notre algorithme est la suivante : on travaille sur l'arbre d'exécution de la CGS donnée en entrée, et on étiquette chaque nœud par les mouvements disponibles aux joueurs. On se focalise ensuite sur les branches qui correspondent à des outcomes des stratégies sélectionnées, puis on vérifie qu'elles satisfont la requête spécifiée dans la formule. Cette technique fut à l'origine présentée dans [18].

Avant de présenter la preuve dans ses détails, nous introduisons les automates d'arbres alternants.

5.5.1 Arbres

Définition 43. *Soient Σ et S deux ensembles finis. Un* S-arbre étiqueté par Σ *est un couple $\mathcal{T} = \langle T, l\rangle$, où :*

1. *$T \subseteq S^*$ est un ensemble non vide de mots finis sur l'alphabet S, qui satisfait la propriété suivante :*
   pour tout mot non vide $n = m \cdot s$ de T avec $m \in S^*$ et $s \in S$, le mot m est aussi dans T ;
2. *$l \colon T \to \Sigma$ est une fonction d'étiquetage.*

Le mot vide, qui constitue la *racine* de l'arbre, sera noté ϵ.

Etant donné un tel arbre $\mathcal{T} = \langle T, l\rangle$ et un nœud n de T, l'ensemble des *directions depuis n dans T* est l'ensemble

$$\mathsf{dir}_T(n) = \{s \in S \mid n \cdot s \in T\}$$

L'ensemble des *successeurs de n dans T* est

$$\mathsf{succ}_T(n) = \{n \cdot s \mid s \in \mathsf{dir}_T(n)\}$$

Nous utiliserons la notation $\mathcal{T}_n$ pour désigner le sous-arbre qui part de n. Un S-arbre est dit *complet* si $T = S^*$, c'est-à-dire si $\mathsf{dir}_T(n) = S$ pour tout $n \in T$. On omettra l'indice T lorsqu'il sera clairement signifié par le contexte.

L'ensemble des *chemins infinis de $\mathcal{T}$* est l'ensemble

$$\mathsf{Path}_{\mathcal{T}} = \{s_0 \cdot s_1 \cdots \in S^\omega \mid \forall i \in \mathbb{N}.\ s_0 \cdot s_1 \cdots s_i \in T\}$$

Si $\pi = (s_i)_{i\in\mathbb{N}}$ est un chemin infini, on écrira $l(\pi)$ pour la suite infinie $\big(l(s_i)\big)_{i\in\mathbb{N}} \in \Sigma^\omega$, et $\mathsf{Inf}\big(l(\pi)\big)$ pour l'ensemble de lettres de Σ qui apparaissent infiniment souvent dans $l(\pi)$.

Supposons maintenant que $\Sigma = \Sigma_1 \times \Sigma_2$, et donnons-nous un S-arbre $\mathcal{T} = \langle T, l\rangle$, étiqueté par Σ. Alors :

- pour tout $n \in T$, on écrit $l(n) = \big(l_1(n), l_2(n)\big)$ avec $l_i(n) \in \Sigma_i$ pour $i \in \{1, 2\}$. De plus, pour $i \in \{1, 2\}$, la *projection de $\mathcal{T}$ sur Σ_i*, que l'on notera $\mathsf{proj}_{\Sigma_i}(\mathcal{T})$, est le S-arbre $\langle T, l_i\rangle$, étiqueté par Σ_i.
- de plus, nous dirons de deux S-arbres étiquetés par Σ sont *Σ_i-équivalents* ssi leurs projections sur Σ_i sont les mêmes.

Ces deux notions s'étendent naturellement à des alphabets plus complexes, de la forme $\prod_{i\in I} \Sigma_i$.

5.5.2 Automates d'arbres alternants

Nous définissons à présent la notion d'automate d'arbres alternant [22], qui sera utilisée dans la preuve. Pour cela, nous avons besoin des définitions suivantes : l'ensemble des *formules booléennes positives* sur un alphabet fini P de

variables propositionnelles, est l'ensemble des formules générées par la syntaxe :

$$\mathsf{PBF}(P) \ni \zeta ::= p \mid \zeta \wedge \zeta \mid \zeta \vee \zeta \mid \top \mid \bot$$

où p décrit l'alphabet P.

On définit la satisfaction d'une formule de $\mathsf{PBF}(P)$ de la manière habituelle, par une valuation

$$v \colon P \to \{\top, \bot\}$$

On écrira abusivement qu'un sous-ensemble P' de P satisfait la formule $\varphi \in \mathsf{PBF}(P)$ ssi la valuation $\mathbb{1}_{P'}$ (qui fait correspondre les éléments de P' à $\top$ et ceux de $P \setminus P'$ à $\bot$) satisfait φ. Puisqu'il n'y a pas de négation, si $P' \models \varphi$ et $P' \subseteq P''$, alors on a aussi $P'' \models \varphi$.

Définition 44. *Soient S et Σ deux ensembles finis. Un* automate de S-arbres alternant sur Σ*, ou $\langle S, \Sigma\rangle$-ATA, est un 4-uple $\mathcal{A} = \langle Q, q_0, \tau, \mathsf{Acc}\rangle$, où :*

- *Q est un ensemble fini d'états,*
- *$q_0 \in Q$ est l'état initial,*
- *$\tau \colon Q \times \Sigma \to \mathsf{PBF}(S \times Q)$ est la fonction de transition, et*
- *$\mathsf{Acc} \colon Q^\omega \to \{\top, \bot\}$ est la fonction d'acceptation.*

Un automate de S-arbres non-déterministe sur Σ*, ou $\langle S, \Sigma\rangle$-NTA, est un $\langle S, \Sigma\rangle$-ATA dans lequel pour tout état q et toute lettre σ, la fonction de transition est de la forme :*

$$\tau(q, \sigma) = \bigvee_{i \in I(q)} \Big(\bigwedge_{s \in S} (s, q_{i,s}) \Big)$$

où $I(q)$ représente un ensemble fini, et les $q_{i,s}$ sont des états (quelconques) de l'automate.

Si $I(q)$ est un singleton depuis tous les états q (la fonction de transition est donc une simple conjonction), alors l'automate sera dit déterministe.

La taille de $\mathcal{A}$, notée $|\mathcal{A}|$, est le nombre d'états de Q.

Soit $\mathcal{A} = \langle Q, q_0, \tau, \mathsf{Acc}\rangle$ un $\langle S, \Sigma\rangle$-ATA, et $\mathcal{T} = \langle T, l\rangle$ un S-arbre étiqueté par Σ. Un *arbre d'exécution* de $\mathcal{A}$ sur $\mathcal{T}$ est un $S \times Q$-arbre étiqueté par $T \times Q$, $\mathcal{E} = \langle E, p\rangle$, tel que :

1. $p(\epsilon) = (\epsilon, q_0)$,
2. pour tout nœud $e \in E$ tel que $p(e) = (t, q)$, les deux conditions suivantes sont satisfaites :
   - l'ensemble $\mathsf{dir}_E(e) = \{(s_0, q_0), (s_1, q_1), ..., (s_n, q_n)\} \subseteq S \times Q$ satisfait $\tau\big(q, l(t)\big)$;
   - pour tout $0 \leq i \leq n$, le nœud $e \cdot (s_i, q_i)$ est étiqueté par $(t \cdot s_i, q_i)$. On notera $p_S\big(e \cdot (s_i, q_i)\big) = t \cdot s_i$ et $p_Q\big(e \cdot (s_i, q_i)\big) = q_i$ les deux composantes de la fonction d'étiquetage.

Un arbre d'exécution est *acceptant* si $\mathsf{Acc}\big(p_Q(\pi)\big) = \top$ pour tout chemin infini $\pi \in (S \times Q)^\omega$ de $\mathsf{Path}_\mathcal{E}$. Un arbre $\mathcal{T}$ est accepté par $\mathcal{A}$ ssi il existe un arbre d'exécution acceptant de $\mathcal{A}$ sur $\mathcal{T}$.

Dans la suite, nous utiliserons *l'acceptation par condition de parité*, donnée sous la forme d'une fonction $\Omega\colon Q \to \{0, ..., k-1\}$, à partir de laquelle la fonction Acc est définie comme suit :

$$\mathsf{Acc}\big(p_Q(\pi)\big) = \top \text{ ssi } \min\{\Omega(q) \mid q \in \mathsf{Inf}\big(p_Q(\pi)\big)\} \text{ est pair}$$

Les $\langle S, \Sigma\rangle$-ATAs qui ont une telle condition d'acceptation sont appelés $\langle S, \Sigma\rangle$-APTs, et étant donné un $\langle S, \Sigma\rangle$-APT $\mathcal{A}$, la taille de l'image de Ω est appelée l'*index* de $\mathcal{A}$, et est notée $\mathsf{idx}(\mathcal{A})$. De la même façon, les $\langle S, \Sigma\rangle$-NPTs sont des $\langle S, \Sigma\rangle$-NTAs dont l'acceptation se fait par condition de parité.

5.5.3 Le déroulement d'une CGS

Soit $\mathcal{C} = \langle \mathsf{Loc}, \mathsf{Lab}, \mathsf{Agt}, \mathcal{M}, \mathsf{Mov}, \mathsf{Edg}\rangle$ une CGS à n joueurs, et ℓ_0 un état de $\mathcal{C}$.

Si q est un état de $\mathcal{C}$, alors on notera $\mathsf{Edg}(q)$ la fonction qui à tout vecteur de mouvements $(m_i)_i \in \mathcal{M}^{\mathsf{Agt}}$, fait correspondre l'état $\mathsf{Edg}(q, (m_i)_i)$, c'est-à-dire le successeur de q selon ce vecteur de mouvements.

