N° ENSC-2008/–

THESE DE DOCTORAT
DE L'ECOLE NORMALE SUPERIEURE DE CACHAN

Présentée par

Ghassan OREIBY (غسان العريبي[1])

Pour obtenir le grade de

DOCTEUR DE L'ECOLE NORMALE SUPERIEURE DE CACHAN

Domaine :
INFORMATIQUE

Sujet de la thèse :

Logiques temporelles pour le contrôle temporisé.

Soutenue à Cachan le 08 décembre 2008 devant le jury composé de :

Eugene Asarin	Président
Thomas Brihaye	Examinateur
François Laroussinie	Co-directeur de thèse
Nicolas Markey	Co-directeur de thèse
Sophie Pinchinat	Rapporteur
Jean-François Raskin	Rapporteur

Laboratoire Spécification & Vérification
(ENS CACHAN/CNRS/UMR 8643)
61, avenue du Président Wilson, 94235 CACHAN CEDEX (France)

[1]Le nom est mentionné en arabe pour la bonne prononciation.

Résumé

La logique ATL a été introduite en 1997 par Alur et al. Grâce à la quantification sur les stratégies, cette logique peut spécifier des propriétés sur les jeux. L'objectif de cette thèse est d'étendre ATL pour des systèmes temporisés concurrents. La première partie de cette thèse présente une étude sur la logique ATL dans un cadre non-temporisé. Nous établissons la complexité exacte du model-checking de cette logique sur différents types de structures de jeu. Nous donnons aussi une étude sur l'expressivité d'ATL et les différents modèles. La deuxième partie est consacrée aux extensions temporisées des structures de jeu. Nous introduisons les structures de jeu concurrents avec durée (DCGS), cette extension nous permet d'avoir des exécutions comprenant des coûts entiers. Pour vérifier ces modèles, TATL présente une extension temporisé d'ATL qui permet de décrire des propriétés quantitatives sur les DCGS. Nous avons montré que le model-checking de la logique TATL sur ces modèles est de complexité exponentielle, et que cette complexité est optimale. Ensuite, nous introduisons les TCGS, des structures de jeu plus générales qui contient des horloges à la manière des automates temporisés. Une action d'un joueur dans ces modèles consiste en une date et une fonction de temps donnant un coup à chaque instant, la transition sera franchie à la date minimale proposée et en fonction des coups de tous les joueurs à cette date. Nous introduisons ensuite la logique TALTL, une logique temporisée qui généralise TATL, qui peut énoncer des propriétés temporisées comprenant des comportements infinis, nous montrons que le problème de model-checking de cette logique est décidable. Nous présentons de même la complexité du model-checking dans un cadre non-zeno.

Mots-clefs. Vérification formelle, ATL, jeux temporisés, model-checking, concurrent games, logiques temporelles.

Abstract

The temporal logic ATL was introduced in the late 90's. More powerful than CTL, using ATL we can express more precise properties with quantification over strategies of different players in a game context. In this thesis, we have studied the complexity of model-checking ATL over different kind of games structures and we have shown that the complexity depends on the relying structure. We studied as well the expressivness power of the logic with respect to the modality "Weak Until". We study also timed extenstions for concurrent games ; we introduce the durational concurrent games structures (DCGS) that extends concurrent games with discrete time (or costs), in a way that in the new model, firing a transition holds a duration. We show that such models benefits from a polynomial model-checking whenever no equality is used in the formulas. We developped another more general extension where clocks are used in a way similar to timed automata, with this richer framework we can express more properties using an extended version of ATL called TATL which can handle the timing properties of the games. We introduced a richer new logic TALTL that extends Timed ATL. TALTL can states timed properties and infinite behaviour while remaining decidable.

Keywords. Formal verification, ATL, timed games, model-checking, concurrent games, temporal logics.

Remerciements

Tout d'abord, je tiens à remercier les membres de jury : Thomas Brihaye, Eugène Asarin, Jean-François Raskin et Sophie Pinchinat d'avoir pris le temps pour lire le manuscrit et être venus pour assister à la soutenance. I would like to thank as well Kim Larsen for receiving me as a post-doc fellow at the departement of computer science in Aalborg university.

Ensuite, j'adresse ma scincère gratitude à mes encadrants Nicolas et François. Je leur suis entièrement reconnaissant pour leur soutien, leurs conseils précieux et leur disponibilité. J'éspère que ce n'était pas très difficile pour eux à me supporter pendant trois longues années.

Ainsi, c'est à mes co-bureautiers qui ont dû me voir quotidiennement : Thomas, Graham, Élie, Akshay, Rémi, Guislain, Arnaud et Ştefan sans oublier Nathalie et Pascal pour leur acceuil quand j'ai changé de bureau, j'espère que je n'étais pas bruyant et gênant des fois. Et bien sûr, j'exprime ma gratitude aux personnes qui ont respecté l'anonymité du *Señor Alvarez* pendant son voyage.

Je remercie Ahmad pour le support technique pendant les trois années de thèse.

Je remercie aussi mes amis Libanais Mehdi, Ziad, Mohammad, Najla, Abdallah, Wafaa, Mehdi et Hicham. Grâce à eux, mon séjour en France a eu une chaleur orientale.

Je souhaite aussi adresser toute ma reconaissance aux membres de ma famille en France, Waël, Myriam, Florence, Hoda, Zeina sans oublier les petits Léa, Raphaël et Florian pour m'avoir supporté durant cette thèse.

Finalement, je reste énormément reconnaissant à mes parents Mohammad et Noha ainsi que ma sœur Maya qui n'ont pas cessé de me soutenir, même dans les situations les plus difficiles. Pendant ma première année de thèse, j'étais profondément touché par le niveau d'atrocité menée par une armée infâme par les coutumes sanglantes et primitives. En effet, je me suis retrouvé à plus de 3000 km loin de mes parents qui étaient en train de survivre des agressions inhumaines et des terreurs non justifiées. De mon côté, les images transmises m'étaient les plus horribles que j'ai jamais vues[2]. Leur courage m'a ému surtout quand ils me demandaient si j'avais besoin de quoi que ce soit alors que le bruit des bombardements précèdait leurs paroles ! J'en profite aussi pour saluer respectueusement tous les Libanais qui ont dû sacrifier leur plus précieux pour contrer cet avilissement de l'homme connu (et parfois glorifié) par le terme "guerre".

[2]Sans tenir en compte des horreurs commises dans la dernière guerre contre Gaza en fin 2008, ces horreurs qui sont les fruits de la "démocracie" du même agresseur.

Table des matières

Résumé **i**

Remerciements **iii**

Table des matières **v**

Introduction **1**

1 Les jeux multi-agents **7**
Introduction 7
1.1 Les structures de jeu 8
1.1.1 Les systèmes de transitions 8
1.1.2 Les Structures de Jeu Concurrent 10
1.1.3 Coalition, stratégie, outcomes d'une stratégie 12
1.1.4 Les différents types de jeux multi-agents 13
1.2 Les logiques temporelles 14
1.2.1 Logiques pour les systèmes de transitions 15
1.2.2 Logiques pour les jeux 18
Conclusion 24
Pour en savoir plus 25

2 Complexité et expressivité d'ATL **27**
Introduction 27
2.1 La complexité du model-checking d'ATL 28
2.1.1 Model-checking d'ATL sur les CGS explicites. 28
2.1.2 Model-checking d'ATL sur les CGS symboliques. 30
2.1.3 Le model-checking d'ATL sur les ATS. 36
2.1.4 Au delà d'ATL 42
2.2 Expressivité 46
2.2.1 Comparaison entre l'expressivité des CGS et celle des ATS 46
2.2.2 Le pouvoir expressif d'ATL 50
Conclusion 54
Pour en savoir plus 54

3 Structures de jeu avec durée **55**
Introduction . . . 55
3.1 Définitions . . . 56
3.1.1 CGS avec durée exacte . . . 56
3.2 Le model-checking de la logique TATL . . . 56
3.2.1 Les contraintes $\leq \zeta$ . . . 57
3.2.2 Les contraintes $\geq \zeta$ . . . 59
3.2.3 Les contraintes $= \zeta$ . . . 63
3.2.4 Résultats pour TATL et $\mathsf{TATL}_{\leq,\geq}$ . . . 65
3.2.5 Les TDCGS unitaires . . . 67
3.3 Les CGSs avec durée . . . 67
3.3.1 Syntaxe et sémantique . . . 67
3.3.2 Le model-checking de la logique $\mathsf{TATL}_{\leq,\geq}$ . . . 69
3.3.3 Le model-checking de la logique TATL . . . 70
Conclusion . . . 71
Pour en savoir plus... . . . 71

4 Les CGS temporisées **73**
4.1 Introduction . . . 73
4.2 Définitions . . . 74
4.2.1 Les automates temporisés . . . 74
4.2.2 Les CGS temporisées . . . 76
4.2.3 Les structures de jeux temporisés . . . 77
4.3 Équivalence des régions et Stratégies . . . 80
4.3.1 Équivalence des régions . . . 80
4.3.2 Des stratégies plus simples . . . 83
4.4 La CGS des régions . . . 86
4.5 La logique ATL temporisée . . . 89
4.5.1 Model-checking . . . 92
4.6 Traitement des stratégies Zeno . . . 95
Conclusion . . . 101
Pour en savoir plus . . . 102

Conclusion et Perspectives **103**
Contribution de la thèse . . . 103
Perspectives . . . 104

A Les classes de complexité **105**
A.1 La hiérarchie polynomiale . . . 105
A.1.1 Quelques problèmes caractérisant la hiérarchie polynomiale . . . 106
A.2 La hiérarchie AC . . . 107

Bibliographie **109**

Introduction

Systèmes critiques. Les systèmes informatiques sont de plus en plus répandus. On note la présence de tels systèmes dans presque tous les domaines ; Des systèmes de positionnements par satellites et des régulateurs de vitesse comportent des logiciels informatiques embarqués. Les domaines aéronautique, nucléaire et autres systèmes critiques s'appuient sur des machines *intelligentes* pour guider leur bon comportement. L'informatique est aussi présente dans d'autres systèmes qui ne sont pas considérés critiques (i.e. si le mauvais comportement n'auraient pas de graves conséquences) mais qui restent important, on peut citer par exemple la téléphonie mobile, les immeubles intelligents, et autres ...

Les *systèmes critiques* sont des systèmes où la moindre faille peut engendrer des conséquences catastrophiques (dans le sens économique ou humain). On peut citer plusieurs accidents dus à des erreurs informatiques :

- par exemple la faille de la sonde *Phobos 1* en 1988 a été causée par l'exécution d'un code annulant le contrôle automatique de l'altitude, les panneaux solaires ont perdu l'orientation, ce qui a épuisé les batteries de la sonde, causant sa perte définitive. Le programme annulant le contrôle automatique de l'altitude a été implémenté dans l'ordinateur de la sonde pour les phases de tests, mais n'a jamais été supprimé puisque les mémoires utilisées étaient des mémoires de lecture unique. Ensuite le contrôle terrestre a envoyé une séquence de commandes dans laquelle une erreur dans un seul caractère a dû déclencher la commande d'exécution du code, causant la perte de la sonde.[Wal89]
- Un autre exemple, en 1993, l'accident du *vol 2904 de la Lufthansa* a été causé par un retard de l'application du système de freinage, ce système étant contrôlé par un logiciel informatique, a fait deux morts et plusieurs blessés. Une enquête sur cet accident a révélé un défaut de conception dans le contrôle automatique de l'appareil (Airbus A320) [War93] :

  > « The aircraft automatics comprises, for basic landing configuration if the aircraft (i.e. with flaps extended to FULL position), the programme which subjects actuation of all braking devices to some specific conditions. Ground spoilers, when selected, will extend provided that either shock absorbers are compressed at both main landing gears (the minimum load to compress one shock absorber being 6300 kgs), or wheel speed are above 72 kts at both main landing gears. Engine reversers, when selected, will deploy provided that shock absorbers are compressed at both main landing gears. Such a logics result in the lack of possibility of immediate actuation of two mentioned above aircraft's braking devices without meeting the conditions described. »

  Ce qui explique que l'activation du système de freinage ne se lance que dans des conditions

précises et qui n'étaient pas satisfaites pendant l'atterrissage de l'avion.

L'introduction des logiciels informatiques dans ce genre de système a donné sujet à de nombreux travaux de recherche. Il est donc nécessaire de traiter et d'analyser les logiciels utilisés dans ce genre de système avant de les utiliser dans l'industrie.

Méthodes formelles. Plusieurs méthodes de vérification formelles ont été développées dans la littérature, et qui s'intéressent à la vérification des différents comportements d'un système. On illustre ci-dessous quelques-unes :

- Les *méthodes de tests* consistent à générer des exécutions dites intéressantes du système et les comparer aux comportements prévus par le système pour vérifier s'il passe ou non les tests. Cette méthode n'est pas exhaustive.
- La *démonstration automatique* qui utilise des systèmes de preuve logiques pour déduire le bon fonctionnement. La démonstration peut être complètement automatique, ou bien il peut s'agir d'un assistant de preuve (semi-automatique).
- Le *model-checking* est une méthode exhaustive, où l'analyse tient à vérifier que *toute* exécution du système vérifie une propriété exprimée dans un langage de spécification (par exemple, la logique temporelle)[Pnu77]. Dans cette thèse, nous nous intéressons à ce cadre là.

Ces trois méthodes fournissent donc un cadre formel pour maintenir à un niveau relativement acceptable la fiabilité de ces systèmes.

Model-checking. Le model-checking consiste en deux phases essentielles (voir figure 1) :

1. La première étape est l'étape de modélisation, puisqu'on souhaite traiter des systèmes réels, on se retrouve obligé de modéliser notre système suivant les modèles mathématiques à notre disposition. On appelle notre modèle de système $\mathcal{S}$. Le cahier des charges doit à son tour être modélisé en utilisant un langage de spécification formel, comme par exemple sur la figure 1, **AG** `safe` veut dire que sur toutes les exécutions du système, la propriété `safe` est toujours vérifiée. Le cahier des charges sera donc transformé en un ensemble de formules logiques.
2. La phase de vérification est la deuxième phase. Elle consiste à donner le modèle du système $\mathcal{S}$ et la propriété à un outil (appelé model-checker), et cet outil donne ensuite une réponse positive si notre modèle satisfait ou négative s'il ne vérifie pas la propriété donnée.

Ces deux phases comportent quelques difficultés. La phase de modélisation doit concevoir un niveau d'abstraction. Donc, on doit passer d'un système réel très général à un modèle mathématique, ce qui va avoir des conséquences en éliminant des comportements qui pourrait être intéressants. Il faut alors s'intéresser dans cette phase à abstraire selon les propriétés qui nous intéressent. Le model-checker, lui aussi, pourra avoir des difficultés, le problème de l'explosion combinatoire est un des problèmes majeurs, en effet, suivant la taille du modèle et de la formule, la vérification tend à prendre un énorme temps de calcul, et dépasse souvent l'espace mémoire disponible.

Le problème de contrôle est un problème qui étend le model-checking : Pour des systèmes ouverts où le comportement dépend aussi de l'environnement, il est nécessaire de synthétiser un contrôleur qui va guider le système pour satisfaire les spécifications quel que soit le comportement de l'environnement. On a donc la notion des *jeux* : Le modèle devient une plate-forme

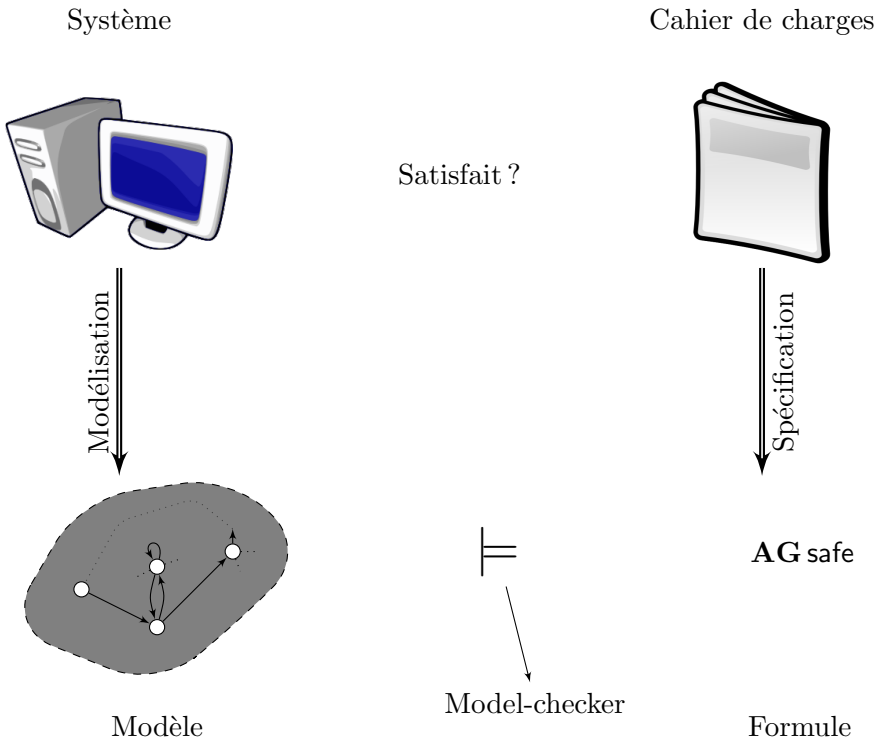


FIG. 1 – La procédure de la vérification

où deux ou plusieurs agents du système interagissent pour que chacun d'eux réalise un objectif donné.

Systèmes temporisés Souvent, il est nécessaire d'étudier des aspects plus compliqués pour les modèles : ajouter du temps est par exemple crucial pour vérifier des spécifications temps-réel. Les *automates temporisés* sont des formalismes qui assurent une description de l'évolution du temps dans les modèles. Un automate va contenir des *horloges* qui servent à mesurer le temps, ensuite, on peut utiliser les valeurs de ces horloges pour autoriser ou refuser l'accès aux différents états par l'intermédiaire des gardes et des invariants présents sur les transitions ou les états. Il est ainsi possible de remettre les horloges à zéro.

Contribution de la thèse

L'objectif principal de la thèse est de développer une extension temporisée des jeux multi-agents tout en fournissant une version temporisée de la logique ATL. Pour aboutir à cet objectif, la thèse s'est déroulée en plusieurs phases.

Nous détaillons, ci-dessous, notre contribution :

Première partie. Dans le cadre non temporisé, nous présentons dans le premier chapitre les structures de jeu concurrentes (ou Concurrent Game Structures). Ces structures sont considérées

comme des extensions des structures de Kripke où franchir chaque transition dépend des choix de *tous* les joueurs. Nous définissons deux variantes des ces structures ; lorsque la table de transitions est donnée explicitement, on appelle la structure une CGS explicite, et lorsque cette table est codée d'une manière plus succincte par l'intermédiaire d'une liste de formules booléennes, on les appelle des CGS symboliques. Un autre type de structure de jeu, appelé les systèmes de transitions alternants (ou ATS) où un choix d'un agent est un sous-ensemble d'états : les joueurs choisissent les états où ils souhaitent se retrouver à l'étape prochaine et l'état suivant est obtenu en calculant l'intersection des choix des différents agents.

Bien que les logiques temporelles classiques, telle la logique du temps arborescent CTL, peuvent être utilisées sur ces structures, on cherche à spécifier plus de propriétés sur les interactions entre les différents agents du système. La logique ATL remplit ce but avec la modalité qui quantifie sur les stratégies des agents, notée par $\langle\langle A \rangle\rangle \varphi$ et qui signifie que "la coalition dans A possède une stratégie qui garantit la formule φ". Nous avons étudié plusieurs aspects de la logique ATL dans le chapitre 2. Notamment, nous avons étudié la complexité du problème du model-checking sur les structures de jeux. Nous avons montré que la complexité[3] varie selon le modèle (ATS, CGS explicite ou symbolique) utilisé. D'autre part, nous avons abordé la question de l'expressivité de la logique. Nous avons montré que la logique ATL sans la modalité $\mathbf{R}$ (le dual de $\mathbf{U}$) est strictement moins expressive que la logique définie avec le dual de l'opérateur $\mathbf{U}$.

Deuxième partie. La deuxième partie est consacrée à étendre les structures de jeu concurrentes avec des durées discrètes. Chaque transition comporte alors une valeur entière positive qui désigne le temps écoulé dans cette transition. Pour ces modèles là, une extension de la logique ATL a été définie pour pouvoir spécifier des propriétés quantitatives. Nous avons cherché des algorithmes efficaces pour le problème de model-checking. Les algorithmes que nous avons présentés, sont polynomiaux pour le fragment de la logique sans la contrainte d'égalité. Pour la logique complète, nous avons montré que le problème est EXPTIME-complet.

Nous avons ensuite étudié le problème d'avoir des intervalles d'entiers sur les transitions au lieu d'avoir une valeur unique. La sémantique pour ces modèles étendues était spéciale, puisqu'il fallait décider qui choisirait la durée de la transition à franchir parmi les valeurs de l'intervalle sur les transitions. Pour cette raison, il y avait des *agents de temps* (un agent par transition) pour décider la durée prise par leur transition. L'avantage d'une telle approche est qu'on peut intégrer ces agents dans les formules TATL, pour qu'ils soient du côté de telle ou telle coalition. Nous avons montré que le problème de model-checking garde la même complexité, c.à.d. EXPTIME-complet pour la logique TATL complète, et PTIME-complet pour la logique TATL sans les contraintes d'égalité. Notons que ces modèles utilisent des durées strictement positives.

Troisième partie. Nous nous intéressons dans le dernier chapitre à définir une sémantique de temps dense avec les notions de jeux. Comme dans le cas des automates temporisés, le système de jeu va comprendre des horloges, des gardes sur les transitions qui autorisent l'action en fonction des valeurs des horloges, et des invariants sur les états. Le point crucial de notre modèle est de définir une sémantique avec le temps dense, mais en gardant la notion de concurrence entre

[3] On suppose que PTIME $\subsetneq$ NP dans toute la suite.

les différents joueurs. Notre approche est la suivante : depuis un état donné, chaque joueur va choisir une action à jouer, et un temps t où il souhaite jouer son action, et il indique aussi ce que serait ses choix pour toutes les dates précédentes au cas où un autre joueur imposeraient une autre date. Ensuite, la plus petite date sera prise, et les actions des différents joueurs du système seront prises au temps choisi. De même, la logique TATL est une extension temporisée de la logique ATL.

Nous avons donc étudié le model-checking de la logique TATL sur les TCGS. Nous avons montré que le problème du model-checking de la logique TATL sur les TCGS est 2EXPTIME-complet, et que cette complexité reste inchangée dans le contexte du non-Zeno. Ensuite, nous définissons la logique TALTL, une extension de la logique TATL, qui est strictement plus expressive, cette logique nous permet de spécifier des propriétés temporisés en même temps que des propriétés d'équité. Le problème du model-checking est aussi un problème 2EXPTIME-complet.

Chapitre 1 Les jeux multi-agents

Introduction

Récemment, les jeux multi-agents ont été l'objet de beaucoup de recherches dans le domaine de la vérification, notamment parce qu'ils permettent de modéliser des problèmes de contrôle. Dans ce chapitre, nous allons présenter des modèles différents de jeux qui peuvent être vus comme des extensions des structures de Kripke. Dans chacun de ces modèles, nous disposons d'un ensemble de joueurs (ou agents) qui interagissent entre eux ; un tour se déroule en deux temps : d'abord, étant donné un état courant q, chaque joueur choisit (indépendamment des autres) un coup à jouer, puis en fonction de ces choix, le jeu évolue dans un nouvel état q'.

Nous étudions les *structures de jeu concurrent* (CGS). Dans les CGS, les coups d'un joueur sont représentés par un alphabet de coups, et l'évolution du système est donnée par une table de transitions qui associe à chaque état q et à chaque vecteur de coups (un coup pour chaque agent) l'état successeur q'. La table de transitions peut se représenter explicitement (on parle alors de CGS *explicites*) ou de manière symbolique pour obtenir un codage succinct (on parle de CGS *symboliques*). Nous allons définir formellement ces modèles et présenter plusieurs exemples d'utilisation.

Dans le cadre du model-checking classique, on modélise le système à vérifier sous la forme d'un système de transitions $\mathcal{K}$ et on s'intéresse à la vérification de propriétés énoncées en logique temporelle comme CTL (Computation Tree Logic [CE81, QS82]), LTL (Linear Temporal Logic, [Pnu77]) ou encore CTL* ([EH86, VS85, CES86]). Ces logiques temporelles sont des formalismes très intéressants pour la spécification de propriétés le long des exécutions d'un système de transitions. On peut énoncer, par exemple, que "tout message est suivi par la réception d'un accusé de réception". En LTL, cette propriété s'écrira $\mathbf{G}\,(\texttt{msg} \rightarrow \mathbf{F}\,\texttt{ack})$ et elle s'interprète sur une exécution de $\mathcal{K}$ (on dit que LTL est une logique temporelle de temps linéaire) : $\mathbf{G}$ signifie "toujours" et $\mathbf{F}$ "un jour dans le futur". Leur emboîtement signifie donc qu'il est toujours vrai (le long d'une exécution ρ) qu'un état vérifiant la proposition "msg" est suivi dans le futur (le long de ρ) par un état vérifiant "ack". En CTL ou CTL*, cette propriété s'écrira $\mathbf{AG}\,(\texttt{msg} \rightarrow \mathbf{AF}\,\texttt{ack})$ et s'interprétera sur l'arbre des exécutions de $\mathcal{K}$ (on dit que CTL et CTL* sont des logiques temporelles de temps arborescent) : ici en plus de $\mathbf{F}$ et $\mathbf{G}$, on utilise le quantificateur universel sur les exécutions ($\mathbf{A}$). Ainsi un état q_0 vérifie cette formule si pour toutes les exécutions ($\mathbf{A}$) issues de q_0, il est toujours vrai ($\mathbf{G}$) que si un état q vérifie "msg" alors le long de toutes les exécutions (A) issues de q il y aura dans le futur ($\mathbf{F}$) un état vérifiant "ack". Dans CTL et CTL*, il existe aussi le quantificateur existentiel ($\mathbf{E}$) sur les exécutions. Par exemple, la formule $\mathbf{AG}\,(P \rightarrow \mathbf{EF}\,P')$ signifie que depuis tout état accessible où P est vrai, *il existe une exécution* menant à P', ce type de propriété est inexprimable en LTL ([Eme90]).

Dans le cas des jeux, on souhaite tenir compte de l'interaction entre agents et énoncer des propriétés plus fines. Par exemple, plutôt que chercher si *une* (ou *toute*) exécution vérifie $P \mathbf{U} P'$ (P est vrai jusqu'à ce que P' soit vérifiée), on souhaite vérifier que l'agent A peut choisir des coups de manière à assurer $P \mathbf{U} P'$ (on dira que A dispose alors d'une stratégie pour garantir $P \mathbf{U} P'$). Dans ce cas, on souhaite donc quantifier sur une partie des exécutions (résultant des différents coups possibles pour les autres joueurs). C'est la motivation de la logique ATL (Alternating-time Temporal Logic [AHK98]). Par exemple, la propriété précédente s'écrira $\langle\langle A \rangle\rangle P \mathbf{U} P'$, où l'opérateur $\langle\langle A \rangle\rangle \varphi$ se lit "les agents dans A disposent d'une *stratégie* pour garantir la propriété φ, quelle que soit la réponse des adversaires". En revenant à l'exemple d'envoi de message, notre système contient deux joueurs A et B, la propriété $\mathbf{AG}(\texttt{msg} \rightarrow \langle\langle A \rangle\rangle \mathbf{F}\, \texttt{ack})$ veut dire qu'une fois un message est envoyé le joueur A dispose d'une stratégie pour assurer un accusé de réception "`ack`" un jour dans le futur.

Ce chapitre est divisé entre les définitions des systèmes multi-agents, où nous présentons les structures possibles : les CGS sous les deux formes *explicites* et *symboliques*. Ensuite, nous montrons la syntaxe et la sémantique de la logique ATL, ainsi que quelques extensions de cette logique.

Les résultats dans chapitre sont basés sur les papiers [LMO07, LMO08]

1.1 Les structures de jeu

Nous généralisons la notion de systèmes de transitions en ajoutant des joueurs à la structure pour obtenir un modèle de système multi-agents appelé Concurrent Games Structrure. Cette nouvelle extension est une structure de jeu, où les joueurs font leur choix simultanément pour avancer à l'état suivant.
Avant de commencer à introduire les structures de jeu, nous rappelons la définition des systèmes de transitions (que l'on peut considérer comme des jeux avec un unique joueur).

1.1.1 Les systèmes de transitions

On se sert des systèmes de transitions pour représenter le comportement d'un système : il consiste en un ensemble d'états et un ensemble de transitions liant un état à un autre. Nous distinguons deux types des systèmes de transitions ; des systèmes de transitions classiques, et des systèmes de transitions pondérés où les transitions portent des valeurs. Nous commençons par définir les systèmes de transitions classiques :

1.1.1.1 Les systèmes de transitions

Définition 1.1
Un système de transitions *est un 4-uplet* $\mathcal{K} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta \rangle$ *tel que :*

- Loc *est l'ensemble des états de la structure, cet ensemble est éventuellement infini ;*
- AP *est un ensemble de propositions atomiques ;*
- $\mathsf{Lab} : \mathsf{Loc} \rightarrow 2^{\mathsf{AP}}$ *est une fonction qui étiquette chaque état par un ensemble de propositions atomiques ;*
- $\delta \subseteq \mathsf{Loc} \times \mathsf{Loc}$ *est la relation de transition.*

1.1.1.2 Le système de transitions pondéré

Dans des systèmes de transitions, les transitions portent parfois des valeurs (entières ou réelles). Ces valeurs désignent des quantités physiques qui dénotent des poids de la transition franchie, ou bien le temps écoulé dans cette transition. On appelle une telle structure un système de transition pondéré. La définition formelle est :

Définition 1.2
Un système de transitions pondéré *est un 4-uplet* $\mathcal{K} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta_{\mathbb{W}} \rangle$ *tel que :*
- Loc *est l'ensemble des états de la structure, cet ensemble est éventuellement infini ;*
- AP *est un ensemble de propositions atomiques ;*
- $\mathsf{Lab} : \mathsf{Loc} \to 2^{\mathsf{AP}}$ *est une fonction qui étiquette chaque état par un ensemble de propositions atomiques ;*
- $\delta_{\mathbb{W}} \subseteq \mathsf{Loc} \times \mathbb{W} \times \mathsf{Loc}$ *est la relation de transition.*

On appelle $\mathbb{W}$ le domaine de valeurs de $\mathcal{K}$, cet ensemble peut désigner des ensembles comme $\mathbb{N}, \mathbb{Q}$ ou $\mathbb{R}$.

Ayant $e = (q, q') \in \delta$ une transition d'un système de transitions (ou bien $e = (q, d, q') \in \delta_{\mathbb{W}}$ s'il s'agit d'un système de transitions pondéré), on note par $e^{\bullet} = q$ l'état de départ de la transition e, et par $e^{\multimap} = q'$ l'état d'arrivée de la transition.

Définition 1.3
Une exécution ρ *d'un système de transitions* $\mathcal{K}$ *est une séquence de transitions* $(\rho_i)_i$ *telle que* $\rho_i \in \delta$, *et pour tout* i, *on a* $\rho_i^{\multimap} = \rho_{i+1}^{\bullet}$. *On note par* $\rho[i] = \rho_i^{\bullet}$ *l'état de départ de la* i*-ème transition de l'exécution* ρ. *On écrit* $\mathsf{Exec}_{\mathcal{K}}(q)$ *pour l'ensemble de toutes les exécutions de* $\mathcal{K}$ *qui commencent par une transition* δ_0 *telle que* $\delta_0^{\bullet} = q$, *et* $\mathsf{Exec}^F_{\mathcal{K}}(q)$ *pour le sous-ensemble des exécutions finies à partir de* q. *On note* $\mathsf{Exec}_{\mathcal{K}}$ *(resp.* $\mathsf{Exec}^F_{\mathcal{K}}$*) pour désigner l'ensemble de toutes les exécutions (resp. finies) dans* $\mathcal{K}$.

On distingue deux types de systèmes de transitions pondérés : les systèmes de transitions pondérés *discrets*, et les systèmes de transitions pondérés *continus*[1] : Pour les systèmes de transitions *classiques* et les systèmes de transitions pondérés *discrets* une exécution $\rho = \rho_0\rho_1\ldots$, l'ensemble des états que contient ρ sont $\rho[0], \rho[1], \ldots$. Quand $\mathcal{K}$ désigne un système de transitions pondéré *continu*, l'ensemble des états que contient l'exécution ρ va être les états $\rho[0], \rho[1], \ldots$ mais aussi les états intermédiaires, c.à.d. si ρ comporte une transition (s, d, s'), l'ensemble des états que comporte ρ va inclure aussi les états situés à une valeur d' t.q. $0 \leq d' < d$ pour des transitions à partir de s.

Pour une exécution ρ, et i, j deux positions le long de ρ, on note par $i \twoheadrightarrow_\rho j$ la *sous-exécution* de ρ commençant par l'état $\rho[i]$ et se terminant par l'état $\rho[j]$.

Soit $\rho = (\rho_i)_{0 \leq i \leq n} \in \mathsf{Exec}^F_{\mathcal{K}}$. La longueur $|\rho|$ de ρ est $n+1$, le dernier état $\mathsf{last}(\rho)$ de ρ est $\rho[n]$ et, pour tout $m \leq n$, le m-ème préfixe $\rho_{\leq m}$ de ρ est l'exécution finie $(\rho_i)_{0 \leq i \leq m}$.

Pour une transition $e = (q, d, q')$ d'un système de transitions pondéré, on appelle $\mathsf{Val}(e) = d$ le *poids* de la transition e. Pour une exécution ρ d'un système de transitions pondéré, on

[1] On peut considérer une exécution comme étant une fonction de l'ensemble des valeurs vers l'ensemble des transitions du système. Pour un système de transitions pondéré *discret* (ou *classique*) une exécution $\rho : \mathbb{N} \to \delta$, et on la note par $\rho = \rho_0\rho_1\ldots$. Pour un système de transitions pondéré *continu*, l'exécution sera la fonction $\rho : \mathbb{R}^+ \to \delta$ tel que pour tout i on a $\mathsf{Cost}(0 \twoheadrightarrow_\rho i) = i$

appelle $\mathsf{Cost}(\rho)$ la somme des valeurs rencontrées le long de l'exécution ρ, on a donc $\mathsf{Cost}(\rho) = \sum_i \mathsf{Val}(\rho_i)$.

Enfin, on note par $s <_\rho s'$ si, et seulement si s précède strictement s' le long de l'exécution ρ, c'est-à-dire pour $|\rho| = 1$ alors $\rho = (s, d, s')$, sinon, il existe des sous-exécutions (éventuellement vides) ρ', ρ'' et ρ''' telle que $\rho = \rho' \cdot (s, d, q) \cdot \rho'' \cdot (q', d', s') \cdot \rho'''$.

1.1.2 Les Structures de Jeu Concurrent

Un système de transitions est considéré comme un jeu à un seul joueur. Pour pouvoir représenter des jeux à plusieurs joueurs, une extension naturelle, la structure de jeu concurrent (Concurrent Game Structure ou simplement CGS), a été introduite par [AHK02]. Elle consiste à étendre le système de transitions en ajoutant plusieurs agents au système, la table de transitions va alors associer à chaque état et à chaque vecteur de coups l'état successeur. Formellement, la définition est la suivante :

Définition 1.4
Une Structure de Jeu Concurrent *(Concurrent Game Structure ou* CGS*) est un* 8*-uplet* $\mathcal{S} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg} \rangle$ *dans lequel :*

- $\langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta \rangle$ *est un système de transitions (éventuellement pondéré) ;*
- $\mathsf{Agt} = \{A_1, \ldots, A_k\}$ *est un ensemble fini d'*agents *(ou bien* joueurs*) ;*
- $\mathbb{M}$ *est l'ensemble (éventuellement infini) de toutes les actions possibles d'agents ;*
- $\mathsf{Mov}\colon \mathsf{Loc} \times \mathsf{Agt} \to \mathcal{P}(\mathbb{M}) \setminus \{\varnothing\}$ *définit l'ensemble (fini) non-vide d'actions possibles pour chaque agent dans chaque état ;*
- $\mathsf{Edg}\colon \mathsf{Loc} \times \mathbb{M}^k \to \delta$ *(où* $k = |\mathsf{Agt}|$*), est une fonction (partielle) qui définit la table de transitions. Elle donne la transition à franchir, à partir d'un état donné et d'un ensemble d'actions des agents, pour arriver au successeur. Cette fonction doit vérifier la condition suivante : pour tout* $\ell \in \mathsf{Loc}$ *et tout* $m_i \in \mathsf{Mov}(\ell, A_i)$*, si* $\mathsf{Edg}(\ell, (m_i)_{i \in \mathsf{Agt}}) = (q, \ell')$*, alors* $q = \ell$*.*[2]

Dans la suite, le chapitre 2 considère les CGS avec des systèmes de transitions classiques et finis. Le chapitre 3 traite le cas des CGS avec des systèmes de transitions finis pondérés discrets. Et finalement, le chapitre 4 utilise des CGS avec des systèmes de transitions infinis pondérés continus.

Le comportement est comme suit [AHK02] : à partir d'un état ℓ, chaque joueur A_i choisit une action possible m_{A_i} dans $\mathsf{Mov}(\ell, A_i)$, l'état suivant est alors donné par $\mathsf{Edg}(\ell, m_{A_1}, ..., m_{A_k})$. On désigne par $\mathsf{Next}(\ell)$ l'ensemble de tous les successeurs possibles à partir de l'état ℓ, et par $\mathsf{Next}(\ell, A_j, m)$, où $m \in \mathsf{Mov}(\ell, A_j)$, la restriction de $\mathsf{Next}(\ell)$ aux états accessibles depuis ℓ quand le joueur A_j exécute l'action m, en d'autres termes :

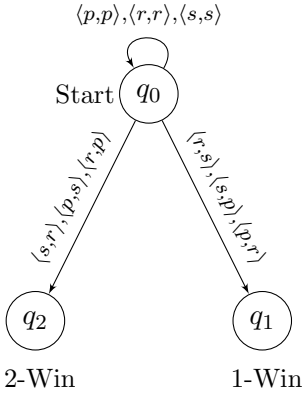
$$\mathsf{Next}(\ell, A_j, m) = \left\{ \mathsf{Edg}(q, m'_1, \ldots, m'_{|\mathsf{Agt}|}) \mid m'_j = m \right\}$$

D'après cette définition, on peut envisager plusieurs méthodes pour coder la table de transitions [JD05]. Ce codage change la concision de la représentation. Nous proposons ainsi deux codages possibles, les CGS explicites et les CGS symboliques :

[2] Parfois au lieu d'écrire $\mathsf{Edg}(\ell, (m_i)_{i \in \mathsf{Agt}}) = (\ell, \ell')$, on simplifie par $\mathsf{Edg}(\ell, (m_i)_{i \in \mathsf{Agt}}) = \ell'$ pour désigner le choix de la transition (ℓ, ℓ').

Définition 1.5
Une CGS explicite *est une CGS où la table de transitions est définie explicitement.*

Exemple 1.1
Prenons l'exemple du jeu chifoumi. *On rappelle d'abord les règles du jeu : les joueurs choisissent simultanément un des trois choix possibles, papier (p), pierre (r) ou ciseaux (s). Le papier bat la pierre, les ciseaux battent le papier, et la pierre bat les ciseaux. Si on tombe sur deux choix similaires on recommence. La figure 1.1 présente ce jeu sous forme d'une CGS à trois états, la table de transition donne l'état qui résulte quand chaque joueur choisit une action. Une transition étiquetée, par exemple, par $\langle r, s\rangle$ veut dire que, simultanément, le premier joueur a fait le choix r et le deuxième a choisi s.*



Joueur 1 \ Joueur 2	q_0	p	r	s
q_0		p	r	s
p	q_0	q_1	q_2	
r	q_2	q_0	q_1	
s	q_1	q_2	q_0	

FIG. 1.1 – Le jeu chifoumi représenté sous forme d'une CGS

Donc, dans une CGS explicite, on est obligé de définir chaque élément de la table de transition de taille $\mathcal{O}(|\mathbb{M}|^{|\mathsf{Agt}|})$, ce qui est exponentiel dans le nombre des agents du système. Mais parfois, il est plus pratique d'avoir une table plus concise. À l'aide des CGS symboliques définies ci-dessous, on peut donner la table de transitions en utilisant des combinaisons booléennes :

Définition 1.6
Une CGS symbolique *est une CGS où, pour chaque état ℓ, la fonction de transition est définie par une séquence finie $((\varphi_0, \ell_0), \ldots, (\varphi_n, \ell_n))$, où $\ell_i \in \mathsf{Loc}$ est un état, et φ_i est une combinaison booléenne de propositions de la forme $A_j \stackrel{?}{=} c$ qui sont évaluées à vrai si, et seulement si, l'agent A_j choisit de jouer l'action c. La table de transition est alors définie comme suit : $\mathsf{Edg}(\ell, m_{A_1}, \ldots, m_{A_k}) = \ell_j$ ssi j est le plus petit indice tel que φ_j est évaluée à vrai quand les joueurs A_1 à A_k choisissent les actions m_{A_1} à m_{A_k}. On demande que la dernière formule booléenne φ_n soit $\top$, pour garantir qu'aucune coalition ne puisse forcer un deadlock.*

La description symbolique des CGS est pratique puisqu'elle permet de ne pas donner toute la table de transition explicitement.

La taille $|\mathcal{C}|$ d'une CGS $\mathcal{C}$ est définie par $|\mathsf{Loc}| + |\mathsf{Edg}|$. Pour les CGS explicites, $|\mathsf{Edg}|$ est la taille de la table de transition. Pour les CGS symboliques, $|\mathsf{Edg}|$ est la somme résultante des tailles des formules utilisées pour définir Edg.

Exemple 1.2
On considère l'exemple suivant, prenons un système qui n'avance que si tous les agents font le même choix, sinon il reste bloqué dans le même état. On peut arriver à décrire ce modèle

en utilisant les CGS explicites, mais en utilisant les CGS symboliques, on peut modéliser ce système plus simplement. La table de transition symbolique est représentée dans la figure 1.2. On prend le cas où on a trois joueurs A_1, A_2 et A_3. Chaque joueur possède deux choix possibles 0 ou 1. On remarque que pour le cas général de n joueurs pour cet exemple, la taille de la table

$$\delta(q_0) = \begin{pmatrix} \Big((A_1 \stackrel{?}{=} 0 \wedge A_2 \stackrel{?}{=} 0 \wedge A_3 \stackrel{?}{=} 0) \vee \\ \qquad (A_1 \stackrel{?}{=} 1 \wedge A_2 \stackrel{?}{=} 1 \wedge A_3 \stackrel{?}{=} 1), q_1\Big) \\ (\top, q_0) \end{pmatrix}, \qquad \delta(q_1) = ((\top, q_1))$$

FIG. 1.2 – La table de transitions de l'exemple 1.2

de transitions pour la CGS symbolique est de l'ordre de $\mathcal{O}(2 \cdot n)$, alors qu'elle devient pour la CGS explicite de l'ordre de $\mathcal{O}(2^n)$.

1.1.3 Coalition, stratégie, outcomes d'une stratégie

Coalition. Une coalition est un sous-ensemble d'agents. Dans des systèmes multi-agents, une coalition A joue contre la coalition de ses adversaires $\mathsf{Agt} \setminus A$ comme s'il s'agit d'un jeu à deux joueurs.

On étend alors la notion de Mov et Next pour des coalitions :

- Étant donné $A \subseteq \mathsf{Agt}$ et $\ell \in \mathsf{Loc}$, on désigne par $\mathsf{Mov}(\ell, A)$ les actions possibles pour la coalition A depuis ℓ. Une telle action m est composée d'actions pour chaque agent de la coalition, c.à.d $m \stackrel{\text{def}}{=} (m_{A_j})_{A_j \in A}$. Alors, étant donnée une action $m' \in \mathsf{Mov}(\ell, \mathsf{Agt} \setminus A)$, on note par $m \oplus m'$ l'action *complète* correspondante (une pour chaque agent). Dans les CGS, il est donné par $\mathsf{Edg}(\ell, m \oplus m')$.
- Next est étendue naturellement pour tenir en compte des coalitions : étant donné $m = (m_a)_{a \in A} \in \mathsf{Mov}(\ell, A)$, on désigne par $\mathsf{Next}(\ell, A, m)$ la restriction de $\mathsf{Next}(\ell)$ aux états accessibles depuis ℓ quand chaque joueur $A_j \in A$ joue l'action m_{A_j}.
  L'ensemble des transitions possibles depuis q pour les choix $(m_i)_{A_i \in A}$ est défini par

$$\mathsf{Next}(q, A, (m_l)_{A_l \in A}) = \{\mathsf{Edg}(q, m'_1, \ldots, m'_{|\mathsf{Agt}|}) \mid \forall A_i \in A.\ m'_i = m_i\}.$$

Définition 1.7
Soit $\mathcal{S} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta_t, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg} \rangle$ une CGS, et $A_i \in \mathsf{Agt}$ un agent. Une stratégie de A_i est une fonction $f_{A_i} : \mathsf{Exec}^F_{\mathcal{S}} \to \mathbb{M}$ t.q., pour tout $r \in \mathsf{Exec}^F_{\mathcal{S}}$, $f_{A_i}(r) \in \mathsf{Mov}(\mathsf{last}(r), A_i)$.

Pour une coalition $A \subseteq \mathsf{Agt}$, une stratégie pour A est une famille $F_A = (f_{A_l})_{A_l \in A}$ de stratégies, une pour chaque agent dans A.

De la même manière, étant donnée une trajectoire finie r et une stratégie $F_A = (f_i)_{A_i \in A}$, on définit

$$\mathsf{Next}(r, A, F_A) = \mathsf{Next}(\mathsf{last}(r), (f_i(r))_{A_i \in A}).$$

Une stratégie est dite *sans mémoire* si elle dépend uniquement de l'état courant (*i.e.*, $f_{A_i}(\rho_0 \cdots \rho_n) = f_{A_i}(\rho_n)$).

Définition 1.8
Soient $\mathcal{S} = \langle \mathit{Loc}, \mathit{AP}, \mathit{Lab}, \delta, \mathit{Agt}, \mathbb{M}, \mathit{Mov}, \mathit{Edg} \rangle$ *une CGS,* $A \subseteq \mathit{Agt}$ *une coalition,* q *un état de* $\mathcal{S}$.

Un outcome *d'une stratégie* F_A *à partir d'une exécution finie* r *de longueur* n *est une exécution* $r' = (r'_i)_i$ *t.q.* r *est un préfixe de* r' *et, pour tout* m, $(r'_{n+m}, r'_{n+m+1}) \in \mathsf{Next}(r'_{\leq n+m}, A, F_A)$. *On note par*[3] $\mathit{Out}_{\mathcal{S}}(r, F_A)$ *(resp.* $\mathit{Out}^1_{\mathcal{S}}(r, F_A)$, $\mathit{Out}^\infty_{\mathcal{S}}(r, F_A)$*) l'ensemble des outcomes de* F_A *depuis* r *(resp. de longueur* $|r| + 1$*, de longueur* $+\infty$*). L'ensemble des exécutions* $\mathit{Exec}_{\mathcal{S}}(q)$ *de* $\mathcal{S}$ *depuis un état* q *est un ensemble d'outcomes depuis* q *pour la coalition vide (donc pour la stratégie vide).*

On note par $\mathsf{Strat}_{\mathcal{S}}(A)$ l'ensemble des stratégies pour A. Finalement, on remarque que $F_\varnothing$ est vide et que $\mathsf{Out}_{\mathcal{S}}(\ell, \varnothing)$ représente l'ensemble de toutes les exécutions depuis ℓ.

Le but d'une stratégie est généralement pour gagner le jeu. Cette notion est donnée dans la définition suivante :

Définition 1.9
Un objectif gagnant Φ *est un ensemble de trajectoires (*$\Phi \subseteq \bigcup_{q \in \mathit{Loc}} \mathit{Exec}_{\mathcal{S}}(q)$*).*

Étant donné un objectif Φ *et un état* ℓ*, on dit que le joueur* A_i *gagne le jeu depuis* ℓ *pour un objectif* Φ*, s'il existe une stratégie* f_{A_i} *telle que* $\mathsf{Out}(\ell, f_{A_i}) \subseteq \Phi$*. On appelle alors la stratégie* f_{A_i} *une* stratégie gagnante.

Remarque 1.1
Un jeu est dit déterminé *si pour toute instance du jeu, un joueur disposera d'une stratégie gagnante (mais pas nécessairement le même joueur dans chaque instance). Il ne peut pas y avoir des stratégies gagnantes pour les deux joueurs, sinon en appliquant les mêmes stratégies en même temps, les deux joueurs vont gagner, ce qui est contradictoire puisqu'un joueur doit jouer contre l'adversaire.*

Il est intéressant de noter que les jeux multi-agents ne sont pas déterminés en général : Ceci est illustré dans l'exemple 1.3.

Par contre, quand les structures de jeu sont turn-based, la propriété d'être déterminées est applicable.

Exemple 1.3
Prenons la figure 1.3 qui représente une CGS à 2 joueurs, telle que, dans ℓ_0 *le joueur* A_1 *n'a pas de stratégie pour arriver dans l'étape suivante à un état étiqueté par* p*, et que le joueur* A_2 *ne dispose non plus d'une stratégie qui lui garantit d'arriver à l'étape suivante à un état qui n'est pas étiqueté par* p*. Dans cette représentation, une transition est étiquetée par* $\langle m_1, m_2 \rangle$ *pour dire que l'action* m_1 *correspond au choix du joueur* A_1 *et l'action* m_2 *un choix du joueur* A_2*.*

1.1.4 Les différents types de jeux multi-agents[AHK02]

On peut distinguer plusieurs sous-classes de jeux multi-agents. On donne la définition de chaque sous-classe pour les CGS mais elle peut être également appliquée sur les autres types de structures.

[3] L'indice $\mathcal{S}$ sera oublié quand c'est évident dans le contexte.

1.1.4.1 Les structures de jeu turn-based

Dans une structure de jeu *turn-based*, un seul joueur a le droit de faire un choix dans chaque état. Formellement, une structure de jeu $\mathcal{S} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg} \rangle$ est dite turn-based si pour tout état $\ell \in \mathsf{Loc}$, il existe un joueur $A_\ell \in \mathsf{Agt}$ tel que pour tout agent $A_b \in \mathsf{Agt} \setminus \{A_\ell\}$, on a $|\mathsf{Mov}(\ell, A_b)| = 1$. Dans l'état ℓ, on dit que le tour est au joueur A_ℓ.

Les structures de jeu turn-based sont des jeux intéressants à étudier et elles bénéficient de la propriété "d'être déterminées" décrite dans la remarque 1.1.

On peut vérifier que la CGS de l'exemple 1.1 n'est pas turn-based puisque dans l'état q_0, chacun des deux joueurs possède trois choix possibles.

1.1.4.2 Les structures de jeu synchrones de Moore

L'ensemble d'états dans une structure de jeu *synchrone de Moore* est le produit des ensembles d'états locaux pour chaque joueur. Les joueurs font simultanément leur choix dans chaque état. Chacun choisit l'état local suivant sans tenir en compte des choix pris par les autres joueurs. Formellement, une structure de jeu $\mathcal{S} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg} \rangle$ est dite synchrone de Moore si elle satisfait les conditions suivantes :

- L'ensemble d'états est de la forme $\mathsf{Loc} = \mathsf{Loc}_1 \times \cdots \times \mathsf{Loc}_{|\mathsf{Agt}|}$,
- Pour chaque joueur $A_i \in \mathsf{Agt}$, dans chaque état $\ell \in \mathsf{Loc}$ et pour toute action $m_i \in \mathsf{Mov}(\ell, A_i)$ il existe $\mathsf{Edg}_i(\ell, m_{A_i})$ tel que
  $\mathsf{Edg}(q, m_1, \ldots, m_{|\mathsf{Agt}|}) = \langle \mathsf{Edg}_1(\ell, m_{A_1}), \ldots, \mathsf{Edg}_{|\mathsf{Agt}|}(\ell, m_{A_{|\mathsf{Agt}|}}) \rangle$.

On a montré dans [dAHM00] que la complexité de la vérification d'invariant pour les structures de Moore est EXPTIME-complet. Ce résultat semble naturel puisqu'il s'agit de vérifier le produit des automates.

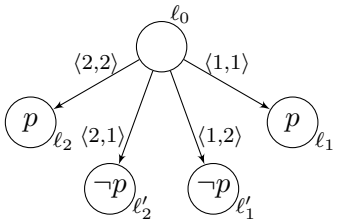


FIG. 1.3 – Une CGS non déterminée.

1.2 Les logiques temporelles

Après avoir défini les structures multi-agents, nous allons consacrer cette section pour définir les logiques qui sont utilisées pour énoncer des propriétés sur des modèles avec plusieurs joueurs. Nous nous intéressons donc à la logique ATL, qui est une extension de la logique CTL. Nous définissons également des extensions de cette logique de la même manière que pour la logique CTL.

Nous commençons par rappeler la définition de la logique de temps arborescent CTL [Eme90], qui a été largement étudiée pour vérifier des propriétés sur les structures à un seul agent comme les systèmes de transitions.

1.2.1 Logiques pour les systèmes de transitions

1.2.1.1 la logique CTL

Définition 1.10
La syntaxe de la logique CTL est donnée par la grammaire suivante :

$$\begin{aligned} \mathsf{CTL} \ni \varphi_s, \psi_s &::= \top \mid P \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \mathbf{E}\varphi_p \mid \mathbf{A}\varphi_p \\ \varphi_p &::= \mathbf{X}\,\varphi_s \mid \varphi_s\,\mathbf{U}\,\psi_s \mid \mathbf{G}\,\varphi_s \end{aligned}$$

avec P un élément de l'ensemble AP. Les formules de la forme φ_s sont appelées des formules d'états, *tandis que les formules de la forme φ_p sont dites des* formules de chemins.

La sémantique de la logique CTL est donnée par :

Définition 1.11
Pour un système de transitions $\mathcal{K}$, un état ℓ de $\mathcal{K}$ (resp. une exécution ρ) satisfait une formule CTL φ_s (resp. une formule de chemin φ_p) qu'on note par $\ell \models_{\mathcal{K}} \varphi_s$ (resp. $\rho \models_{\mathcal{K}} \varphi_p$), si les conditions suivantes sont satisfaites :

$$\begin{aligned} \ell &\models_{\mathcal{K}} \top \\ \ell &\models_{\mathcal{K}} P && \Leftrightarrow && P \in \mathsf{Lab}(\ell), \\ \ell &\models_{\mathcal{K}} \varphi_s \vee \psi_s && \Leftrightarrow && \ell \models_{\mathcal{K}} \varphi_s \text{ ou } \ell \models_{\mathcal{K}} \psi_s, \\ \ell &\models_{\mathcal{K}} \neg\varphi_s && \Leftrightarrow && \ell \not\models_{\mathcal{K}} \varphi_s, \\ \ell &\models_{\mathcal{K}} \mathbf{A}\varphi_p && \Leftrightarrow && \forall \rho \in \mathsf{Exec}(\ell).\ \rho \models_{\mathcal{K}} \varphi_p, \\ \ell &\models_{\mathcal{K}} \mathbf{E}\varphi_p && \Leftrightarrow && \exists \rho \in \mathsf{Exec}(\ell).\ \rho \models_{\mathcal{K}} \varphi_p, \\ \rho &\models_{\mathcal{K}} \mathbf{X}\,\varphi_s && \Leftrightarrow && \rho[1] \models_{\mathcal{K}} \varphi_s, \\ \rho &\models_{\mathcal{K}} \varphi_s\,\mathbf{U}\,\psi_s && \Leftrightarrow && \exists i.\ \rho[i] \models_{\mathcal{K}} \psi_s \text{ et } \forall 0 \leq j < i.\ \rho[j] \models_{\mathcal{K}} \varphi_s, \\ \rho &\models_{\mathcal{K}} \mathbf{G}\,\varphi_s && \Leftrightarrow && \forall i.\ \rho[i] \models_{\mathcal{K}} \varphi_s. \end{aligned}$$

Pour les formules d'états, la signification des opérateurs booléens est claire. La signification de $\mathbf{A}\varphi_p$ est que sur *tous les chemins*, on a φ_p. $\mathbf{E}\varphi_p$ signifie qu'*il existe un chemin* qui vérifie φ_p.
Pour les formules de chemins, on désigne par $\varphi_s\,\mathbf{U}\,\psi_s$ qu'on va *visiter des états satisfaisants φ_s jusqu'à arriver un jour* dans un état qui satisfait ψ_s. $\mathbf{X}\,\varphi_s$ signifie que l'état *suivant* satisfait φ_s.

Les raccourcis syntaxiques À part les modalités $\mathbf{X}$ et $\mathbf{U}$, il est courant d'utiliser les raccourcis syntaxiques, on présente ci-dessous une liste non-exhaustive de quelques opérateurs classiques utilisés dans la litérature.

- Les formules booléennes classiques $\bot \stackrel{def}{=} \neg\top$, $\varphi \wedge \psi \stackrel{def}{=} \neg(\neg\varphi \vee \neg\psi)$, $\varphi \rightarrow \psi \stackrel{def}{=} (\neg\varphi) \vee \psi$ et $\varphi \leftrightarrow \psi \stackrel{def}{=} \varphi \rightarrow \psi \wedge \psi \rightarrow \varphi$;
- $\mathbf{F}\,\varphi \stackrel{def}{=} \top\,\mathbf{U}\,\varphi$ signifie qu'on va avoir *un jour* dans le futur la formule φ satisfaite,
- $\mathbf{G}\,\varphi \equiv \neg\mathbf{F}\,\neg\varphi$,

- $\varphi \mathbf{R} \psi \stackrel{def}{=} \neg((\neg\varphi) \mathbf{U} (\neg\psi))$ est l'opérateur dual à l'opérateur $\mathbf{U}$, et se lit "*Release*". La signification de cet opérateur est que sur un chemin donné, la présence d'un état satisfaisant la formule φ *relâche* la contrainte d'avoir toujours ψ,
- $\varphi \mathbf{W} \psi \stackrel{def}{=} \mathbf{G}\,\varphi \vee \varphi \mathbf{U} \psi$ est une forme faible de la modalité $\mathbf{U}$. À l'aide de cette modalité, on n'a plus l'obligation, le long d'un chemin, de satisfaire ψ dans le futur tant qu'on satisfait φ. Cette modalité est une réécriture de l'opérateur précédent $\mathbf{R}$: $\varphi \mathbf{W} \psi \equiv \psi \mathbf{R} (\varphi \vee \psi)$. On l'appelle un "*Until faible*" (ou *Weak Until*).

Définition 1.12
La taille d'une formule φ, qu'on note $|\varphi|$, est le nombre d'opérateurs, de modalités et de propositions atomiques utilisés pour construire cette formule.

Par exemple, la taille d'une formule $\mathbf{E} P_1 \mathbf{U} P_2$ est 4, puisqu'on a P_1 et P_2 deux propositions atomiques, l'opérateur $\mathbf{U}$ et la modalité $\mathbf{E}$, d'où la taille obtenue.

Par contre, si on utilise des formules comprenant des raccourcis syntaxiques, comme la formule $\mathbf{EX}\ (P_1 \mathbf{U}\ \mathbf{AF} P_2)$, il faut calculer la taille de la formule équivalente avec des opérateurs définis par la logique. Notre formule devient donc $\mathbf{EX}\ (P_1 \mathbf{U}\ (\mathbf{A} \top \mathbf{U} P_2))$ qui a la taille 8. Mais en général, cette différence n'est pas importante, puisque la taille de la formule obtenue, en remplaçant les raccourcis syntaxiques, reste polynomiale dans la taille de la formule de départ.

On note, qu'une formule peut être représentée par un arbre, chaque nœud étant un opérateur ou une modalité, les feuilles seront des propositions atomiques. Le nombre de fils issus de chaque noeud étant l'arité de l'opérateur représenté par ce nœud. Ainsi, la taille d'une formule sera la taille de l'arbre correspondant. Si on représente cet arbre par un DAG (graphe orienté acyclique), on définit la taille DAG comme la taille du DAG qui représente cette formule.

Model-checking. Le model-checking de la logique CTL sur des systèmes de transitions ou les structures de Kripke est PTIME-complet. Le model-checking est basé sur un algorithme d'étiquetage (où on étiquette les états par des sous-formules) et sur les méthodes de calcul de points fixes. Plusieurs outils efficaces ont été développés pour ce but [CCGR99].

1.2.1.2 La logique TCTL

Cette logique est une extension de la logique CTL. On constate l'existence de deux logiques différentes selon la structure et les propriétés qu'on traite.

On commence par définir la logique TCTL, cette logique a été introduite dans [EMSS92, ACD93], Elle est généralement utilisé pour vérifier des propriétés pour le temps discret, ainsi que des propriétés pour le temps dense. Formellement,

Définition 1.13
La syntaxe de la logique TCTL est donnée par la grammaire suivante :

$$\begin{aligned} TCTL \ni \varphi_s, \psi_s &::= \top \mid P \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \mathbf{E}\varphi_p \mid \mathbf{A}\varphi_p \\ \varphi_p &::= \mathbf{X}\,\varphi_s \mid \varphi_s \mathbf{U}_{\sim\zeta} \psi_s \mid \varphi_s \mathbf{R}_{\sim\zeta} \psi_s \end{aligned}$$

avec $P \in AP$, $\sim \in \{<, \leq, =, \geq, >\}$, et $\zeta \in \mathbb{N}$.

Définition 1.14
Les clauses suivantes définissent quand un état ℓ (resp. une exécution $\rho = (\ell_0, d_0, \ell_1)(\ell_1, d_1, \ell_2)\ldots$) d'un système de transitions pondéré $\mathcal{K}$ satisfait une formule TCTL φ_s (resp. une formule de chemin φ_p), qu'on écrit $\ell \models_{\mathcal{K}} \varphi_s$ (resp. $\rho \models_{\mathcal{K}} \varphi_p$), par induction sur la structure de la formule :

$$\begin{aligned}
\ell \models_{\mathcal{K}} \mathbf{E}\varphi_p &\Leftrightarrow \exists \rho \in \mathit{Exec}(\ell).\ \rho \models_{\mathcal{K}} \varphi_p,\\
\ell \models_{\mathcal{K}} \mathbf{A}\varphi_p &\Leftrightarrow \forall \rho \in \mathit{Exec}(\ell).\ \rho \models_{\mathcal{K}} \varphi_p,\\
\rho \models_{\mathcal{K}} \mathbf{X}\,\varphi_s &\Leftrightarrow \rho[1] \models_{\mathcal{A}} \varphi_s,\\
\rho \models_{\mathcal{K}} \varphi_s\,\mathbf{U}_{\sim\zeta}\,\psi_s &\Leftrightarrow \exists i \in \mathbb{N}.\ \rho[i] \models_{\mathcal{K}} \psi_s,\ \mathit{Cost}(0 \twoheadrightarrow_\rho i) \sim \zeta\\
&\qquad \text{et } \rho[j] \models_{\mathcal{K}} \varphi_s \text{ pour tout } \rho[0] \leq_\rho \rho[j] <_\rho \rho[i],\\
\rho \models_{\mathcal{K}} \varphi_s\,\mathbf{R}_{\sim\zeta}\,\psi_s &\Leftrightarrow \rho \models_{\mathcal{K}} \neg(\neg\varphi_s\,\mathbf{U}_{\sim\zeta}\,\neg\psi_s).
\end{aligned}$$

Le model-checking de TCTL a été étudié pour différents modèles : les structures de Kripke classiques où on compte le nombre de transitions comme poids [EMSS92], les automates temporisés [ACD93] et les DTG qui sont des graphes avec des durées entières [LMS06].

On présente maintenant une autre logique basée sur les horloges, pour cette raison, on rappelle la notion des valuations des horloges :
Étant donné un ensemble $\mathcal{C}$ de *variables d'horloge*, une *valuation* est une fonction $v\colon \mathcal{C} \to \mathbb{R}^+$. Étant donnée une valuation v, un délai $t \in \mathbb{R}^+$ et un sous-ensemble $Z \subseteq \mathcal{C}$, une valuation $v' = v + t$ est définie par $v'(c) = v(c) + t$ pour toute $c \in \mathcal{C}$, et la valuation $v'' = v[Z \leftarrow 0]$ est définie par $v''(c) = v(c)$ si $c \notin Z$ et $v''(c) = 0$ sinon. On écrit v_0 pour la valuation t.q. $v_0(c) = 0$ pour toute $c \in \mathcal{C}$.

Les états d'un système de transitions pondérés peuvent être représentés par des couples de forme (q, v) où q est un état de la structure de base et v une valuation. Ces systèmes de transitions sont utilisés pour décrire le comportement d'un automate temporisé.

Pour ce cas, on définit la logique $\mathsf{TCTL}_{\mathcal{C}}$ dans laquelle interviennent des horloges pour mesurer le temps écoulé.

Définition 1.15
La syntaxe de la logique $\mathit{TCTL}_{\mathcal{C}}$ est donnée par la grammaire suivante :

$$\begin{aligned}
\mathit{TCTL}_{\mathcal{C}} \ni \varphi_s, \psi_s &::= P \mid c \sim n \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \mathbf{A}\varphi_p \mid \mathbf{E}\varphi_p \mid c.\varphi_s\\
\varphi_p &::= \varphi_s\,\mathbf{U}\,\psi_s \mid \mathbf{G}\,\varphi_s
\end{aligned}$$

où P varie dans AP, et c varie dans un ensemble fini des horloges de formule, $\sim \in \{<, \leq, =, \geq, >\}$ et $n \in \mathbb{N}$.

Définition 1.16
Soit $\mathcal{K} = \langle \mathit{Loc}, \mathit{AP}, \mathit{Lab}, \delta_{\mathbb{W}} \rangle$ un système de transitions pondéré. Étant donnée une trajectoire infinie ρ dans $\mathcal{K}$ et soit (ℓ, d, ℓ') une transition le long de ρ qui correspond à un état ℓ. Soient $\varphi \in \mathit{TCTL}_{\mathcal{C}}$, et w une valuation des horloges de formule qui apparaissent dans φ. Un état ℓ satisfait

la formule φ sous la valuation w dans $\mathcal{K}$, qu'on note par $\ell \models^w_{\mathcal{K}} \varphi$, est définie récursivement par :

$$
\begin{array}{rcl}
\ell \models^w_{\mathcal{K}} P & \Leftrightarrow & P \in \mathit{Lab}(\ell) \\
\ell \models^w_{\mathcal{K}} c \sim n & \Leftrightarrow & w(c) \sim n \\
\ell \models^w_{\mathcal{K}} \neg\varphi_s & \Leftrightarrow & \ell \not\models \varphi_s \\
\ell \models^w_{\mathcal{K}} \varphi_s \vee \psi_s & \Leftrightarrow & \ell \models^w_{\mathcal{K}} \varphi_s \text{ ou } \ell \models^w_{\mathcal{K}} \psi_s \\
\ell \models^w_{\mathcal{K}} \mathbf{A}\varphi_p & \Leftrightarrow & \forall \rho \in \mathit{Exec}(\ell).\rho \models^w_{\mathcal{K}} \varphi_p \\
\ell \models^w_{\mathcal{K}} \mathbf{E}\varphi_p & \Leftrightarrow & \exists \rho \in \mathit{Exec}(\ell).\rho \models^w_{\mathcal{K}} \varphi_p \\
\ell \models^w_{\mathcal{K}} c.\varphi_s & \Leftrightarrow & \ell \models^{w[c\leftarrow 0]}_{\mathcal{K}} \varphi_s \\
\rho \models^w_{\mathcal{K}} \varphi_s \mathbf{U} \psi_s & \Leftrightarrow & \exists j''.\rho[0] <_\rho \rho[j''] \text{ et } \rho[j''] \models^{w''}_{\mathcal{K}} \psi_s \text{ et} \\
 & & \forall j'.\rho[0] <_\rho \rho[j'] <_\rho \rho[j''] \text{ et } \rho[j'] \models^{w'}_{\mathcal{K}} \varphi_s, \\
 & & \text{où } w^{(i)} = w + \mathit{Cost}(0 \twoheadrightarrow_\rho j^{(i)}) \\
\rho \models^w_{\mathcal{K}} \varphi_s \mathbf{R} \psi_s & \Leftrightarrow & \rho \models^w_{\mathcal{K}} \neg(\neg\varphi_s \mathbf{U} \neg\psi_s).
\end{array}
$$

Comparaison entre TCTL et TCTL$_\mathcal{C}$. La différence entre les deux logiques se situe au niveau des horloges de formules. On note qu'il est facile de traduire des formules TCTL à des formules TCTL$_\mathcal{C}$. On peut traduire une formule TCTL en utilisant une seule horloge : $\mathbf{E}\varphi_1 \mathbf{U}_{\sim\zeta} \varphi_2 \equiv c.\, \mathbf{E}\varphi_1 \mathbf{U} (\varphi_2 \wedge c \sim \zeta)$. À l'aide de cette traduction, on peut voir que la logique TCTL$_\mathcal{C}$ est au moins aussi expressive que TCTL. Dans le sens inverse, il a été montré que TCTL$_\mathcal{C}$ est strictement plus expressive que TCTL dans les automates temporisés [BCM05, AH93].

Puisque la quantification dans les formules CTL se fait sur les chemins (et non sur l'existence de stratégies), cette logique ne convient pas pour caractériser des interactions entre plusieurs agents dans des structures de jeu à plusieurs joueurs. La section suivante introduit une logique récente, la logique ATL, où la quantification sur les stratégies des joueurs est possible.

1.2.2 Logiques pour les jeux

1.2.2.1 La logique ATL

On s'intéresse maintenant à définir la logique ATL. L'intérêt de cette logique est d'exprimer des propriétés sur des CGS et des ATS (qui représentent d'autres structures de jeux définies dans la définition 2.1) sur l'existence de stratégies pour une coalition afin de garantir des propriétés temporelles. Notre définition est légèrement différente de celle proposée dans [AHK02]. Nous soulignerons cette différence plus tard dans la section 2.2.2.

Définition 1.17
La syntaxe de la logique ATL est définie par la grammaire suivante :

$$
\begin{array}{rcl}
\mathit{ATL} \ni \varphi_s, \psi_s & ::= & \top \mid P \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \langle\!\langle A \rangle\!\rangle \varphi_p \\
\varphi_p & ::= & \neg\varphi_p \mid \mathbf{X}\varphi_s \mid \varphi_s \mathbf{U} \psi_s
\end{array}
$$

avec P un élément de l'ensemble AP et A un sous-ensemble de Agt.

On définit la taille d'une formule ATL de la même manière que pour CTL.

En comparant avec la syntaxe de la logique du temps arborescent, on peut voir que la syntaxe de ATL diffère de celle de CTL par l'introduction de la modalité $\langle\langle A\rangle\rangle \varphi_p$ (que l'on peut lire comme "*la coalition des joueurs A possède une stratégie pour satisfaire* φ_p"). Donc, informellement, à l'aide de cette nouvelle modalité, la quantification se fait sur les stratégies des joueurs, alors que, dans le cas de la logique CTL, on quantifie uniquement sur les chemins.

Avec cette nouvelle logique, on peut exprimer des propriétés qui n'étaient pas exprimables à l'aide de la logique CTL. Par exemple, avec la formule ci-dessous, on énonce dans l'exemple 2.1 la propriété qui dit que lorsque le train passe la barrière, le contrôleur ne peut pas le forcer à y rentrer. La formule ATL correspondante est la suivante :

$$\langle\langle \varnothing \rangle\rangle \, \mathbf{G} \, (\texttt{out_of_gate} \Rightarrow \neg \langle\langle \texttt{ctrl} \rangle\rangle \, \mathbf{F} \, \texttt{in_gate})$$

Formellement, la sémantique de la logique ATL est la suivante :

Définition 1.18
Les formules ATL sont interprétées sur des états d'une structure de jeu $\mathcal{S}$*. On dit qu'un état* ℓ *satisfait une formule* φ *dans* $\mathcal{S}$*, et qu'on note par* $\ell \models_{\mathcal{S}} \varphi$*, s'il satisfait les conditions ci-dessous :*

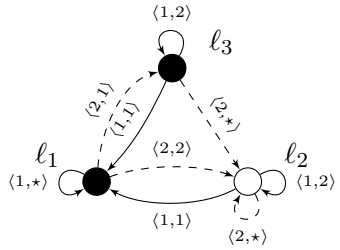
$$\begin{array}{rcl}
\ell \models_{\mathcal{S}} \top & & \\
\ell \models_{\mathcal{S}} P & \Leftrightarrow & P \in \mathit{Lab}(\ell), \\
\ell \models_{\mathcal{S}} \varphi_s \vee \psi_s & \Leftrightarrow & \ell \models_{\mathcal{S}} \varphi_s \text{ ou } \ell \models_{\mathcal{S}} \psi_s, \\
\ell \models_{\mathcal{S}} \neg\varphi_s & \Leftrightarrow & \ell \not\models_{\mathcal{S}} \varphi_s, \\
\ell \models_{\mathcal{S}} \langle\langle A\rangle\rangle \varphi_p & \Leftrightarrow & \exists F_A \in \mathsf{Strat}(A).\ \forall \rho \in \mathsf{Out}(\ell, F_A).\ \rho \models_{\mathcal{S}} \varphi_p, \\
\rho \models_{\mathcal{S}} \neg\varphi_p & \Leftrightarrow & \rho \not\models_{\mathcal{S}} \varphi_p, \\
\rho \models_{\mathcal{S}} \mathbf{X}\, \varphi_s & \Leftrightarrow & \rho[1] \models_{\mathcal{S}} \varphi_s, \\
\rho \models_{\mathcal{S}} \varphi_s \, \mathbf{U} \, \psi_s & \Leftrightarrow & \exists i.\ \rho[i] \models_{\mathcal{S}} \psi_s \text{ et } \forall 0 \leq j < i.\ \rho[j] \models_{\mathcal{S}} \varphi_s.
\end{array}$$

On sait que pour la logique ATL, il est suffisant de se restreindre à des stratégies sans mémoire (*i.e.*, $\langle\langle A\rangle\rangle \varphi_p$ est satisfaite ssi il existe une stratégie sans mémoire dont tous les outcomes satisfont φ_p) [AHK02, Sch04].