Pour chaque état $\ell \in \mathsf{Loc}$, on définit :

- $\mathsf{Strat}(\ell) = \{\ell\} \times \{\mathsf{Lab}(\ell)\} \times \{\mathsf{Edg}(\ell)\}$;
- $\mathsf{Strat}^+(\ell) = \mathsf{Strat}(\ell) \times (\mathcal{M} \cup \{\bot\})^{\mathsf{Agt}} \times 2^{\{p_o, p_l, p_r\}}$,

où $\bot$ est un symbole spécial qui n'apparaît pas dans $\mathcal{M}$, et p_o, p_l et p_r sont trois nouvelles propositions qui ne sont pas dans AP.

On notera :

$$\mathsf{Strat}_{\mathcal{C}} = \bigcup_{\ell \in \mathsf{Loc}} \mathsf{Strat}(\ell)$$

$$\mathsf{Strat}^+_{\mathcal{C}} = \bigcup_{\ell \in \mathsf{Loc}} \mathsf{Strat}^+(\ell)$$

Remarquons que les tailles $|\mathsf{Strat}_{\mathcal{C}}|$ et $|\mathsf{Strat}^+_{\mathcal{C}}|$ sont linéaires en fonction de la taille de l'entrée, puisque nous supposons que l'entrée contient une représentation explicite de la fonction de transition Edg [21].

Définition 45. *Le* déroulement *de $\mathcal{C}$ depuis ℓ_0 est le Loc-arbre complet étiqueté par $\mathsf{Strat}_{\mathcal{C}}$, $\mathcal{U} = \langle U, v\rangle$, où :*

- $U = \mathsf{Loc}^*$
- *pour tout $u \in U$, $v(u) \in \mathsf{Strat}(\mathsf{last}(\ell_0 \cdot u))$*

L'ensemble $\mathsf{Strat}(\mathsf{last}(\ell_0 \cdot u)$ étant un singleton, $v(u)$ est défini de façon unique.

Un déroulement étendu *de $\mathcal{C}$ depuis ℓ_0 est un Loc-arbre complet $\mathcal{U}'$ étiqueté par $\mathsf{Strat}^+_{\mathcal{C}}$ tel que $\mathcal{U}$ et $\mathcal{U}'$ sont $\mathsf{Strat}_{\mathcal{C}}$-équivalents.*

Pour chaque lettre σ de $\mathsf{Strat}^+_{\mathcal{C}}$, on écrit σ_{Loc}, σ_{AP}, σ_{Edg}, σ_{str} et σ_p pour les cinq composantes, et on étend cette notation indicielle aux fonctions d'étiquetage des arbres (on écrit l_{Loc}, l_{AP}, l_{Edg}, l_{str} et l_p). Ainsi, on a $\sigma_{\mathsf{Loc}} = l_{\mathsf{Loc}}(\sigma)$, $\sigma_{\mathsf{AP}} = l_{\mathsf{AP}}(\sigma)$, etc.

Dans la suite, nous identifierons un nœud u de $\mathcal{U}$ (qui est un mot fini sur l'alphabet Loc) avec le chemin fini $\ell_0 \cdot u$ de $\mathcal{C}$. Remarquons que cette suite d'états

de $\mathcal{C}$ peut ne correspondre à aucun chemin *réel* de $\mathcal{C}$, dans le cas où elle implique une transition qui n'est pas dans Edg.

Définition 46. *A partir d'une structure $\mathcal{C}$ et d'un état ℓ_0, on définit l'automate déterministe de parité $\mathcal{A}_{\mathcal{C},\ell_0} = \langle \overline{\mathsf{Loc}}, \overline{\ell_0}, \tau, \Omega \rangle$, de Loc-arbres sur l'alphabet $\mathsf{Strat}_{\mathcal{C}}^{+}$, tel que :*

- $\overline{\mathsf{Loc}} = \{\overline{\ell} \mid \ell \in \mathsf{Loc}\}$,
- $\overline{\ell_0}$ *est l'état initial,*
- Ω *est la fonction constante égale à* 0 *(par conséquent, tout arbre d'exécution valide est acceptant),*
- *étant donnés un état $\overline{\ell} \in \overline{\mathsf{Loc}}$ et une lettre $\sigma \in \mathsf{Strat}_{\mathcal{C}}^{+}$, la fonction de transition est définie comme suit :*

$$\tau(\overline{\ell}, \sigma) = \begin{cases} \bigwedge_{\ell' \in \mathsf{Loc}} (\ell', \overline{\ell'}) & \text{si } \sigma \in \mathsf{Strat}^{+}(\ell) \\ \bot & \text{sinon} \end{cases}$$

Lemme 27. *Soit $\mathcal{C}$ une CGS et ℓ_0 un état de $\mathcal{C}$. Soit $\mathcal{T} = \langle T, l \rangle$ un Loc-arbre étiqueté par $\mathsf{Strat}_{\mathcal{C}}^{+}$. Alors $\mathcal{A}_{\mathcal{C},\ell_0}$ accepte $\mathcal{T}$ ssi $\mathsf{proj}_{\mathsf{Strat}_{\mathcal{C}}}(\mathcal{T})$ est le déroulement de $\mathcal{C}$ depuis ℓ_0.*

Démonstration. Supposons que $\mathcal{T}$ est accepté par $\mathcal{A}_{\mathcal{C},\ell_0}$. Soit alors $\mathcal{E} = \langle E, p \rangle$ un arbre d'exécution acceptant de $\mathcal{A}_{\mathcal{C},\ell_0}$ sur $\mathcal{T}$.

Nous montrons par induction que tout nœud n de E est tel que $n = (\ell_i, \overline{\ell_i})_i$, et que pour tout $\ell \in \mathsf{Loc}$, $n \cdot (\ell, \overline{\ell})$ est un successeur de n.

Par conséquent chaque nœud n de $\mathcal{T}$ a $|\mathsf{Loc}|$ successeurs, et est étiqueté par $l(n) \in \mathsf{Strat}^{+}(\mathsf{last}(\ell_0 \cdot n))$. Il s'ensuit que $\mathsf{proj}_{\mathsf{Strat}_{\mathcal{C}}}(\mathcal{T})$ est le déroulement de $\mathcal{C}$ depuis ℓ_0.

On prouve facilement l'autre direction en construisant explicitement un arbre d'exécution acceptant. □

Par la suite nous utiliserons également l'automate $\mathcal{A}_{\mathcal{C}}$, qui accepte la réunion de tous les $\mathcal{L}(\mathcal{A}_{\mathcal{C},\ell_0})$ où ℓ_0 décrit l'ensemble d'états Loc. On construit un tel automate à l'aide du Lemme 30 que l'on trouvera ci-dessous.

5.5.4 Les quantifications sur les stratégies

Soit $\mathcal{T} = \langle T, l \rangle$ un Loc-arbre complet étiqueté par $\mathsf{Strat}_{\mathcal{C}}^{+}$, qui est accepté par $\mathcal{A}_{\mathcal{C},\ell_0}$. Un tel arbre définit des stratégies *partielles* pour chacun des joueurs : pour $A_i \in \mathsf{Agt}$, et pour chaque nœud $n \in T$, on définit

$$\mathsf{strat}_{A_i}^{\mathcal{T}}(\ell_0 \cdot n) = l_{\mathsf{str}}(n)(A_i) \in \mathcal{M} \cup \{\bot\}$$

Si $A \subseteq \mathsf{Agt}$ est une coalition, nous écrirons $\mathsf{strat}_A^{\mathcal{T}}$ pour désigner la famille de stratégies $(\mathsf{strat}_{A_i}^{\mathcal{T}})_{A_i \in A}$.

Dans un premier temps, pour toutes les coalitions $A \subseteq \mathsf{Agt}$, nous allons construire un $\langle \mathsf{Loc}, \mathsf{Strat}_{\mathcal{C}}^{+} \rangle$-APT $\mathcal{A}_{\mathrm{strat}}(A)$ qui garantit que pour tout joueur A_i de A, $\mathsf{strat}_{A_i}^{\mathcal{T}}$ est *effectivement* une stratégie du joueur A_i, c'est-à-dire qu'elle ne retourne jamais $\bot$.

Définition 47. *On définit l'automate $\mathcal{A}_{strat}(A)$: il n'a qu'un état q_0 (qui est donc l'état initial). On pose*

$$\tau(q_0, \sigma) = \bigwedge_{\ell \in \mathsf{Loc}} (\ell, q_0)$$

dans le cas où $\sigma_{\mathsf{str}}(A_i) \neq \bot$ pour tous les joueurs A_i de A. Si ce n'est pas le cas, on pose

$$\tau(q_0, \sigma) = \bot$$

Enfin, $\mathcal{A}_{\mathsf{strat}}$ accepte tous les arbres qui ont un arbre d'exécution valide (c'est-à-dire que Ω vaut toujours 0).