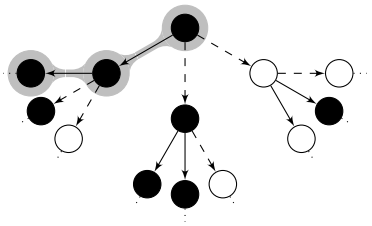
On remarque que $\langle\langle \varnothing \rangle\rangle \varphi_p$ correspond à la formule CTL $\mathbf{A}\varphi_p$ (*i.e.*, quantification universelle sur toutes les exécutions à partir de l'état courant), et $\langle\langle \mathsf{Agt} \rangle\rangle \varphi_p$ correspond à la quantification existentielle $\mathbf{E}\varphi_p$. En général, $\neg \langle\langle A \rangle\rangle \varphi_p$ n'est pas équivalente à $\langle\langle \mathsf{Agt} \setminus A \rangle\rangle \neg\varphi_p$ [AHK02, GvD06] : en effet, l'absence d'une stratégie pour la coalition A pour assurer φ n'implique pas l'existence d'une stratégie pour la coalition $\mathsf{Agt} \setminus A$ pour assurer $\neg\varphi$. Cependant, cette propriété devient valide si on restreint la structure à des jeux turn-based.

Exemple 1.4
Prenons la CGS de la figure 1.4(a), cette CGS comprend deux étiquettes d'état ; l'une est représentée par un état noir, et l'autre par l'état blanc. Considérons les formules ATL $\varphi = \langle\langle A \rangle\rangle \, \mathbf{G} \bullet$ *et* $\psi = \langle\langle A \rangle\rangle \, \mathbf{F} \circ$*. Dans chaque état, chaque joueur dispose de deux coups, on s'intéresse notamment aux choix du joueur A, notre protagoniste. Dans la figure, on distingue graphiquement les actions de ce joueur par des transitions solides (qui désignent le coup* 1 *pris par le*

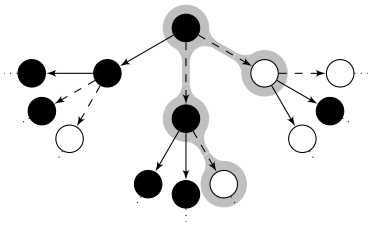
joueur A) ou des transitions en pointillé (désignant le coup 2). On représente ensuite, dans la figure 1.4(b), l'arbre d'exécution de cette CGS à partir de l'état ℓ_1. En examinant cette figure, on se convainc facilement que l'état ℓ_1 satisfait les formules φ (le joueur A dispose d'une stratégie pour rester toujours *dans des états noirs) et ψ (le joueur A possède une stratégie pour arriver* un jour *à un état blanc). En effet, la stratégie à suivre pour satisfaire la formule φ est de jouer toujours dans l'état ℓ_1 le coup* 1 *(la transition solide). Pour satisfaire la formule ψ, la stratégie gagnante λ_A est la suivante : $\lambda_A(\ell_1) = 2$, $\lambda_A(\ell_2) = 2$ et $\lambda_A(\ell_3) = 2$ (en d'autres termes, choisir toujours la transition en pointillé)*[4].



(a) La CGS de l'exemple 1.4



(b) soulignant la stratégie garantissant φ



(c) soulignant la stratégie garantissant ψ

FIG. 1.4 – Un arbre d'exécution à partir de l'état ℓ_1 vérifiant les formules ATL φ et ψ

1.2.2.2 Les extensions

Comme pour les logiques temporelles classiques, on considère dans cette section plusieurs extensions d'ATL qui donnent plus de possibilités dans la manière de combiner les quantifications sur les stratégies et les modalités temporelles.

La logique ATL* . On commence par définir la logique ATL* . La logique ATL, ainsi que les logiques ATL$^+$ et EATL (qu'on va définir dans la suite) sont des fragments de cette logique plus large. La définition d'ATL* [AHK02] est la suivante :

[4]Les parties de l'arbre éclairées en gris, représentent les outcomes de la stratégie gagnante pour chacune des deux formules

Définition 1.19
La syntaxe de la logique ATL* est définie par la grammaire suivante :

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

avec $P \in \mathit{AP}$ et $A \subseteq \mathit{Agt}$.

La taille et la taille DAG d'une formule ATL* sont définies de la même manière que pour ATL. On interprète les formules ATL* sur les états d'une structure de jeu $\mathcal{S}$, la sémantique des modalités est la suivante (si $\rho = \rho_0\,\rho_1 \ldots$ avec $\rho_i \in \delta$, on désigne par ρ^i le $i+1$-ème suffixe $\rho_i\rho_{i+1}\ldots$) :

$$\begin{aligned} \ell \models_{\mathcal{S}} \langle\langle A \rangle\rangle \varphi_p &\quad\Leftrightarrow\quad \exists F_A \in \mathsf{Strat}(A).\ \forall \rho \in \mathsf{Out}(\ell, F_A).\ \rho \models_{\mathcal{S}} \varphi_p, \\ \rho \models_{\mathcal{S}} \varphi_s &\quad\Leftrightarrow\quad \rho[0] \models_{\mathcal{S}} \varphi_s \\ \rho \models_{\mathcal{S}} \mathbf{X}\,\varphi_p &\quad\Leftrightarrow\quad \rho^1 \models_{\mathcal{S}} \varphi_p, \\ \rho \models_{\mathcal{S}} \varphi_p \,\mathbf{U}\, \psi_p &\quad\Leftrightarrow\quad \exists i.\rho^i \models_{\mathcal{S}} \psi_p \text{ et } \forall 0 \leq j < i.\ \rho^j \models_{\mathcal{S}} \varphi_p \end{aligned}$$

ATL est le fragment de la logique ATL* où chaque quantification sur les stratégies $\langle\langle A \rangle\rangle$ doit être suivie par une modalité $\mathbf{U}$ ou $\mathbf{X}$. On peut définir également d'autres fragments :

La logique ATL$^+$. ATL$^+$ est la restriction d'ATL* où un quantificateur sur la stratégie $\langle\langle A \rangle\rangle$ doit toujours être inséré entre deux modalités temporelles $\mathbf{U}$ ou $\mathbf{X}$.

Définition 1.20
La syntaxe de la logique ATL$^+$ est définie par la grammaire suivante :

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

avec $P \in \mathit{AP}$ et $A \subseteq \mathit{Agt}$.

Notons que la logique ATL$^+$ bénéficie du même pouvoir expressif que celui de la logique ATL [HRS02]. Et donc, à l'aide de cette logique, on exprime des propriétés exprimables en ATL, sauf que les formules obtenues par la logique ATL$^+$ sont généralement exponentiellement plus concises.

La logique EATL. EATL étend ATL en autorisant les opérateurs $\langle\langle A \rangle\rangle\, \mathbf{G}\,\mathbf{F}$ (noté $\langle\langle A \rangle\rangle \overset{\infty}{\mathbf{F}}$) et $\langle\langle A \rangle\rangle\, \mathbf{F}\,\mathbf{G}$ (noté $\langle\langle A \rangle\rangle \overset{\infty}{\mathbf{G}}$). Ces nouveaux opérateurs sont utilisés pour exprimer des propriétés d'équité.

Définition 1.21
La syntaxe de la logique EATL est définie par la grammaire suivante :

$$\begin{aligned} \mathit{EATL} \ni \varphi_s, \psi_s &::= \top \mid P \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \langle\langle A \rangle\rangle \varphi_p \\ \varphi_p, \psi_p &::= \neg\varphi_p \mid \mathbf{X}\,\varphi_s \mid \varphi_s \,\mathbf{U}\, \psi_s \mid \overset{\infty}{\mathbf{F}}\,\varphi_s \end{aligned}$$

avec $P \in \mathit{AP}$ et $A \subseteq \mathit{Agt}$.

L'opérateur $\overset{\infty}{\mathbf{G}}\,\varphi_s$ est donc un raccourci syntaxique exprimant le dual de l'opérateur $\overset{\infty}{\mathbf{F}}$, il peut donc être exprimé par $\overset{\infty}{\mathbf{G}}\,\varphi_s \equiv \neg\overset{\infty}{\mathbf{F}}\,\neg\varphi_s$.

Dans le cas de la logique EATL, on tient à préciser que la logique est strictement plus expressive que la logique ATL (et par suite la logique ATL$^+$). Le sujet de l'expressivité sera abordé dans le chapitre suivant.

Exemple 1.5
La formule φ ci-dessous exprime que le joueur A dispose d'une stratégie qui garantit qu'à partir d'un état satisfaisant une propriété $\bullet$ (la couleur noire dans la figure 1.5), le joueur B dispose d'une stratégie pour passer infiniment souvent dans un état ayant la propriété $\circ$ (la couleur blanche dans la figure) :

$$\varphi = \langle\!\langle A \rangle\!\rangle\, \mathbf{G}\, (\bullet \rightarrow \langle\!\langle B \rangle\!\rangle\, \overset{\infty}{\mathbf{F}}\, \circ)$$

D'autre part, la formule ψ ci-dessous énonce qu'il existe un chemin où, un jour, le joueur A possède une stratégie pour passer toujours infiniment souvent dans des états satisfaisant $\bullet$.

$$\psi = \langle\!\langle A, B \rangle\!\rangle\, \mathbf{F}\, \langle\!\langle A \rangle\!\rangle\, \overset{\infty}{\mathbf{G}}\, \bullet$$

On peut vérifier que ces deux formules sont satisfaites dans l'état ℓ_1 de la figure 1.5 .

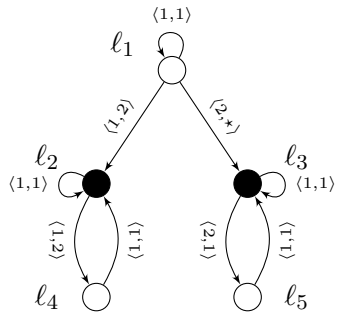


FIG. 1.5 – Une CGS vérifiant les formules de l'exemple 1.5

Exemples. Quelques formules soulignant la différence entre ces extensions :

$$\langle\!\langle A \rangle\!\rangle \left(\mathbf{F}\, P_1 \wedge \mathbf{F}\, P_2 \wedge P_3\, \mathbf{U}\, P_4\right) \text{ est une formule ATL}^+,$$
$$\langle\!\langle A \rangle\!\rangle\, \mathbf{F} \left(P_1 \wedge \langle\!\langle A' \rangle\!\rangle\, \overset{\infty}{\mathbf{F}}\, P_2\right) \text{ est une formule EATL},$$
$$\langle\!\langle A \rangle\!\rangle \left(\mathbf{F}\, P_1 \wedge P_2\, \mathbf{U}\, (P_3 \wedge \mathbf{F}\, P_4)\right) \text{ est une formule ATL}^\star.$$

1.2.2.3 La logique ATL temporisée

Maintenant, on étend la logique ATL avec la notion du temps, pour obtenir deux fragments similaires à ce qu'on vient de présenter pour la logique CTL.

La logique TATL. Nous commençons par définir la logique TATL, l'extension de TCTL avec jeu :

Définition 1.22
La syntaxe de la logique TATL est définie par la grammaire suivante :

$$\begin{aligned} \mathit{TATL} \ni \varphi_s, \psi_s &::= \top \mid P \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \langle\langle A\rangle\rangle \varphi_p \\ \varphi_p &::= \mathbf{X}\,\varphi_s \mid \varphi_s \,\mathbf{U}_{\sim\zeta}\, \psi_s \mid \varphi_s \,\mathbf{R}_{\sim\zeta}\, \psi_s \end{aligned}$$

avec $P \in \mathit{AP}$, $A \subseteq \mathit{Agt}$, $\sim \in \{<, \leq, =, \geq, >\}$, *et* $\zeta \in \mathbb{N}$.

Définition 1.23
les clauses suivantes définissent quand un état ℓ *(resp. une exécution* $\rho = (\ell_0, d_0, \ell_1)(\ell_1, d_1, \ell_2)\ldots$*) d'une CGS* $\mathcal{S}$ *satisfait une formule TATL* φ_s *(resp. une formule de chemin* φ_p*), qu'on écrit* $\ell \models_{\mathcal{S}} \varphi_s$ *(resp.* $\rho \models_{\mathcal{S}} \varphi_p$*), par induction sur la structure de la formule :*

$$\begin{aligned} \ell \models_{\mathcal{S}} \langle\langle A\rangle\rangle \varphi_p &\Leftrightarrow \exists \lambda_A \in \mathsf{Strat}(A).\ \forall \rho \in \mathsf{Out}(\ell, \lambda_A).\ \rho \models_{\mathcal{S}} \varphi_p \\ \rho \models_{\mathcal{S}} \mathbf{X}\,\varphi_s &\Leftrightarrow \rho[1] \models_{\mathcal{S}} \varphi_s \\ \rho \models_{\mathcal{S}} \varphi_s \,\mathbf{U}_{\sim\zeta}\, \psi_s &\Leftrightarrow \exists i \in \mathbb{N}.\ \rho[i] \models_{\mathcal{S}} \psi_s,\ \mathit{Cost}(0 \twoheadrightarrow_\rho i) \sim \zeta \\ &\qquad \text{et } \rho[j] \models_{\mathcal{S}} \varphi_s \text{ pour tout } \rho[0] \leq_\rho \rho[j] <_\rho \rho[i] \\ \rho \models_{\mathcal{S}} \varphi_s \,\mathbf{R}_{\sim\zeta}\, \psi_s &\Leftrightarrow \rho \models_{\mathcal{S}} \neg(\neg\varphi_s \,\mathbf{U}_{\sim\zeta}\, \neg\psi_s) \end{aligned}$$

Quand le contexte est clair, l'indice $\mathcal{S}$ ne figure pas et on se contente d'écrire $\ell \models \varphi$.

Intuitivement, $\varphi_s \,\mathbf{R}_{\sim\zeta}\, \psi_s$ nécessite que ψ_s soit vraie tant que "$\sim \zeta$" est satisfaite, mais cette contrainte sera relâchée une fois que φ_s devient vraie. Formellement :

$$\begin{aligned} \rho \models_{\mathcal{S}} \varphi_s \,\mathbf{R}_{\sim\zeta}\, \psi_s \quad \Leftrightarrow \quad & \forall i \in \mathbb{N}.\ (\mathsf{Cost}(0 \twoheadrightarrow_\rho i) \sim \zeta \rightarrow \rho[i] \models_{\mathcal{A}} \psi_s) \\ & \text{ou } \exists \rho[j] <_\rho \rho[i].\ \rho[j] \models_{\mathcal{S}} \varphi_s. \end{aligned}$$

$\mathsf{TATL}_{\leq,\geq}$ désigne le fragment de TATL où on interdit l'utilisation des indices de la forme "$= \zeta$" dans les contraintes temporelles pour $\mathbf{U}$ et $\mathbf{R}$.

La logique $\mathsf{TATL}_{\mathcal{C}}$. On définit une autre logique quantitative pour les jeux, nommée $\mathsf{TATL}_{\mathcal{C}}$. On trouve des horloges qui interviennent dans les formules de cette logique qui étend la logique $\mathsf{TCTL}_{\mathcal{C}}$.

Définition 1.24
Étant donné un ensemble d'horloges de formules $\mathcal{X}$*, La syntaxe de la logique* $\mathit{TATL}_{\mathcal{C}}$ *est donnée par la grammaire suivante :*

$$\begin{aligned} \mathit{TATL}_{\mathcal{C}} \ni \varphi_s, \psi_s &::= P \mid c \sim n \mid \neg\varphi_s \mid \varphi_s \vee \varphi_s \mid \langle\langle A\rangle\rangle \varphi_p \mid c.\varphi_s \\ \varphi_p &::= \varphi_s \,\mathbf{U}\, \varphi_s \mid \mathbf{G}\,\varphi_s \end{aligned}$$

où P *varie dans* AP, $A \subseteq \mathit{Agt}$, *et* c *varie dans* $\mathcal{X}$, $\sim \in \{<, \leq, =, \geq, >\}$ *et* $n \in \mathbb{N}$.

Définition 1.25
Soit $\mathcal{S} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta_{\mathbb{W}}, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg} \rangle$ une CGS construite sur un système de transitions pondéré continu. Étant donnée une trajectoire infinie ρ dans $\mathcal{S}$ et soit (ℓ, d, ℓ') une position le long de r qui correspond à un état ℓ. Soient $\varphi \in \mathsf{TATL}_{\mathcal{C}}$, et w une valuation de formules d'horloge qui apparaissent dans φ. Un état ℓ (ou une exécution r) satisfait une formule φ sous la valuation w, qu'on note par $\ell \models^w_{\mathcal{S}} \varphi$ (ou $\rho \models^w_{\mathcal{S}} \varphi$), est définie récursivement par :

$$
\begin{aligned}
\ell \models^w_{\mathcal{S}} P &\Leftrightarrow P \in \mathsf{Lab}(\ell) \\
\ell \models^w_{\mathcal{S}} c \sim n &\Leftrightarrow w(c) \sim n \\
\ell \models^w_{\mathcal{S}} \neg\varphi_s &\Leftrightarrow \ell \not\models \varphi_s \\
\ell \models^w_{\mathcal{S}} \varphi_s \vee \psi_s &\Leftrightarrow \ell \models^w_{\mathcal{S}} \varphi_s \text{ ou } \ell \models^w_{\mathcal{S}} \psi_s \\
\ell \models^w_{\mathcal{S}} \langle\!\langle A \rangle\!\rangle \varphi_p &\Leftrightarrow \exists F_A \in \mathsf{Strat}(A). \forall \rho \in \mathsf{Out}^{\infty}(\ell, F_A). \rho \models^w_{\mathcal{S}} \varphi_p \\
\ell \models^w_{\mathcal{S}} c.\varphi_s &\Leftrightarrow \ell \models^{w[c \leftarrow 0]}_{\mathcal{S}} \varphi_s \\
\rho \models^w_{\mathcal{S}} \varphi_s \mathbf{U} \psi_s &\Leftrightarrow \exists j''. \rho[0] <_\rho \rho[j''] \text{ et } \rho[j''] \models^{w''}_{\mathcal{S}} \psi_s \text{ et} \\
&\quad \forall j'. \rho[0] <_\rho \rho[j'] <_\rho \rho[j''] \text{ et } \rho[j'] \models^{w'}_{\mathcal{S}} \varphi_s, \\
&\quad \text{où } w^{(i)} = w + \mathsf{Cost}(0 \twoheadrightarrow_\rho j^{(i)}) \\
\rho \models^w_{\mathcal{S}} \varphi_s \mathbf{R} \psi_s &\Leftrightarrow \rho \models^w_{\mathcal{S}} \neg(\neg\varphi_s \mathbf{U} \neg\psi_s).
\end{aligned}
$$

Chacune de ces logiques (ATL, TATL et $\mathsf{TATL}_{\mathcal{C}}$) sera étudiée dans un chapitre séparé : Le chapitre 2 est consacré au model-checking de la logique ATL, le chapitre 3 s'intéresse au model-checking de la logique TATL, et finalement, dans le chapitre 4, le model-checking de la logique $\mathsf{TATL}_{\mathcal{C}}$ est étudié.

Conclusion

Dans ce chapitre on a présenté les types de structures pour modéliser les systèmes multi-agents (les CGS sous deux formes explicites et symboliques), ces structures sont basées sur les systèmes de transitions qui peuvent éventuellement désigner des systèmes pondérés. Nous avons aussi introduit la logique ATL qui étend la logique CTL en quantifiant sur les stratégies des joueurs. À l'aide de cette logique, on peut exprimer des propriétés pour décrire des comportements sur les interactions des différents joueurs, ce qui n'était pas possible avec les logiques antérieures. Des extensions temporisées de cette logique ont également été introduites.

Le model-checking de la logique CTL faisait l'objet de nombreuses études. Or, le cas du model-checking d'ATL a été étudié dans le cas des CGS explicites avec des systèmes de transitions non pondérés. Dans le chapitre suivant, nous allons étudier et comparer les CGS explicites et symboliques ainsi qu'une autre type de structures de jeu (les systèmes de transitions alternants), et établir un lien entre eux, ainsi que présenter la complexité du model-checking de la logique ATL sur ces différents modèles. De même, nous établissons un lien entre les différentes structures étudiées, et nous présentons les algorithmes (ainsi que leur complexité) pour faire la traduction entre les différentes structures de jeu non-pondérées.

Pour en savoir plus ...

La logique ATL est une logique dont la quantification se fait sur les stratégies des joueurs. Or, en imbriquant les quantifications de stratégies, on perd toute information sur les stratégies des joueurs qui ont été prises avant l'introduction de la nouvelle quantification. Récemment, des papiers s'intéressant à ce type de jeu ont été publiés, on note le papier de Henzinger et autres [CHP07] où on trouve une logique sur les stratégies qui ressemble à la logique du premier ordre. Le papier [Pin07] est un travail similaire développé indépendamment. À l'aide de ces deux formalismes, on peut exprimer des propriétés plus intéressantes qui peuvent engendrer des conditions de Nash (équilibre de Nash) et bien d'autres.

Par ailleurs, on trouve une étude [BDLM08] dans ce même cadre mais en s'intéressant surtout à des stratégies sans-mémoire ou bien à des mémoires bornées.

Chapitre 2 Complexité et expressivité d'ATL

Introduction

Après avoir établi les définitions dans le chapitre précédent, nous allons étudier la complexité du model-checking de ATL et ses extensions sur les différents types de structures de jeux.

Le model-checking de CTL a été prouvé PTIME-complet [EH86, Eme90, CE81, QS82]. Cet algorithme est un algorithme d'étiquetage, c.à.d. qui étiquette chaque sous formule par une proposition atomique, et qui consiste à calculer une formule de point fixe à l'aide du calcul de l'ensemble des prédécesseurs. Nous montrerons que, dans le cas de la logique ATL, la complexité est supérieure (sous l'hypothèse que PTIME $\subsetneq$ NP). Toutefois, on note que cette complexité diffère selon la structure utilisée comme modèle. En effet, on montre qu'elle est directement liée au calcul des prédécesseurs contrôlables (l'ensemble des états depuis lesquels on peut avoir une stratégie qui garantit l'arrivée dans un ensemble donné) : dans le cas des CGS explicites, la complexité du model-checking reste polynomiale puisque le calcul du prédécesseur contrôlable consiste à un parcours simple de la table de transitions. Ce calcul est différent dans le cas des CGS symboliques, puisque la table de transitions est concise, le calcul des prédécesseurs contrôlables devient Σ_2^{P}-complet, ce qui rend la complexité générale du model-checking Δ_3^{P}-complet. On introduit un différent type de structure, nommé ATS, où un coup d'un agent est un sous-ensemble d'états (intuitivement ce sont les états vers lesquels le joueur va mener le jeu à l'étape suivante) et l'état successeur est alors l'intersection de tous les coups des agents. La complexité pour calculer les prédécesseurs contrôlables dans ces structures se situe dans NP, et on verra que la complexité du model-checking pour les ATS est Δ_2^{P}-complet. Nous montrerons ces résultats, ainsi que la complexité du model-checking d'autres extensions dans la suite du chapitre.

D'un autre coté, nous nous sommes intéressés à l'étude de l'expressivité des modèles d'une part, et de la logique d'autre part. Nous avons construit des algorithmes optimaux pour traduire les modèles entre eux, la borne inférieure de la complexité de ces algorithmes étant une conséquence directe des résultats du model-checking. Pour la logique, la grammaire de la logique ATL ressemble à celle de la logique CTL, et donc dans la syntaxe, les formules de chemins possibles consistent à avoir des formules avec *Until* (e.g. $\langle\!\langle A \rangle\!\rangle P_1 \mathbf{U} P2$) et des formules avec la modalité *toujours* (e.g. $\langle\!\langle A \rangle\!\rangle \mathbf{G} P$). Dans le cas de la logique CTL, on pouvait, à l'aide de ces deux modalités, exprimer le dual de $\mathbf{U}$ qui est une modalité classique dans les logiques temporelles (qu'on appelle *Release* et qu'on note par $\mathbf{R}$), or, nous montrons que la logique ATL ne préserve pas cette propriété, et donc qu'il est important d'inclure cette modalité dans la définition de ATL.

Le plan du chapitre est le suivant : dans la première section, nous cherchons la complexité

du model-checking de ATL sur les différentes structures de jeu. La deuxième section est destinée à l'étude du pouvoir expressif de la logique ATL.

2.1 La complexité du model-checking d'ATL

Dans cette section, nous allons établir la complexité exacte du model-checking de la logique ATL. Ce problème a été déjà abordé dans les articles introduisant la logique ATL, sur les ATS [AHK98] et les CGS [AHK02]. On a montré que la complexité en temps du problème du model-checking est de l'ordre de $\mathcal{O}(m \cdot l)$, où m est le nombre de transitions et l est la taille de la formule. On a donc montré que cette complexité est dans PTIME (et est clairement PTIME-dur, puisque c'est le cas pour CTL). Mais ce résultat n'est valable que dans le cas des CGS explicites. Pour les ATS, le nombre de transitions peut être exponentiel par rapport à la taille du système (plus précisément, dans le nombre d'agents). Le cas des CGS symboliques ne se situe pas dans une meilleure position pour la complexité du model-checking. La concision de la table de transitions entraîne une explosion combinatoire du nombre des transitions. Par conséquent, les algorithmes classiques nécessitant un temps $\mathcal{O}(m \cdot l)$ dans le cas des CGS définies avec une table de transitions explicites, ne s'exécutent plus en temps polynomial quand cette dernière est donnée sous forme symbolique.

L'algorithme de model-checking de la logique ATL est identique à celui utilisé pour CTL : il consiste à calculer récursivement les points fixes en se basant *e.g.* sur l'équivalence suivante :

$$\langle\langle A\rangle\rangle\, p\, \mathbf{U}\, q \equiv \mu Z.(q \vee (p \wedge \langle\langle A\rangle\rangle\, \mathbf{X}\, Z)) \tag{2.1}$$

La différence avec CTL est qu'on doit traiter les modalités $\langle\langle A\rangle\rangle\, \mathbf{X}$ —qui correspondent à la *pre-image* d'un ensemble d'états *pour une coalition donnée*— au lieu des modalités classiques EX. Dans le cas de contrôle, $\langle\langle A\rangle\rangle\, \mathbf{X}$ correspond aux *prédécesseurs contrôlables* d'un ensemble d'états pour une coalition : la définition de $CPre(A, S)$, où $A \subseteq \mathsf{Agt}$ et $S \subseteq \mathsf{Loc}$, est la suivante :

$$CPre(A, S) \stackrel{\text{def}}{=} \{\ell \in \mathsf{Loc} \mid \exists m_A \in \mathsf{Mov}(\ell, A) \text{ t.q. } \mathsf{Next}(\ell, A, m_A) \subseteq S\}$$

Effectivement, nous verrons que l'élément crucial dans l'algorithme du model-checking est le calcul de l'ensemble $CPre(A, S)$.

Dans la suite, on établit la complexité exacte du calcul de *CPre* (plus précisément, on considère le problème de décision associé : étant donnés $A \subseteq \mathsf{Agt}$, $S \subseteq \mathsf{Loc}$, et $\ell \in \mathsf{Loc}$, décider si $\ell \in CPre(A, S)$), puis on établit la complexité de model-checking de ATL pour les différents types de systèmes multi-agents.

2.1.1 Model-checking d'ATL sur les CGS explicites.

Dans la partie précédente, nous avons discuté de la complexité du model-checking de ATL sur les CGS explicites qui a été établie dans [AHK02] :

Théorème 2.1 [AHK02]

Le model-checking de ATL sur les CGS explicites est PTIME-complet.

À notre connaissance, la complexité exacte de calcul de *CPre* pour les CGS explicites n'a pas encore été étudiée. L'algorithme standard (polynomial) est suffisant pour trouver la complexité du model-checking d'ATL.

En effet, étant donné un état ℓ, un ensemble d'états S et une coalition A, la complexité pour décider si $\ell \in \mathit{CPre}(A, S)$ est largement plus basse que PTIME (voir l'annexe A pour avoir une intuition de ces classes de complexité) :

Proposition 2.1
Calculer le CPre sur les CGS explicites est dans AC^0.

Démonstration: On commence par définir le codage de l'entrée comme une séquence de bits :
- les premiers $k = |\mathsf{Agt}|$ bits définissent la coalition : le i-ème bit est 1 si, et seulement si l'agent A_i appartient à A – on appelle cette sous-séquence "Entrée 3" ;
- les $|\mathsf{Loc}|$ bits suivants définissent l'ensemble S – on appelle cette sous-séquence "Entrée 1" ;
- Pour simplifier, on suppose que tous les agents ont le même nombre d'actions dans ℓ. On écrit p pour ce nombre, on suppose qu'il est au moins 2. La table de transitions $\mathsf{Edg}(\ell)$ est donc donnée comme une séquence de p^k ensembles de $m = \log(|\mathsf{Loc}|)$ bits – on appelle cette sous-séquence "Entrée 2".

Avant de commencer à décrire les différentes étapes, nous décrirons brièvement quelques circuits que nous utilisons pour notre construction et qu'on trouve dans la figure 2.1.
- le rôle du *circuit 1* est de renvoyer pour une entrée b de k-bits une sortie de taille 2^k-bits où le seul bit mis à zéro est le bit numéro b (une flèche en pointillés partante d'un bit désigne qu'elle porte la négation de ce bit) ;
- le *circuit 2* renvoie 1 si et seulement si, pour au moins une même position, les deux bits dans chacune des deux entrées sont 1.
- la construction du *circuit 3* ne dépend pas seulement de la taille de l'entrée mais aussi d'une coalition B. L'entrée est considérée comme l'ensemble des réponses pour toutes les actions possibles depuis un état donné, et la sortie donne une réponse positive s'il existe un ensemble d'actions contrôlées par la coalition B.

La première étape consiste à modifier notre entrée pour avoir la forme suivante :
- les k premiers bits définissent la coalition ;
- ensuite, une séquence de p^k bits nous indique si l'état résultant appartient à S. Pour obtenir cette séquence, on utilise le circuit 1 à chaque m bits de l'entrée 2, la sortie est ensuite liée avec l'entrée 1, avec le circuit 2. L'ensemble des p^k bits donnés par les circuits 2 produisent la séquence en question.

On construit maintenant le circuit qui calcule si la coalition A possède une stratégie pour arriver dans S. Puisque les circuits dépendent uniquement de la taille de l'entrée, on ne peut pas construire de circuit pour la coalition A. On construit plutôt un circuit pour chaque coalition possible (la taille est exponentielle en fonction du nombre d'agents, mais reste polynomiale en fonction de la taille de l'entrée, sachant que $p \geq 2$), et on choisit le résultat correspondant à la coalition A.

Pour chaque coalition possible B, on construit un circuit (circuit 3 de la figure 2.1) dont l'élément final répond 1 ssi $\ell \in \mathit{CPre}(B, S)$. On arrive à ceci en utilisant des circuits avec un nombre d'entrées non borné de profondeur 2 : au premier niveau, on utilise $p^{|B|}$ portes ET (les portes g_i), qui représentent chacune des $p^{|B|}$ actions possibles de la coalition B (on note

par $action_i(B)$ l'action numéro i de la coalition B). Chacune des ces portes est liée à $p^{k-|B|}$ bits de la table de transitions qui correspondent à l'ensemble des $p^{k-|B|}$ actions possibles des adversaires. Au deuxième niveau, une porte OU est liée à toutes les portes du niveau 1.

Clairement, la porte OU au niveau 2 est évaluée à vrai ssi la coalition B possède une stratégie pour arriver à S. De plus, il y a $\binom{k}{l}$ coalitions de taille l, chacune est traitée par un circuit possédant p^l+1 portes. Le circuit résultant possède alors $(p+1)^k+2^k$ portes, ce qui est polynomial dans la taille de l'entrée. Ce circuit est donc un circuit AC^0.