La preuve du lemme suivant est triviale :

Lemme 28. *Soit $\mathcal{C}$ une CGS, un état ℓ_0 de cette structure, et une coalition $A \subseteq \mathsf{Agt}$. Soit $\mathcal{T} = \langle T, l \rangle$ un Loc-arbre complet étiqueté par $\mathsf{Strat}_{\mathcal{C}}^{+}$ accepté par $\mathcal{A}_{\mathcal{C},\ell_0}$. Alors $\mathcal{T}$ est accepté par $\mathcal{A}_{\mathsf{strat}}(A)$ ssi pour tout joueur A_i de la coalition A, $\mathsf{strat}_{A_i}^{\mathcal{T}}$ ne vaut jamais $\bot$, c'est-à-dire qu'elle représente une vraie stratégie du joueur A_i, au sens où celle-ci est complète, puisqu'elle associe bien un état à tout préfixe d'exécution de $\mathcal{C}$.*

Nous allons à présent construire un automate afin de vérifier que la proposition p_o étiquette les outcomes de $\mathcal{T}$. Plus formellement, soit $A \subseteq \mathsf{Agt}$ un ensemble de joueurs. L'automate $\mathcal{A}_{\mathsf{out}}(A)$ acceptera $\mathcal{T}$ ssi p_o étiquette exactement les outcomes des stratégies $\mathsf{strat}_{A_i}^{\mathcal{T}}$ des joueurs A_i de A.

Pour cela, nous avons besoin d'introduire, pour tout étiquetage σ de $\mathsf{Strat}_{\mathcal{C}}^{+}$, la notation $\mathsf{Next}(\sigma, A)$, qui désigne l'ensemble des états ℓ de Loc satisfaisant :

$$\exists (m_i)_i \in \mathcal{M}^{\mathsf{Agt}} \text{ t.q. } \ell = \sigma_{\mathsf{Edg}}((m_i)_i) \text{ et } \forall A_i \in A.\ \sigma_{\mathsf{str}}(A_i) = m_i$$

En d'autres termes, $\mathsf{Next}(\sigma, A)$ est l'ensemble des états successeurs de l'état σ_{Loc} si les joueurs de A suivent les stratégies qui leur sont données par σ_{str}, en accord avec la table de transition σ_{Edg}. On remarque que $\mathsf{Next}(\sigma, A)$ est non vide ssi $\sigma_{\mathsf{str}}(A_i) \neq \bot$ pour tout les joueurs A_i de A.

Définition 48. *On réalise cela au moyen de l'automate à deux états, $\mathcal{A}_{out}(A) = \langle Q, q_\in, \tau, \Omega \rangle$, tel que :*

- *$Q = \{q_\in, q_\notin\}$,*
- *$q_\in$ est l'état initial,*
- *Ω est la fonction constante égale à 0,*
- *la fonction de transition τ est définie comme suit : si $p_o \notin \sigma$, alors $\tau(q_\in, \sigma) = \bot$ et $\tau(q_\notin, \sigma) = \bigwedge_{\ell \in \mathsf{Loc}} (\ell, q_\notin)$; sinon, $\tau(q_\notin, \sigma) = \bot$ et*

$$\tau(q_\in, \sigma) = \bigwedge_{\ell \in \mathsf{Next}(\sigma, A)} (\ell, q_\in) \wedge \bigwedge_{\ell \notin \mathsf{Next}(\sigma, A)} (\ell, q_\notin)$$

On a alors :

Lemme 29. *Soit $\mathcal{C}$ une CGS, ℓ_0 l'un de ses états, et une coalition $A \subseteq \mathsf{Agt}$. Soit $\mathcal{T} = \langle T, l\rangle$ un Loc-arbre complet étiqueté par $\mathsf{Strat}_{\mathcal{C}}^{+}$ accepté par $\mathcal{A}_{\mathcal{C},\ell_0}$, ainsi que par $\mathcal{A}_{\mathsf{strat}}(A)$. Alors $\mathcal{T}$ est accepté par $\mathcal{A}_{\mathsf{out}}(A)$ ssi en chaque nœud n de T, les deux énoncés suivants sont équivalents :*

1. *$p_o \in l_p(n)$*
2. *le chemin fini $\ell_0 \cdot n$ est une outcome finie de $\mathsf{strat}_D^{\mathcal{T}}$ depuis ℓ_0.*

Démonstration. Supposons que $\mathcal{T}$ est accepté par $\mathcal{A}_{\mathcal{C},\ell_0}$, $\mathcal{A}_{\mathsf{strat}}(A)$ et par $\mathcal{A}_{\mathsf{out}}(A)$. Nous allons prouver l'équivalence par récurrence sur la profondeur du nœud n.

Dans le cas d'initialisation $n = \epsilon$, $\ell_0 \cdot n$ est une outcome de n'importe quelle stratégie depuis ℓ_0. D'autre part, puisque l'état initial est $q_\in$, on a nécessairement $p_o \in l(\epsilon)$, car $\tau(q_\in, \sigma) = \bot$ si $p_o \notin \sigma$. Il y a donc équivalence entre ces deux assertions qui sont toujours vraies.

Prenons maintenant un nœud quelconque $n = m \cdot \ell$ de $\mathcal{T}$, distinct de la racine, et supposons que l'équivalence est vérifiée au niveau du prédécesseur m de n.

(1) $\Rightarrow$ (2) : On suppose que $p_o \in l_p(n)$. On remarque que $\mathcal{A}_{\mathsf{out}}(A)$ est un automate d'arbres déterministe. L'arbre d'exécution de $\mathcal{A}_{\mathsf{out}}(A)$ sur $\mathcal{T}$ est donc un arbre $\mathcal{U} = \langle T, v\rangle$ où $v\colon T \to T \times \{q_\in, q_\notin\}$. Par construction de la fonction de transition τ, comme $p_o \in l_p(n)$, on ne peut pas avoir $v(n) = (n, q_\notin)$, par conséquent $v(n) = (n, q_\in)$. Or si $v(m) = (m, q_\notin)$, alors tous les successeurs de ce nœud, et en particulier n, seraient des $q_\notin$-nœuds. On en déduit que $v(m) = (m, q_\in)$. Par induction, ceci entraîne que l'exécution finie $\ell_0 \cdot m$ est une outcome de $\mathsf{strat}_A^{\mathcal{T}}$ depuis ℓ_0. Enfin, on a forcément $\ell \in \mathsf{Next}(l(m), A)$, ce qui implique précisément que $\ell_0 \cdot n$ est également une outcome de $\mathsf{strat}_A^{\mathcal{T}}$ depuis ℓ_0.

(2) $\Rightarrow$ (1) : Supposons que $p_o \notin l_p(n)$; alors l'arbre d'exécution contient nécessairement le nœud $(n, q_\notin)$. Nous distinguons deux cas, selon que le nœud prédécesseur est $(m, q_\in)$ ou $(m, q_\notin)$. Dans le premier cas, on obtient que $\ell_0 \cdot m$ est une outcome, mais que $\ell \notin \mathsf{Next}(l(m), A)$, ce qui signifie que $\ell_0 \cdot n$ n'est pas une outcome des stratégies pour A qui sont codées dans $\mathcal{T}$. Dans le second cas, $\ell_0 \cdot m$ n'est déjà pas une outcome, donc $\ell_0 \cdot n$ ne l'est pas non plus.

Nous prouvons à présent l'implication réciproque. Soit $\mathcal{T}$ un Loc-arbre complet étiqueté par $\mathsf{Strat}_{\mathcal{C}}^{+}$, accepté par $\mathcal{A}_{\mathcal{C},\ell_0}$ et par $\mathcal{A}_{\mathsf{strat}}(A)$, et tel que les nœuds étiquetés par p_o forment précisément les outcomes de $\mathsf{strat}_A^{\mathcal{T}}$ depuis ℓ_0.

Considérons l'arbre $\mathcal{U} = \langle U, v\rangle$ où U est l'ensemble des couples $(n_i, q_i)_i$ tels que :

– les $(n_i)_i$ sont dans T

– pour tout i, $q_i = \begin{cases} q_\in & \text{si } p_o \in l_p((n_j)_{j\le i}) \\ q_\notin & \text{sinon} \end{cases}$

et v est telle que $v((n_i, q_i)_{i\le m}) = ((n_i)_{i\le m}, q_m)$.

Nous allons montrer qu'il s'agit d'un arbre d'exécution valide de $\mathcal{A}_{\mathsf{out}}(A)$ sur $\mathcal{T}$. En effet, il part de l'état initial $q_\in$, puisque ℓ_0 est une outcome de n'importe quelle stratégie depuis lui-même. Prenons maintenant un nœud $n' = (n_i, q_i)_i$ dans U, qui correspond au nœud $n = (n_i)_i$ de T. Si $p_o \in l_p(n)$, alors $v(n')$ est

$(n, q_\in)$, et $\ell_0 \cdot n$ est une outcome de $\mathsf{strat}_A^{\mathcal{T}}$ depuis ℓ_0. Par définition de $\mathsf{Next}(\sigma, A)$, pour tout $\ell \in \mathsf{Loc}$, on a $p_o \in l_p(n \cdot \ell)$ ssi $\ell \in \mathsf{Next}(l(n), A)$. Alors U contient $n' \cdot (\ell, q_\in)$ pour tout $\ell \in \mathsf{Next}(l(n), A)$, et $n' \cdot (\ell, q_\notin)$ pour tout $\ell \notin \mathsf{Next}(l(n), A)$. Autrement dit, la fonction de transition est satisfaite au niveau de ce nœud. Maintenant, si $p_o \notin l_p(n)$, alors $v(n')$ est $(n, q_\notin)$, et $\ell_0 \cdot n$ n'est pas une outcome, et donc pour tout état $\ell \in \mathsf{Loc}$, $\ell_0 \cdot n \cdot \ell$ n'est pas non plus une outcome. Il s'ensuit que U contient tous les successeurs de n' requis pour satisfaire la condition de transition. □

5.5.5 Opérations booléennes, non-déterminisation, projection, ...

Dans cette section, nous allons rappeler quelques résultats connus qui se rapportent aux automates d'arbres alternants, dont nous aurons besoin pour notre construction finale. Les trois premiers lemmes sont des résultats classiques, et nous ne donnerons une preuve que pour le quatrième.

Lemme 30. *[25, 26] Soit $\mathcal{A}$ et $\mathcal{B}$ deux $\langle S, \Sigma\rangle$-APTs qui acceptent respectivement les langages A et B. Nous pouvons construire deux $\langle S, \Sigma\rangle$-APTs $\mathcal{C}$ et $\mathcal{D}$ qui reconnaissent respectivement les langages $A \cap B$ et $\overline{A}$ (le complémentaire de A dans l'ensemble des S-arbres étiquetés par Σ). La taille et l'index de $\mathcal{C}$ valent au plus $(|\mathcal{A}| + |\mathcal{B}|)$ et $\max(\mathsf{idx}(\mathcal{A}), \mathsf{idx}(\mathcal{B})) + 1$, tandis que ceux de $\mathcal{D}$ sont $|A|$ et $\mathsf{idx}(\mathcal{A})$.*

Lemme 31. *[26] Soit $\mathcal{A}$ un $\langle S, \Sigma\rangle$-APT. On peut construire un $\langle S, \Sigma\rangle$-NPT $\mathcal{N}$ qui accepte le même langage que $\mathcal{A}$, et tel que $|\mathcal{N}| \in 2^{O\left(a \cdot \log(a)\right)}$ et $\mathsf{idx}(\mathcal{N}) \in O(a)$, où a représente $|\mathcal{A}|\mathsf{idx}(\mathcal{A})$.*

Lemme 32. *[24] Soit $\mathcal{A}$ un $\langle S, \Sigma\rangle$-NPT, avec $\Sigma = \Sigma_1 \times \Sigma_2$. Pour tout $i \in \{1, 2\}$, on peut construire un $\langle S, \Sigma\rangle$-NPT $\mathcal{B}_i$ tel que, pour tout arbre $\mathcal{T}$, on a :*

$$\mathcal{T} \in \mathcal{L}(\mathcal{B}_i) \text{ ssi } \exists \mathcal{T}' \in \mathcal{L}(\mathcal{A}).\ \mathsf{proj}_{\Sigma_i}(\mathcal{T}) = \mathsf{proj}_{\Sigma_i}(\mathcal{T}')$$

La taille et l'index de $\mathcal{B}_i$ sont les mêmes que ceux de $\mathcal{A}$.

Lemme 33. *Soit $\mathcal{A}$ un $\langle S, \Sigma \times 2^{\{p\}}\rangle$-APT tel que pour toute paire de S-arbres $\mathcal{T}$ et $\mathcal{T}'$ étiquetés par $\Sigma \times 2^{\{p\}}$ tels que $\mathsf{proj}_\Sigma(\mathcal{T}) = \mathsf{proj}_\Sigma(\mathcal{T}')$, on a $\mathcal{T} \in \mathcal{L}(\mathcal{A})$ ssi $\mathcal{T}' \in \mathcal{L}(\mathcal{A})$. On peut alors construire un $\langle S, \Sigma \times 2^{\{p\}}\rangle$-APT $\mathcal{B}$ t.q. pour tout S-arbre $\mathcal{T} = \langle T, l\rangle$ étiqueté par $\Sigma \times 2^{\{p\}}$, $\mathcal{T}$ est accepté par $\mathcal{B}$ ssi :*

$$\forall n \in T.\ \big(p \in l(n) \Leftrightarrow \mathcal{T}_n \in \mathcal{L}(\mathcal{A})\big).$$

De plus, $\mathcal{B}$ a une taille $O(|\mathcal{A}|)$ et un index $\mathsf{idx}(\mathcal{A}) + 1$.

Démonstration. La construction de $\mathcal{B}$ se fait en deux étapes : d'abord, en appliquant le Lemme 30, on construit un automate $\mathcal{D}$ qui accepte exactement les arbres de $\big(\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{P})\big) \cup \big(\mathcal{L}(\overline{\mathcal{A}}) \cap \mathcal{L}(\overline{\mathcal{P}})\big)$, où $\mathcal{P}$ est un automate qui accepte exactement les arbres dont la racine est étiquetée par p. Alors $\mathcal{D}$ a une taille $O(|\mathcal{A}|)$ et un index $\mathsf{idx}(\mathcal{A}) + 1$. De plus, un arbre $\mathcal{T} = \langle T, l\rangle$ est accepté par $\mathcal{D}$ ssi :

$$p \in l(\epsilon) \Leftrightarrow \mathcal{T} \in \mathcal{L}(\mathcal{A})$$

Puis $\mathcal{B}$ est obtenu en fourchant une exécution de $\mathcal{D}$ à chaque nœud : formellement, si $\mathcal{D} = \langle Q_D, q_D, \tau_D, \Omega_D \rangle$, alors $\mathcal{B} = \langle Q_D \cup \{q_B\}, q_B, \tau_B, \Omega_B \rangle$ (on suppose $q_B \notin Q_D$) où τ_B et Ω_B coïncident respectivement avec τ_D et Ω_D sur Q_D. On complète ces fonctions en posant $\Omega_B(q_B) = 0$ (ou n'importe quelle autre entier pair), et :

$$\tau_B(q_B, \sigma) = \tau_D(q_D, \sigma) \wedge \bigwedge_{\ell \in \mathsf{Loc}} (\ell, q_B)$$

Nous allons montrer que cet automate $\mathcal{B}$ accepte bien les arbres annoncés. Si un S-arbre $\mathcal{T} = \langle T, l \rangle$ étiqueté par $\Sigma \times 2^{\{p\}}$ est accepté par $\mathcal{B}$, alors pour tout nœud n de T, tout arbre d'exécution contient un nœud étiqueté par (n, q_B). Depuis un tel nœud, $\mathcal{B}$ fourche de nouvelles branches aux q_B-nœuds d'une part, et reproduit le comportement de $\mathcal{D}$ depuis q_D d'autre part. Ce dernier point implique que le sous-arbre $\mathcal{T}_n$ est accepté par $\mathcal{D}$, ce qui justifie que l'on ait l'équivalence voulue.

Réciproquement, soit $\mathcal{T}$ un S-arbre étiqueté par $\Sigma \times 2^{\{p\}}$ qui satisfait la partie droite de l'équivalence. Alors pour tout nœud n de $\mathcal{T}$, le sous-arbre $\mathcal{T}_n$ est accepté par $\mathcal{D}$, et admet par conséquent un arbre d'exécution acceptant. A partir de ces arbres d'exécution, il est facile de construire un arbre d'exécution acceptant de $\mathcal{B}$ sur $\mathcal{T}$. □

5.5.6 Construire un automate d'arbres alternant à partir d'une formule de $\mathsf{ATL}_{sc,\infty}$

Lemme 34. *Soit $\mathcal{C}$ une CGS avec un ensemble fini d'états Loc. Soit ψ une formule de $\mathsf{ATL}_{sc,\infty}$, et $D \subseteq \mathsf{Agt}$ une coalition. On peut construire un $\langle \mathsf{Loc}, \mathsf{Strat}^+_{\mathcal{C}} \rangle$-APT $\mathcal{A}_{\psi,D}$ t.q.*

- *pour tout Loc-arbre complet étiqueté par $\mathsf{Strat}^+_{\mathcal{C}}$ $\mathcal{T}$ accepté par $\mathcal{A}_{\mathcal{C}}$ et par $\mathcal{A}_{\mathsf{strat}}(D)$, on a :*

$$\mathcal{T} \in \mathcal{L}(\mathcal{A}_{\psi,D}) \quad \Leftrightarrow \quad \big(\mathcal{C}, l_{\mathsf{Loc}}(\epsilon)\big) \models_{\mathsf{strat}^{\mathcal{T}}_D} \psi$$

- *pour toute paire de Loc-arbres complets étiquetés par $\mathsf{Strat}^+_{\mathcal{C}}$, $\mathcal{T}$ and $\mathcal{T}'$, t.q. $\mathsf{proj}_{\mathsf{Strat}'_{\mathcal{C}}}(\mathcal{T}) = \mathsf{proj}_{\mathsf{Strat}'_{\mathcal{C}}}(\mathcal{T}')$, avec $\mathsf{Strat}'_{\mathcal{C}} = \mathsf{Strat}_{\mathcal{C}} \times (\mathcal{M} \cup \{\bot\})^{\mathsf{Agt}}$, on a :*

$$\mathcal{T} \in \mathcal{L}(\mathcal{A}_{\psi,D}) \quad \Leftrightarrow \quad \mathcal{T}' \in \mathcal{L}(\mathcal{A}_{\psi,D}).$$

La taille de $\mathcal{A}_{\psi,D}$ est au plus d-exponentielle, où d représente le nombre de quantificateurs emboîtés de stratégies dans ψ. Son index est $(d-1)$-exponentiel.

Démonstration. Nous allons procéder par induction sur la structure de la formule ψ. Le cas des propositions atomiques est facile. En appliquant le Lemme 30, on obtient immédiatement le résultat dans les cas où φ est une combinaison booléenne de sous-formules.

Nous donnons une idée de la preuve pour le cas $\psi = \ll A \gg \mathbf{X}\,\varphi$. Les cas des formules $\ll A \gg \varphi_1\,\mathbf{U}\,\varphi_2$ et $\ll A \gg \varphi_1\,\mathbf{W}\,\varphi_2$ peuvent se démontrer de manière similaire, en utilisant les deux propositions atomiques p_l et p_r.