Il reste simplement à renvoyer le résultat qui correspond à la coalition A. Ce qui est fait facilement dans AC^0. Le circuit final est représenté dans la figure 2.2. □

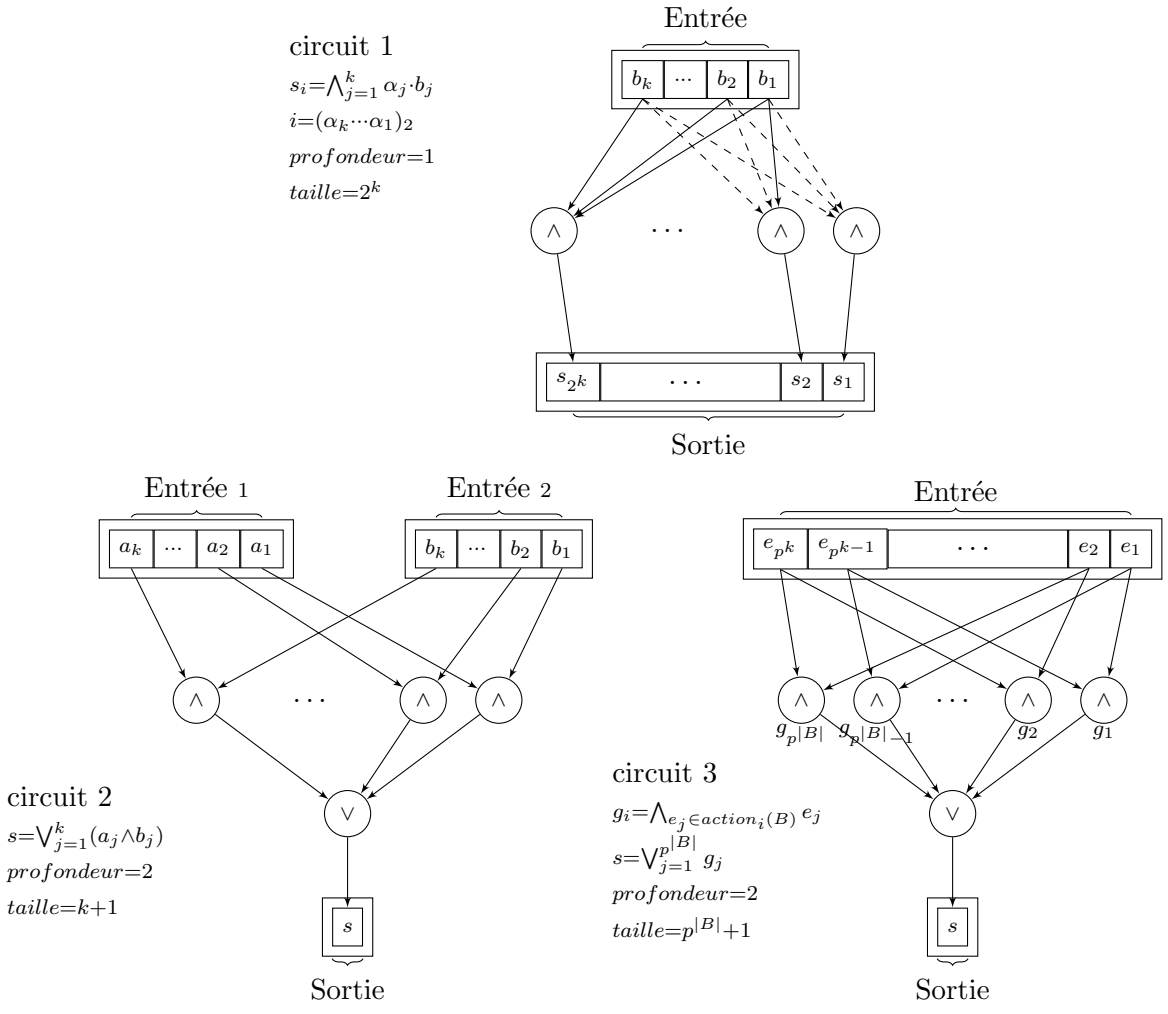


FIG. 2.1 – Les circuits utilisés dans la construction du circuit AC^0

2.1.2 Model-checking d'ATL sur les CGS symboliques.

Au dessus, on a montré que le model-checking de ATL pour les CGS explicites est polynomial. On s'intéresse maintenant à la complexité du model-checking quand il s'agit des CGS

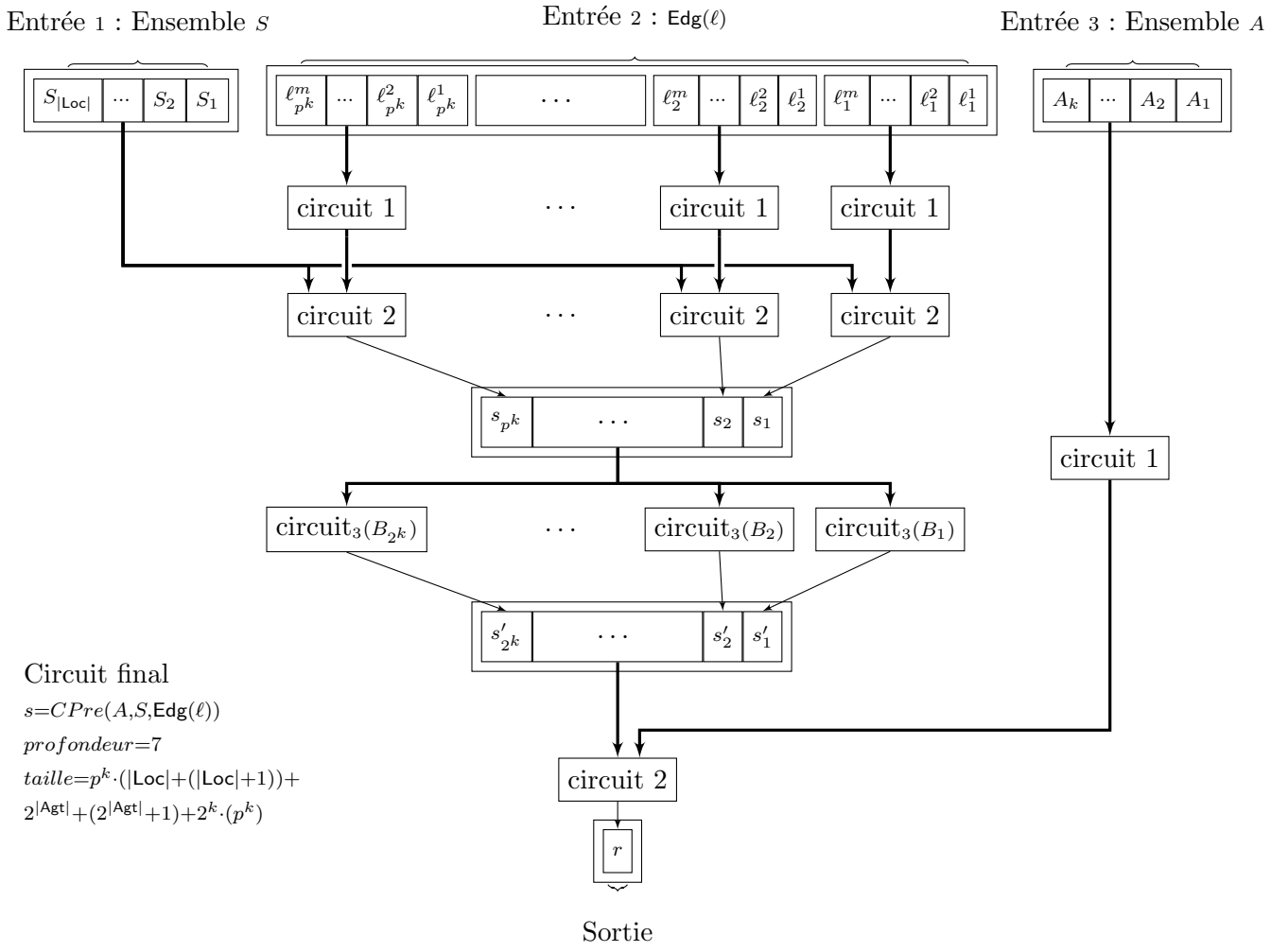


FIG. 2.2 – Les circuits pour calculer le $CPre$

symboliques.

Supposons que les transitions issues de ℓ sont données —dans la table de transition— par la séquence $((\varphi_0,\ell_0),(\varphi_1,\ell_1),\ldots,(\varphi_n,\ell_n))$, on a : $\ell \in CPre(A,S)$ ssi $\exists m_A \in \mathsf{Mov}(\ell,A)$, t.q. il n'existe pas un $m_{\bar{A}} \in \mathsf{Mov}(\ell,\mathsf{Agt}\setminus A)$ et $\ell_i \in \mathsf{Loc}\setminus S$ t.q. $\varphi_i[m_A \oplus m_{\bar{A}}] \equiv \top$ et $\varphi_j[m_A \oplus m_{\bar{A}}] \equiv \bot$ pour tout $j<i$. On cherche donc une action $m_A \in \mathsf{Mov}(\ell,A)$ t.q. pour tout $m_{\bar{A}} \in \mathsf{Mov}(\ell,\bar{A})$, la négation de $\bigvee_{\ell_i \in \mathsf{Loc}\setminus S}(\varphi_i[m_A] \wedge \bigwedge_{j<i} \neg\varphi_j[m_A])$ reste valide.

Ce problème correspond à une instance du problème EQSAT$_2$ qui est un problème Σ_2^{P}-complet :

EQSAT$_2$:

Entrée: deux familles de variables $X = \{x^1, \ldots, x^n\}$ et $Y = \{y^1, \ldots, y^n\}$, une formule booléenne φ sur l'ensemble des variables $X \cup Y$.

Sortie: Vrai ssi $\exists X.\ \forall Y.\ \varphi$.

En effet, on obtient comme corollaire direct de [JD05, Lemma 1] :

Proposition 2.2
Calculer CPre dans les CGS symboliques est Σ_2^{P}-complet.

Démonstration: L'appartenance dans Σ_2^{P} suit directement la remarque ci-dessus. Une procédure Σ_2^{P} est donnée explicitement par l'algorithme 1.

Procedure `co-strategy`$(q, (\varphi_i, \ell_i)_i, (m_a)_{a\in A}, S)$
`//vérifier si l'adversaire dispose d'une co-stratégie pour que` $(m_a)_{a\in A}$ `s'échappent à` S
begin
  foreach $\bar{a} \in \bar{A}$ **do**
    $m_{\bar{a}} \leftarrow guess(q, \bar{a})$;
  $i \leftarrow 0$;
  while $\neg\varphi_i(m_a, m_{\bar{a}})$ **do**
    $i \leftarrow i + 1$;
  if $\ell_i \notin S$ **then**
    return *yes*;
  else
    return *no*;
end
Procedure `CPre`(A, S) **begin**
  $W \leftarrow \varnothing$;
  foreach $q \in \mathcal{C}$ **do**
    foreach $a \in A$ **do**
      $m_a \leftarrow guess(q, a)$;
    if *not* `co-strategy`$(q, (\varphi_i, \ell_i)_i, (m_a)_{a\in A}, S)$ **then**
      $W \leftarrow W \cup \{q\}$;
  return W;
end

Algorithme 1 : Le problème de décision de *CPre* sur les CGS symboliques.

Pour montrer la dureté dans Σ_2^{P}, on utilise directement la construction de [JD05, Lemma 1] : À partir d'une instance $\exists X.\ \forall Y.\ \varphi$ de EQSAT$_2$, on construit une CGS symbolique avec 3 états q_1, $q_\top$ et $q_\bot$, et $2 \cdot n$ agents : A^1, ..., A^n, B^1, ..., B^n, chacun a deux choix possibles dans q_1 et un seul choix dans $q_\top$ et $q_\bot$. Les transitions issues de $q_\top$ et $q_\bot$ sont des boucles. Les transitions issues de q_1 sont données par : $\delta(q_1) = ((\varphi[x^j \leftarrow (A^j \stackrel{?}{=} 1), y^j \leftarrow (B^j \stackrel{?}{=} 1)], q_\top), (\top, q_\bot))$.

Il s'ensuit que, q_1 appartient à $CPre(\{A^1, \ldots, A^n\}, \{q_\top\})$ ssi il existe une affectation pour les variables dans X t.q. φ est vraie quel que soit le choix des agents B dans Y.

□

Maintenant on peut chercher la complexité du model-checking. On montre que la complexité du model-checking d'ATL sur les CGS symboliques est supérieure à celle sur les CGS explicites : La preuve de dureté dans Σ_2^{P} de $\mathit{CPre}(A, S)$ pourrait être adaptée facilement pour montrer la dureté dans Π_2^{P}. En effet, si on considère le problème dual (c.à.d le problème Π_2^{P}-complet) AQSAT$_2$ $\forall X.\ \exists Y.\ \varphi$, et la même CGS symbolique, la formule $\neg \langle\!\langle A^1, \ldots, A^n \rangle\!\rangle \mathbf{X} \neg q_\top$ est vérifiée si et seulement si l'instance de AQSAT$_2$ est négative. Cela exprime le fait que les joueurs dans A^1 à A^n n'ont pas de stratégie pour échapper à $q_\top$: quelque soit leur choix, les joueurs B^1 à B^n pourront forcer φ.

Ceci contredit le résultat dans [JD05] qui annonce que le model-checking d'ATL est Σ_2^{P}-complet. En fait, la manière dont leur algorithme traite la négation est erronée : les jeux sur les CGS (et les ATS) sont généralement non déterminés, et donc si un joueur n'a pas de stratégie pour forcer φ, cela n'implique pas que les autres joueurs ont une stratégie pour forcer $\neg\varphi$. En réalité, ceci veut dire que les autres joueurs possèdent une *co-stratégie* pour forcer $\neg\varphi$ (on désigne par *co-stratégie* la manière dont ils réagissent contre les actions de leurs adversaires [GvD06]).

En utilisant l'expression des modalités d'ATL sous la forme des formules de points fixes (voir Eq. (2.1)), on peut calculer l'ensemble des états qui satisfont une formule d'ATL en temps polynomial par rapport au nombre d'appel de *CPre*, qui mène à un algorithme dans Δ_3^{P} :

Proposition 2.3
Le model-checking d'ATL sur des CGS symboliques est dans Δ_3^{P}.

Pour montrer la dureté dans Δ_3^{P}, on introduit le problème suivant qui est Δ_3^{P}-complet [LMS01, Sch04].

SNSAT$_2$:

Entrée: m familles de variables $X_i = \{x_i^1, ..., x_i^n\}$, m familles de variables $Y_i = \{y_i^1, ..., y_i^n\}$, m variables z_i, m formules booléennes φ_i, les variables qui apparaissent dans φ_i sont dans $X_i \cup Y_i \cup \{z_1, ..., z_{i-1}\}$.

Sortie: la valeur de z_m, est définie par

$$\begin{cases} z_1 \stackrel{\text{def}}{=} \exists X_1.\ \forall Y_1.\ \varphi_1(X_1, Y_1) \\ z_2 \stackrel{\text{def}}{=} \exists X_2.\ \forall Y_2.\ \varphi_2(z_1, X_2, Y_2) \\ \quad\ldots \\ z_m \stackrel{\text{def}}{=} \exists X_m.\ \forall Y_m.\ \varphi_m(z_1, ..., z_{m-1}, X_m, Y_m) \end{cases}$$

Alors, on obtient :

Proposition 2.4
Le model-checking d'ATL sur les CGS symboliques est un problème Δ_3^{P}*-dur.*

Démonstration: On choisit une instance $\mathcal{I}$ de SNSAT$_2$, et on la réduit à une instance du problème de model-checking de ATL. On note qu'une telle instance définit d'une manière unique les valeurs des variables z_i. On désigne par $v_\mathcal{I} \colon \{z_1, ..., z_m\} \to \{\top, \bot\}$ cette valuation. De plus, si $v_\mathcal{I}(z_i) = \top$, il existe une valuation valide pour les variables dans X_i. On étend $v_\mathcal{I}$ à $\{z_1, ..., z_m\} \cup \bigcup_i X_i$, avec $v_\mathcal{I}(x_i^j)$ une valuation valide si $v_\mathcal{I}(z_i) = \top$.

On définit une CGS symbolique $\mathcal{C}$ comme suit : elle contient $m \cdot n$ agents A_i^j (un pour chaque x_i^j), $m \cdot n$ agents B_i^j (un pour chaque y_i^j), m agents C_i (un pour chaque z_i), et un agent supplémentaire D. La structure est construite de m états q_i, m états $\overline{q_i}$, m états s_i, et deux états $q_\top$ et $q_\bot$. On a trois propositions atomiques : $s_\top$ et $s_\bot$, (qui étiquettent les états $q_\top$ et $q_\bot$ resp.), et une proposition atomique s (qui étiquette les états s_i). Les autres états ne sont pas étiquetés.

À l'exception de D, les agents représentent des variables booléennes, ils ont donc deux choix possibles (0 et 1). L'agent D possède m choix (0 à $m-1$). La relation de transition est définie par : pour chaque i,

$$\begin{array}{l}\delta(\overline{q_i}) = ((\top, s_i));\\ \delta(s_i) = ((\top, q_i));\\ \delta(q_\top) = ((\top, q_\top));\\ \delta(q_\bot) = ((\top, q_\bot));\end{array} \qquad \delta(q_i) = \left(\begin{array}{l} ((D \stackrel{?}{=} 0) \wedge \varphi_i[x_i^j \leftarrow (A_i^j \stackrel{?}{=} 1),\\ \qquad y_i^j \leftarrow (B_i^j \stackrel{?}{=} 1), z_k \leftarrow (C_k \stackrel{?}{=} 1)], q_\top)\\ ((D \stackrel{?}{=} 0), q_\bot)\\ ((D \stackrel{?}{=} k) \wedge (C_k \stackrel{?}{=} 1), q_k) \text{ pour tout } k < i\\ ((D \stackrel{?}{=} k) \wedge (C_k \stackrel{?}{=} 0), \overline{q_k}) \text{ pour tout } k < i\\ (\top, q_\top)\end{array}\right)$$

Intuitivement, à partir de l'état q_i, les agents booléens choisissent une valuation pour la variable qu'ils représentent, et l'agent D peut, soit choisir une valuation pour vérifier si elle valide φ_i (en choisissant l'action 0), soit "défier" le joueur C_k, avec l'action $k < i$.

La formule ATL construite récursivement est la suivante :

$$\begin{aligned}\psi_0 &\stackrel{\text{def}}{=} \top\\ \psi_{k+1} &\stackrel{\text{def}}{=} \langle\!\langle \mathsf{AC} \rangle\!\rangle (\neg s)\, \mathbf{U}\, (q_\top \vee \mathbf{EX}\, (s \wedge \mathbf{EX}\, \neg\psi_k))\end{aligned}$$

on note par AC la coalition $\{A_1^1, ..., A_m^n, C_1, ..., C_m\}$.

Soit $f_\mathcal{I}(A)$ une stratégie sans mémoire pour l'agent $A \in \mathsf{AC}$ qui consiste à jouer selon la valuation $v_\mathcal{I}$ (*i.e.* l'action 0 si la variable associée avec A est évaluée à 0 dans $v_\mathcal{I}$, et l'action 1 sinon). On complète la preuve de la Proposition 2.4 par le lemme suivant :

Lemme 2.1
Pour tout $i \leq m$ et pour tout $k \geq i$, les trois propositions suivantes sont équivalentes :

1. $\mathcal{C}, q_i \models \psi_k$;
2. *Les stratégies $f_\mathcal{I}$ sont valides pour que* $\mathcal{C}, q_i \models \psi_k$;
3. *la variable z_i est évaluée à $\top$ dans $v_\mathcal{I}$.*

Démonstration: Clairement, *(2)* implique *(1)*. On montre par induction sur i que *(1)* implique *(3)* et que *(3)* implique *(2)*.

Tout d'abord, on suppose que $q_1 \models \psi_j$, pour un $j \geq 1$. Puisqu'un seul $q_\top$ et $q_\bot$ est accessible depuis q_1, on a $q_1 \models \langle\!\langle \mathsf{AC} \rangle\!\rangle \mathbf{X}\, q_\top$. Dans ce cas, on a (presque) la même réduction $\mathbf{\Sigma}_2^\mathsf{P}$ que celle décrite dans [JD05] : il existe une valuation des variables x_1^1 à x_1^n t.q., quel que soit le choix des joueurs D et B_1^1 à B_m^n, l'exécution se termine dans $q_\top$. Ceci est vrai en particulier si le joueur D choisit de jouer l'action 0 : pour toute valuation des variables y_1^1 à y_1^n, $\psi_1(X_1, Y_1)$ reste vraie, et z_1 est évaluée à vrai dans $v_\mathcal{I}$.

Deuxièmement, si z_1 est évaluée à vrai, alors $v_{\mathcal{I}}(x_1^1),\ldots, v_{\mathcal{I}}(x_1^n)$ sont t.q., quelle que soit la valeur de y_1^1 à y_1^n, ψ_1 reste vraie. Si les joueurs A_1^1 à A_1^n jouent selon la stratégie $f_{\mathcal{I}}$, alors les joueurs D et B_1^1 à B_1^n ne pourront pas échapper à l'état $q_\top$, et $q_1 \models \langle\langle \mathsf{AC} \rangle\rangle \mathbf{X}\, q_\top$, et donc le même raisonnement persiste pour ψ_k quand $k \geq 1$.

On continue à montrer maintenant le cas inductif : si le résultat est vrai pour un indice $i \geq 1$, alors c'est aussi vrai pour $i+1$. Supposons que $q_{i+1} \models \psi_{k+1}$, avec $k \geq i$. Il existe donc une stratégie qui garantit ψ_{k+1}, *i.e.*, tous les outcomes suivant cette stratégie satisfont $(\neg s)\,\mathbf{U}\,(q_\top \vee \mathbf{EX}\,(s \wedge \mathbf{EX}\,\neg\psi_k))$. En fonction du choix du joueur D dans l'état q_{i+1}, on obtient les informations suivantes :

Premièrement, si le joueur D choisit de jouer l'action l, avec $1 \leq l \leq i$, le jeu continue vers l'état q_l ou $\overline{q_l}$, selon le choix du joueur C_l.

- Si le joueur C_l joue l'action 0, l'exécution se termine dans $\overline{q_l}$. Puisque la seule façon pour sortir de l'état est d'entrer dans l'état s_l, étiqueté par s, on obtient $\overline{q_l} \models \mathbf{EX}\,(s \wedge \mathbf{EX}\,\neg\psi_k)$, *i.e.*, donc $q_l \models \neg\psi_k$. Par hypothèse d'induction, on obtient que z_l est évaluée à faux pour notre instance de SNSAT_2.
- Si le joueur C_l choisit de jouer 1, l'exécution passe à q_l. Dans cet état, les joueurs dans AC pourraient continuer de jouer en appliquant leur stratégie, qui assure $q_l \models \psi_{k+1}$, et donc, par hypothèse d'induction, z_l est évaluée à vrai dans $\mathcal{I}$.

Alors, la stratégie, pour AC pour forcer ψ_{k+1} dans q_{i+1}, nécessite que les joueurs C_1 à C_i jouent suivant $v_{\mathcal{I}}$ et la validité de leur choix pourrait être assurée par l'"adversaire" D.

Maintenant, si le joueur D joue l'action 0, tous les outcomes possibles vont nécessairement mener immédiatement à $q_\top$ (puisque ψ_{k+1} est valide, et puisque $q_\bot \not\models \mathbf{EX}\,(s \wedge \mathbf{EX}\,\neg\psi_k)$). On obtient immédiatement que les joueurs B_{i+1}^1 à B_{i+1}^n ne pourront pas rendre ψ_{i+1} faux, donc z_{i+1} est évaluée à vrai dans $\mathcal{I}$.

Deuxièmement, si z_{i+1} est évaluée à vrai, on suppose que les joueurs dans AC jouent suivant $f_{\mathcal{I}}$, et on considère les choix possibles du joueur D :

- Si le joueur D choisit l'action 0, puisque z_{i+1} est évaluée à vrai et puisque les joueurs C_1 à C_i et A_{i+1}^1 à A_{i+1}^n ont joué suivant $v_{\mathcal{I}}$, les joueurs B_{i+1}^1 à B_{i+1}^n ne peuvent pas échapper l'état $q_\top$.
- Si le joueur D choisit une action l comprise entre 1 et i, l'exécution va arriver à l'état q_l ou l'état $\overline{q_l}$, selon le choix du joueur C_l.
  - Si C_l joue l'action 0, *i.e.*, si z_l est évaluée à faux dans $v_{\mathcal{I}}$, l'exécution arrive à l'état $\overline{q_l}$, et par hypothèse d'induction on a $q_l \models \neg\psi_k$. donc $\overline{q_l} \models \mathbf{EX}\,(s \wedge \mathbf{EX}\,\neg\psi_k)$, et la stratégie remplit sa tâche.
  - Si C_l joue 1, *i.e.*, si z_l est évaluée à vrai, alors l'exécution arrive dans l'état q_l, dans lequel et par hypothèse d'induction, la stratégie $f_{\mathcal{I}}$ peut forcer ψ_{k+1}.
- Si le joueur D joue l'action l avec $l > i$, l'exécution arrivera directement à $q_\top$, et la formule sera satisfaite.

□ □

Avec la Proposition 2.3, on obtient :

Théorème 2.2

Le model-checking d'ATL sur les CGS symboliques est Δ_3^{P}-complet.

2.1.3 Le model-checking d'ATL sur les ATS.

2.1.3.1 Les systèmes de transitions alternants

On présente maintenant des structures de jeux multi-agents légèrement différentes que les CGS. Dans ces types de structures, au lieu d'avoir une table de transitions qui renvoie le successeur après le choix du coup de chaque agent du système, le coup de chaque joueur dans un état donné est l'ensemble des transitions qu'il souhaite prendre dans l'étape suivante.

Dans les travaux originaux sur ATL [AHK98], la logique était interprétée sur les ATS, qu'on définit formellement par :

Définition 2.1
Un système de transitions alternant *(Alternating Transition System ou* ATS*)* $\mathcal{A}$ *est un* 6*-uplet* $\langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathsf{Agt}, \mathsf{Mov} \rangle$ *avec :*

- $\langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta \rangle$ *un système de transitions ;*
- Agt *est un ensemble d'agents (comme pour les CGS) ;*
- $\mathsf{Mov}\colon \mathsf{Loc} \times \mathsf{Agt} \to \mathcal{P}(\mathcal{P}(\delta))$ *associe pour chaque état* ℓ *et pour chaque agent* A_i *un ensemble d'actions possibles, chaque action étant un sous-ensemble de* δ*. Dans chaque état* ℓ*, on demande que, pour tout* $\delta_i \in \mathsf{Mov}(\ell, A_i)$, $\bigcap_{i \leq k} \delta_i$ *soit un singleton. Cette fonction doit aussi vérifier que l'état de départ des transitions données soit l'état* ℓ*, c.à.d. pour tout état* ℓ*, pour chaque* $\delta_i \in \mathsf{Mov}(\ell, A_i)$ *si* $(q, \ell') = \delta_i$ *alors* $q = \ell$*.*

L'idée est la suivante : dans un état ℓ, tous les agents ayant choisi leurs actions (*i.e.*, un sous-ensemble de transitions), l'exécution procède vers l'*unique* transition qui appartient à tous les ensembles choisis par les agents. De nouveau, on désigne par $\mathsf{Next}(\ell)$ (resp. $\mathsf{Next}(\ell, A_j, m)$) l'ensemble de toutes les transitions possibles qui mènent vers les états successeurs (resp. l'ensemble des transitions possibles quand le joueur A_j choisit l'action m).

La taille d'un ATS est $|\mathsf{Loc}| + |\mathsf{Mov}|$ où $|\mathsf{Mov}|$ est la somme du nombre transition dans chaque action possible pour tout agent et dans tout état.

Exemple 2.1 ([AHK98])
Considérons l'exemple du passage à niveau, on s'intéresse à représenter ce modèle par un ATS. Notre structure comprend alors 2 *joueurs : Le train (A) et le contrôleur (B). L'ensemble d'états est* $\mathsf{Loc} = \{\ell_1, \ell_2, \ell_3, \ell_4\}$*.*

- *Les propositions atomiques appliquées aux états sont :*
  - $\mathsf{Lab}(\ell_1) = \{\texttt{out_of_gate}\}$ *: le train n'est pas entré.*
  - $\mathsf{Lab}(\ell_2) = \{\texttt{out_of_gate}, \texttt{request}\}$ *: le train n'est pas entré mais il a demandé la permission au contrôleur.*
  - $\mathsf{Lab}(\ell_3) = \{\texttt{out_of_gate}, \texttt{grant}\}$ *: le contrôleur a donné au train la permission d'entrer.*
  - $\mathsf{Lab}(\ell_4) = \{\texttt{in_gate}\}$ *: le train est entré dans le passage à niveau.*
- *La table de transition est donnée par :*

- $\mathit{Mov}(\ell_1, A) = \{\{(\ell_1, \ell_1)\}, \{(\ell_1, \ell_2)\}\}$
  $\mathit{Mov}(\ell_1, B) = \{\{(\ell_1, \ell_1), (\ell_1, \ell_2), (\ell_1, \ell_3), (\ell_1, \ell_4)\}\}$
- $\mathit{Mov}(\ell_2, A) = \{\{(\ell_2, \ell_1), (\ell_2, \ell_2), (\ell_2, \ell_3), (\ell_2, \ell_4)\}\}$
  $\mathit{Mov}(\ell_2, B) = \{\{(\ell_2, \ell_1)\}, \{(\ell_2, \ell_2)\}, \{(\ell_2, \ell_3)\}\}$
- $\mathit{Mov}(\ell_3, A) = \{\{(\ell_3, \ell_1)\}, \{(\ell_3, \ell_4)\}\}$
  $\mathit{Mov}(\ell_3, B) = \{\{(\ell_3, \ell_1), (\ell_3, \ell_2), (\ell_3, \ell_3), (\ell_3, \ell_4)\}\}$
- $\mathit{Mov}(\ell_4, A) = \{\{(\ell_4, \ell_1), (\ell_4, \ell_2), (\ell_4, \ell_3), (\ell_4, \ell_4)\}\}$
  $\mathit{Mov}(\ell_4, B) = \{\{(\ell_4, \ell_1)\}, \{(\ell_4, \ell_4)\}\}$

On peut vérifier que l'ATS de cet exemple est un jeu turn-based puisque chaque état est contrôlé par un seul joueur.

Remarque 2.1
Comme pour les CGS, les ATS ne sont pas déterminés non plus. La figure 2.3 représente un ATS non-déterminé "équivalent" à la CGS de l'exemple 1.3.

$$\mathsf{Loc} = \{\ell_0, \ell_1, \ell_2, \ell'_1, \ell'_2\}$$

$$\mathsf{Mov}(\ell_0, A_1) = \{\{\ell_1, \ell'_1\}, \{\ell_2, \ell'_2\}\}$$
$$\mathsf{Mov}(\ell_0, A_2) = \{\{\ell_1, \ell'_2\}, \{\ell_2, \ell'_1\}\}$$

$$\text{avec } \begin{cases} \mathsf{Lab}(\ell_1) = \mathsf{Lab}(\ell_2) = \{p\} \\ \mathsf{Lab}(\ell'_1) = \mathsf{Lab}(\ell'_2) = \varnothing \end{cases}$$

FIG. 2.3 – Un ATS non déterminé.

On montre dans la section 2.2.1 que les CGS et les ATS ont la même expressivité (pour la bisimulation alternante [AHKV98] que nous définissons plus tard).

2.1.3.2 Le model-checking d'ATL sur les ATS

Dans le cas des ATS, le calcul de *CPre* (et donc le model-checking d'ATL) ne peut pas non plus se faire dans PTIME. Un corollaire direct de [JD05, Lemma 4] est le suivant :

Proposition 2.5
Calculer CPre dans les ATS est NP*-complet.*

Démonstration: L'algorithme 2 nous montre comment calculer *CPre* dans NP pour les ATS : il consiste à deviner le choix pour chaque joueur dans la coalition, et vérifier ensuite si les états obtenus sont dans S.

De même, le côté NP-dur suit de [JD05, Lemme 4]. On propose ici une preuve légèrement différente qui va être une première étape pour le caractère Δ_2^{P}-dur du model-checking d'ATL sur les ATS.

La preuve est une réduction directe de 3-SAT : Soit $\mathcal{I} = (S^1, ..., S^n)$ une instance de 3-SAT sur les variables $X = \{x^1, ..., x^m\}$. On suppose que $S^j = \alpha^{j,1} s^{j,1} \vee \alpha^{j,2} s^{j,2} \vee \alpha^{j,3} s^{j,3}$ avec $s^{j,k} \in X$

```
Procedure CPre(A, S) begin
    W ← ∅;
    foreach q ∈ C do
        foreach a ∈ A do
            m_a ← guess(q, a);
        if ⋂_{a∈A} m_a ⊆ S then
            W ← W ∪ {q};
    return W;
end
```

Algorithme 2 : Problème de décision du calcul de *CPre* dans un ATS

et $\alpha^{j,k} \in \{0,1\}$ pour indiquer si la variable $s^{j,k}$ est prise négativement (0) ou positivement (1). Pour simplifier, on suppose qu'aucune clause ne contient une proposition et sa négation.

Avec cette instance, on associe l'ATS $\mathcal{A}$ suivant. Une illustration de cette construction est montré dans l'exemple 2.2. Il contient $8n+1$ états : un état q, et, pour chaque clause S^j, huit états $q^{j,0}$ à $q^{j,7}$. Intuitivement, l'état $q^{j,k}$ correspond à la clause $B^{j,k} = k_1 s^{j,1} \vee k_2 s^{j,2} \vee k_3 s^{j,3}$, où $k_1 k_2 k_3$ correspond à la notation binaire de k. On n'utilise qu'une seule proposition atomique α dans notre ATS : un état $q^{j,k}$ est étiqueté par α ssi il ne correspond pas à la clause S^j. Par construction, pour chaque j, un seul état $q^{j,0}$ à $q^{j,7}$ n'est pas étiqueté par α.

On a $m+1$ joueurs, où m est le nombre de variables qui apparaissent dans $\mathcal{I}$. On associe à chaque x^i un joueur A^i. On nomme le joueur supplémentaire D. Seules les transitions issues de q sont utiles pour cette réduction. On peut donc supposer que les autres états contiennent des boucles. Dans q, le joueur A^i décide la valeur de x^i. Il peut alors choisir entre deux ensembles de transitions suivants, ce sont donc les états qui correspondent aux clauses qui ne sont validées pas par son choix :

$$\{q^{j,k} \mid \forall l \leq 3.\ s^{j,l} \neq x^i \text{ ou } \alpha^{i,l} = 0\} \qquad \text{si } x^i = \top$$
$$\{q^{j,k} \mid \forall l \leq 3.\ s^{j,l} \neq x^i \text{ ou } \alpha^{i,l} = 1\} \qquad \text{si } x^i = \bot$$

Finalement, le joueur D possède n choix qui sont $\{q^{1,0}, ..., q^{1,7}\}$ à $\{q^{n,0}, ..., q^{n,7}\}$.

Tout d'abord, montrons que l'intersection des différents choix des agents forme un singleton. On associe une valuation au choix pris par les différents joueurs $A^1 \dots A^m$. Pour cette valuation, et pour chaque $j \leq n$, exactement une clause entre $B^{j,0}$ et $B^{j,7}$ est évaluée à faux (puisqu'on a supposé qu'un littéral n'apparaît pas avec sa négation dans la même clause). Donc, l'intersection de ces choix avec le choix du joueur D donne un unique état.

Maintenant, soit $\varphi = \langle\langle A^1, \dots, A^m \rangle\rangle \mathbf{X}\, \alpha$. Si l'état $q \models \varphi$ alors les joueurs A^1 à A^m peuvent choisir une valuation pour x^1 à x^m pour que le joueur D n'ait pas la possibilité de trouver une clause de l'instance originale (*i.e.*, qui n'est pas étiquetée avec α) qui est évaluée à faux(*i.e.*, qui ne peut pas être vraie pour aucun choix pris par les joueurs A^1 à A^m). Dans ce cas, l'instance 3-CNF est satisfaisable.

D'un autre côté, si l'instance est satisfaisable, il suffit donc que les joueurs A^1 à A^m jouent en accord avec la valuation des variables x^1à x^m qui rend l'instance satisfaisable. Puisque cette valuation rend toutes les clauses originales vraies, on obtient une stratégie qui aboutit seulement à des états qui sont étiquetés avec α. □

Exemple 2.2
Supposons qu'on souhaite modéliser une instance de 3-SAT en utilisant les ATS déjà définies. Prenons l'instance $\mathcal{I} = \left(x^1 \vee \neg x^2 \vee \neg x^3\right) \wedge \left(x^1 \vee x^2 \vee \neg x^4\right) \wedge \left(\neg x^1 \vee x^3 \vee x^4\right)$. *On reprend la construction précédente pour obtenir l'ATS suivant :*

- $\mathsf{Agt} = \left\{A^1, A^2, A^3, A^4, D\right\}$,
- $\mathsf{Loc} = \left\{q_0, q^{1,0}, \ldots, q^{1,7}, q^{2,0}, \ldots, q^{2,7}, q^{3,0}, \ldots, q^{3,7}\right\}$,
- *La table de transition est donnée par :*

$$
\begin{array}{rcl}
\mathsf{Mov}(q_0, A^1) & = & \left\{\left\{q^{1,0}, q^{1,2}, q^{1,4}, q^{1,6}, q^{2,0}, q^{2,2}, q^{2,4}, q^{2,6}, q^{3,1}, q^{3,3}, q^{3,5}, q^{3,7}\right\},\right. \\
& & \left.\left\{q^{1,1}, q^{1,3}, q^{1,5}, q^{1,7}, q^{2,1}, q^{2,3}, q^{2,5}, q^{2,7}, q^{3,0}, q^{3,2}, q^{3,4}, q^{3,6}\right\}\right\} \\
\mathsf{Mov}(q_0, A^2) & = & \left\{\left\{q^{1,2}, q^{1,3}, q^{1,6}, q^{1,7}, q^{2,0}, q^{2,1}, q^{2,4}, q^{2,5}, q^{3,0}, \ldots, q^{3,7}\right\},\right. \\
& & \left.\left\{q^{1,0}, q^{1,1}, q^{1,4}, q^{1,5}, q^{2,2}, q^{2,3}, q^{2,6}, q^{2,7}, q^{3,0}, \ldots, q^{3,7}\right\}\right\} \\
\mathsf{Mov}(q_0, A^3) & = & \left\{\left\{q^{1,4}, q^{1,5}, q^{1,6}, q^{1,7}, q^{2,0}, \ldots, q^{2,7}, q^{3,0}, q^{3,1}, q^{3,4}, q^{3,5}\right\},\right. \\
& & \left.\left\{q^{1,0}, q^{1,1}, q^{1,2}, q^{1,3}, q^{2,1}, \ldots, q^{2,7}, q^{3,2}, q^{3,3}, q^{3,6}, q^{3,7}\right\}\right\} \\
\mathsf{Mov}(q_0, A^4) & = & \left\{\left\{q^{1,0}, \ldots, q^{1,7}, q^{2,4}, q^{2,5}, q^{2,6}, q^{2,7}, q^{3,0}, q^{3,1}, q^{3,2}, q^{3,3}\right\},\right. \\
& & \left.\left\{q^{1,0}, \ldots, q^{1,7}, q^{2,0}, q^{2,1}, q^{2,2}, q^{2,3}, q^{3,4}, q^{3,5}, q^{3,6}, q^{3,7}\right\}\right\} \\
\mathsf{Mov}(q_0, D) & = & \left\{\left\{q^{1,0}, \ldots, q^{1,7}\right\}, \left\{q^{2,0}, \ldots, q^{2,7}\right\}, \left\{q^{3,0}, \ldots, q^{3,7}\right\}\right\} \\
\mathsf{Mov}(q, A) & = & \{\{q\}\} \qquad \forall A \in \mathsf{Agt} \text{ and } \forall q \in \mathsf{Loc} \setminus \{q_0\}
\end{array}
$$

  On peut vérifier que pour tout choix possible de chacun des joueurs, l'intersection résultante est toujours un singleton.
- *on donne* $\mathsf{Lab}(q) = \alpha$ *ssi* $q \notin \{q^{1,1}, q^{2,3}, q^{3,6}\}$

Le choix de chaque joueur $A_1, \ldots, A_4$ *est lié à la valeur de la variable booléenne* $x_1, \ldots, x_4$ *resp. (e.g. si* $x_1 = \top$*, le joueur* A_1 *prend le premier ensemble parmi son ensemble de choix.)*
La figure 2.4 illustre le cas où les valeurs booléennes choisies sont $(x^1, x^2, x^3, x^4) = (\top, \top, \top, \top)$. *Graphiquement chaque joueur remplit un quart de cercle en gris (le choix de* A^1 *est rempli par* ◣,A^2 *par* ◢,A^3 *par* ◥ *et* A^4 *par* ◤*). Par ex.* ◓ *signifie que l'état représenté par ce dessin, est pris par les choix des joueurs 1 et 3. Ainsi, les noeuds qui correspondent au choix des 4 joueurs* A^1, A^2, A^3 *et* A^4 *seront des cercles remplis entièrement en gris. Notre formule est donc satisfaisable si la coalition des joueurs* $A^1, \ldots, A^4$ *a une stratégie pour que l'état suivant soit toujours un état étiqueté par* α*. ce qui est le cas de cet exemple.*

Comme dans le cas des CGS symboliques, on utilise les expressions sous la forme de points fixes des modalités de ATL et ainsi avec l'algorithme NP de calcul des *CPre*, on obtient un algorithme Δ_2^{P} pour ATL :

Proposition 2.6
*Le model-checking d'*ATL *sur les ATS est dans* Δ_2^{P}.