L'idée de la construction est la suivante : on utilise les automates

1. $\mathcal{A}_{\mathsf{out}}(D \cup A)$ pour étiqueter les outcomes avec p_o
2. $\mathcal{A}_{\varphi, D\cup A}$ pour étiqueter les nœuds qui satisfont φ
3. un automate intermédiaire $\mathcal{A}_f$ qui vérifie que toutes les outcomes satisfont $\mathbf{X}\,\varphi$
4. enfin on projette selon la stratégie de la coalition A afin d'obtenir notre nouvel automate pour $\ll A \gg \mathbf{X}\,\varphi$.

Supposons qu'on ait déjà construit l'automate $\mathcal{A}_{\varphi, D\cup A}$ (c'est l'hypothèse d'induction). En appliquant le Lemme 33 à $\mathcal{A}_{\varphi, D\cup A}$ et en notant p_r la proposition supplémentaire, on obtient un automate $\mathcal{B}_{p_r,\varphi,D\cup A}$ tel que, si un arbre $\mathcal{T} = \langle T, l\rangle$ est accepté par $\mathcal{A}_{\mathcal{C}}$ et par $\mathcal{A}_{\mathsf{strat}}(D \cup A)$, il sera également accepté par $\mathcal{B}_{p_r,\varphi,D\cup A}$ ssi :

$$\forall n \in T.\Big(\; p_r \in l(n) \Leftrightarrow \big(\mathcal{C}, l_{\mathsf{Loc}}(n)\big) \models_{\mathsf{strat}^{\mathcal{T}_n}_{D\cup A}} \varphi\Big) \tag{5.1}$$

Pour vérifier que toutes les outcomes satisfont $\mathbf{X}\,\varphi$, il ne nous reste plus qu'à construire un automate $\mathcal{A}_f$ qui vérifie la propriété de $\mathsf{CTL}^\star$ $\mathbf{A}(\mathbf{G}\,p_o \rightarrow \mathbf{X}\,p_r)$. Voir [19] pour cette construction classique. Cet automate $\mathcal{A}_f$ a la propriété suivante : pour tout Loc-arbre étiqueté par $\mathsf{Strat}^+_{\mathcal{C}}$ $\mathcal{T} = \langle T, l\rangle$, on a

$$\mathcal{T} \in \mathcal{L}(\mathcal{A}_f) \quad \Leftrightarrow \quad (\mathcal{T}, \epsilon) \models \mathbf{A}(\mathbf{G}\,p_o \rightarrow \mathbf{X}\,p_r) \tag{5.2}$$

A présent soit $\mathcal{H}$ le produit des automates $\mathcal{A}_{\mathsf{strat}}(A)$, $\mathcal{A}_{\mathsf{out}}(D \cup A)$, $\mathcal{A}_f$ et $\mathcal{B}_{p_r,\varphi,D\cup A}$, et soit $\mathcal{T}$ un arbre accepté par $\mathcal{A}_{\mathcal{C}}$ et par $\mathcal{A}_{\mathsf{strat}}(D)$. Si $\mathcal{T}$ est accepté par $\mathcal{H}$, alors $D \cup A \subseteq \mathsf{dom}(\mathcal{T})$ et toutes les outcomes de la stratégie $\mathsf{strat}^{\mathcal{T}}_{D\cup A}$ depuis $l_{\mathsf{Loc}}(\epsilon)$ satisfont $\mathbf{X}\,\varphi$.

La réciproque n'est pas vraie en général, mais nous allons montrer quelque chose de plus faible : à partir de $\mathcal{T} = \langle T, l\rangle$, accepté par $\mathcal{A}_{\mathcal{C}}$ et $\mathcal{A}_{\mathsf{strat}}(D)$, et tel que $D \cup A \subseteq \mathsf{dom}(\mathcal{T})$ et les outcomes de $\mathsf{strat}^{\mathcal{T}}_{D\cup A}$ depuis $l_{\mathsf{Loc}}(\epsilon)$ satisfont $\mathbf{X}\,\varphi$, on construit $\mathcal{T}' = \langle T, l'\rangle$ tel que $\mathsf{proj}_{\mathsf{Strat}'_{\mathcal{C}}}(\mathcal{T}) = \mathsf{proj}_{\mathsf{Strat}'_{\mathcal{C}}}(\mathcal{T}')$, et $\mathcal{T}'$ est accepté par $\mathcal{H}$.

Pour cela, il suffit de modifier les étiquetages de $\mathcal{T}$ par p_o et p_r, de façon à ce que $\mathcal{T}'$ soit accepté par $\mathcal{A}_{\mathsf{out}}(D \cup A)$ et $\mathcal{B}_{p_r,\varphi,D\cup A}$. Cela garantit que $\big(\mathcal{C}, l_{\mathsf{Loc}}(\epsilon)\big) \models_{\mathsf{strat}^{\mathcal{T}'}_{D\cup A}} \ll \varnothing \gg \mathbf{X}\,\varphi$, et que $\mathcal{T}'$ est aussi accepté par $\mathcal{A}_f$. Finalement, on obtient l'équivalence :

pour tout arbre $\mathcal{T} = \langle T, l\rangle$ accepté par $\mathcal{A}_{\mathcal{C}}$ et par $\mathcal{A}_{\mathsf{strat}}(D)$,

$$\begin{aligned} D \cup A \subseteq \mathsf{dom}(\mathcal{T}) \quad \text{et} \quad \big(\mathcal{C}, l_{\mathsf{Loc}}(\epsilon)\big) \models_{\mathsf{strat}^{\mathcal{T}}_{D\cup A}} \ll \varnothing \gg \mathbf{X}\,\varphi \qquad \Leftrightarrow \\ \exists \mathcal{T}' \text{ t.q. } \mathsf{proj}_{\mathsf{Strat}'_{\mathcal{C}}}(\mathcal{T}') = \mathsf{proj}_{\mathsf{Strat}'_{\mathcal{C}}}(\mathcal{T}) \text{ et } \mathcal{T}' \in \mathcal{L}(\mathcal{H}) \end{aligned} \tag{5.3}$$

A présent, en appliquant le Lemme 31, on obtient un $\langle \mathsf{Strat}^+_{\mathcal{C}}, \mathsf{Loc}\rangle$-NPT $\mathcal{N}$ tel que $\mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{H})$. On peut alors appliquer le Lemme 32 pour $\mathsf{Strat}^+_{\mathcal{C}} =$

$(\mathsf{Strat}_{\mathcal{C}} \times (\mathcal{M} \cup \{\bot\})^{\mathsf{Agt} \smallsetminus A}) \times ((\mathcal{M} \cup \{\bot\})^A \times 2^{\{p_o, p_l, p_r\}})$ au NPT $\mathcal{N}$; le nouvel automate $\langle \mathsf{Strat}_{\mathcal{C}}^+, \mathsf{Loc} \rangle$-NPT $\mathcal{P}$ accepte tous les arbres $\mathcal{T}$ dont l'étiquetage par $(\mathcal{M} \cup \{\bot\})^A \times 2^{\{p_o, p_l, p_r\}}$ peut être modifié pour que l'arbre soit accepté par $\mathcal{N}$.

Alors $\mathcal{P}$ satisfait les deux propriétés du Lemme 34 :

– la seconde propriété est une conséquence directe de l'usage du Lemme 32.

– pour la première, prenons $\mathcal{T} = \langle T, l \rangle$ accepté par $\mathcal{A}_{\mathcal{C}}$ et par $\mathcal{A}_{\mathsf{strat}}(D)$. Nous établissons une équivalence.

($\Rightarrow$) : si $\mathcal{T}$ est accepté par $\mathcal{P}$, alors d'après le Lemme 32, il existe un arbre $\mathcal{T}' = \langle T, l' \rangle$, qui a le même étiquetage que $\mathcal{T}$ sur $\mathsf{Strat}_{\mathcal{C}} \times (\mathcal{M} \cup \{\bot\})^{\mathsf{Agt} \smallsetminus A})$, et qui est accepté par $\mathcal{N}$. On déduit de (5.3), et du fait que $\mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{H})$:

$$D \cup A \subseteq \mathsf{dom}(\mathcal{T}') \text{ et } \big(\mathcal{C}, l_{\mathsf{Loc}}(\epsilon)\big) \models_{\mathsf{strat}_{D \cup A}^{\mathcal{T}'}} \ll \varnothing \gg \mathbf{X}\, \varphi$$

Par conséquent $\mathsf{strat}_A^{\mathcal{T}'}$ est une stratégie pour la coalition A, qui témoigne du fait que

$$\big(\mathcal{C}, l_{\mathsf{Loc}}(\epsilon)\big) \models_{\mathsf{strat}_D^{\mathcal{T}'}} \ll A \gg \mathbf{X}\, \varphi$$

Cela prouve le résultat annoncé puisque $\mathsf{strat}_D^{\mathcal{T}} = \mathsf{strat}_D^{\mathcal{T}'}$.