On prouve que cette solution est optimale :

Proposition 2.7
*Le model-checking d'*ATL *sur les ATS est* Δ_2^{P}*-dur.*

Démonstration: La preuve est faite par réduction du problème SNSAT

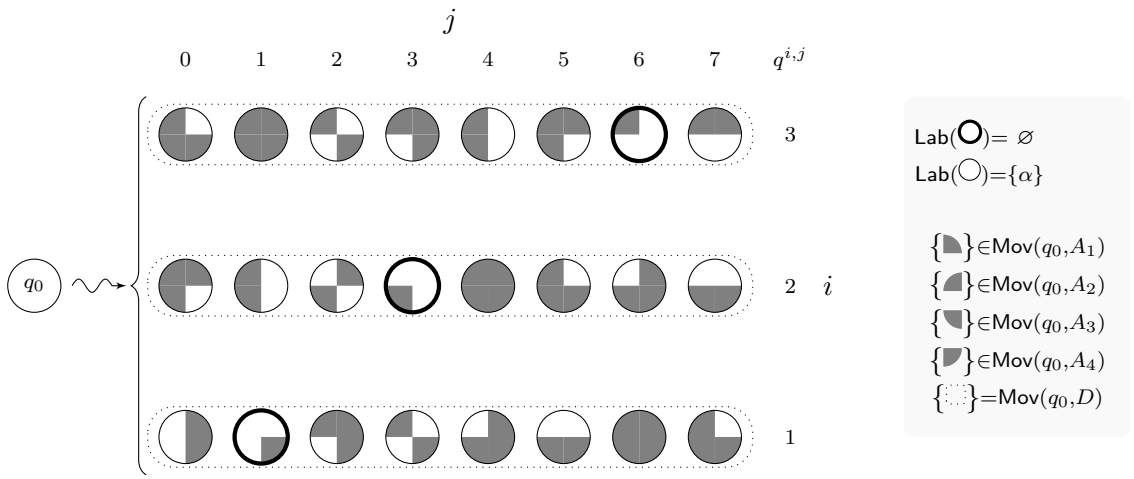


FIG. 2.4 – L'ATS de l'exemple 2.2

SNSAT:

Entrée: p familles des variables $X_r = \{x_r^1, ..., x_r^m\}$, p variables z_r, p formules booléennes φ_r dans 3-CNF, avec φ_r ayant comme variables qui interviennent $X_r \cup \{z_1, ..., z_{r-1}\}$.

Sortie: La valeur de z_p, est définie par

$$\begin{cases} z_1 \stackrel{\text{def}}{=} \exists X_1.\ \varphi_1(X_1) \\ z_2 \stackrel{\text{def}}{=} \exists X_2.\ \varphi_2(z_1, X_2) \\ z_3 \stackrel{\text{def}}{=} \exists X_3.\ \varphi_3(z_1, , z_2, X_3) \\ \quad\ \ldots \\ z_p \stackrel{\text{def}}{=} \exists X_p.\ \varphi_p(z_1, ..., z_{p-1}, X_p) \end{cases}$$

Soit $\mathcal{I}$ une instance de SNSAT, on suppose que chaque φ_r est construite de n clauses S_r^1 à S_r^n, avec $S_r^j = \alpha_r^{j,1} s_r^{j,1} \vee \alpha_r^{j,2} s_r^{j,2} \vee \alpha_r^{j,3} s_r^{j,3}$. Une telle instance définit une unique valuation $v_{\mathcal{I}}$ pour les variables z_1 à z_r, qui peut être étendue sur l'ensemble de toutes les variables en choisissant une valuation positive pour x_r^1 à x_r^n quand z_r sont évaluées à vrai.

On décrit maintenant l'ATS $\mathcal{A}$: il contient $(8n + 3)p$ états :

- p états $\overline{q_r}$ et p états q_r,
- p états s_r,
- et pour toute formule φ_r, pour chaque clause S_r^j de φ_r, huit états $q_r^{j,0}, \ldots, q_r^{j,7}$, comme pour la réduction précédente.

Les états s_r sont étiquetés avec une proposition atomique s, et les états $q_r^{j,k}$ qui ne correspondent pas à la clause S_r^j sont étiquetés par α. On associe un joueur A_r^j pour chaque variable x_r^j, un joueur C_r pour chaque z_r, et on ajoute ainsi un joueur supplémentaire D. Dans la table de transitions, on a une boucle sur chaque état $q_r^{j,k}$, une transition de chaque $\overline{q_r}$ vers l'état correspondant s_r, et depuis chaque s_r vers le q_r correspondant. À partir l'état q_r :

- le joueur A_r^j va choisir une valeur de la variable x_r^j en prenant un ensemble parmi les deux

ensembles de transitions suivants :

$$\{q_r^{g,k} \mid \forall l \leq 3.\ s_r^{g,l} \neq x_r^j \text{ ou } \alpha_r^{g,l} = 0\} \cup \{q_t, \overline{q_t} \mid t < r\} \qquad \text{si } x_r^j = \top$$
$$\{q_r^{g,k} \mid \forall l \leq 3.\ s_r^{g,l} \neq x_r^j \text{ ou } \alpha_r^{g,l} = 1\} \cup \{q_t, \overline{q_t} \mid t < r\} \qquad \text{si } x_r^j = \bot$$

Ces deux choix permettent d'accéder à un des deux états q_t ou $\overline{q_t}$. Dans q_r, les joueurs A_t^j avec $t \neq r$ ont un seul choix qui est l'ensemble de tous les états.

– Le joueur C_t va choisir la valeur de la variable qu'il représente. Pour les joueurs A_r^j, ce choix va être exprimé en choisissant entre deux ensembles d'états correspondant à la clause qui n'a pas été évaluée à vrai. Mais, comme pour la preuve de la Proposition 2.4, les joueurs C_t vont avoir la possibilité de "vérifier" leur choix, en allant soit à l'état q_t ou bien à l'état $\overline{q_t}$. Formellement, ceci donne deux ensembles :

$$\{q_r^{g,k} \mid \forall l \leq 3.\ s_r^{g,l} \neq z_t \text{ ou } \alpha_r^{g,l} = 0\} \cup \{q_u, \overline{q_u} \mid u \neq t\} \cup \{q_t\} \qquad \text{si } z_t = \top$$
$$\{q_r^{g,k} \mid \forall l \leq 3.\ s_r^{g,l} \neq z_t \text{ ou } \alpha_r^{g,l} = 1\} \cup \{q_u, \overline{q_u} \mid u \neq t\} \cup \{\overline{q_t}\} \qquad \text{si } z_t = \bot$$

– Finalement, le joueur D choisit soit de défier un joueur C_t, avec $t < r$, en prenant comme action l'ensemble $\{q_t, \overline{q_t}\}$, soit de vérifier qu'une clause S_r^j est satisfaite, en prenant comme action l'ensemble $\{q_r^{j,0}, \ldots, q_r^{j,7}\}$.

On montre tout d'abord que de n'importe quel choix de tous les joueurs fournit un seul état successeur. À l'exception des états q_r, la preuve est évidente. Pour un état q_r, au début on se restreint aux choix des joueurs A_r^j à C_r, alors :

– Si on considère les états $q_r^{1,0}$ à $q_r^{n,7}$, le même argument que la preuve précédente assure qu'exactement un état par clause est choisi,
– Si on considère les états q_t et $\overline{q_t}$, les choix des joueurs B_t assurent qu'exactement un état a été choisi dans chaque paire $\{q_t, \overline{q_t}\}$, pour tout $t < r$.

Clairement, le choix du joueur D va donner exactement un état parmi les états restants.

Maintenant, on construit la formule ATL. C'est une formule définie par induction (similaire à celle utilisée dans la preuve de la Proposition 2.4), qui est définie par $\psi_0 = \top$ et (on note par AC pour l'ensemble des joueurs $\{A_1^1, \ldots, A_p^m, C_1, \ldots, C_p\}$) :

$$\psi_{r+1} \stackrel{\text{def}}{=} \langle\langle \mathsf{AC} \rangle\rangle (\neg s)\, \mathbf{U}\, (\alpha \vee \mathbf{EX}\, (s \wedge \mathbf{EX}\, \neg \psi_r)).$$

En notant $f_{\mathcal{I}}$ la stratégie sans-mémoire associée à $v_{\mathcal{I}}$, on obtient donc :

Lemme 2.2
Pour tout $r \leq p$ et $t \geq r$, les propositions suivantes sont équivalentes :

1. *$q_r \models \psi_t$;*
2. *la stratégie $f_{\mathcal{I}}$ garantit que $q_r \models \psi_t$;*
3. *la variable z_r est évaluée à vrai dans $v_{\mathcal{I}}$.*

Démonstration: On montre par induction sur r que *(1)* implique *(3)* et que *(3)* implique *(2)*, puisque *(2)* implique *(1)* étant évidente. Pour $r = 1$, puisqu'il n'y a aucun état s qui est accessible, on remonte à la preuve précédente sur la dureté dans NP.

Supposons que le résultat est vrai jusqu'à l'indice r. Alors, si $q_{r+1} \models \psi_{t+1}$ pour un $t \geq r$, on choisit une stratégie pour la coalition AC qui valide cette propriété. On considère encore les différents choix possibles pour le joueur D :

– si le joueur D fait le choix d'aller vers q_u ou $\overline{q_u}$, avec $u < r+1$: l'exécution se termine dans q_u si le joueur C_u choisit de mettre z_u à vrai. Mais dans ce cas, la formule ψ_{t+1} reste valide dans q_u, qui donne par hypothèse d'induction que z_u est évaluée à vrai dans $v_{\mathcal{I}}$. Réciproquement, l'exécution se termine dans $\overline{q_u}$ si le joueur C_u met z_u à faux. Dans ce cas, on obtient que $q_u \models \neg\psi_t$, avec $t \geq u$, qui nécessite par hypothèse d'induction que z_u est évaluée à faux.
Ce premier cas nécessite que le joueur C_1 à C_r choisit la valeur correcte des variables z_1 à z_r.
– si le joueur D choisit un ensemble de huit états qui correspond à une clause S^j_{r+1}, donc la stratégie des autres joueurs assure que l'exécution va arriver à un état étiqueté par α. Comme dans la réduction précédente, ceci indique que la clause correspondante s'est mise à vrai par les choix des autres joueurs.

Tout cela montre que la variable z_{r+1} est évaluée à vrai.

Maintenant, si la variable z_{r+1} est évaluée à vrai, on pose que les joueurs dans AC jouent en accord à la valuation $f_{\mathcal{I}}$. Alors

– si le joueur D choisit d'aller vers un ensemble d'états qui correspond à une clause de φ_{r+1}, il va nécessairement arriver dans un état qui est étiqueté par α, puisque la clause est vraie par la valuation choisie.
– si le joueur D choisit d'aller vers q_u ou $\overline{q_u}$, pour un u quelconque, alors il va défier le joueur B_u pour montrer que son choix était correct. Par hypothèse d'induction, et puisque le joueur B_u a jouer selon $f_{\mathcal{I}}$, la formule $(\neg s)\,\mathbf{U}\,(\alpha \vee \mathbf{EX}\,(s \wedge \mathbf{EX}\,\neg\psi_{t+1}))$ va être satisfaite, pour $t \geq u$.

□ □

On arrive donc à avoir la complexité exacte du model-checking des formules de ATL sur les ATS :

Théorème 2.3

Le model-checking de ATL sur les ATS est Δ_2^{P}-complet.

2.1.4 Au delà d'ATL

2.1.4.1 Le model-checking d'ATL$^+$

On commence par noter que ATL^+ est une extension de ATL qui permet d'exprimer des propriétés avec des formules plus succinctes [Wil99, AI01] mais ces deux logiques ont le même pouvoir d'expressivité : toute formule d'ATL^+ peut être traduite en une formule ATL équivalente [HRS02].

Il a été montré que la complexité du model-checking d'ATL^+ sur les ATSs est Δ_3^{P}-complet dans [Sch04]. Mais la preuve de la difficulté dans Δ_3^{P} de [Sch04] est basée sur une réduction en espace logarithmique seulement par rapport à la taille DAG de la formule. On montre ci-dessous que le model-checking d'ATL^+ est Δ_3^{P}-complet (en prenant la définition classique de la taille d'une formule) pour les 3 différents types de structure de jeu.

Proposition 2.8
Le model-checking d'ATL$^+$ sur les CGS symboliques est dans Δ_3^{P}.

Démonstration: L'algorithme Δ_3^{P} utilisé dans [Sch04] est utilisé sur les CGS explicites. Dans la suite on va étendre cet algorithme pour pouvoir traiter les CGS symboliques : Pour chaque sous-formule de la forme $\langle\langle A\rangle\rangle\,\varphi$, on devine la stratégie (sans-mémoire) pour les joueurs dans A. Dans chaque état, les choix de chaque joueur dans A peuvent être remplacés dans la fonction de transition. On calcule ensuite l'ensemble des états où la formule CTL^+ $\mathbf{A}\varphi$ est satisfaite. On peut faire cela en temps Δ_2^{P} [CES86, LMS01], mais il est nécessaire de calculer, au début, les transitions possibles dans la structure restante, *i.e.*, vérifier laquelle des formules associées à la transition est satisfaite. On peut accomplir cela en temps polynomial indépendamment des appels à l'oracle NP, et donc on n'augmente pas la complexité de notre algorithme. □

Proposition 2.9
Le model-checking d'ATL^+ *sur les CGS explicites* turn-based *à deux joueurs est* Δ_3^{P}*-difficile.*

Démonstration: La réduction suivante est une extension directe de celle présentée dans [LMS01] dans le cas de CTL^+. En particulier, la différence avec les réductions précédentes vient du fait que les formules booléennes sont maintenant codées par des formules ATL^+, et non pas dans le modèle.

On code une instance $\mathcal{I}$ de SNSAT_2, tout en conservant les notations utilisées dans les preuves des Propositions 2.4 (pour le problème SNSAT_2) et 2.7 (pour énumérer les clauses). La figure. 2.5 montre la CGS turn-based $\mathcal{C}$ à deux joueurs associés à $\mathcal{I}$. Les états s_1 à s_m sont

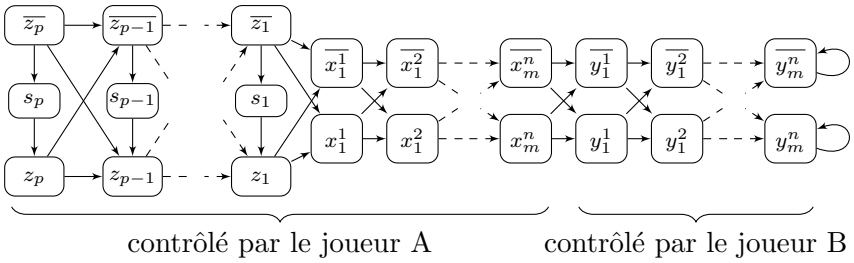


FIG. 2.5 – Le CGS $\mathcal{C}$

étiquetés par la proposition atomique s, les états $\overline{z_1}$ à $\overline{z_m}$ sont étiquetés par la proposition atomique $\overline{z}$, et les autres états sont étiquetés par leur nom comme le montre la figure 2.5.

La formule ATL^+ est construite récursivement avec $\psi_0 = \top$ et

$$\psi_{k+1} = \langle\langle A\rangle\rangle\,[\mathbf{G}\,\neg s \wedge \mathbf{G}\,(\overline{z} \rightarrow \mathbf{EX}\,(s \wedge \mathbf{EX}\,\neg\psi_k)) \wedge \bigwedge_{w\leq p} [(\mathbf{F}\,z_w) \rightarrow \bigwedge_{j\leq n} \bigvee_{k\leq 3} \mathbf{F}\,l_w^{j,k}]]$$

où $l_w^{j,k} = v$ quand $s_w^{j,k} = v$ et $\alpha_w^{j,k} = 1$, et $l_w^{j,k} = \overline{v}$ quand $s_w^{j,k} = v$ et $\alpha_w^{j,k} = 0$. On obtient donc :

Lemme 2.3
Pour tout $r \leq p$ *et* $t \geq r$*, les propositions suivantes sont équivalentes :*
- $z_r \models \psi_t$ *;*
- *les stratégies* $f_{\mathcal{I}}$ *valident* $q_r \models \psi_t$ *;*
- *la variable* z_r *sont évaluées à vrai dans* $v_{\mathcal{I}}$*.*

Quand $r = 1$, les états s ou $\overline{z}$ ne sont pas accessibles depuis z_1, avoir $z_1 \models \psi_t$, avec $t \geq 1$, est équivalent à $z_1 \models \langle\langle A \rangle\rangle \bigwedge_j \bigvee_k \mathbf{F}\, l_1^{j,k}$. Qui est équivalent à son tour à avoir z_1 évaluée à vrai dans $\mathcal{I}$.

On montre maintenant le cas inductif. Si $z_{r+1} \models \psi_{t+1}$ avec $t \geq r$, on considère la stratégie pour A t.q. tous les outcomes satisfont cette propriété, et on choisit un outcome qu'on nomme ρ. Puisqu'il ne peut passer dans aucun état s, il définit une valuation v_ρ des variables z_1 à z_{r+1} et x_1^1 à x_m^n de manière évidente. Chaque fois qu'un outcome passe par un état $\overline{z_u}$, il satisfait $\mathbf{EX}\,(s \wedge \mathbf{EX}\,\psi_t)$. Chaque fois que l'outcome passe par un état z_u, le suffixe de l'outcome satisfait la formule ψ_{t+1} dans z_u. Grâce à l'hypothèse d'induction, ces deux cas conduisent à avoir $v_\rho(z_u) = v_\mathcal{I}(z_u)$ pour tout $u < r+1$. Maintenant, quand $w = r+1$, la sous-formule $\bigwedge_w \left[(\mathbf{F}\, z_w) \rightarrow \bigwedge_{j \leq n} \bigvee_{k \leq 3} \mathbf{F}\, l_w^{j,k}\right]$ conduit à avoir que φ_{r+1} est satisfaite pour n'importe quelle valeur des y_{r+1}^j, *i.e.*, z_{r+1} est évaluée à vrai dans $\mathcal{I}$.

Réciproquement, si z_r est évaluée à vrai, la stratégie $f_\mathcal{I}$ nous mène clairement à avoir $z_r \models \psi_t$. □

On obtient donc le corollaire immédiat suivant :

Théorème 2.4

Le model-checking d'ATL$^+$ est Δ_3^{P}-complet sur les ATSs, les CGS explicites et les CGS symboliques.

2.1.4.2 Le model-checking de EATL

Comme pour la logique de temps arborescent CTL [Eme90], ajouter la modalité $\overset{\infty}{\mathbf{F}}$ augmente de l'expressivité de ATL : EATL est plus expressif que ATL (voir la section 2.2.2.2).

Mais la complexité théorique du model-checking ne change pas en passant de ATL vers EATL :

Théorème 2.5

Le model-checking d'EATL est :
- PTIME-complet sur les CGS explicites ;
- Δ_2^{P}-complet sur les ATSs ;
- Δ_3^{P}-complet sur les CGS symboliques.

Démonstration: On étend l'algorithme du model-checking d'ATL. Ceci se fait en exprimant les modalités $\langle\langle A \rangle\rangle \overset{\infty}{\mathbf{F}}$ et $\langle\langle A \rangle\rangle \overset{\infty}{\mathbf{G}}$ comme des formules de points fixes [dAHM01] :

$$\langle\langle A \rangle\rangle \overset{\infty}{\mathbf{F}}\, p \equiv \nu y.\mu x.\,(\langle\langle A \rangle\rangle \mathbf{X}\,(x) \vee (p \wedge \langle\langle A \rangle\rangle \mathbf{X}\,(y)))$$
$$\langle\langle A \rangle\rangle \overset{\infty}{\mathbf{G}}\, p \equiv \mu y.\nu x.\,(\langle\langle A \rangle\rangle \mathbf{X}\,(x) \wedge (p \vee \langle\langle A \rangle\rangle \mathbf{X}\,(y)))$$

Le calcul de ces points fixes nécessite un nombre polynomial d'exécutions de *CPre*.
Et la dureté découle directement de la dureté du model-checking d'ATL. □

2.1.4.3 Le model-checking d'ATL* .

La complexité du model-checking de la logique CTL* est connue d'être EXPTIME-complète. Cette explosion dans la complexité est due à la traduction d'une formule LTL en un automate de Büchi.

Dans le cas de la logique ATL* , le model-checking d'une formule sur les CGS explicites a été montŕé 2EXPTIME-complet :

Théorème 2.6 [AHK02]

Le model-checking d'une formule ATL* sur les CGS explicites est 2EXPTIME-complet.

Démonstration: Puisque vérifier une formule ATL* de la forme $\langle\langle A\rangle\rangle\,\varphi$ (où φ est une formule LTL) correspond à montrer l'existence d'une stratégie gagnante pour la coalition A, le model-checking de la logique ATL* sur les CGS explicites nécessite donc l'utilisation des automates d'arbres. On construit un automate d'arbres reconnaissant les arbres dans lesquels tous les chemins satisfont la propriété φ et un automate qui accepte les arbres correspondants aux stratégies possibles de la coalition A.

Prenons une CGS $\mathcal{S} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg}\rangle$ et une formule ATL* φ. On suppose qu'on a étiqueté chaque état ℓ de $\mathcal{S}$ par toutes les sous-formules de φ qui y sont satisfaites. Pour les formules qui ne sont pas de la forme $\varphi' = \langle\langle A\rangle\rangle\,\varphi$, avec φ une formule LTL, on utilise l'algorithme de model-checking de CTL*. Sinon, on construit un automate d'arbres de Rabin $\mathcal{A}_\varphi$ qui reconnaît exactement les arbres qui satisfont la formule $\mathbf{A}\varphi$. Pour tout état ℓ dans $\mathcal{S}$, on construit l'automate d'arbres de Büchi $\mathcal{A}_{\mathcal{S},\ell,A}$ qui reconnaît les arbres d'exécutions pour des stratégies de la coalition A. On calcule ensuite le produit des automates d'arbres $\mathcal{A}_\varphi$ et $\mathcal{A}_{\mathcal{S},\ell,A}$, on obtient donc un automate de Rabin qui reconnaît les exécutions des stratégies des coalitions de A qui satisfont la formule $\mathbf{A}\varphi$.

Puisqu'un état $\ell \models \langle\langle A\rangle\rangle\,\varphi$ si et seulement s'il existe une stratégie pour les joueurs dans A pour satisfaire φ. On en déduit que $\ell \models \langle\langle A\rangle\rangle\,\varphi$ ssi l'automate produit est non vide.

On précise que la taille de l'automate $\mathcal{A}_\varphi$ est de $2^{2^{\mathcal{O}(|\varphi|)}}$, et il comprend $2^{\mathcal{O}(|\varphi|)}$ paires de Rabin. On s'intéresse maintenant à définir l'automate $\mathcal{A}_{\mathcal{S},\ell,A}$. On définit tout d'abord $Post(A,\ell') \subseteq 2^{\mathsf{Loc}}$ qui représente l'ensemble des ensembles d'états ρ tel que les joueurs dans A se coopèrent pour garantir que le successeur de ℓ' est un membre de ρ, en d'autres termes $Post(A,\ell') = \{\mathsf{Next}(\ell', A, \lambda_A) \mid \lambda_A \in \mathsf{Strat}_{\mathcal{S}}(A)\}$. L'automate d'arbres $\mathcal{A}_{\mathcal{S},\ell,A}$ correspond à l'alphabet d'entrée 2^{AP}, un ensemble d'états Loc, l'état initial ℓ, la fonction de transition non-déterministe η telle que $\eta(\ell', \mathsf{Lab}(\ell')) = Post(A,\ell')$, et $\eta(\ell', p) = \varnothing$ si $p \neq \mathsf{Lab}(\ell')$. De cette manière chaque ensemble dans $Post(A,\ell')$ détermine l'ensemble des successeurs de l'entrée $\mathsf{Lab}(\ell')$. Finalement, on note que l'ensemble des états acceptants dans cet automate de Büchi est l'ensemble Loc de tous les états. Donc l'automate d'arbres obtenu garde la même taille de la structure $\mathcal{S}$ de départ.
Ensuite, résoudre le problème de vide avec un automate d'arbres de Rabin de taille n avec r paires de Rabin nécessite un temps $\mathcal{O}(n \cdot r)^{3r}$. Ainsi, étiqueter un état avec une formule φ' nécessite un temps $(|\mathcal{S}| \cdot 2^{2^{\mathcal{O}(|\varphi|)}})^{2^{\mathcal{O}(|\varphi|)}} = |\mathcal{S}|^{2^{\mathcal{O}(|\varphi|)}}$. Ce qui donne un algorithme dans 2EXPTIME.

Pour montrer la dureté, on présente une réduction directe du problème de décision des jeux LTL qui est 2EXPTIME-complet [ATM03]. □

La complexité du model-checking des formules d'$\mathsf{ATL}^\star$ reste la même sur les différentes structures CGS explicite, CGS symbolique et ATS, puisque cela provient notamment de la formule à vérifier :

Corollaire 2.1
Le model-checking de $\mathsf{ATL}^\star$ est 2EXPTIME*-complet sur les ATS, les CGS explicites et les CGS symboliques.*

Démonstration: On étend l'algorithme de [AHK02] pour les CGS explicites. On s'intéresse notamment à la construction de $\mathcal{A}_{\mathcal{S},\ell,A}$.

Les états de $\mathcal{A}_{\mathcal{S},\ell,A}$ sont des états de $\mathcal{S}$. Depuis l'état ℓ, on a un nombre de transitions égal au nombre d'actions possibles $m = (m_{A_i})_{A_i \in A}$ pour la coalition A. Chaque transition est un ensemble d'états qui doivent apparaître dans le prochain niveau de l'arbre. Formellement, étant donné $p \in 2^{\mathsf{AP}}$,

$$\delta(\ell, p) = \{\mathsf{Next}(\ell, A, m) \mid m = (m_{A_i})_{A_i \in A} \text{ avec } \forall A_i \in A.\ m_{A_i} \in \mathsf{Mov}(\ell, A_i)\}$$

quand $p = \mathsf{Lab}(\ell)$, et $\delta(\ell, p) = \varnothing$ sinon.

Pour les CGS explicites, la fonction de transition est calculée facilement en temps polynomial. Pour les ATS et les CGS symboliques, la fonction de transition est calculée en énumérant l'ensemble (de taille exponentielle) des actions de la coalition A (calculer $\mathsf{Next}(\ell, A, m)$ se fait en temps polynomial si on fixe le choix de la coalition).

On peut calculer $\mathcal{A}_{\mathcal{S},\ell,A}$ en temps exponentiel, et le test de vide du produit d'automates nécessite un temps double-exponentiel. Ce qui rend l'algorithme complet dans 2EXPTIME. La borne inférieure est une conséquence de la borne inférieure des CGS explicites.

□

On note finalement que nos résultats pourraient être adaptés facilement au μ-calcul alternant (AMC) [AHK02] : L'algorithme PTIME proposé dans [AHK02] pour les CGS explicites, demande encore un nombre polynomial de calculs de *CPre*, on peut l'adapter facilement pour les ATS et les CGS symboliques : on obtient donc que le model-checking du fragment sans alternation a la même complexité que le model-checking de ATL, et le model-checking du μ-calcul alternant est dans EXPTIME pour les trois types de modèles.

2.2 Expressivité

Jusqu'à maintenant, nous avons étudié comment la quantification sur les stratégies possibles des agents peut augmenter la complexité du model-checking.

On s'intéresse maintenant aux problèmes de l'expressivité. On étudie tout d'abord la traduction entre les différents modèles (CGS explicite, CGS symbolique et ATS). Ensuite, on regarde l'expressivité des modalités "Until" and "Always", en montrant qu'ils ne peuvent pas exprimer le dual de "Until".

2.2.1 Comparaison entre l'expressivité des CGS et celle des ATS

On montre dans cette section que les CGS et les ATS sont équivalents : ils peuvent modéliser les *mêmes* jeux concurrents. Pour formaliser cette notion, on introduit la définition suivante :

Définition 2.2 ([AHKV98])
Soient $\mathcal{A}$ et $\mathcal{B}$ deux modèles de concurrents (des ATS ou des CGS) sur un même ensemble d'agents Agt. Soit $R \subseteq \mathsf{Loc}_{\mathcal{A}} \times \mathsf{Loc}_{\mathcal{B}}$ une relation (non-vide) entre les états de $\mathcal{A}$ et les états de $\mathcal{B}$. On appelle cette relation une bisimulation alternante *quand, pour tout $(\ell, \ell') \in R$, les conditions suivantes sont vraies :*

- $\mathsf{Lab}_{\mathcal{A}}(\ell) = \mathsf{Lab}_{\mathcal{B}}(\ell')$;
- *pour toute coalition $A \subseteq \mathsf{Agt}$, on a*

$$\forall m \colon A \to \mathsf{Mov}_{\mathcal{A}}(\ell, A).\ \exists m' \colon A \to \mathsf{Mov}_{\mathcal{B}}(\ell', A).$$
$$\forall q' \in \mathsf{Next}(\ell', A, m').\ \exists q \in \mathsf{Next}(\ell, A, m).\ (q, q') \in R.$$

- *symétriquement, pour toute coalition $A \subseteq \mathsf{Agt}$, on a*

$$\forall m' \colon A \to \mathsf{Mov}_{\mathcal{B}}(\ell', A).\ \exists m \colon A \to \mathsf{Mov}_{\mathcal{A}}(\ell, A).$$
$$\forall q \in \mathsf{Next}(\ell, A, m).\ \exists q' \in \mathsf{Next}(\ell', A, m').\ (q, q') \in R.$$

où $\mathsf{Next}(\ell, A, m)$ est l'ensemble d'états qui sont accessibles depuis ℓ quand chaque joueur $A_i \in A$ joue $m(A_i)$.

Deux modèles sont dits bisimilaires alternants s'il existe une bisimulation alternante entre tous leurs états.

En utilisant cette définition comme critère d'équivalence, on peut déduire que les CGS (symbolique et explicite) et les ATS possèdent le même pouvoir expressif.

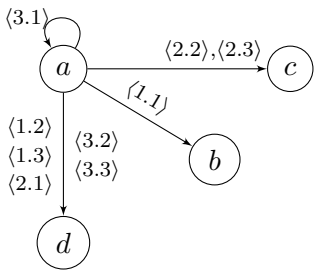
Théorème 2.7

1. Étant donnée une CGS explicite, construire une CGS symbolique équivalente se fait en temps linéaire ;
2. Étant donnée une CGS symbolique, construire une CGS explicite équivalente se fait en temps exponentiel ;
3. Étant donnée une CGS explicite, construire une ATS équivalente se fait en temps polynomial (cubique) ;
4. Étant donnée une ATS, construire une CGS explicite équivalente se fait en temps exponentiel ;
5. Étant donnée une CGS symbolique, construire une ATS équivalente se fait en temps exponentiel ;
6. Étant donnée une ATS, construire une CGS symbolique équivalente se fait en temps polynomial (quadratique) ;

On note que les traductions sont optimales : les traductions exponentielles ne pourront pas être achevées en un temps polynomial à cause des résultats de complexité du model-checking de ATL.

Démonstration: On note que les points 4 et 3 ont été étudiés dans [GJ04].

En effet, pour montrer le point 1, on donne un algorithme polynomial qui transforme la table de transitions d'une CGS explicite en une table de transitions d'une CGS symbolique.



Les choix à partir l'état a :

Joueur 1	
choix 1 :	$\{b_{a,1,1}, d_{a,1,2}, d_{a,1,3}\}$
choix 2 :	$\{c_{a,2,2}, c_{a,2,3}, d_{a,2,1}\}$
choix 3 :	$\{a_{a,3,1}, d_{a,3,2}, d_{a,3,3}\}$

Joueur 2	
choix 1 :	$\{a_{a,3,1}, b_{a,1,1}, d_{a,2,1}\}$
choix 2 :	$\{c_{a,2,2}, d_{a,1,2}, d_{a,3,2}\}$
choix 3 :	$\{c_{a,2,3}, d_{a,1,3}, d_{a,3,3}\}$

FIG. 2.6 – Conversion d'une CGS explicite vers un ATS

Soit $\mathcal{A} = \langle \mathsf{Loc}_\mathcal{A}, \mathsf{AP}, \mathsf{Lab}_\mathcal{A}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}_\mathcal{A}, \mathsf{Edg}_\mathcal{A} \rangle$ une CGS explicite, on cherche à construire la CGS symbolique $\mathcal{B} = \langle \mathsf{Loc}_\mathcal{A}, \mathsf{AP}, \mathsf{Lab}_\mathcal{A}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}_\mathcal{A}, \mathsf{Edg}_\mathcal{B} \rangle$ correspondante. Tous les autres éléments étant les mêmes, on cherche à déterminer la table de transitions de $\mathsf{Edg}_\mathcal{B}$. L'algorithme linéaire est le suivant : Pour chaque état ℓ, étant donnée la table de transitions explicite, on construit la table symbolique $\mathsf{Edg}_\mathcal{B}(\ell) = ((\varphi_m, \mathsf{Edg}_\mathcal{A}(\ell, m)))_{m \in \mathbb{M}^{|\mathsf{Agt}|}}$, avec $m = \langle m_1, \ldots, m_{|\mathsf{Agt}|} \rangle$ un vecteur de choix, et $\varphi_m = \bigwedge_{1 \leq i \leq |\mathsf{Agt}|} \left(A_i \stackrel{?}{=} m_i \right)$. L'ordre des apparitions des formules n'est pas important dans la traduction puisqu'on obtient une formule pour chaque choix possible des agents, et donc une formule φ_i est vérifiée pour un choix unique d'actions des différents joueurs.

Le point 2 consiste à montrer un algorithme exponentiel. L'idée est d'énumérer tous les choix possibles qui correspondent aux formules booléennes. Ci-dessous, on décrit l'algorithme. On suppose qu'on souhaite transformer la CGS symbolique $\mathcal{B} = \langle \mathsf{Loc}_\mathcal{B}, \mathsf{AP}, \mathsf{Lab}_\mathcal{B}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}_\mathcal{B}, \mathsf{Edg}_\mathcal{B} \rangle$ en une CGS explicite $\mathcal{A} = \langle \mathsf{Loc}_\mathcal{B}, \mathsf{AP}, \mathsf{Lab}_\mathcal{B}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}_\mathcal{B}, \mathsf{Edg}_\mathcal{A} \rangle$. Pour chaque état ℓ, on construit la table de transitions explicites $\mathsf{Edg}_\mathcal{A}$ tel que pour chaque $m \in \mathbb{M}^{|\mathsf{Agt}|}$, on cherche la formule (φ_i, ℓ_i) avec le plus petit indice qui est vérifiée. Le résultat sera $\mathsf{Edg}_\mathcal{A}(\ell, m) = \ell_i$. Le temps nécessaire est exponentiel puisque la table obtenue contient tous les choix possibles (qui est exponentiel dans le nombre des agents).