($\Leftarrow$) : si $\big(\mathcal{C}, l_{\mathsf{Loc}}(\epsilon)\big) \models_{\mathsf{strat}_D^{\mathcal{T}}} \ll A \gg \mathbf{X}\, \varphi$, alors on peut modifier l'étiquetage de $\mathcal{T}$ avec une stratégie témoignante pour A, et on obtient un arbre $\mathcal{T}'$ tel que $\big(\mathcal{C}, l_{\mathsf{Loc}}(\epsilon)\big) \models_{\mathsf{strat}_{D \cup A}^{\mathcal{T}'}} \ll \varnothing \gg \mathbf{X}\, \varphi$. D'après (5.3), $\mathcal{T}'$ peut à son tour être modifié en un arbre $\mathcal{T}''$, avec $\mathsf{proj}_{\mathsf{Strat}'_{\mathcal{C}}}(\mathcal{T}'') = \mathsf{proj}_{\mathsf{Strat}'_{\mathcal{C}}}(\mathcal{T}')$, de telle façon que $\mathcal{T}'' \in \mathcal{L}(\mathcal{H})$. Enfin, comme les projections de $\mathcal{T}''$ et de $\mathcal{T}$ coïncident sur $(\mathsf{Strat}_{\mathcal{C}} \times (\mathcal{M} \cup \{\bot\})^{\mathsf{Agt} \smallsetminus A})$, on a que $\mathcal{T}$ est accepté par $\mathcal{P}$. Ceci termine la preuve pour le cas $\ll A \gg \mathbf{X}\, \varphi$.

Enfin, si $\psi = \rangle A \langle\, \varphi$, on construit $\mathcal{A}_{\rangle A \langle\, \varphi, D} = \mathcal{A}_{\varphi, D \smallsetminus A}$, dont on peut facilement prouver qu'il satisfait les deux points requis.

A moins qu'on soit dans le cas où A est la coalition vide, la construction de l'automate pour $\ll A \gg \mathbf{X}\, \varphi$ (aussi bien que celles de $\ll A \gg \varphi_1\, \mathbf{U}\, \varphi_2$ et de $\ll A \gg \varphi_1\, \mathbf{W}\, \varphi_2$) entraîne une explosion exponentielle du nombre d'états par rapport aux nombres d'états et aux index des automates des sous-formules ; de plus, son index est bilinéaire en fonction de la taille et de l'index de ces automates. On en déduit que, pour une formule qui contient d quantificateurs non vides de stratégies, l'automate a pour taille une d-exponentielle et pour index une $(d-1)$-exponentielle. □

Corollaire 8. *Etant donnés une formule φ de $\mathsf{ATL}_{sc,\infty}$, une CGS $\mathcal{C}$ et un état ℓ_0 de $\mathcal{C}$, on peut construire un automate d'arbre de parité alternant $\mathcal{A}$ t.q.*

$$\mathcal{L}(\mathcal{A}) \neq \varnothing \qquad \Leftrightarrow \qquad (\mathcal{C}, \ell_0) \models_{\varnothing} \varphi$$

De plus, $\mathcal{A}$ a une taille d-exponentielle et un index $(d-1)$-exponentiel, où d est le nombre de quantificateurs de stratégies non vides emboîtés.

Démonstration. Il suffit de prendre le produit de l'automate $\mathcal{A}_{\varphi,\varnothing}$ (donné par le Lemme 34) avec $\mathcal{A}_{\mathcal{C},\ell_0}$. Dans le cas où cet $\langle \mathsf{Loc}, \mathsf{Strat}_{\mathcal{C}}^+ \rangle$-APT accepte un arbre $\mathcal{T}$, le Lemme 34 implique que $\big(\mathcal{C}, \ell_0\big) \models_{\varnothing} \varphi$. Réciproquement, si $\big(\mathcal{C}, \ell_0\big) \models \varphi$, alors l'arbre de déroulement étendu $\mathcal{T} = \langle T, l \rangle$ de $\mathcal{C}$ depuis ℓ_0 dans lequel $l_{\mathsf{str}}(n) = \bot$

pour tout $n \in T$ est accepté par $\mathcal{A}_{\mathcal{C},\ell_0}$ (et, trivialement, par $\mathcal{A}_{\mathsf{strat}}(\varnothing)$), et d'après le Lemme 34, il est également accepté par$\mathcal{A}_{\varphi,\varnothing}$. □

Démonstration du Théorème 16. La première assertion du Théorème 16 se prouve facilement, puisque le vide du langage reconnu par un automate d'arbres alternant de parité $\mathcal{A}$ se vérifie en temps $\exp\big(O(|\mathcal{A}| \times \mathsf{idx}(\mathcal{A}))\big)$.

Pour la deuxième affirmation, remarquons que la taille et l'index de $\mathcal{A}_{\varphi,\varnothing}$ dans la preuve du Corollaire 8 ne dépend pas de la CGS $\mathcal{C}$. Par conséquent l'automate $\mathcal{A}$ du Corollaire 8 a une taille linéaire en fonction de $|\mathcal{C}|$, et s'effectue en un temps exponentiel en fonction de $|\mathcal{C}|$ (car $\mathcal{A}_{out}(D)$ requiert le calcul de $\mathsf{Next}(\sigma, D)$). Le fait que le langage soit non vide est alors vérifié en un temps exponentiel en fonction de $|\mathcal{C}|$. □

Notre algorithme peut facilement être modifié afin de s'appliquer à $\mathsf{ATL}^*_{sc,\infty}$. Une solution est de s'appuyer sur le Corollaire 3, mais notre traduction de $\mathsf{ATL}^*_{sc,\infty}$ vers $\mathsf{ATL}_{sc,\infty}$ peut doubler le nombre de quantificateurs emboîtés de stratégies non vides. L'algorithme se retrouverait par conséquent dans $(2k+1)$-EXPTIME, où k est le nombre de quantificateurs de stratégies emboîtés. Une autre solution est d'adapter notre construction, en replaçant chaque sous-formule d'état par une nouvelle proposition atomique, et de construire un automate $\mathcal{A}_f$ pour une formule quelconque de $\mathsf{CTL}^\star$, éventuellement plus compliquée. Cela constituerait un algorithme $(k+1)$-EXPTIME. Dans les deux cas, la complexité en programme reste la même, dans EXPTIME.

De plus, notre algorithme pourrait être modifié pour traiter SL [12]. Une différence importante est que SL peut nécessiter de stocker plusieurs stratégies d'un même joueur dans l'arbre, tandis que $\mathsf{ATL}_{sc,\infty}$ ne stocke dans le contexte qu'une stratégie par joueur. On devrait donc construire une version modifiée de la fonction Next que l'on utilise pour construire $\mathcal{A}_{\mathsf{out}}(D)$, où il faudrait indiquer explicitement *quelle* stratégie considérer pour chaque joueur.

5.6 Dureté du problème MC $\mathsf{ATL}^*_{sc,\infty}$

Nous allons à présent formuler une borne inférieure pour la difficulté de ce problème.

Théorème 17. *Le model-checking de $\mathsf{ATL}^*_{sc,\infty}$ est k-EXPSPACE-dur pour n'importe quel entier k.*

Pour montrer cela, nous allons effectuer une réduction depuis le problème de satisfaisabilité de la logique QPTL (pour "quantified propositionnal temporal logic"), dont nous rappelons ici la définition [34].

Définition 49. *La syntaxe de QPTL est définie par la grammaire suivante :*

$$\mathsf{QPTL} \ni \varphi_p \quad ::= \quad p \mid \neg\varphi_p \mid \varphi_p \vee \psi_p \mid \mathbf{X}\,\varphi_p \mid \mathbf{F}\,\varphi_p \mid \exists p \varphi_p$$

où p décrit l'ensemble AP.

On dit de deux exécutions λ et λ' qu'elles sont *p-différentes* si elles ont le même étiquetage, sauf (éventuellement) en ce qui concerne la proposition p.

La sémantique de QPTL est définie comme suit :

Soit λ une exécution.

$$\begin{array}{rcl} \lambda \models p & \text{ssi} & p \in \mathsf{Lab}(\lambda[0]) \\ \lambda \models \neg\varphi_p & \text{ssi} & \lambda \not\models \varphi_p \\ \lambda \models \varphi_p \vee \psi_p & \text{ssi} & \lambda \models \varphi_p \text{ ou } \lambda \models \psi_p \\ \lambda \models \mathbf{X}\,\varphi_p & \text{ssi} & \lambda[1,\infty] \models \varphi_p \\ \lambda \models \mathbf{F}\,\varphi_p & \text{ssi} & \exists i \geq 0.\ \lambda[i,\infty] \models \varphi_p \\ \lambda \models (\exists p)\varphi_p & \text{ssi} & \exists \lambda'\ p\text{-différente de } \lambda,\ \lambda' \models \varphi_p \end{array}$$

On utilisera $(\forall p)\varphi$ comme abréviation de $\neg(\exists p)\neg\varphi$.

On dira d'une formule de QPTL qu'elle est *sous forme normale* si elle s'écrit :

$$(Q_1 p_1)(Q_2 p_2)...(Q_n p_n)\varphi,$$

où chaque Q_i représente soit $\forall$ soit $\exists$, et φ est une formule sans quantificateurs. On sait que toute formule de QPTL est équivalente à une formule sous forme normale.

Théorème 18. *Le problème de satisfaisabilité d'une formule φ (sous forme normale) de QPTL est dans k-EXPSPACE, où k représente le nombre d'alternation de quantificateurs de φ.*

Proposition 5. *Soit φ une formule de QPTL sous forme normale. On peut réduire le problème de satisfaisabilité de φ au problème de model-checking d'une certaine formule Φ de $\mathsf{ATL}^*_{sc,\infty}$, sur une certaine structure $\mathcal{C}$. De plus, la taille de Φ, et celle de la structure $\mathcal{C}$, sont toutes les deux linéaires en fonction de celle de φ.*

Démonstration. Soit $\varphi = (Q_1 p_1)(Q_2 p_2)(Q_3 p_3)...(Q_n p_n)\psi$, où $\psi \in \mathsf{LTL}$ et les Q_i sont des quantificateurs existentiels ou universels, une formule de QPTL sous forme normale.