On décrit maintenant le point 4. Pour pouvoir remarquer l'explosion dans la taille, il faut voir que la taille de la table de transitions d'un ATS est de l'ordre $\mathcal{O}\left(|\mathbb{M}| \times |\mathsf{Agt}| \times |\mathsf{Loc}|^2\right)$, alors que la taille de celle d'une CGS est de $\mathcal{O}\left(|\mathbb{M}|^{|\mathsf{Agt}|} \times |\mathsf{Loc}|\right)$
La démarche de l'algorithme est assez simple, on commence par le même ensemble d'états que celui de l'ATS, et construit la table de transitions de la manière suivante : Dans l'ATS $\mathcal{A}$, et à partir d'un état ℓ, on numérote chaque choix de chaque joueur i par un coup $m_i \in \mathbb{M} \subset \mathbb{N}$ $(0 < m \leq |\mathsf{Mov}_\mathcal{A}(\ell, A_i)|$, bien entendu, on a supposé que $\mathsf{Mov}_\mathcal{A}(\ell, A_i) = \{Q_{\ell,m_i} \mid 1 \leq m_i \leq |\mathsf{Mov}_\mathcal{A}(\ell, A_i)|\})$, et on note ce choix par Q_{ℓ,m_i}. On calcule ensuite l'élément $\mathsf{Mov}_\mathcal{C}\left(\ell, m_1, \ldots, m_{|\mathsf{Agt}|}\right) = \bigcap_{1 \leq i \leq |\mathsf{Agt}|} Q_{\ell,m_i}$.

Pour montrer le point 6, il suffit d'écrire pour chaque état possible, la conjonction (pour chaque agent) des disjonctions des choix qui contiennent les états suivants. Par exemple, si on a $\mathsf{Mov}_\mathcal{A}(\ell_0, A_1) = \{\{\ell_1, \ell_2\}, \{\ell_1, \ell_3\}\}$ et $\mathsf{Mov}_\mathcal{A}(\ell_0, A_2) = \{\{\ell_2, \ell_3\}, \{\ell_1\}\}$ dans l'ATS $\mathcal{A}$, alors

chaque joueur va avoir deux choix dans la CGS $\mathcal{B}$ associé, et

$$\mathsf{Edg}_{\mathcal{B}}(\ell_0) = \begin{pmatrix} ((A_1 = 1 \vee A_1 = 2) \wedge (A_2 = 2), & \ell_1)\,, \\ ((A_1 = 1) \wedge (A_2 = 1), & \ell_2)\,, \\ ((A_1 = 2) \wedge (A_2 = 1), & \ell_3) \end{pmatrix}$$

Formellement, soit $\mathcal{A} = \langle \mathsf{Loc}_{\mathcal{A}}, \mathsf{AP}, \mathsf{Lab}_{\mathcal{A}}, \delta, \mathsf{Agt}, \mathsf{Mov}_{\mathcal{A}} \rangle$ un ATS.
On définit alors $\mathcal{B} = \langle \mathsf{Loc}_{\mathcal{B}}, \mathsf{AP}, \mathsf{Lab}_{\mathcal{B}}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}_{\mathcal{B}}, \mathsf{Edg}_{\mathcal{B}} \rangle$ par :
- $\mathsf{Loc}_{\mathcal{B}} = \mathsf{Loc}_{\mathcal{A}}$, $\mathsf{Lab}_{\mathcal{B}} = \mathsf{Lab}_{\mathcal{A}}$;
- $\mathsf{Mov}_{\mathcal{B}} \colon \ell \times A_i \to [1, |\mathsf{Mov}_{\mathcal{A}}(\ell, A_i)|]$;
- $\mathsf{Edg}_{\mathcal{B}}$ une fonction qui donne pour chaque état ℓ une séquence $((\varphi_{\ell'}, \ell'))_{\ell' \in \mathsf{Loc}_{\mathcal{A}}}$ (l'ordre n'est pas important ici puisque les formules vont s'exclure mutuellement) avec

$$\varphi_{\ell'} = \bigwedge_{A_i \in \mathsf{Agt}} \left(\bigvee_{\substack{\ell' \text{ apparaît dans } j\text{-ème} \\ \text{ensemble de } \mathsf{Mov}_{\mathcal{A}}(\ell, A_i)}} A_i \stackrel{?}{=} j \right)$$

Le calcul de $\mathsf{Edg}_{\mathcal{B}}$ nécessite un temps quadratique ($\mathcal{O}(|\mathsf{Loc}_{\mathcal{A}}| \times |\mathsf{Mov}_{\mathcal{A}}|)$). On peut donc montrer que l'identité $\mathrm{Id} \subseteq \mathsf{Loc}_{\mathcal{A}} \times \mathsf{Loc}_{\mathcal{B}}$ est une bisimulation alternante puisqu'il existe une correspondance directe entre les choix des deux structures.

On explique maintenant la transformation des CGS explicites à des ATS, présentée dans le point 3. Soit $\mathcal{A} = \langle \mathsf{Loc}_{\mathcal{A}}, \mathsf{AP}, \mathsf{Lab}_{\mathcal{A}}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}_{\mathcal{A}}, \mathsf{Edg}_{\mathcal{A}} \rangle$ une CGS explicite. On définit l'ATS suivant $\mathcal{B} = \langle \mathsf{Loc}_{\mathcal{B}}, \mathsf{AP}, \mathsf{Lab}_{\mathcal{B}}, \delta, \mathsf{Agt}, \mathsf{Mov}_{\mathcal{B}} \rangle$ par (voir la Figure 2.6 pour avoir une idée de la construction) :
- $\mathsf{Loc}_{\mathcal{B}} \subseteq \mathsf{Loc}_{\mathcal{A}} \times \mathsf{Loc}_{\mathcal{A}} \times \mathbb{M}^k$, où $k = |\mathsf{Agt}|$, avec $(\ell, \ell', m_{A_1}, \ldots, m_{A_k}) \in \mathsf{Loc}_{\mathcal{B}}$ ssi $\ell = \mathsf{Edg}_{\mathcal{A}}(\ell', m_{A_1}, \ldots, m_{A_k})$;
- $\mathsf{Lab}_{\mathcal{B}}(\ell, \ell', m_{A_1}, \ldots, m_{A_k}) = \mathsf{Lab}_{\mathcal{A}}(\ell)$;
- Depuis un état $q = (\ell, \ell', m_{A_1}, \ldots, m_{A_k})$, le joueur A_j possède $|\mathsf{Mov}_{\mathcal{A}}(\ell, A_j)|$ actions possibles :

$$\mathsf{Mov}_{\mathcal{B}}(q, A_j) = \Big\{ \big\{ (\ell'', \ell, m'_{A_1}, \ldots, m'_{A_j} = i, \ldots, m'_{A_k}) \;\mid\; m'_{A_n} \in \mathsf{Mov}_{\mathcal{A}}(\ell, A_n) \\ \text{et } \ell'' = \mathsf{Edg}_{\mathcal{A}}(\ell, m_{A_1}, \ldots, m_{A_j} = i, \ldots, m_{A_k}) \big\} \;\mid\; i \in \mathsf{Mov}_{\mathcal{A}}(\ell, A_j) \Big\}$$

Cet ATS est construit en temps $\mathcal{O}(|\mathsf{Loc}_{\mathcal{A}}|^2 \cdot |\mathsf{Edg}_{\mathcal{A}}|)$. Il reste à montrer la bisimulation alternante entre ces deux structures. On définit alors la relation

$$R = \{ (\ell, (\ell, \ell', m_{A_1}, \ldots, m_{A_k})) \;\mid\; \ell \in \mathsf{Loc}_{\mathcal{A}}, (\ell, \ell', m_{A_1}, \ldots, m_{A_k})) \in \mathsf{Loc}_{\mathcal{B}} \}.$$

En utilisant les CGS explicites, on peut montrer immédiatement le point 5. Mais il est possible de suivre le même raisonnement dans le point 3 pour le prouver. □

La figure 2.7 résume ces résultats. On note que les coûts de ces traductions sont optimaux en utilisant les résultats sur la complexité (et en supposant que NP est différent de PTIME).

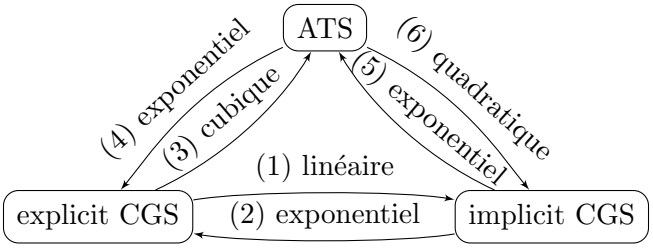


FIG. 2.7 – Les coûts de la traduction entre les trois modèles

2.2.2 Le pouvoir expressif d'ATL

2.2.2.1 $\langle\langle A\rangle\rangle$ R ne peut pas être exprimé avec $\langle\langle A\rangle\rangle$ U et $\langle\langle A\rangle\rangle$ G

Dans les papiers originaux où ATL [AHK97, AHK02] a été proposé, la syntaxe de la logique est présentée différemment de la définition utilisée ici : en effet la définition classique de ATL était la suivante :

$$\begin{array}{rcl} \mathsf{ATL}_{\text{orig}} \ni \varphi_s, \psi_s & ::= & \top \mid p \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \langle\langle A\rangle\rangle \varphi_p \\ \varphi_p & ::= & \mathbf{X}\,\varphi_s \mid \mathbf{G}\,\varphi_s \mid \varphi_s\,\mathbf{U}\,\psi_s. \end{array}$$

La différence qu'on peut noter entre cette définition et la définition 1.17 est l'absence de la négation dans la grammaire des formules de chemin, et la modalité *toujours* ($\mathbf{G}\,\varphi$) à sa place. Ceci rappelle la manière de définir les formules de chemin de CTL.

Or, la dualité est un concept fondamental dans les logiques modale et temporelle : par exemple, le dual de la modalité **U**, noté par **R** (*release*), est définie par $p\,\mathbf{R}\,q \stackrel{\text{def}}{\equiv} \neg((\neg p)\,\mathbf{U}\,(\neg q))$. Les modalités duales nous permettent par exemple de propager les négations à l'intérieur des formules, c'est une propriété importante pour manipuler des formules.

On peut exprimer la modalité **R** dans LTL en n'utilisant que **U** et **G** :

$$p\,\mathbf{R}\,q \equiv \mathbf{G}\,q \vee q\,\mathbf{U}\,(p \wedge q). \tag{2.2}$$

De la même façon, ceci s'exprime dans CTL avec les modalités **EX**, **EG** et **EU**, par :

$$\mathbf{E}p\,\mathbf{R}\,q \equiv \mathbf{EG}\,q \vee \mathbf{E}q\,\mathbf{U}\,(p \wedge q) \qquad\qquad \mathbf{A}p\,\mathbf{R}\,q \equiv \neg\,\mathbf{E}(\neg p)\,\mathbf{U}\,(\neg q).$$

On peut facilement remarquer que ces équivalences ne sont pas toujours vraies pour la logique ATL. En général, $\langle\langle A\rangle\rangle\, p\,\mathbf{R}\,q$ n'est pas équivalent à $\langle\langle A\rangle\rangle\,\mathbf{G}\,q \vee \langle\langle A\rangle\rangle\, q\,\mathbf{U}\,(p \wedge q)$: on peut avoir une partie des outcomes satisfaisant $\mathbf{G}\,q$ et d'autres parties satisfaisant $q\,\mathbf{U}\,(p \wedge q)$. En effet, on montre que $\mathsf{ATL}_{\text{orig}}$ est strictement moins expressive que ATL :

Théorème 2.8

Il n'existe pas de formule dans $\mathsf{ATL}_{\text{orig}}$ équivalente à $\Phi = \langle\langle A\rangle\rangle\,(a\,\mathbf{R}\,b)$.

La preuve du Théorème 2.8 est basée sur des techniques similaires à celles utilisées pour montrer les résultats d'expressivité pour des logiques temporelles comme CTL et ECTL [Eme90] :

on construit deux familles de modèles $(s_i)_{i\in\mathbb{N}}$ et $(s'_i)_{i\in\mathbb{N}}$ t.q. (1) $s_i \not\models \Phi$, (2) $s'_i \models \Phi$ pour tout i, et (3) s_i et s'_i satisfont la même formule $\mathsf{ATL}_{\text{orig}}$ de taille inférieure à i. Le Théorème 2.8 est une conséquence directe de l'existence de telles familles de modèles. Pour simplifier la présentation, on montre le théorème pour la formule $\Phi = \langle\langle A\rangle\rangle (b \,\mathbf{R}\, (a \vee b)) \equiv \langle\langle A\rangle\rangle a \,\mathbf{W}\, b$.

Ces modèles sont décrits par une CGS. Dans cette CGS (voir la figure 2.8), les états s_i et s'_i

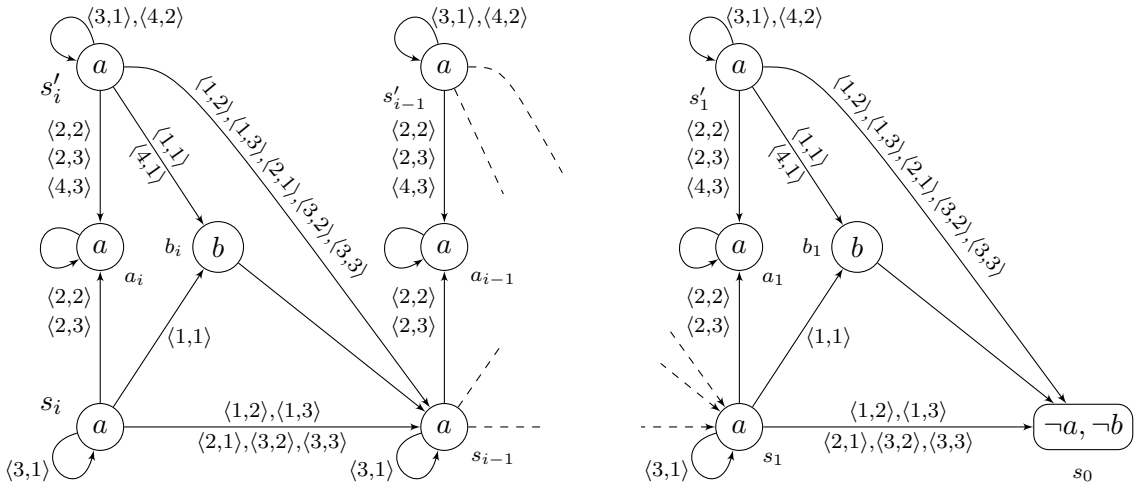


FIG. 2.8 – La CGS $\mathcal{C}$

diffèrent seulement par la quatrième action que le joueur A_1 possède dans s'_i. Ceci assure que, depuis l'état s'_i (pour tout i), le joueur A_1 possède une stratégie (jouer toujours l'action 4) pour forcer $a \,\mathbf{W}\, b$. Or, ceci n'est pas possible à partir de l'état s_i : par induction sur i, on peut montrer que $s_i \not\models \langle\langle A_1\rangle\rangle a \,\mathbf{W}\, b$. Le cas de base est trivial. On suppose maintenant que cette propriété est vraie pour i : depuis s_{i+1}, pour toute stratégie de A_1 qui commence par une action dans $\{1, 2, 3\}$ et pour tous ces choix, le joueur A_2 pourrait choisir l'action (2, 1 et 2 resp.) qui force d'avoir l'état suivant s_i où par hypothèse d'induction A_1 n'a pas de stratégie pour avoir $a \,\mathbf{W}\, b$.

On montre maintenant que s_i et s'_i satisfont les mêmes formules de taille inférieure à i. On montre tout d'abord les équivalences suivantes :

Lemme 2.4
Pour tout $i > 0$, et tout $\psi \in \mathsf{ATL}_{orig}$ avec $|\psi| \leq i$:

$$b_i \models \psi \text{ ssi } b_{i+1} \models \psi \tag{2.3}$$
$$s_i \models \psi \text{ ssi } s_{i+1} \models \psi \tag{2.4}$$
$$s'_i \models \psi \text{ ssi } s'_{i+1} \models \psi \tag{2.5}$$

Démonstration: La preuve se fait par induction sur la structure de la formule ψ et sur i.

Cas de base : $i = 1$. Puisqu'on demande d'avoir $|\psi| \leq i$, ψ n'a donc qu'une proposition atomique. Le résultat est alors évident.

Étape d'induction. On suppose que le résultat est vrai jusqu'à un certain $i-1 \geq 1$, et on montre qu'il reste vrai pour i. Étant donné ψ t.q. $|\psi| \leq i$, on procède par induction sur la structure de ψ :

- Le résultat est évident pour les propositions atomiques et pour les combinaisons booléennes des sous-formules.
- Sinon, on considère la modalité de la quantification sur les stratégies dans ψ. Si c'est une modalité CTL ($\langle\langle\varnothing\rangle\rangle$ ou $\langle\langle A_1, A_2\rangle\rangle$), les résultats sont directs. De plus, puisqu'il existe une seule transition issue de b_i, on peut donc exprimer toute modalité de $\mathsf{ATL}_{\text{orig}}$ par une modalité de CTL dans cet état, et on obtient (2.3).
- Si $\psi = \langle\langle A_1\rangle\rangle \mathbf{X}\,\psi_1$: Supposons $s_i \models \psi$. Alors, selon la stratégie, soit b_i et s_{i-1}, ou a_i et s_{i-1}, ou s_i et s_{i-1}, vont satisfaire ψ_1. Par hypothèse d'induction, ceci se propage à l'étape suivante, et donc la même stratégie pourrait être jouée dans s_{i+1}.
  La réciproque est similaire (et donc (2.4)), ainsi que la preuve pour (2.5).
- Si $\psi = \langle\langle A_1\rangle\rangle \mathbf{G}\,\psi_1$: Si $s_i \models \psi$, donc s_i, et alors s_{i+1}, satisfont ψ_1. Jouer l'action 3 est une stratégie pour le joueur A_1 pour forcer $\mathbf{G}\,\psi_1$ depuis s_{i+1}, puisque le jeu ne peut pas rester dans s_{i+1} ou aller à s_i, quand le joueur A_1 possède une stratégie gagnante.
  La réciproque est immédiate, comme le joueur A_1 ne peut pas s'éviter l'état s_i quand il joue depuis s_{i+1}. On a donc (2.4) pour les formules $\langle\langle A_1\rangle\rangle \mathbf{G}\,\psi_1$.

  Si $s'_i \models \psi$, alors s'_i et s'_{i+1} satisfont ψ_1. De plus, le joueur A_1 ne peut pas s'abstenir de jouer pour arriver à l'état s_{i-1}. Et donc, $s_{i-1} \models \psi_1$ —et par hypothèse d'induction, la même chose pour s_i— et $s_i \models \psi$, comme ci-dessus. Maintenant, en suivant la même stratégie dans s'_{i+1} pour une stratégie gagnante de s'_i va clairement forcer $\mathbf{G}\,\psi_1$. La réciproque est pareille : il suffit d'imiter, depuis s'_i, la stratégie qui valide $s'_{i+1} \models \psi$. on montre ainsi (2.5), et on conclut ce cas.
- Si $\psi = \langle\langle A_1\rangle\rangle \psi_1 \mathbf{U}\,\psi_2$: Si $s_i \models \psi$, alors soit ψ_2 ou ψ_1 sont vrai dans s_i, et donc dans s_{i+1}. Le premier cas est trivial. Dans le dernier, le joueur A_1 peut imiter la stratégie gagnante dans s_{i+1} : le jeu arrive dans s_i, en passant par des états intermédiaires satisfaisant ψ_1 (ou ψ_2), et il peut appliquer donc sa stratégie originale.
  La réciproque est évidente, puisque depuis s_{i+1}, le joueur A_1 ne peut pas s'échapper de l'état s_i, depuis lequel il doit avoir une stratégie gagnante.

  Si $s'_i \models \psi$, à part le cas trivial où s'_i satisfait ψ_2, on a $s_{i-1} \models \psi$. Puisqu'une stratégie (sans-mémoire) dans s'_i qui valide ψ consiste nécessairement à jouer l'action 1 ou 2. Et donc a_i et b_i satisfont ψ, et la même stratégie (choix 1 ou 2, resp.) force $\mathbf{G}\,\psi_1$ depuis s_i. Il est maintenant donc facile de voir que la même stratégie est correcte depuis s'_{i+1}.
  Réciproquement, à part les cas triviaux, la stratégie consiste seulement à jouer 1 ou 2. Dans les deux cas, le jeu se termine dans s_i, et donc dans s_{i-1}. Alors $s_{i-1} \models \psi$, et la même stratégie dans s'_{i+1} pourrait être utilisée dans s'_i pour valider ψ.
- Les preuves des modalités $\langle\langle A_2\rangle\rangle \mathbf{X}\,\psi_1$, $\langle\langle A_2\rangle\rangle \mathbf{G}\,\psi_1$, et $\langle\langle A_2\rangle\rangle \psi_1 \mathbf{U}\,\psi_2$ sont similaires aux preuves précédentes.

□

Lemme 2.5
$\forall i > 0, \forall \psi \in \mathsf{ATL}_{orig}$ *avec* $|\psi| \leq i : s_i \models \psi$ *ssi* $s'_i \models \psi$.

Démonstration: La démonstration se fait par induction sur i, et sur la structure de la formule ψ.

Le cas où $i = 1$ est trivial puisque s_1 et s'_1 portent les mêmes propositions atomiques. Pour l'étape d'induction, raisonner sur les modalités de CTL ($\langle\langle\varnothing\rangle\rangle$ et $\langle\langle A_1, A_2\rangle\rangle$) est aussi direct, on considère donc juste le cas des modalités $\langle\langle A_1\rangle\rangle$ et $\langle\langle A_2\rangle\rangle$.

On considère tout d'abord les modalités $\langle\langle A_1\rangle\rangle$. On sait qu'on peut se restreindre à des stratégies sans mémoire dans cette configuration. Si le joueur A_1 possède une stratégie dans s_i pour forcer une propriété, il peut donc utiliser la même stratégie depuis s'_i. Réciproquement, si le joueur A_1 possède une stratégie dans s'_i pour forcer une propriété donnée, on distingue deux cas : soit cette stratégie qui consiste à jouer les actions 1, 2 ou 3, ce qui peut être imité depuis s_i. Ou bien la stratégie consiste à jouer l'action 4 et on distingue trois cas :

- $\psi = \langle\langle A_1\rangle\rangle \mathbf{X}\,\psi_1$: jouer 4 étant une stratégie gagnante conduit à avoir que s'_i, a_i et b_i doivent satisfaire ψ_1. Alors s_i (par hypothèse d'induction sur la formule) et s_{i-1} (par la Lemme 2.4) satisfont ψ_1. Jouer 1 (ou 3) dans s_i assure que l'état suivant va satisfaire ψ_1.
- $\psi = \langle\langle A_1\rangle\rangle \mathbf{G}\,\psi_1$: en jouant 4, le jeu arrive dans s_{i-1} (en passant par b_i), et dans a_i et s'_i. Donc $s_{i-1} \models \psi$, et en particulier ψ_1. Par hypothèse d'induction, $s_i \models \psi_1$, et en jouant 1 (ou 3) dans s_i, et donc en copiant la stratégie originale (depuis s'_i), cela force $\mathbf{G}\,\psi_1$.
- $\psi = \langle\langle A_1\rangle\rangle \psi_1 \mathbf{U}\,\psi_2$: une stratégie commençant avec l'action 4 implique $s'_i \models \psi_2$ (le jeu peut rester toujours dans s'_i). Donc $s_i \models \psi_2$ par hypothèse d'induction, et on obtient le résultat voulu.

On s'intéresse maintenant aux modalités $\langle\langle A_2\rangle\rangle$: clairement si $\langle\langle A_2\rangle\rangle \psi_1$ est satisfaite dans s'_i, elle sera encore satisfaite dans s_i. Réciproquement, si le joueur A_2 joue selon une stratégie (sans-mémoire) pour satisfaire une propriété dans s_i : Si elle consiste à jouer 1 ou 3, cette même stratégie devra donc marcher dans s'_i. Si cette stratégie consiste à jouer 2, ce choix a le même effet que de jouer 3 dans s'_i et donc satisfera la même propriété. □

Remarque 2.2
ATL_{orig} et ATL ont le même pouvoir de distinction puisqu'on a besoin seulement de la modalité $\langle\langle\cdot\rangle\rangle \mathbf{X}$ (voir [AHKV98, la preuve du Th. 6]). Ce qui implique qu'on ne peut pas avoir deux modèles M et M' une formule $\Phi \in \mathsf{ATL}$ t.q. (1) $M \models \Phi$, (2) $M' \not\models \Phi$, et (3) M et M' satisfont la même formule ATL_{orig}.

Remarque 2.3
Les jeux turn-based de la section 1.1.4.1 où chaque état est contrôlé par un seul joueur sont des structures qui possèdent la propriété d'être déterminée *: étant donné un ensemble d'agents A, cette coalition possède une stratégie gagnante pour un objectif Φ donné, ou alors, les adversaires($\mathsf{Agt} \setminus A$) ont une stratégie pour forcer $\neg\Phi$. La modalité $\mathbf{R}$ pourrait donc être exprimée dans ce genre de structure par :*

$$\langle\langle A\rangle\rangle \varphi\,\mathbf{R}\,\psi \equiv_{\text{TB-CGS}} \neg\langle\langle \mathsf{Agt} \setminus A\rangle\rangle (\neg\varphi)\,\mathbf{U}\,(\neg\psi).$$

2.2.2.2 $\langle\langle A\rangle\rangle \overset{\infty}{\mathbf{G}}$ et $\langle\langle A\rangle\rangle \overset{\infty}{\mathbf{F}}$ ne sont pas exprimables dans ATL

Il est bien connu qu'on ne peut pas exprimer les formules ECTL de la forme $\mathbf{E}\overset{\infty}{\mathbf{F}}\,P$ (et son dual $\mathbf{A}\overset{\infty}{\mathbf{G}}\,P$) dans CTL [Eme90]. D'un autre côté, on a les équivalences suivantes :

$$\mathbf{E}\overset{\infty}{\mathbf{G}}\,P \equiv \mathbf{EF}\ \mathbf{EG}\,P \qquad\qquad \mathbf{A}\overset{\infty}{\mathbf{F}}\,P \equiv \mathbf{AG}\ \mathbf{AF}\,P.$$

Dans ATL on se retrouve dans une situation légèrement différente : aucune des formules $\langle\langle A\rangle\rangle \overset{\infty}{\mathbf{F}}$ ou $\langle\langle A\rangle\rangle \overset{\infty}{\mathbf{G}}$ n'est exprimable dans ATL. En effet, si on suppose que $\langle\langle A\rangle\rangle \overset{\infty}{\mathbf{F}}$ est exprimable par

une formule $\varphi \in \mathsf{ATL}$, en particulier dans des jeux de 1 joueur (*i.e.*, les structures de Kripke), on se retrouve avec des formules équivalentes aux formules de CTL de la forme $\mathbf{E}\overset{\infty}{\mathbf{F}}$, qui ne sont pas exprimables. On raisonne de manière similaire pour $\langle\langle A\rangle\rangle \overset{\infty}{\mathbf{G}}$.

Conclusion

Nous avons étudié dans ce chapitre la complexité du model-checking de la logique ATL sur des différents types de structures multi-agents. La figure 2.9 récapitule ces résultats dans une table.

	CGS	ATS	CGS symbolique
$CPre$	AC^0	NP-complet	Σ_2^{P}-complet
ATL	PTIME-complet	Δ_2^{P}-complet	Δ_3^{P}-complet
ATL^+	Δ_3^{P}-complet	Δ_3^{P}-complet	Δ_3^{P}-complet
$\mathsf{ATL}^\star$	2-EXPTIME	2-EXPTIME	2-EXPTIME

FIG. 2.9 – La complexité du model-checking

Nous avons montré ensuite que la logique ATL présentée dans [AHK02] n'est pas aussi expressive que prévue. Nous avons proposé donc d'ajouter la modalité "Release" dans la définition. De même, nous avons donné la complexité des traductions entre les différentes structures de jeu par rapport à la bisimulation alternante.

Puisque, dans ce chapitre, nous nous sommes intéressés uniquement au model-checking des propriétés temporelles, le chapitre suivant sera consacré au model-checking des propriétés quantitatives sur des jeux. Les modèles seront basés donc sur des systèmes de transitions pondérés finis discrets, et la logique utilisée sera la logique TATL.

Pour en savoir plus ...

Dans ce chapitre, nous avons étudié le problème du model-checking d'ATL sous l'hypothèse que nous disposons des informations complètes sur le système. Or ce cas n'est pas toujours vrai comme dans les papiers [JÅ06, Sch04] où on suppose que les agents ont accès seulement à une partie de l'espace d'états du système. Dans ces papiers une étude, sur la complexité du model-checking a été faite dans ce cadre-là.

D'un autre côté, on trouve une étude intéressante [BGMR08] sur le problème de la coalition minimum, où on cherche à trouver la plus petite coalition pour qu'une formule ATL sera satisfaite. La complexité de ce problème est DP.

Finalement, le papier [GvD06] s'intéresse au problème de l'axiomatisation de la logique. Ensuite, Le problème de la satisfaisabilité d'ATL—c.à.d. étant donnée une formule ATL, trouver s'il existe une structure de jeu qui satisfait cette formule— a été montré EXPTIME-complet [GvD06] (c'est-à-dire la même complexité du problème que la satisfaisabilité de CTL [EH85]) dans le cas où le nombre d'agents est fixé, sinon, ce problème est étudié dans [WLWW06] où il est montré que la satisfaisabilité de la logique reste EXPTIME-complète même si le nombre d'agents n'est pas fixé. Par ailleurs, le problème de la satisfaisabilité de la logique $\mathsf{ATL}^\star$ garde la même complexité 2EXPTIME-complet que le model-checking [Sch08] qui est également la même complexité que le problème de satisfaisabilité de $\mathsf{CTL}^\star$ [VS85].

Chapitre 3 Structures de jeu avec durée

Introduction

Le développement des systèmes critiques a imposé d'enrichir les modèles existants dans la littérature, soit en ajoutant un aspect temporisé, soit par l'intermédiaire de l'addition de quelques aspects quantitatifs comme des poids discrets, des probabilités, etc. On trouve donc dans la littérature des modèles qui traitent le temps d'une manière discrète [LMS06, MPS95]. De tels modèles bénéficient d'algorithmes de model-checking efficaces (par rapport à ceux qui utilisent une notion dense du temps) [CC95, LMS06, MS04].

Nous présentons ici un modèle de jeu concurrent où les transitions du système portent des valeurs entières désignant le temps écoulé lorsqu'une transition est franchie. De même, nous pouvons également considérer ces valeurs entières comme des poids (ou des coûts) associés à chaque transition. Et à la fin de chaque exécution, nous pouvons de cette manière calculer la durée (ou bien le coût) totale qui a été nécessaire pour arriver à l'état final.

Mélanger l'aspect temporisé et l'aspect du jeu reste toujours un sujet de recherche, pour une extension de ce type de jeu où des intervalles de temps interviennent, la définition de la sémantique n'est plus naturelle. En effet, on se retrouve en train d'ajouter des agents extérieurs qui contrôlent la durée, ces agents pourront éventuellement être utilisés plus tard dans les formules pour donner le pouvoir du contrôle le temps à une coalition donnée.

Dans ce chapitre, nous introduisons des modèles qui disposent de la sémantique des jeux concurrents munie de temps discret. Ces modèles sont considérés comme une extension des CGS définies dans le chapitre 1, mais en ajoutant à chaque transition un entier qu'on considère comme une durée (ou bien, il peut être considéré également comme un coût).
Ces modèles sont vérifiés à l'aide d'une version temporisée de la logique ATL qu'on appelle TATL. Cette logique va donc quantifier sur les stratégies des agents, mais aussi, vérifier le temps écoulé (ou bien le coût payé) le long d'une exécution.

La première partie de ce chapitre introduit ces modèles et la logique. Ensuite, on montre les algorithmes du model-checking des différentes modalités de la logique ainsi qu'une étude de la complexité de ces algorithmes. Finalement, on présente une extension de ces modèles avec des intervalles d'entiers.

Les résultats dans chapitre sont basés sur le papier [LMO06]

3.1 Définitions

3.1.1 CGS avec durée exacte

Les CGS avec durée exacte sont un cas particulier des CGS qu'on a introduites dans le chapitre 1. Ce sont des structures de jeux où on peut mesurer le temps écoulé dans les transitions. Formellement :

Définition 3.1
Une CGS avec durée exacte *(Tight Durational Concurrent Game Structure ou simplement TDCGS) est un* 8*-uplet* $\mathcal{A} = \langle \mathit{Loc}, \mathit{AP}, \mathit{Lab}, \delta_{\mathbb{N}>0}, \mathit{Agt}, \mathbb{M}, \mathit{Mov}, \mathit{Edg} \rangle$ *avec :*

- $\langle \mathit{Loc}, \mathit{AP}, \mathit{Lab}, \delta_{\mathbb{N}>0} \rangle$ *est un système de transitions pondéré et* fini *;*
- $\langle \mathit{Loc}, \mathit{AP}, \mathit{Lab}, \delta_{\mathbb{N}>0}, \mathit{Agt}, \mathbb{M}, \mathit{Mov}, \mathit{Edg} \rangle$ *est une CGS ;*

Une TDCGS est un cas particuliers d'une CGS construite sur un système de transitions pondéré et fini (voir le chapitre 1). Là, dans les TDCGS, chaque transition porte une durée (ou un coût) entière positive non nulle. Étant donnée une transition $\mathsf{Edg}\left(\ell, m_1, \ldots, m_{|\mathsf{Agt}|}\right) = (\ell, d, \ell')$, on désigne par $\mathsf{Edg}_{\mathsf{Loc}}\left(\ell, m_1, \ldots, m_{|\mathsf{Agt}|}\right)$ pour noter l'état d'arrivée ℓ' et par $\mathsf{Edg}_{\tau}\left(\ell, m_1, \ldots, m_{|\mathsf{Agt}|}\right)$ la durée d écoulée dans la transition.

La sémantique des TDCGS est similaire à celle des CGS : Dans un état ℓ, chaque joueur A_i choisit une action m_i tel que $m_i \in \mathsf{Mov}(\ell, A_i)$. La table de transitions indique ensuite la transition à prendre $\mathsf{Edg}\left(\ell, m_1, \ldots, m_{|\mathsf{Agt}|}\right)$ qui mène à l'état suivant et la durée passée dans la transition.

Logiques temporelles. Bien que les logiques CTL, TCTL et ATL peuvent être utilisées sur les TDCGS, nous nous intéressons à vérifier des propriétés quantitatives sur les stratégies. Pour cette raison, on utilise la logique TATL (voir définition 1.22) pour spécifier des propriétés sur ces structures.

Nous allons étudier la complexité du model-checking de la logique TATL sur les TDCGS.

3.2 Le model-checking de la logique TATL

Dans le chapitre 2, nous avons vu que la complexité du model-checking d'une formule ATL sur des CGS est linéaire dans la taille de la structure et de la formule [AHK02]. D'un autre côté, la complexité du model-checking des formules TCTL sur les systèmes de transitions à temps discret est complète dans PTIME si les contraintes temporelles ne contiennent pas d'égalité, sinon, la complexité sera Δ_2^{P}-complet [LMS06].

Nous présentons dans cette section l'algorithme du model-checking pour la logique TATL sur les TDCGS. Nous expliquerons les méthodes pour traiter les modalités $\mathbf{U}_{\sim\zeta}$ et $\mathbf{R}_{\sim\zeta}$. Ensuite, nous regroupons ces algorithmes pour avoir un algorithme d'étiquetage qui est PTIME-complet sans les contraintes d'égalité, et EXPTIME-complet sinon.

3.2.1 Les contraintes $\leq \zeta$

3.2.1.1 La modalité $\mathbf{U}_{\leq\zeta}$

On considère le cas des formules de la forme $\langle\!\langle A\rangle\!\rangle\, \varphi_1\, \mathbf{U}_{\leq\zeta}\, \varphi_2$, qui signifie que la coalition A possède une stratégie pour vérifier φ_2 dans moins de ζ unités de temps et vérifie la formule φ_1 dans les états intermédiaires. On suppose que les états sont déjà étiquetés par les formules φ_1 et φ_2, et donc elles pourront être considérées comme des propositions atomiques.

Définissons la nouvelle modalité $\mathbf{U}^{\leq i}_{\leq n}$, avec la sémantique suivante :

$$\begin{aligned}\rho \models_{\mathcal{A}} P_1\, \mathbf{U}^{\leq i}_{\leq n}\, P_2 \quad \Leftrightarrow \quad & \exists j \leq i.\ \rho[j] \models_{\mathcal{A}} P_2,\ \mathsf{Cost}(0 \twoheadrightarrow_\rho j) \leq n, \\ & \text{et } \rho[k] \models_{\mathcal{A}} P_1 \text{ pour tout } 0 \leq k < j\end{aligned}$$

En supposant que la coallition A est fixée, on définit maintenant les fonctions $v_i(\ell)$, pour $i \in \mathbb{N}$ et $\ell \in \mathsf{Loc}$, avec les règles récursives suivantes :

$$\begin{cases} \text{si } \ell \models P_2 : & v_0(\ell) = 0 \\ \text{si } \ell \models \neg P_2 : & v_0(\ell) = +\infty \end{cases}$$

$$\begin{cases} \text{si } \ell \models P_2 : & v_{i+1}(\ell) = 0 \\ \text{si } \ell \models \neg P_1 \wedge \neg P_2 : & v_{i+1}(\ell) = +\infty \\ \text{sinon :} & v_{i+1}(\ell) = \min\limits_{c \in \mathsf{Mov}(\ell, A)} \max\limits_{\bar{c} \in \mathsf{Mov}(\ell, \bar{A})} \Big(\mathsf{Edg}_\tau(\ell, c \oplus \bar{c}) + v_i(\mathsf{Edg}_{\mathsf{Loc}}(\ell, c \oplus \bar{c})) \Big) \end{cases}$$

Nous montrons le lemme suivant :

Lemme 3.1
Pour tout $i \in \mathbb{N}$, pour tout $n \in \mathbb{N}$ et pour tout $\ell \in \mathsf{Loc}$, on a :

$$n \geq v_i(\ell) \quad \Leftrightarrow \quad \ell \models \langle\!\langle A\rangle\!\rangle\, P_1\, \mathbf{U}^{\leq i}_{\leq n}\, P_2.$$

Démonstration: La preuve est par induction sur i : Le cas de base est direct, l'étape d'induction est aussi triviale quand $\ell \models P_2$ et quand $\ell \models \neg P_1 \wedge \neg P_2$. Nous montrons donc le dernier cas, quand $\ell \models P_1 \wedge \neg P_2$: Supposons que l'hypothèse d'induction est vraie pour l'étape i. Si $n \geq v_{i+1}(\ell)$, alors (par définition de $v_{i+1}(\ell)$) il existe un choix $c \in \mathsf{Mov}(\ell, A)$ tel que, pour toute action $\bar{c} \in \mathsf{Mov}(\ell, \bar{A})$, on a

$$n \geq \mathsf{Edg}_\tau(\ell, c \oplus \bar{c}) + v_i(\mathsf{Edg}_{\mathsf{Loc}}(\ell, c \oplus \bar{c})).$$

Par hypothèse d'induction, pour tout $\bar{c} \in \mathsf{Mov}(\ell, \bar{A})$, on a

$$\mathsf{Edg}_{\mathsf{Loc}}(\ell, c \oplus \bar{c}) \models \langle\!\langle A\rangle\!\rangle\, P_1\, \mathbf{U}^{\leq i}_{\leq n - \mathsf{Edg}_\tau(\ell, c \oplus \bar{c})}\, P_2.$$

La stratégie σ_A qui garantit cette propriété complétée avec l'action initiale $c \in \mathsf{Mov}(\ell, A)$, est une stratégie qui force $P_1\, \mathbf{U}^{\leq i+1}_{\leq n}\, P_2$ à partir de ℓ.

La réciproque suit les mêmes démarches : étant donnée une stratégie σ_A qui force $P_1\, \mathbf{U}^{\leq i+1}_{\leq n}\, P_2$ depuis ℓ, et soit $c = \sigma_A(\ell)$, on déduit que n satisfait les mêmes inégalités ci-dessus. □

On obtient alors la correction de l'algorithme.

Lemme 3.2
Soient $\mathcal{A}$ une TDCGS, et $\varphi = \langle\langle A\rangle\rangle P_1 \mathbf{U}_{\leq\zeta} P_2$ une formule TATL (avec $P_1, P_2 \in AP$). On peut calculer l'ensemble des états de $\mathcal{A}$ où φ est satisfaite en temps $\mathcal{O}(|Loc| \cdot |Edg|)$.

Démonstration: On note tout d'abord que la modalité $\mathbf{U}^{\leq i}_{\leq k}$ nécessite que la partie droite de la formule soit satisfaite dans i étapes au plus. Il est évident que pour tout $n \in \mathbb{N}$,

$$\ell \models \langle\langle A\rangle\rangle P_1 \mathbf{U}_{\leq n} P_2 \quad \Leftrightarrow \quad \ell \models \langle\langle A\rangle\rangle P_1 \mathbf{U}^{\leq |\mathsf{Loc}|}_{\leq n} P_2. \tag{3.1}$$

En effet, si tous les outcomes d'une stratégie satisfont $P_1 \mathbf{U}_{\leq n} P_2$, il est possible d'adapter cette stratégie pour que chaque état soit visité au plus une fois dans chaque outcome (puisque les coûts sont positifs).

En utilisant l'équation (3.1) et le lemme 3.1, il suffit de calculer $v_{|\mathsf{Loc}|}(\ell)$, pour chaque $\ell \in \mathsf{Loc}$, pour déduire l'ensemble d'états où φ est satisfaite. Cet algorithme nécessite un temps d'exécution $\mathcal{O}(|\mathsf{Loc}| \cdot |\mathsf{Edg}|)$. □

3.2.1.2 La modalité $\mathbf{R}_{\leq\zeta}$

La modalité release est traitée d'une manière similaire.
Nous définissons $v'_i(\ell)$ par :

$$\begin{cases} \text{si } \ell \models \neg P_2 : & v'_0(\ell) = 0 \\ \text{si } \ell \models P_2 : & v'_0(\ell) = +\infty \end{cases}$$

$$\begin{cases} \text{si } \ell \models \neg P_2 : & v'_{i+1}(\ell) = 0 \\ \text{si } \ell \models P_1 \wedge P_2 : & v'_{i+1}(\ell) = +\infty \\ \text{sinon :} & v'_{i+1}(\ell) = \max\limits_{c \in \mathsf{Mov}(\ell, A)} \min\limits_{\bar{c} \in \mathsf{Mov}(\ell, \bar{A})} \Big(\mathsf{Edg}_\tau(\ell, c \oplus \bar{c}) + v'_i(\mathsf{Edg}_{\mathsf{Loc}}(\ell, c \oplus \bar{c}))\Big) \end{cases}$$

On définit l'opérateur $\mathbf{R}^{\leq i}_{\leq n}$ comme étant le dual de $\mathbf{U}^{\leq i}_{\leq n}$, c.à.d.

$$\rho \models_{\mathcal{A}} P_1 \mathbf{R}^{\leq i}_{\leq n} P_2 \quad \Leftrightarrow \quad \forall j \leq i.\, [(\mathsf{Cost}(0 \twoheadrightarrow_\rho j) \leq n \rightarrow \rho[j] \models_{\mathcal{A}} P_2) \text{ ou } \exists k < j. \rho[k] \models_{\mathcal{A}} P_1]$$

Alors, et en considérant que seules les durées strictement positives sont possibles, l'équivalence similaire à celle de l'équation (3.1) sera satisfaite pour tout $n \in \mathbb{N}$

$$\ell \models \langle\langle A\rangle\rangle P_1 \mathbf{R}_{\leq n} P_2 \quad \Leftrightarrow \quad \ell \models \langle\langle A\rangle\rangle P_1 \mathbf{R}^{\leq |\mathsf{Loc}|}_{\leq n} P_2. \tag{3.2}$$

et on obtient le lemme suivant :

Lemme 3.3
Pour tout $i \in \mathbb{N}$, pour tout $n \in \mathbb{N}$ et tout $\ell \in Loc$, on a :

$$n < v'_i(\ell) \quad \Leftrightarrow \quad \ell \models \langle\langle A\rangle\rangle P_1 \mathbf{R}^{\leq i}_{\leq n} P_2.$$

Démonstration: Comme dans le cas précédent, on procède dans la preuve par induction sur i. Les cas de base étant triviaux, ainsi que les cas $\ell \models \neg P_2$ et $\ell \models P_1 \wedge P_2$, nous nous intéressons au dernier cas quand $\ell \models \neg P_1 \wedge P_2$: Supposons que l'hypothèse est vraie pour l'étape i. Si $n < v'_{i+1}(\ell)$, alors par définition, il existe un choix $c \in \mathsf{Mov}(\ell, A)$ tel que, pour toute action $\bar{c} \in \mathsf{Mov}(\ell, \bar{A})$, on a,

$$n < \mathsf{Edg}_\tau(\ell, c \oplus \bar{c}) + v'_i(\mathsf{Edg}_{\mathsf{Loc}}(\ell, c \oplus \bar{c}))$$

D'après notre hypothèse d'induction, on a pour tout $\bar{c} \in \mathsf{Mov}\left(\ell, \bar{A}\right)$,

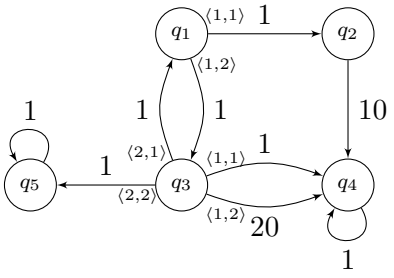
$$\mathsf{Edg}_{\mathsf{Loc}}(\ell, c \oplus \bar{c}) \models \langle\!\langle A \rangle\!\rangle\, P_1\, \mathbf{R}^{\leq i}_{\leq n - \mathsf{Edg}_\tau(\ell, c \oplus \bar{c})}\, P_2$$

En combinant la stratégie qui garantit cette dernière propriété avec l'action c, on obtient une stratégie qui garantit, à partir de l'état ℓ, la propriété $P_1\, \mathbf{R}^{\leq i+1}_{\leq n}\, P_2$. La réciproque est aussi valable pour montrer le sens inverse. □

Ce qui nous donne le lemme suivant :

Lemme 3.4
Soient $\mathcal{A}$ une TDCGS, et $\varphi = \langle\!\langle A \rangle\!\rangle\, P_1\, \mathbf{R}_{\leq \zeta}\, P_2$ une formule TATL (avec $P_1, P_2 \in AP$). On peut calculer l'ensemble des états de $\mathcal{A}$ où φ est satisfaite dans un temps $\mathcal{O}(|\mathsf{Loc}| \cdot |\mathsf{Edg}|)$.

Démonstration: Puisque le calcul de la nouvelle modalité $\mathbf{R}^{\leq i}_{\leq k}$ se fait un temps polynomial. La preuve est donc évidente, ayant que la séquence v' nécessite un temps $|\mathsf{Loc}|$ au plus (en suivant l'équation 3.2) □



$v_i(\ell)$	0	1	2	3
q_1 $(P_1, \neg P_2)$	$+\infty$	$+\infty$	21	21
q_2 (P_1, P_2)	0	0	0	0
q_3 $(P_1, \neg P_2)$	$+\infty$	20	20	20
q_4 $(\neg P_1, P_2)$	0	0	0	0
q_5 $(P_1, \neg P_2)$	$+\infty$	$+\infty$	$+\infty$	$+\infty$

FIG. 3.1 – L'algorithme pour $\mathbf{U}_{\leq \zeta}$.

Exemple 3.1
Considérons l'exemple de la figure 3.1. Dans cette TDCGS, la durée est l'entier qui figure au milieu de chaque transition. Les vecteurs qui sont écrits près des états nous indiquent les choix des agents pour tirer cette transition. Quand chaque agent ne possède qu'un seul choix possible, les transitions ne seront pas étiquetées par ces choix.

Les valuations des propositions atomiques sont données dans la table à droite de la figure. Cette table montre les valeurs de $v_i(\ell)$ pour chaque état en exécutant l'algorithme. Comme on le remarque, cette exécution converge dans trois étapes. Par exemple, $v_3(q_1) = 21$ indique que $q_1 \models \langle\!\langle A_1 \rangle\!\rangle\, P_1\, \mathbf{U}_{\leq 21}\, P_2$ est satisfaite, mais $q_1 \not\models \langle\!\langle A_1 \rangle\!\rangle\, P_1\, \mathbf{U}_{\leq 20}\, P_2$.

3.2.2 Les contraintes $\geq \zeta$

Cette section est dédiée à l'étude des modalités avec des contraintes $\geq \zeta$. Achever un objectif avec ces modalités peut nécessiter le passage dans les mêmes états plusieurs fois pour qu'une exécution soit plus longue, les stratégies des joueurs dans ce cas ne sont plus nécessairement des stratégies sans mémoire. En effet, un joueur peut avoir une stratégie pour boucler dans un état jusqu'à garantir une valeur donnée, et ensuite changer la stratégie pour arriver à son objectif.

On trouve dans la TDCGS de la figure 3.2 que la stratégie du premier joueur dans l'état vérifiant P_1, pour satisfaire la formule $\langle\!\langle A \rangle\!\rangle P_1 \mathbf{U}_{\geq 30} P_2$, est de de jouer 14 fois le coup 1 et puis jouer le coup 2 pour arriver à l'état vérifiant P_2. Il passe donc une durée supérieure à 30 unités de temps. (Dans cet exemple, on note que seulement deux stratégies sans mémoire sont possibles, l'une bouclant dans l'état satisfaisant P_1 ne vérifiant pas la formule, et l'autre menant vers l'état satisfaisant P_2 dans 3 unités de temps.)

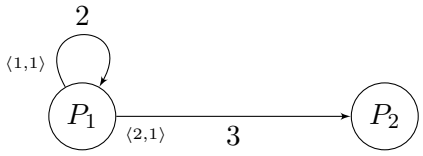


FIG. 3.2 – $\langle\!\langle A \rangle\!\rangle P_1 \mathbf{U}_{\geq 30} P_2$ nécessite une stratégie avec mémoire

3.2.2.1 La modalité $\mathbf{U}_{\geq\zeta}$

On considère maintenant les formules de la forme $\langle\!\langle A \rangle\!\rangle \varphi_1 \mathbf{U}_{\geq\zeta} \varphi_2$ qui exprime que la coalition A possède une stratégie pour rester dans des états φ_1 au moins ζ unités de temps avant d'arriver dans φ_2.

On introduit la modalité suivante :

$$\rho \models_{\mathcal{A}} P_1 \mathbf{U}^{\geq i} P_2 \quad \Leftrightarrow \quad \exists j \geq i.\ \rho[j] \models_{\mathcal{A}} P_2 \text{ et } \rho[k] \models_{\mathcal{A}} P_1 \text{ pour tout } 0 \leq k < j$$

On calcule ensuite la séquence des valeurs, définie par les règles récursives suivantes :

$$\begin{cases} \text{si } \ell \models \neg \langle\!\langle A \rangle\!\rangle P_1 \mathbf{U} P_2 : & w_0(\ell) = -\infty \\ \text{si } \ell \models \langle\!\langle A \rangle\!\rangle P_1 \mathbf{U} P_2 \wedge \neg \langle\!\langle A \rangle\!\rangle P_1 \mathbf{U}^{\geq 1} P_2 : & w_0(\ell) = 0 \\ \text{si } \ell \models \langle\!\langle A \rangle\!\rangle P_1 \mathbf{U}^{\geq 1} P_2 : & w_0(\ell) = +\infty \end{cases}$$

$$\begin{cases} \text{si } \ell \models \neg \langle\!\langle A \rangle\!\rangle P_1 \mathbf{U} P_2 : & w_{i+1}(\ell) = -\infty \\ \text{si } \ell \models \langle\!\langle A \rangle\!\rangle P_1 \mathbf{U} P_2 \wedge \neg \langle\!\langle A \rangle\!\rangle P_1 \mathbf{U}^{\geq 1} P_2 : & w_{i+1}(\ell) = 0 \\ \text{si } \ell \models \langle\!\langle A \rangle\!\rangle P_1 \mathbf{U}^{\geq 1} P_2 : & w_{i+1}(\ell) = \max\limits_{c \in \mathsf{Mov}(\ell, A)} \min\limits_{\bar{c} \in \mathsf{Mov}(\ell, \bar{A})} \Big(\mathsf{Edg}_\tau(\ell, c \oplus \bar{c}) + w_i(\mathsf{Edg}_{\mathsf{Loc}}(\ell, c \oplus \bar{c})) \Big) \end{cases}$$

Ces valeurs satisfont le lemme suivant :

Lemme 3.5
Pour tout $i \in \mathbb{N}$, pour tout $n \in \mathbb{N}$ et $\ell \in \mathsf{Loc}$, on a

$$n \leq w_i(\ell) \quad \Leftrightarrow \quad \ell \models \langle\!\langle A \rangle\!\rangle \big[(P_1 \mathbf{U}_{\geq n} P_2) \vee (P_1 \mathbf{U}^{\geq i+1} P_2) \big].$$

Démonstration: Les cas simples étant triviaux, on s'intéresse à l'étape inductive dans le cas où $\ell \models \langle\!\langle A \rangle\!\rangle P_1 \mathbf{U}^{\geq 1} P_2$. En particulier, on a $\ell \models P_1$. Premièrement, choisissons $n \leq w_{i+1}(\ell)$, supposons que le résultat est vrai à l'étape i. Par définition de $w_{i+1}(\ell)$, il existe $c \in \mathsf{Mov}(\ell, A)$ tel que, pour tout $\bar{c} \in \mathsf{Mov}(\ell, \bar{A})$, on a

$$n \leq \mathsf{Edg}_\tau(\ell, c \oplus \bar{c}) + w_i(\mathsf{Edg}_{\mathsf{Loc}}(\ell, c \oplus \bar{c})).$$

Par hypothèse d'induction, cela signifie que

$$\mathsf{Edg}_{\mathsf{Loc}}(\ell, c \oplus \bar{c}) \models \langle\langle A \rangle\rangle \left[(P_1 \, \mathbf{U}_{\geq n - \mathsf{Edg}_\tau(\ell, c \oplus \bar{c})} \, P_2) \vee (P_1 \, \mathbf{U}^{\geq i+1} \, P_2) \right].$$

La stratégie qui garantit cette propriété et le choix $c \in \mathsf{Mov}(\ell, A)$, nous donnent une stratégie qui garantit que

$$\ell \models \langle\langle A \rangle\rangle \left[(P_1 \, \mathbf{U}_{\geq n} \, P_2) \vee (P_1 \, \mathbf{U}^{\geq i+2} \, P_2) \right].$$

La réciproque suit les mêmes étapes (dans le sens inverse). □

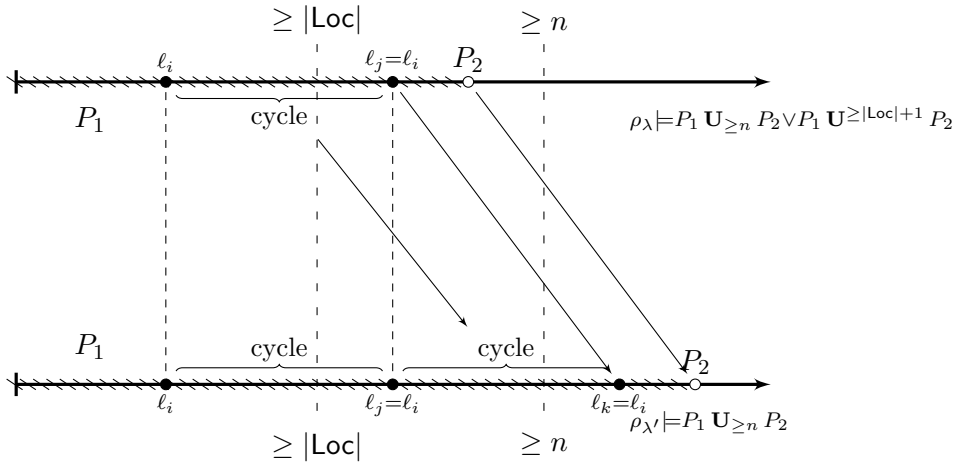


FIG. 3.3 – Schéma expliquant la preuve pour la modalité $\mathbf{U}_{\geq \zeta}$

Lemme 3.6
Soient $\mathcal{A}$ une TDCGS, et $\varphi = \langle\langle A \rangle\rangle P_1 \, \mathbf{U}_{\geq \zeta} \, P_2$ une formule TATL (avec $P_1, P_2 \in AP$). Le calcul de l'ensemble d'états de $\mathcal{A}$ où φ est satisfaite se fait en un temps $\mathcal{O}(|\mathsf{Loc}| \cdot |\mathsf{Edg}|)$.

Démonstration: En utilisant le lemme 3.5, on a l'équivalence suivante :

$$\ell \models \langle\langle A \rangle\rangle P_1 \, \mathbf{U}_{\geq n} \, P_2 \;\Leftrightarrow\; \ell \models \langle\langle A \rangle\rangle \left[(P_1 \, \mathbf{U}_{\geq n} \, P_2) \vee (P_1 \, \mathbf{U}^{\geq |\mathsf{Loc}|+1} \, P_2) \right].$$

Cette équivalence repose sur le fait que toutes les durées sont strictement positives : Si un outcome de la stratégie satisfait $P_1 \, \mathbf{U}^{\geq |\mathsf{Loc}|+1} \, P_2$, alors un état est visité deux fois (comme l'indique la figure 3.3), il est donc possible d'adapter cette stratégie pour qu'il soit visité n fois (et donc pour une durée supérieure à n) avant d'arriver dans un état satisfaisant P_2. On note que la stratégie utilisée dans ce cas est une stratégie *avec* mémoire, on applique la stratégie extraite de l'algorithme jusqu'à dépasser la durée de la formule, ensuite on utilise la stratégie pour vérifier la formule $\langle\langle A \rangle\rangle P_1 \, \mathbf{U} \, P_2$.

L'idée est similaire à celle de la preuve du lemme 3.2. Cet algorithme nécessite de calculer premièrement l'ensemble des états qui satisfont $\langle\langle A \rangle\rangle P_1 \, \mathbf{U} \, P_2$ et ceux qui satisfont $\langle\langle A \rangle\rangle P_1 \, \mathbf{U}^{\geq 1} \, P_2$. Ce calcul se fait dans un temps $\mathcal{O}(|\mathsf{Edg}|)$ en utilisant les algorithmes du model-checking de la logique ATL. Ensuite, le calcul de $w_i(\ell)$ pour chaque $\ell \in \mathsf{Loc}$ et pour chaque $i \leq |\mathsf{Loc}|$ se fait en un temps $\mathcal{O}(|\mathsf{Loc}| \cdot |\mathsf{Edg}|)$.

□

3.2.2.2 La modalité $\mathbf{R}_{\geq\zeta}$

Le cas de la modalité release est similaire au cas précédent.

On définit la modalité suivante :

$$\rho \models_{\mathcal{A}} P_1 \,\mathbf{R}^{\geq i}\, P_2 \quad \Leftrightarrow \quad \begin{array}{l}\text{pour tout } j \geq i.\ \rho[j] \models_{\mathcal{A}} P_2 \\ \qquad\qquad \text{ou il existe } k < j.\ \rho[k] \models_{\mathcal{A}} P_1\end{array}$$

Ensuite, on calcule la séquence des valeurs w' définie par :

$$\begin{cases} \text{si } \ell \models \langle\!\langle A \rangle\!\rangle P_1 \,\mathbf{R}\, P_2 : & w'_0(\ell) = -\infty \\ \text{si } \ell \models \neg\langle\!\langle A \rangle\!\rangle P_1 \,\mathbf{R}\, P_2 \wedge \langle\!\langle A \rangle\!\rangle P_1 \,\mathbf{R}^{\geq 1}\, P_2 : & w'_0(\ell) = 0 \\ \text{si } \ell \models \neg\langle\!\langle A \rangle\!\rangle P_1 \,\mathbf{R}^{\geq 1}\, P_2 : & w'_0(\ell) = +\infty \end{cases}$$

$$\begin{cases} \text{si } \ell \models \langle\!\langle A \rangle\!\rangle P_1 \,\mathbf{R}\, P_2 : & w'_{i+1}(\ell) = -\infty \\ \text{si } \ell \models \neg\langle\!\langle A \rangle\!\rangle P_1 \,\mathbf{R}\, P_2 \wedge \langle\!\langle A \rangle\!\rangle P_1 \,\mathbf{R}^{\geq 1}\, P_2 : & w'_{i+1}(\ell) = 0 \\ \text{si } \ell \models \neg\langle\!\langle A \rangle\!\rangle P_1 \,\mathbf{R}^{\geq 1}\, P_2 : & w'_{i+1}(\ell) = \min\limits_{c \in \mathsf{Mov}(\ell, A)} \max\limits_{\bar{c} \in \mathsf{Mov}(\ell, \bar{A})} \Big(\mathsf{Edg}_\tau(\ell, c \oplus \bar{c}) + w'_i(\mathsf{Edg}_{\mathsf{Loc}}(\ell, c \oplus \bar{c})) \Big) \end{cases}$$

On obtient le lemme :

Lemme 3.7
Pour tout $i \in \mathbb{N}$, pour tout $n \in \mathbb{N}$ et $\ell \in \mathsf{Loc}$, on a

$$n > w'_i(\ell) \quad \Leftrightarrow \quad \ell \models \langle\!\langle A \rangle\!\rangle \big[(P_1 \,\mathbf{R}_{\geq n}\, P_2) \wedge (P_1 \,\mathbf{R}^{\geq i+1}\, P_2) \big].$$

Démonstration: Le cas où $\ell \models \langle\!\langle A \rangle\!\rangle P_1 \,\mathbf{R}\, P_2$ est évident puisque dans ce cas $\ell \models \langle\!\langle A \rangle\!\rangle P_1 \,\mathbf{R}_{\geq n}\, P_2$ pour tout $n \geq 0$, et donc $w'_i(\ell) = -\infty$ pour tout i.

On prouve le lemme pour les cas où $\ell \models \neg\langle\!\langle A \rangle\!\rangle P_1 \,\mathbf{R}^{\geq 1}\, P_2$ par induction. Les cas de base sont faciles. Supposons que l'équivalence du lemme 3.7 est satisfaite pour i et montrons que le lemme est vrai pour l'étape $i+1$. Posons $n > w'_{i+1}(\ell)$, alors, par définition, il existe un choix c depuis l'état ℓ pour la coalition A, tel que pour tout $\bar{c} \in \mathsf{Mov}(\ell, \bar{A})$, on a :

$$n > \mathsf{Edg}_\tau(\ell, c \oplus \bar{c}) + w'_i(\mathsf{Edg}_{\mathsf{Loc}}(\ell, c \oplus \bar{c})).$$

Or, on a que

$$\mathsf{Edg}_{\mathsf{Loc}}(\ell, c \oplus \bar{c}) \models \langle\!\langle A \rangle\!\rangle \big[(P_1 \,\mathbf{R}_{\geq n - \mathsf{Edg}_\tau(\ell, c \oplus \bar{c})}\, P_2) \wedge (P_1 \,\mathbf{R}^{\geq i+1}\, P_2) \big].$$

On adapte alors facilement la stratégie qui garantit cette propriété, on obtient alors :

$$\ell \models \langle\!\langle A \rangle\!\rangle \big[(P_1 \,\mathbf{R}_{\geq n}\, P_2) \wedge (P_1 \,\mathbf{R}^{\geq i+2}\, P_2) \big].$$

□

Lemme 3.8
Soient $\mathcal{A}$ une TDCGS, et $\varphi = \langle\!\langle A \rangle\!\rangle P_1 \,\mathbf{R}_{\geq\zeta}\, P_2$ une formule TATL (avec $P_1, P_2 \in \mathsf{AP}$). Le calcul de l'ensemble d'états de $\mathcal{A}$ où φ est satisfaite se fait en un temps $\mathcal{O}(|\mathsf{Loc}| \cdot |\mathsf{Edg}|)$.

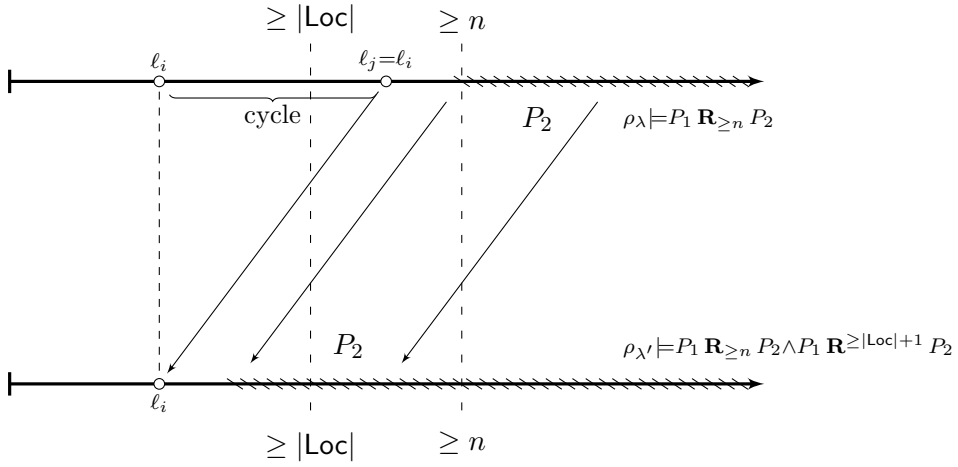


FIG. 3.4 – Schéma expliquant la preuve pour la modalité $\mathbf{R}_{\geq\zeta}$

Démonstration: Montrons tout d'abord l'équivalence suivante :

$$\ell \models \langle\!\langle A \rangle\!\rangle\, P_1\, \mathbf{R}_{\geq n}\, P_2 \quad \Leftrightarrow \quad \ell \models \langle\!\langle A \rangle\!\rangle \left[(P_1\, \mathbf{R}_{\geq n}\, P_2) \wedge (P_1\, \mathbf{R}^{\geq |\mathsf{Loc}|+1}\, P_2) \right].$$

Le sens inverse étant évident, on s'intéresse à montrer l'autre direction dans l'équivalence. Ayant un état $\ell \models \langle\!\langle A \rangle\!\rangle\, P_1\, \mathbf{R}_{\geq n}\, P_2$ signifie que la coalition A dispose d'une stratégie qui garantit $P_1\, \mathbf{R}_{\geq n}\, P_2$, on considère donc les cas suivants : si cette même stratégie garantit que le chemin satisfait $\mathsf{P}_1\, \mathbf{R}^{\geq |\mathsf{Loc}|+1}\, P_2$ l'équivalence est satisfaite, sinon, on peut conclure que le chemin en question passe par un même état au moins deux fois créant des cycles, ceci est illustré dans le chemin en haut de la figure 3.4, et donc on peut modifier cette stratégie pour ne plus trouver des cycles (chemin $\rho_{\lambda'}$ dans la figure) et donc réduire cette stratégie pour une stratégie sans mémoire, vérifiant ainsi la deuxième partie de l'équivalence.

En se basant sur l'équivalence ci-dessus et le lemme 3.7, on obtient l'équivalence :

$$n > w'_{|\mathsf{Loc}|+1}(\ell) \Leftrightarrow \ell \models \langle\!\langle A \rangle\!\rangle\, P_1\, \mathbf{R}_{\geq n}\, P_2 \quad \Leftrightarrow \quad \ell \models \langle\!\langle A \rangle\!\rangle \left[(P_1\, \mathbf{R}_{\geq n}\, P_2) \wedge (P_1\, \mathbf{R}^{\geq |\mathsf{Loc}|+1}\, P_2) \right].$$

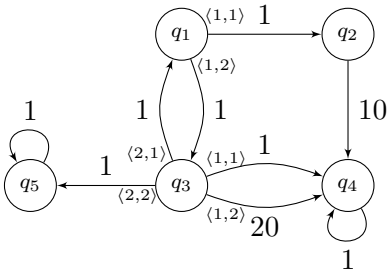
Comme dans le cas de la modalité $\mathbf{U}_{\geq\zeta}$, pour calculer la séquence w'_i, on doit premièrement identifier les états qui satisfont la formule ATL $\langle\!\langle A \rangle\!\rangle\, P_1\, \mathbf{R}\, P_2$, ce qui nécessite un temps polynomial dans la taille de la table de transitions. Ensuite, on calcule les valeurs $w'_i(\ell)$ pour tout $i \leq |\mathsf{Loc}|$ et tout $\ell \in \mathsf{Loc}$, d'où le temps de calcul $\mathcal{O}(|\mathsf{Loc}| \cdot |\mathsf{Edg}|)$.

□

Exemple 3.2
Dans la figure 3.5, on applique l'algorithme pour la même TDCGS de l'exemple précédent. La table montre l'exécution de $w_i(\ell)$, pour chaque état. L'exécution converge dans trois étapes. Par exemple, $w_3(q_1) = 2$ indique que $q_1 \models \langle\!\langle A_1 \rangle\!\rangle\, P_1\, \mathbf{U}_{\geq 2}\, P_2$ est vraie, mais $q_1 \not\models \langle\!\langle A_1 \rangle\!\rangle\, P_1\, \mathbf{U}_{\geq 3}\, P_2$.

3.2.3 Les contraintes $= \zeta$

Dans cette section, on s'intéresse à étudier les contraintes d'égalité dans les formules TATL séparément des contraintes rencontrées précédemment. La raison principale d'avoir faire cette



$w_i(\ell)$	0	1	2	3
q_1 $(P_1, \neg P_2)$	$+\infty$	$+\infty$	2	2
q_2 (P_1, P_2)	$+\infty$	10	10	10
q_3 $(P_1, \neg P_2)$	$+\infty$	1	1	1
q_4 $(\neg P_1, P_2)$	0	0	0	0
q_5 $(P_1, \neg P_2)$	$-\infty$	$-\infty$	$-\infty$	$-\infty$

FIG. 3.5 – L'algorithme pour $\mathbf{U}_{\geq\zeta}$.

séparation est, comme on le prouve ultérieurement dans cette section, vérifier une propriété ayant ces contraintes est de complexité exponentielle.

3.2.3.1 La modalité $\mathbf{U}_{=\zeta}$

Lemme 3.9
Soient $\mathcal{A}$ une TDCGS, et $\varphi = \langle\!\langle A \rangle\!\rangle P_1 \mathbf{U}_{=\zeta} P_2$ une formule TATL. On peut calculer l'ensemble des états de $\mathcal{A}$ où φ est satisfaite dans un temps $\mathcal{O}(\zeta \cdot |\mathsf{Edg}|)$

Puisque ζ est codé en binaire, cet algorithme nécessite un temps exponentiel dans la taille de la formule.
Démonstration: On utilise la programmation dynamique, et on construit récursivement la table $T\colon \mathsf{Loc} \times \{0, \ldots, \zeta\} \to \{\top, \bot\}$, telle que

$$T(\ell, i) = \top \quad \Leftrightarrow \quad \ell \models_{\mathcal{A}} \langle\!\langle A \rangle\!\rangle P_1 \mathbf{U}_{=i} P_2. \tag{3.3}$$

Quand $i = 0$, avoir $T(\ell, 0) = \top$ ssi $\ell \models P_2$ vérifie l'équation (3.3) (puisque toutes les durées sont non nulles). Maintenant, choisissons $i < \zeta$, et supposons que tous les $T(\ell, j)$ sont calculés pour $j \leq i$. Alors,

$T(\ell, i+1) = \top \quad \Leftrightarrow \quad \ell \models P_1$ et $\exists c \in \mathsf{Mov}(\ell, A).\ \forall \bar{c} \in \mathsf{Mov}(\ell, \bar{A}).$
$\mathsf{Edg}(\ell, c \cdot \bar{c}) = (\ell, d, \ell')$ avec $T(\ell', i-d) = \top$.

Cette étape peut être faite puisque toutes les durées sont non-nulles. Ce calcul sera terminé en parcourant la table de transitions, et donc dans un temps linéaire dans la taille de Edg. Il est évident que l'équation (3.3) est vérifiée dans cette construction, et donc finalement, $\ell \models \varphi$ ssi $T(\ell, \zeta) = \top$. Cet algorithme s'exécute en temps $\mathcal{O}(\zeta \cdot |\mathsf{Edg}|)$.

□

3.2.3.2 La modalité $\mathbf{R}_{=\zeta}$

Un algorithme similaire est défini pour traiter les modalités de la forme $\langle\!\langle A \rangle\!\rangle \varphi_1 \mathbf{R}_{=\zeta} \varphi_2$: la table $T'(\ell, i)$ est initialisée de la même manière (i.e., $T'(\ell, 0) = \top \Leftrightarrow \ell \models P_2$), et chaque étape est calculée suivant la règle :

$T'(\ell, i) = \top \quad \Leftrightarrow \quad \ell \models P_1$ ou $\exists c \in \mathsf{Mov}(\ell, A).\ \forall \bar{c} \in \mathsf{Mov}(\ell, \bar{A}).$
$\mathsf{Edg}(\ell, c \cdot \bar{c}) = (\ell, d, \ell')$ avec $T'(\ell', i-d) = \top$.