A cette formule, nous associons une *CGS* $\mathcal{C}$ et une formule Φ de $\mathsf{ATL}^*_{sc,\infty}$ de la manière suivante :

1. $\mathsf{Agt} = \{P_1, ..., P_n\} \cup \{P\}$, où n est le nombre d'alternations de quantificateurs de φ ;
2. L'ensemble d'états Loc est une réunion de trois sous-ensembles d'états : à chaque *proposition atomique* p_i, on associe un *état* t_i, un état $\overline{t_i}$ et un état s_i. De plus, on ajoute un état initial s_0 :
$$\mathsf{Loc} = \{t_i \mid\ 1 \leq i \leq n\} \cup \{\overline{t_i} \mid\ 1 \leq i \leq n\} \cup \{s_i \mid\ 0 \leq i \leq n\}\,;$$
3. L'ensemble des propositions atomiques AP est constitué des propositions p_i et $\overline{p_i}$, et d'une proposition supplémentaire s ;

$$\mathsf{AP} = \{p_i \mid \ 1 \leq i \leq n\} \cup \{\overline{p_i} \mid \ 1 \leq i \leq n\}\,;$$

4. On définit trois mouvements différents : $\mathcal{M} = \{1, 2, 3\}$;
5. On étiquette les états t_i par p_i, les états $\overline{t_i}$ par $\overline{p_i}$, et s_0 par s. Les autres états s_i ne sont pas étiquetés.

$$\mathsf{Lab}(t_i) = \{p_i\},$$
$$\mathsf{Lab}(\overline{t_i}) = \{\overline{p_i}\},$$
$$\mathsf{Lab}(s_0) = \{s\}.$$

6. La CGS est une structure *turn-based*, où le joueur P contrôle s_0, et chaque état s_i est contrôlé par le joueur P_i. Aucun autre état n'est contrôlé :

$$\forall 1 \leq i, k \leq n, \mathsf{Mov}(t_i, P_k) = \mathsf{Mov}(\overline{t_i}, P_k) = \mathsf{Mov}(s_0, P_k) = \{1\},$$
$$\mathsf{Mov}(s_0, P) = \{1, 2\},$$
$$\forall 1 \leq i, k \leq n, \mathsf{Mov}(s_i, P) = \{1\},$$
$$\forall 1 \leq i \leq n, \forall 1 \leq k \leq n, \mathsf{Mov}(s_i, P_k) = \begin{cases} \{1, 2\} & \text{si } i = k \\ \{1\} & \text{sinon} \end{cases}$$

7. Edg relie s_0 à tous les s_i (y compris lui-même), et les autres s_i sont reliés à la fois à $\overline{t_i}$ par le mouvement 1 et à t_i par le mouvement 2 (voir la Figure. 5.4).

Dans la Figure 5.4, les vecteurs de mouvements ne sont pas indiqués pour ne pas alourdir le dessin. Nous n'avons pas fait figurer de propositions atomiques pour la même raison.

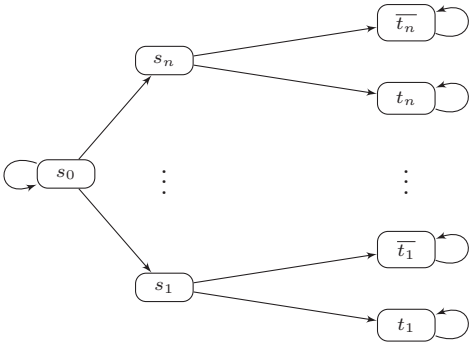

FIG. 5.4 – Une représentation simplifiée de la CGS $\mathcal{C}$

A partir d'une formule ψ de LTL, on construit Υ en remplaçant toute proposition atomique $p_i \in \mathsf{AP}$ par $\ll P \gg \mathbf{X}\,\mathbf{X}\,p_i$.

Par exemple, la formule $\mathbf{G}\,(p_2 \Leftrightarrow \mathbf{X}\,p_1)$ sera transformée en :

$$\mathbf{G}\,(\ll P \gg \mathbf{X}\,\mathbf{X}\,p_2 \Leftrightarrow \mathbf{X}\ \ll P \gg \mathbf{X}\,\mathbf{X}\,p_1)$$

Sur la structure $\mathcal{C}$, on définit une bijection entre les étiquetages d'un chemin nu (c'est-à-dire sans aucun étiquetage) λ par toute proposition atomique p_i, et l'ensemble des stratégies du joueur correspondant P_i. Pour tout entier k, on pose :

$$F_{|P_i}(s_0^k s_i) = \begin{cases} 2 & \text{si } p_i \in \lambda[k] \\ 1 & \text{sinon.} \end{cases}$$

On notera λ_F le chemin associé par cette bijection à un contexte stratégique F. D'autre part, appelons λ_0 l'exécution de la structure $\mathcal{C}$ qui reste en s_0.

Quel que soit le contexte stratégique F, on a :

$$\lambda_F \models \psi \Leftrightarrow (\mathcal{C}, \lambda_0) \models_F \Upsilon$$

On montre cela par induction sur les formules de LTL.

On ne détaille que le cas des propositions atomiques, les étapes d'induction étant faciles. Si $\psi = p_i$, on veut montrer :

$$\lambda_F \models p_i \Leftrightarrow (\mathcal{C}, \lambda_0) \models_F \ll P \gg \mathbf{X}\,\mathbf{X}\, p_i$$

On sait par définition que si $\lambda_F \models p_i$, alors $F_{|P_i}(s_i) = t_i$. Et sous ce contexte, la stratégie du joueur P qui bouge en s_i depuis l'état s_0, garantit l'objectif $\mathbf{X}\,\mathbf{X}\, p_i$. Dans le cas où $\lambda_F \not\models p_i$, on a $F_{|P_i}(s_i) = \overline{t_i}$, et P ne peut pas garantir $\mathbf{X}\,\mathbf{X}\, p_i$ sous ce contexte.

Maintenant, à partir d'une formule φ de QPTL sous forme normale, on obtient Φ en :

1. remplaçant la partie non quantifiée ψ par $\ll P \gg (\mathbf{G}\, s \wedge \Upsilon)$
2. remplaçant les $(\exists p_i)$ par $\ll P_i \gg$

Par exemple, la formule $\forall p_1 \exists p_2 \mathbf{G}\,(p_2 \Leftrightarrow \mathbf{X}\, p_1)$ sera transformée en :

$$\neg \ll P_1 \gg \neg \ll P_2 \gg \ll P \gg \Big(\mathbf{G}\, s \wedge \mathbf{G}\,(\ll P \gg \mathbf{X}\,\mathbf{X}\, p_2 \Leftrightarrow \mathbf{X}\ \ll P \gg \mathbf{X}\,\mathbf{X}\, p_1)\Big)$$

On veut maintenant montrer que pour tout contexte stratégique F, on a :

$$\lambda_F \models \varphi \Leftrightarrow (\mathcal{C}, s_0) \models_F \Phi$$

Prouvons cela par récurrence sur le rang de quantification n de φ. Si φ n'est pas quantifiée, c'est une formule de LTL, et il s'agit de prouver que

$$\lambda_F \models \varphi \Leftrightarrow (\mathcal{C}, s_0) \models_F \ll P \gg (\mathbf{G}\, s \wedge \Upsilon)$$

C'est une conséquence directe du résultat précédent car λ_0 est la seule exécution qui vérifie $\mathbf{G}\, s$.

Maintenant prouvons l'hérédité : $\varphi = \exists p_k.\ \varphi'$. Il s'agit de montrer :

$$\lambda_F \models \exists p_k.\ \varphi' \Leftrightarrow (\mathcal{C}, s_0) \models_F \ll P_k \gg \Phi'$$

C'est-à-dire que l'existence d'un étiquetage par p_k est équivalente à l'existence d'une stratégie pour le joueur P_k. Il suffit pour cela de nous servir de notre bijection entre les étiquetages et les stratégies.

Finalement, on a prouvé que :

$$\varphi \text{ est satisfaisable} \Leftrightarrow (\mathcal{C}, s_0) \models \Phi$$

□

Nous avons comme corollaire de ce théorème que :

Corollaire 9. *Le problème de model-checking de $\mathsf{ATL}_{sc,\infty}$ est non élémentaire*

De cela, du théorème 9, et du fait que $\mathsf{ATL}_{sc,\infty} \leq_{ex} \mathsf{ATL}^*_{sc,\infty} \setminus \rangle \cdot \langle$ par une injection canonique, on déduit que le problème de model-checking de la logique SL est également non élémentaire. Ceci contredit un résultat de [23], où un algorithme 2EXPTIME est proposé pour ce problème. Il est clair que cet algorithme n'est correct que pour un fragment de la logique proposée.

Chapitre 6

Conclusion

Initialement, la problématique qui sous-tendait ce travail s'est formée sur la constatation d'une limite des logiques ATL et ATL*, inhérente à leur sémantique. L'emboîtement de quantificateurs de stratégie revêtait un sens que l'on jugeait discutable, et il nous a semblé pertinent de proposer une sémantique qui assimilerait les stratégies à des contextes, et les traiteraient comme tels lors de l'évaluation de la valeur de vérité d'une formule.

De nombreuses études sont parues pendant la durée de ce travail, et ont abordé plus ou moins directement le problème des contextes stratégiques. Par leur intermédiaire, des solutions différentes ont été soumises à la communauté scientifique. Certaines nuances les distinguent, et un travail fécond consisterait à comparer précisément leurs formalismes, saisir leurs qualités relatives, pour déterminer le compromis le plus satisfaisant possible.

Pour ce qui est de notre apport personnel, par rapport à la logique ATL, qui fut notre point de départ, nos ajouts de plusieurs quantificateurs de stratégies ont beaucoup augmenté l'expressivité. Cette modification est conséquente, et permet de spécifier des propriétés dont l'utilité est avérée, notamment l'existence d'un équilibre de Nash. De plus, de nombreux problèmes naturels de contrôle relèvent de notre sémantique plutôt que de celle d'ATL.

Nous avons également défini une notion de mémoire de stratégie comme nombre entier, et notre sémantique permet des quantifications explicites sur des stratégies à mémoire bornée. Si cela peut refléter certaines contraintes matérielles, en particulier la limitation des ressources, il serait probablement intéressant d'explorer les issues théoriques de notre définition formelle de la mémoire. C'est-à-dire d'exhiber des paramètres théoriques pertinents, de la structure ou bien de la formule, à partir desquels on pourrait calculer une limite sur la mémoire suffisante d'une stratégie.

Un fait remarquable est que sous le paradigme des contextes stratégiques, le fragment sans étoile a la même expressivité que la logique complète, dans le cas de la mémoire infinie.

La relation d'équivalence définie par "les deux états vérifient exactement les mêmes formules de ATL_{sc}" est strictement plus fine que la bisimulation alternante. Nous avons donc affaire à une nouvelle relation qu'il serait intéressant de caractériser. Dans ce sens, nous avons cherché à caractériser la relation binaire "vérifier les mêmes formules de GL", et proposé une nouvelle définition qui constitue selon nous un premier pas vers la relation associée à ATL_{sc}.

Dans le cadre de la mémoire non bornée, nous avons démontré que les problèmes de model-checking ont une complexité non élémentaire, et apparaissent donc absolument irréalisables en pratique, même en se restreignant au fragment sans étoile. Cela constituait pourtant un recours habituel pour réduire la difficulté du model-checking (ATL, CTL). Il semble y avoir des liens très profonds entre les logiques où l'on peut quantifier sur les propositions atomiques d'une part (par exemple QPTL), et nos logiques avec quantifications stratégiques sans restrictions sur la mémoire d'autre part. Ceux-ci restent à établir avec précision.

Nous avons également prouvé que nos logiques avec contextes stratégiques sans mémoire, elles, ont leur problème de model-checking qui est PSPACE-complet.

En ce qui concerne les logiques avec quantifications sur des stratégies à mémoire bornée par un entier k (codé en binaire), nous avons montré que leur problème de model-checking est dans k-EXPSPACE. En revanche, nous ne sommes pas en mesure de fournir une borne inférieure autre que la PSPACE-dureté qui est une conséquence du résultat sur les stratégies sans mémoire.

Une autre série de problèmes sur lesquels nous pourrions nous pencher sont ceux qui ont trait à la *satisfaisabilité* de nos logiques. Il serait notamment intéressant de comprendre comment se comporte le degré de branchement minimal d'un arbre satisfaisant une formule donnée, en fonction de la taille de cette formule.

Bibliographie

[1] Ågotnes, Thomas and Goranko, Valentin and Jamroga, Wojciech. Alternating-time Temporal Logics with Irrevocable Strategies. In Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge (TARK'07), pages 15-24, Samet, Dov, 2007

[2] Ågotnes, Thomas and Goranko, Valentin and Jamroga, Wojciech. Strategic Commitment and Release in Logics for Multi-Agent Systems. Technical Report, Clausthal U. of Technology, Germany, 2008

[3] Alur, Rajeev and Henzinger, Thomas A. and Kupferman, Orna. Alternating-time Temporal Logic. In Journal of the ACM, pages 672-713, ACM Press, 2002

[4] Alur, Rajeev and Henzinger, Thomas A. and Kupferman, Orna and Vardi, Moshe Y. Alternating Refinement Relations. In Proceedings of the 9th International Conference on Concurrency Theory (CONCUR'98), pages 163-178, Springer, 1998

[5] Baier, Christel and Brázdil, Tomáš and Größer, Marcus and Kučera, Antonín. Stochastic Game Logic. In Proceedings of the 4th International Conference on Quantitative Evaluation of Systems (QEST'07), pages 227-236, IEEE Comp. Soc. Press, 2007

[6] Christel Baier and Joost-Pieter Katoen. Principles of model checking, MIT Press, 2008

[7] Berard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., Schnoebelen, P. Systems and Software Verification, 1999

[8] Brihaye, Thomas and Da Costa, Arnaud and Laroussinie, François and Markey, Nicolas. ATL with Strategy Contexts and Bounded Memory. In Proceedings of the Symposium on Logical Foundations of Computer Science (LFCS'09), pages 92-106, Springer, 2009

[9] Brihaye, Thomas and Da Costa, Arnaud and Laroussinie, François and Markey, Nicolas. ATL with Strategy Contexts and Bounded Memory, LSV, 2009

[10] K. Chatterjee, T. A. Henzinger, and M. Jurdzinski. Games with secure equilibria. In *Theoretical Computer Science, pages 67-82, 2006*

[11] Chatterjee, Krishnendu and Henzinger, Thomas A. and Piterman, Nir. Strategy Logic. In Proceedings of the 18th International Conference on Concurrency Theory (CONCUR'07), pages 59-73, Springer-Verlag, 2007

[12] Chatterjee, Krishnendu and Henzinger, Thomas A. and Piterman, Nir. Strategy Logic. In Inf. & Comp., pages 677-693, Elsevier Science, 2010

[13] Clarke, Edmund M. and Emerson, E. Allen. Design and Synthesis of Synchrization Skeletons using Branching-Time Temporal Logic. In Proceedings of the 3rd Workshop on Logics of Programs (LOP'81), pages 52-71, Springer-Verlag, 1982

[14] E. M. Clarke, Jr., E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. In ACM Transactions on Programming Languages and Systems, pages 244-263, April 1986

[15] E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, volume B, chapitre 16, pages 995-1072. Elsevier, 1990

[16] E. A. Emerson and J. Y. Halpern. "Sometimes" and "Not Never" revisited : On branching versus linear time temporal logic. In *J. ACM, pages 151-178, 1986*

[17] Martin J. Osborne and Ariel Rubinstein. A course in game theory. MIT Press, 1994

[18] O. Kupferman, M. Y. Vardi, and P. Wolper. Module checking. In *Information and Computation, pages 322-344, 2001*

[19] Kupferman, Orna and Vardi, Moshe Y. and Wolper, Pierre. An Automata-Theoretic Approach to Branching-Time Model-Checking. In ACM Press, pages 312-360, 2000

[20] Laroussinie, François and Markey, Nicolas and Oreiby, Ghassan. On the Expressiveness and Complexity of ATL. In Proceedings of the 10th International Conference on Foundations of Software Science and Computation Structure (FoSSaCS'07), pages 243-257, Springer-Verlag, 2007

[21] Laroussinie, François and Markey, Nicolas and Oreiby, Ghassan. On the Expressiveness and Complexity of ATL. In Logical Methods in Computer Science 4 (2:7), 2008

[22] René Mazala. Infinite Games. In Automata, Logics, and Infinite Games, LNCS 2500, pages 23-42, Springer, 2001

[23] Fabio Mogavero and Aniello Murano and Moshe Y. Vardi. Reasoning About Strategies. In IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010), pages 133–144, Leibniz International Proceedings in Informatics (LIPIcs), 2010

[24] Muller, David and Schupp, Paul. Alternating automata on infinite objects, determinacy and Rabin's theorem. In EPIT'84, pages 99-107, Springer, 1985

[25] Muller, David and Schupp, Paul. Alternating Automata on Infinite Trees. In TCS, pages 267-276, Elsevier Science, oct 1987

[26] Muller, David and Schupp, Paul. Simulating alternating tree automata by nondeterministic automata : New results and new proofs of the theorems of Rabin, McNaughton and Safra. In TCS, pages 69-107, Elsevier Science, apr 1995

[27] Christos Papadimitriou. Computational Complexity, 1994

[28] Pinchinat, Sophie. A Generic constructive Solution for Concurrent Games with Expressive Constraints on Strategies. In Proceedings of the 5th International Symposium on Automated Technology for Verification and Analysis (ATVA'07), pages 253-267, Springer-Verlag, 2007

[29] Pinchinat, Sophie. Quantified Mu-Calculus with Decision Modalities for Concurrent Game Structures. ANU Computer Science Technical Reports TR-CS-07-02 (http ://cs.anu.edu.au/techreports/2007/TR-CS-07-02.pdf)

[30] Pnueli, Amir. The Temporal Logic of Programs. In Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS'77), pages 46-57, IEEE Comp. Soc. Press, 1977

[31] Queille, Jean-Pierre and Sifakis, Joseph. Specification and verification of concurrent systems in CESAR. In Proceedings of the 5th International Symposium on Programming (SOP'82), pages 337-351, Springer-Verlag, 1982

[32] P.-Y. Schobbens. Alternating-time logic with imperfect recall. In *Proceedings of the 1st Workshop on Logic and Communication in Multi-Agent Systems (LCMAS'03), ENTCS 85. Elsevier, 2004*

[33] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. In Journal of the ACM, pages 733-749, ACM Press, 1985

[34] A.P. Sistla, M. Vardi and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. In Theoret. Comput. Sci. 49, pages 217-238, 1987

[35] Walther, Dirk and van der Hoek, Wiebe and Wooldridge, Michael. Alternating-time Temporal Logic with Explicit Strategies. In Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge (TARK'07), pages 269-278, Samet, Dov, 2007