D'où le lemme suivant :

Lemme 3.10
Soient $\mathcal{A}$ une TDCGS, et $\varphi = \langle\langle A \rangle\rangle P_1 \mathbf{R}_{=\zeta} P_2$ une formule TATL. On peut calculer l'ensemble des états de $\mathcal{A}$ où φ est satisfaite dans un temps $\mathcal{O}(\zeta \cdot |Edg|)$.

3.2.4 Résultats pour TATL et TATL$_{\leq,\geq}$

En utilisant les lemmes 3.2, 3.4, 3.6, 3.8, 3.9 et 3.10, on déduit les procédures pour traiter toutes les modalités, ce qui donne le résultat suivant :

Théorème 3.1

Le model-checking d'une formule TATL φ sur une TDCGS $\mathcal{A}$ se fait en temps $\mathcal{O}\left(|\mathcal{A}|^2 \cdot |\varphi| \cdot \zeta_{\max}\right)$, où $\zeta_{\max}$ est la constante maximale qui apparaît dans φ.

Cet algorithme est polynomial dans la taille de la TDCGS, mais Il est exponentiel à cause du codage en binaire des constantes qui apparaissent dans la formule. On va voir que cette explosion dans la complexité est inévitable. Pour montrer cela, nous définissons premièrement les countdown games.

Définition 3.2
Un countdown game *est un jeu à deux joueurs. Il consiste à avoir un graphe avec poids (V, E), avec V l'ensemble des sommets et $E \subseteq V \times \mathbb{N} \times V$ la relation de transition qui associe un poids à chaque transition. Une configuration d'un countdown game (V, E) est une paire $(v, C) \in V \times \mathbb{N}$. À chaque tour, le Joueur 1 choisit, depuis la configuration courante (v, C), une durée d t.q. (1) $0 < d \leq C$ et (2) il existe au moins une transition $(v, d, v') \in E$. Le Joueur 2 choisit alors une de ces transitions (issue de v et qui possède la durée d) et la nouvelle configuration est $(v', C - d)$. Toute configuration $(v, 0)$ est terminale et elle est une configuration gagnante pour le Joueur 1. Toute configuration (v, C) t.q. (1) $C > 0$ et (2) il n'existe aucune transition $(v, d, -)$ avec $d \leq C$ est terminale et est gagnante pour le Joueur 2.*

Ci-dessous, on donne un exemple d'un countdown game.

Exemple 3.3
La figure 3.6 montre un countdown game à trois états. On peut vérifier que ce jeu est gagnant pour le premier joueur pour la configuration $(\ell_1, 13)$, alors qu'il sera perdant pour la configuration $(\ell_1, 10)$.

Théorème 3.2 [JLS07]

Décider si le Joueur 1 possède une stratégie gagnante pour une configuration (v, C) d'un countdown game est un problème EXPTIME-complet.

Démonstration: La dureté du problème dans EXPTIME se montre par une réduction du problème d'acceptation d'un mot par une machine de Turing alternante bornée linéairement en espace. □

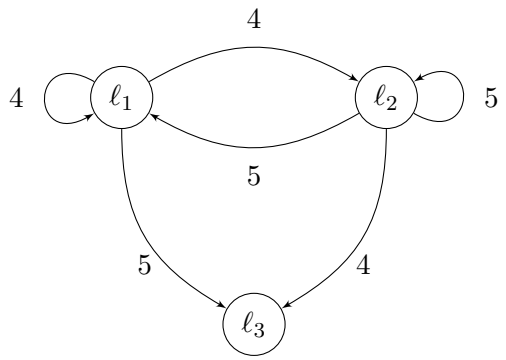


FIG. 3.6 – Le countdown game de l'exemple 3.3

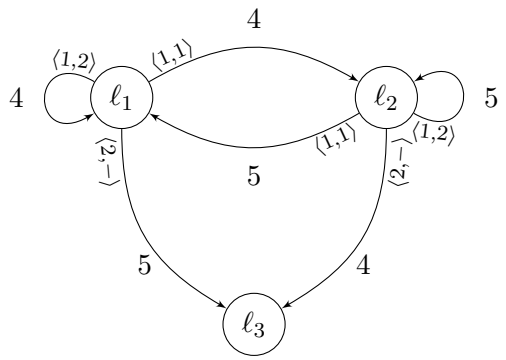


FIG. 3.7 – La TDCGS correspondante

Ayant ce résultat, on peut maintenant montrer le théorème suivant :

Théorème 3.3

Le model-checking de la logique TATL sur des TDCGS est EXPTIME-complet.

Démonstration: La preuve est basée sur la réduction du problème de décision des countdown games vers des TDCGS.

On construit la TDCGS $\mathcal{A} = \langle V, \mathsf{AP}, \mathsf{Lab}, \delta_{\mathbb{N}}, \{A_1, A_2\}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg}\rangle$ qui correspond à un countdown game (V, E) de la manière suivante : Pour tout état $\ell \in V$, posons l'ensemble des transitions issues de ℓ de durée d qu'on note $F_{\ell,d} = \{(\ell, d, \ell') \mid (\ell, d, \ell') \in E\}$ et soit $F_\ell = \{F_{\ell,d} \mid d \in \mathbb{N}\}$ l'ensemble des $F_{\ell,d}$ pour toute durée d. Le nombre de choix possible pour le premier joueur est $\mathsf{Mov}(\ell, A_1) = |F_\ell|$ et l'ensemble des choix possibles pour le deuxième joueur est donné par $\mathsf{Mov}(\ell, A_2) = \max_{d\in\mathbb{N}} |F_{\ell,d}|$. Pour chaque durée d, on numérote les transitions qui portent cette durée avec l'action $1 \leq c_d \leq \mathsf{Mov}(\ell, A_1)$ ($c_d \neq c_{d'}$ si $d \neq d'$), et on numérote de 1 jusqu'à $\mathsf{Mov}(\ell, A_2)$ chaque transition $(\ell, d, \ell') \in E$ partante de ℓ et ayant comme durée d[1], on construit la transition correspondante dans $\mathcal{A}$ tel que, $\mathsf{Edg}(\ell, \langle c_d, c_2\rangle) = (\ell', d)$. Et donc, décider si une configuration (v, C) avec $C \in \mathbb{N}$ est gagnante pour le Joueur 1 se réduit au problème du model-checking : $v \models_{\mathcal{A}} \langle\langle A_1\rangle\rangle \mathbf{F}_{=C} \top$.

□

Exemple 3.4
Ainsi, si on revient à l'exemple 3.3, la figure 3.7 montre la TDCGS correspondante à la réduction de ce countdown game. Dans cet exemple, $\ell_1 \models \langle\langle A\rangle\rangle \mathbf{F}_{=13} \top$*, et on peut donc déduire que la configuration* $(\ell_1, 13)$ *est gagnante pour le premier joueur dans le countdown game.*

L'algorithme qu'on a présenté est exponentiel, or, on peut avoir un algorithme efficace si on se restreint au fragment de la logique $\mathsf{TATL}_{\leq,\geq}$:

[1] si le nombre de transitions ayant la durée d est inférieur à $\mathsf{Mov}(\ell, A_2)$, on numérote la dernière transition avec les numéros restants

Théorème 3.4

Le model-checking d'une formule $\mathsf{TATL}_{\leq,\geq}$ φ sur une TDCGS $\mathcal{A}$ se fait en temps $\mathcal{O}(|\mathcal{A}|^2 \cdot |\varphi|)$, et donc, dans PTIME.

C'est une conséquence immédiate des lemmes 3.2, 3.4, 3.2 et 3.8. La dureté en PTIME suit le fait que le model-checking de la logique CTL sur les structures de Kripke est PTIME-complet. On obtient alors :

Corollaire 3.1
Le model-checking de la logique $\mathsf{TATL}_{\leq,\geq}$ sur les TDCGS est PTIME*-complet.*

3.2.5 Les TDCGS unitaires

Les TDCGS unitaires sont des TDCGS où toutes les durées sont égales à 1. Intuitivement, les TDCGS unitaires sont plus faciles à manipuler que les TDCGS en utlisant l'équivalence $\langle\langle A\rangle\rangle \varphi_1 \mathbf{U}_{=\zeta} \varphi_2 \equiv \langle\langle A\rangle\rangle \varphi_1 \mathbf{U}_{=\zeta/2} (\langle\langle A\rangle\rangle \varphi_1 \mathbf{U}_{=d-\zeta/2} \varphi_2)$ pour tout $d < \zeta$. De cette manière, la complexité de l'algorithme du model-checking des formules de la forme $\mathbf{U}_{=\zeta}$ devient de l'ordre de $\mathcal{O}(\log(\zeta) \cdot |\mathcal{A}|)$. On obtient donc :

Théorème 3.5

Le model-checking d'une formule φ de la logique TATL sur une TDCGS unitaire $\mathcal{A}$ se fait en temps $\mathcal{O}(|\mathcal{A}|^2 \cdot |\varphi|)$, et donc PTIME-complet.

3.3 Les CGSs avec durée

On propose dans cette section une extension des modèles qu'on vient d'introduire au début de ce chapitre. Dans cette extension, ces nouveaux modèles permettent d'avoir des transitions étiquetées avec des intervalles (au lieu d'avoir un seul entier). De cette manière, les agents ne connaissent pas d'avance la durée de la transition. Des agents spéciaux, un par transition, vont décider la durée. Ce qui nous donne un cadre plus riche dans lequel on peut avoir des agents "classiques" avec des agents "de temps".

3.3.1 Syntaxe et sémantique

On désigne par $\mathcal{I}$ l'ensemble des intervalles dans $\mathbb{N}^{>0} \cup \{+\infty\}$.

Définition 3.3
Une CGS avec durée *(Durational CGS, ou DCGS) est un* 8*-uplet* $\mathcal{S} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta_{\mathcal{I}}, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg}\rangle$ *telle que :*

- $\langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta_{\mathcal{I}}\rangle$ *est un système de transitions pondéré fini où associe à chaque transition un intervalle de durée ;*
- $\mathsf{Agt} = \{A_1, \ldots, A_k\}$ *est l'ensemble des agents du système ;*
- $\mathbb{M}$ *est l'ensemble d'actions possibles ;*

- $\mathit{Mov} : \mathit{Loc} \times \mathit{Agt} \to \mathcal{P}(\mathbb{M}) \setminus \{\varnothing\}$ *définit l'ensemble non-vide d'actions possibles pour chaque agent dans chaque état ;*
- $\mathit{Edg} : \mathit{Loc} \times \mathbb{M}^k \to \delta_{\mathcal{I}}$ *est une fonction partielle qui définit la table de transitions ;*
- *Cette CGS comporte des agents implicites (non inclus dans* Agt*), on appelle ces agents des agents de temps.*

La taille de la table de transitions est encore l'espace nécessaire pour l'écrire en binaire, et la taille d'une DCGS est $|\mathsf{Loc}| + |\mathsf{Edg}|$. Là encore, on utilise $\mathsf{Edg}_\tau(\ell, c)$ pour noter l'intervalle de durées de la transition $\mathsf{Edg}(\ell, c)$, et $\mathsf{Edg}_{\mathsf{Loc}}(\ell, c)$ pour noter son état d'arrivée.

La syntaxe ne semble pas très différente de celle des TDCGS mais la différence est que les agents doivent choisir une transition qui doit être tirée par le système, ainsi qu'une durée pour cette transition dans l'intervalle correspondant. Ceci est accompli à l'aide des "agents de temps" : On considère un agent de temps $ta_{\ell,c}$ par états ℓ et par action c dans $\mathsf{Mov}(\ell, \mathsf{Agt})$. Formellement, la sémantique d'une DCGS $\mathcal{S}$ est définie par la TDCGS $\mathcal{A}[\mathcal{S}] = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta_{\mathbb{N}}, \mathsf{Agt}', \mathbb{M}', \mathsf{Mov}', \mathsf{Edg}' \rangle$ telle que :

- $\mathbb{M}' = \mathbb{N} \cup \mathbb{M}$
- $\mathsf{Agt}' = \mathsf{Agt} \cup \{ta_{\ell,c} \mid \ell \in \mathsf{Loc} \text{ et } c \in \mathsf{Mov}(\ell, \mathsf{Agt})\}$,
- $\mathsf{Mov}'(\ell, a) = \mathsf{Mov}(\ell, a)$ pour tout $a \in \mathsf{Agt}$; $\mathsf{Mov}'(\ell, ta_{\ell,c}) = \mathsf{Edg}_\tau(\ell, c)$ pour tout $ta_{\ell,c} \in \mathsf{Agt}'$, et $\mathsf{Mov}'(\ell, ta_{\ell',c}) = \{0\}$ pour tout $ta_{\ell,c} \in \mathsf{Agt}'$ avec $\ell' \neq \ell$,
- $\mathsf{Edg}'(\ell, c, t_{\ell_0,c_0}, \ldots, t_{\ell_n,c_m}) = (\ell, t, \ell')$ ssi $c \in \mathsf{Mov}'(\ell, \mathsf{Agt})$ pour tout i, $t = t_{\ell,c}$ et $t_{\ell,c} \in \mathsf{Mov}'(\ell, ta_{\ell,c})$, et $t_{\ell',c'} = 0$ quand $\ell' \neq \ell$ ou $c \neq c'$.

Comme dans le cas des TDCGSs, on utilise les mêmes notions de coalition d'agents (qui comprennent éventuellement des agents de temps), les stratégies et les outcomes.

On note que la table de transitions qui correspond à la TDCGS $\mathcal{A}[\mathcal{S}]$ est infinie quand il existe des intervalles infinis dans la définition de $\mathcal{S}$. Et quand il est fini, la taille $|\mathsf{Edg}'|$ est bornée par $|\mathsf{Edg}| \cdot b_M$ où b_M est la constante maximale qui figure dans les intervalles des durées dans $\mathcal{S}$: dans Edg', on remplace chaque entrée (ℓ, c) de Edg par $(b - a) + 1$ entrées quand $\mathsf{Edg}_\tau(\ell, c) = [a; b]$. Donc la taille de $\mathcal{A}[\mathcal{S}]$ est potentiellement exponentielle dans $|\mathcal{S}|$ (puisqu'on utilise le codage en binaire). Noter que dans chaque entrée de Edg', un seul agent de temps peut avoir plusieurs choix.

Les agents de temps peuvent être utilisés dans les modalités de TATL pour exprimer l'existence des stratégies pour des coalitions qui peuvent contrôler les durées de certaines transitions. Étant donnés une DCGS $\mathcal{S}$, un état ℓ, et une formule φ, on écrit $\ell \models_{\mathcal{S}} \varphi$ quand $\ell \models_{\mathcal{A}[\mathcal{S}]} \varphi$. On supprime l'indice quand il n'y a pas d'ambiguïté.

Exemple 3.5
Prenons l'exemple du jeu de Nim simplifié qui illustre ces modèles. Dans ce jeu, on dispose d'un ensemble de N allumettes alignées sur une table, et chaque joueur, à son tour, ramasse une à trois allumettes. Le joueur qui prend la dernière allumette gagne le jeu. Ce jeu pourrait être représenté facilement à l'aide des DCGS (où les "durées" vont représenter le nombre d'allumettes pris par les joueurs), comme l'illustre la figure 3.8. Le joueur A gagne ssi la formule $\langle\!\langle A, ta_{A,\langle 1,1\rangle} \rangle\!\rangle \mathbf{F}_{=N} B$ *est satisfaite.*

Les agents de temps. Au lieu de prendre le côté d'un agent donné en le donnant le contôle du choix des durées dans les transitions, deux solutions plus générales sont possibles : associer un agent de temps pour chaque état du système, ou bien avoir un agent de temps pour chaque

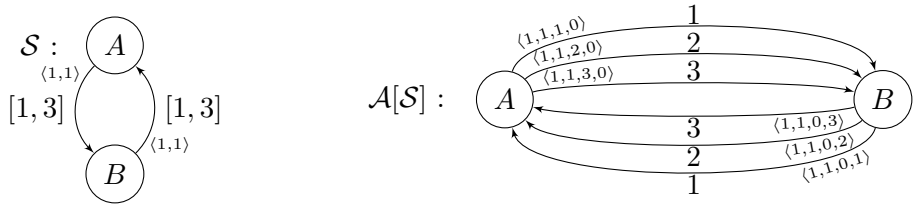


FIG. 3.8 – Une DCGS pour le jeu de Nim simplifié, et la TDCGS correspondante.

transition du système. Ces agents pourront être utilisés comme n'importe quel autre agents dans les coallitions lors de la vérification des formules.

Clairement, avoir un agent de temps par transition est plus général qu'avoir un agent de temps par état : En effet, dans la première approche, un agent de temps ta_ℓ (qui contrôle les durées de toutes les transitions issues de ℓ) peut être simulé facilement par la coalition $\{ta_{\ell,c_1}, \ldots, ta_{\ell,c_m}\}$ qui contient tous les agents de temps des transitions issues de ℓ. Pour cette raison, nous avons adopté la définition où un agent de temps est donné pour chaque transition.

3.3.2 Le model-checking de la logique $\mathsf{TATL}_{\leq,\geq}$

La vérification des formules TATL qui contient des modalités avec des contraintes de la forme "$\leq c$" ou "$\geq c$" ne nécessite pas de considérer toutes les transitions de $\mathcal{A}[\mathcal{S}]$. On se restreint à l'analyse d'une abstraction de $\mathcal{A}[\mathcal{S}]$:

Définition 3.4
Soient $\mathcal{S} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg}\rangle$ une DCGS et $\mathcal{A}[\mathcal{S}] = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta', \mathsf{Agt}', \mathbb{M}', \mathsf{Mov}', \mathsf{Edg}'\rangle$ la TDCGS correspondante à la sémantique de $\mathcal{S}$. Étant donné un entier C, on définit la C-abstraction de $\mathcal{S}$ comme étant la TDCGS $\mathcal{A}[\mathcal{S}]_C = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta', \mathsf{Agt}', \mathsf{Mov}'', \mathsf{Edg}''\rangle$ avec :
- *$\mathsf{Mov}''(\ell, ta_{\ell,c})$ est $\{a, b\}$ (resp. $\{a; C+1\}$) si $\mathsf{Edg}_\tau(\ell, c) = [a, b]$ (resp. $\mathsf{Edg}_\tau(\ell, c) = [a, +\infty)$) ; et Mov'' coïncide avec Mov' pour les autres cas ($a \in \mathsf{Agt}$ ou $ta_{\ell',c}$ avec $\ell' \neq \ell$).*
- *Edg'' est définie par Edg' mais en remplaçant Mov' par Mov''.*

Dans la TDCGS $\mathcal{A}[\mathcal{S}]_C$, on remplace une transition avec un intervalle de temps λ par deux transitions avec durées fixes : une transition correspondant à la valeur gauche de λ, et une transition qui correspond à la valeur droite de λ quand λ est fini, sinon elle sera remplacé par une transition portant la valeur $C+1$ (ou par la valeur gauche de λ si $C+1 \notin \lambda$). Effectivement, un intervalle ouvert, est intéressant pour la valeur de vérité de quelques propriétés puisqu'il peut laisser des durées arbitrairement longues. Mais des délais de $C+1$ unités de temps sont toujours suffisant pour traiter des formules de $\mathsf{TATL}_{\leq,\geq}$ ayant des constantes inférieures à C :

Lemme 3.11
Soient $\mathcal{S}$ une DCGS et C un entier. Pour tout $\zeta \leq C$ et $\ell \in \mathsf{Loc}$, on a :

$$\ell \models_{\mathcal{A}[\mathcal{S}]} \langle\!\langle A\rangle\!\rangle\, P_1\, \mathbf{U}_{\leq\zeta}\, P_2 \quad \Leftrightarrow \quad \ell \models_{\mathcal{A}[\mathcal{S}]_C} \langle\!\langle A\rangle\!\rangle\, P_1\, \mathbf{U}_{\leq\zeta}\, P_2 \tag{3.4}$$

$$\ell \models_{\mathcal{A}[\mathcal{S}]} \langle\!\langle A\rangle\!\rangle\, P_1\, \mathbf{U}_{\geq\zeta}\, P_2 \quad \Leftrightarrow \quad \ell \models_{\mathcal{A}[\mathcal{S}]_C} \langle\!\langle A\rangle\!\rangle\, P_1\, \mathbf{U}_{\geq\zeta}\, P_2 \tag{3.5}$$

$$\ell \models_{\mathcal{A}[\mathcal{S}]} \langle\!\langle A\rangle\!\rangle\, P_1\, \mathbf{R}_{\leq\zeta}\, P_2 \quad \Leftrightarrow \quad \ell \models_{\mathcal{A}[\mathcal{S}]_C} \langle\!\langle A\rangle\!\rangle\, P_1\, \mathbf{R}_{\leq\zeta}\, P_2 \tag{3.6}$$

$$\ell \models_{\mathcal{A}[\mathcal{S}]} \langle\!\langle A\rangle\!\rangle\, P_1\, \mathbf{R}_{\geq\zeta}\, P_2 \quad \Leftrightarrow \quad \ell \models_{\mathcal{A}[\mathcal{S}]_C} \langle\!\langle A\rangle\!\rangle\, P_1\, \mathbf{R}_{\geq\zeta}\, P_2 \tag{3.7}$$

Démonstration: Le nombre de comportements possibles dans $\mathcal{A}[\mathcal{S}]$ est supérieur à celui dans $\mathcal{A}[\mathcal{S}]_C$, mais les exécutions supplémentaires ne changent pas la valeur de vérité des formules $\mathsf{TATL}_{\leq,\geq}$.

En effet, supposons que $\rho = (\ell_0, d_0, \ell_1)\dots(\ell_i, d_i, \ell_{i+1})\dots$ soit une exécution dans $\mathcal{A}[\mathcal{S}]$ et soit c_i^a (resp. c_i^t) la $i+1$-ème action des agents Agt (resp. les agents de temps) dans ρ [2]. On peut changer l'action de l'agent ta_{ℓ_i,c_i} pour obtenir une autre exécution ρ' avec le *même préfixe et le même suffixe* que ρ : la durée passée dans l'état ℓ_i a changé (et donc les dates globales des actions) mais pas la durée passée dans les autres états. Cette propriété nous permet de faire des changements locaux dans les délais sans affecter la séquence des états visités.

Considérons la stratégie σ_A pour la coalition A dans $\mathcal{A}[\mathcal{S}]$ pour assurer $\psi = \mathsf{P}_1\,\mathbf{U}_{\leq\zeta}\,P_2$. En partant de σ_A, on peut construire une stratégie σ'_A qui assure ψ dans $\mathcal{A}[\mathcal{S}]_C$. En effet, le seul changement qu'on doit faire c'est avec les choix des agents de temps quand, dans l'état ℓ avec une action c pour la coalition Agt, σ_A nécessite d'attendre une durée de $t_{\ell,c}$ unités de temps lorsque $t_{\ell,c}$ n'est pas dans les choix restreints de $\mathcal{A}[\mathcal{S}]_C$ (*i.e.* $t_{\ell,c} \notin \mathsf{Mov}''(\ell, ta_{\ell,c})$). Dans ce cas, la stratégie σ'_A peut proposer la durée minimale dans $\mathsf{Mov}''(\ell, ta_{\ell,c})$: l'état final satisfaisant ψ va être accessible plus tôt que dans ρ et alors ψ va être satisfaite. Si la formule à vérifier est de la forme $\mathbf{U}_{\geq c}$ la durée maximale devrait être choisie pour garantir que la formule va être satisfaite. On traite l'opérateur release d'une manière similaire.

Considérons une stratégie σ'_A pour la coalition A dans $\mathcal{A}[\mathcal{S}]_C$ pour garantir la formule $\psi = \mathsf{P}_1\,\mathbf{U}_{\leq\zeta}\,P_2$. On complète cette stratégie pour garantir φ dans la TDCGS complète : Considérons une exécution finie ρ dans $\mathcal{A}[\mathcal{S}]$, on peut définir une exécution correspondante $\bar{\rho}$ dans $\mathcal{A}[\mathcal{S}]_C$ où tout choix de l'agent de temps $ta_{\ell,c}$ depuis l'état ℓ reste, soit inchangé si $ta_{\ell,c} \in A$ —c'est agent contrôlé par A, et ayant appliqué σ_A depuis le début de l'exécution garantit que son action est dans $\mathcal{A}[\mathcal{S}]_C$—, soit remplacé par la durée maximale dans $\mathsf{Mov}''(\ell, ta_{\ell,c})$ si $ta_{\ell,c}$ n'est pas un agent de temps A-contrôlable. Donc, il est suffisant de définir $\sigma_A(\rho)$ comme $\sigma'_A(\bar{\rho})$.

Maintenant, si on considère $\psi = \mathsf{P}_1\,\mathbf{U}_{\geq\zeta}\,P_2$, on construit $\bar{\rho}$ d'une manière différente et en considérant les durées minimales. □

On note que la taille de $\mathcal{A}[\mathcal{S}]_C$ est bornée par $2 \cdot |\mathcal{S}|$. On obtient donc :

Théorème 3.6

Le model-checking de la logique $\mathsf{TATL}_{\leq,\geq}$ sur les DCGS est PTIME-complet.

3.3.3 Le model-checking de la logique TATL

Pour la logique TATL, on réduit le problème à celui des TDCGS finies. Étant donnée une DCGS $\mathcal{S}$ et une formule $\varphi = \langle\!\langle A \rangle\!\rangle\, P_1\,\mathbf{U}_{=\zeta}\,P_2$, où on ajoute explicitement un agent supplémentaire par transition, avec les actions dans $[a,b]$ si la transition correspondante est étiquetée par $[a,b]$, et les actions dans $[a, \max(a, \zeta+1)]$ si elle est étiquetée par $[a,+\infty)$. À l'aide de cette restriction notre TDCGS correspondante est finie, sa taille est de l'ordre de $\mathcal{O}(|\mathcal{S}|\cdot\max(b_M,\zeta))$ avec b_M l'entier maximal qui apparaît dans les bornes des intervalles dans $\mathcal{S}$. Et en appliquant l'algorithme du théorème 3.1 à cette TDCGS, on obtient un algorithme EXPTIME pour le model-checking

[2] *i.e.* $(\ell_i, t_{i+1} - t_i, \ell_{i+1}) = \mathsf{Edg}'(\ell_i, c_i^a \oplus c_i^t)$ avec $c_i^a \oplus c_i^t \in \mathsf{Mov}(\ell_i, \mathsf{Agt}')$.

d'une formule φ sur notre DCGS $\mathcal{S}$. Cet algorithme est similaire pour la modalité release. On répète cette procédure un nombre polynomial de fois pour vérifier une formule TATL, pour obtenir un algorithme en temps $\mathcal{O}(|\mathcal{S}|^2 \cdot b_M \cdot \zeta_{\max} \cdot |\varphi|)$ (avec $\zeta_{\max}$ la constante maximale dans la formule φ), et donc, en temps exponentiel dans la taille de la structure et de la formule.

On note que la dureté du problème de model-checking est une conséquence d'avoir codé les entiers en binaires. On obtient alors le théorème suivant qui est une conséquence immédiate de ce résultat (avec celui du théorème 3.3)

Théorème 3.7

Le model-checking de la logique TATL sur les DCGS est EXPTIME-complet.

Conclusion

Les modèles introduits dans ce chapitre représentent une extension des CGS non-temporisées. Les transitions dans les TDCGS portent des durées entières positives représentant le temps écoulé lorsqu'une transition est franchie. Pour ces modèles nous utilisons la logique TATL pour vérifier des propriétés quantitatives sur les stratégies des joueurs. Nous avons montré que la complexité du model-checking de la logique TATL sur les TDCGS est EXPTIME-complet, mais cette explosion est liée au codage en binaire des constantes. Conséquemment, les TDCGS sont étendues pour avoir des intervalles d'entiers sur les transitions (c'est ce qu'on note par DCGS). Et puisqu'il s'agissait des systèmes à plusieurs agents, une possibilité était de considérer implicitement des agents supplémentaires dans la sémantique dont le rôle est de décider la durée à prendre dans chaque transition. La complexité du model-checking est inchangée dans ce cas pour la logique TATL et reste EXPTIME-complet.

On note que ces modèles bénéficient d'un algorithme de model-checking efficace quand les propriétés à vérifier ne comprennent pas des contraintes avec égalités ($= \zeta$), et le problème de model-checking sera désormais PTIME-complet.

L'étude des modèles quantitatifs des CGS ouvre la voie pour une étude plus approfondie de l'aspect temporisé. Effectivement, le chapitre suivant est dédié à l'aspect temps dense des systèmes de jeux. Nous allons définir une sémantique proche de celle des automates temporisés.

Pour en savoir plus...

Notre problème est lié au μ-calcul quantitatif qu'on peut trouver dans [dAFS04] cette logique est étudiée sur des systèmes de transitions métriques. On trouve aussi que ce formalisme a été étendu avec les jeux, il a été montré dans [FGK08] qu'une correspondance dans les deux sens entre les jeux de parité quantitatifs et le μ-calcul quantitatif.

Les jeux que nous avons présentés s'intéressent aux comportements finis des exécutions. Par ailleurs, on trouve dans la littérature des systèmes similaires mais qui traitent des comportements infinis d'une manière quantitative. Les jeux mean-payoff [CHJ05, ZP96], par exemple, consistent en une structure à deux joueurs, où un joueur cherche à maximiser la valeur moyenne du jeu et le but de l'autre est de la minimiser. Ces jeux sont des structures turn-based, les

transitions portent un poids comme dans le cas des TDCGS, et on cherche à savoir si la valeur moyenne des exécutions infinies est supérieure à une valeur donnée. Il a été montré que ce problème est dans $\mathsf{UP} \cap \mathsf{coUP}$, et qu'il est lié aux jeux de parité [CHJ05, Jur98].

Chapitre 4 Les CGS temporisées

« How amazing time is, and how amazing we are. Time has been transformed, and we have changed ; it has advanced and set us in motion ; it has unveiled its face, inspiring us with bewilderment and exhilaration.
Yesterday we complained of time and feared it, but today we love and embrace it. Indeed, we have begun to perceive its purposes and characteristics, and to comprehend its secrets and enigmas.»

Khalil Gibran, *The Vision* (1883–1931)

4.1 Introduction

Les systèmes que nous avons étudiés jusqu'à maintenant, n'ont pas représenté le temps d'une manière continue, c'est l'objet de ce chapitre. Or, dans la littérature, il existe beaucoup de travaux théoriques et appliqués sur ce sujet. Ces recherches ont donné naissance à de nombreuses théories qui ont contribué avec succès dans des outils dans le milieu industriel.

En particulier, les *automates temporisés* sont considérés comme un des formalismes les plus intéressants pour traiter les systèmes manipulant des *aspects temps-réel*. Ce formalisme consiste en un système de transitions enrichi avec des *horloges*. Les valeurs de ces horloges, à valeur dans $\mathbb{R}$, vont imposer des restrictions sur le comportement du modèle, notamment, les transitions vont contenir des *gardes* qui décrivent quand une transition peut être exécutée, lors du franchissement d'une transition. Des *invariants* dans chaque états exigent des contraintes sur les valeurs des horloges. Pour spécifier des propriétés sur le comportement de ces modèles, plusieurs logiques ont été proposées comme TPTL et TCTL qui étendent respectivement les logiques non temporisées LTL et CTL. Ces deux logiques comprennent des horloges de formule à l'aide desquelles on peut avoir des propriétés quantitatives sur les délais séparant les actions. Considérons la modélisation d'un canal de communication, à l'aide d'un automate temporisé nous pouvons maintenant préciser le temps qu'un message met pour arriver à sa destination. De plus, nous avons la possibilité de vérifier des propriétés du genre : Chaque message envoyé est suivi d'un accusé de réception dans un intervalle de 5 secondes. Cette propriété sera exprimée par la formule TCTL suivante : $\mathbf{AG}\,(\mathtt{msg} \rightarrow \mathbf{AF}_{\leq 5}\,\mathtt{ack})$.

On note que les automates temporisés ont fait l'objet d'une énorme étude théorique dans les vingt dernières années [AD90, AH94, AD94, AH92, ACD93, HNSY94] dans lesquelles les algorithmes de model-checking reposent sur une équivalence des valuations des horloges, ap-

pelée *équivalence des régions*. Cette méthode nous permet de construire un automate fini qui est "équivalent" (pour la propriété à vérifier) à l'automate temporisé d'origine. Et ces études ont donné naissance à plusieurs outils [LPY97, BDM+98, DOTY96].

Par ailleurs, l'aspect "jeu" n'a été ajouté aux systèmes temporisés que récemment, les *automates à jeux temporisés* sont parmi les premiers modèles concernés [MPS95, AMPS98, AM99, dAFH+03, CDF+05, HP06]. Ce genre de modèle est considéré comme étant un automate temporisé dont les transitions ont été partitionnées entre les différents agents du système. À chaque étape du jeu, chaque joueur va choisir une transition et un délai qu'il souhaite attendre avant d'exécuter la transition, le joueur ayant le délai le plus petit est choisi, et son choix sera exécuté (la transition choisie et le délai). Quand on se retrouve face à plusieurs délais égaux, le joueur sera choisi d'une manière non déterministe ou bien en précisant à l'avance une hiérarchie des joueurs, mais la symétrie sera cassée dans ce cas. De même, et dans ce contexte, la logique ATL a été étendue pour traiter des propriétés temporisées. Les formules maintenant comprennent des horloges pour pouvoir exprimer des propriétés temporisées. Le model-checking de cette logique sur les automates de jeux temporisés est un problème EXPTIME-complet [HP06, dAFH+03].

Dans ce chapitre, nous regardons le problème des jeux temporisés sous un autre angle. Nous avons vu que pour obtenir les automates à jeux temporisés [dAFH+03], nous avons étendu les automates temporisés avec des jeux. Maintenant, nous attaquons ce problème d'une manière différente, ayant une structure de jeu concurrente, nous allons l'étendre pour avoir du temps réel pour obtenir finalement des structures temporisés avec des jeux "plus" concurrents appelées *TCGS*. Une action pour un joueur devient alors, un délai où il souhaite exécuter son action, et une fonction qui associe un coup (ou bien un mouvement) pour toute durée. La transition sera exécutée au délai minimum choisi par les joueurs, et la transition sera prise par les actions données par les fonctions de chaque joueur prises au délai minimum. On peut maintenant représenter l'exemple des canaux de communication en se servant de cette nouvelle approche. Depuis l'état de départ, un joueur souhaite envoyer son message au temps d_1, l'autre joueur choisit de l'envoyer au temps d_2, et le choix du joueur avec durée minimum sera exécuté. Le cas où les deux joueurs ont fait le même choix de la date d'envoi nous mène vers une collision (un autre état du système), cela représente mieux ce qui se passe dans le cas réel. Alors que dans une modélisation à l'aide des automates à jeux temporisés, un message sera envoyé en choisissant un joueur d'une manière non déterministe.

Les résultats dans chapitre sont basés sur le papier [BLMO07]

4.2 Définitions

Nous donnons dans cette section une définition modifiée des automates temporisés. Ensuite, et en se basant sur les CGS pondérées continues présentées dans le chapitre 1, nous présentons les CGS temporisées, qui combinent les CGS et les automates temporisés.

4.2.1 Les automates temporisés

La définition des CGS temporisées est basée sur celle des automates temporisés [AD94]. Pour cette raison, on consacre cette partie pour décrire ces modèles.

Les automates temporisés ont été définis au début des années 90. Ces machines représentent des automates classiques augmentés avec des horloges qui évoluent d'une manière continue avec le temps. La version des automates temporisés que nous définissons est légèrement différente des automates définis dans [AD94]. Nous présentons notre définition des automates temporisés, ensuite on expliquera comment adapter le modèle des CGS avec des aspects temps-réel.

Tout d'abord, on rappelle quelques notions :

Étant donné un ensemble $\mathcal{C}$ de *variables d'horloge*, une *valuation* est une fonction $v\colon \mathcal{C} \to \mathbb{R}^+$ (notée aussi par $v \in (\mathbb{R}^+)^{\mathcal{C}}$). Étant donnés une valuation v, un délai $t \in \mathbb{R}^+$ et sous-ensemble $Z \subseteq \mathcal{C}$, la valuation $v' = v+t$ est définie par $v'(c) = v(c)+t$ pour tout $c \in \mathcal{C}$, et la valuation $v'' = v[Z \leftarrow 0]$ est définie par $v''(c) = v(c)$ si $c \notin Z$ et $v''(c) = 0$ sinon. On écrit v_0 pour la valuation t.q. $v_0(c) = 0$ pour tout $c \in \mathcal{C}$.

Définition 4.1
L'ensemble des contraintes d'horloge *est l'ensemble des formules définies par la grammaire suivante :*

$$\mathsf{Constr}(\mathcal{C}) \ni \varphi ::= c \sim n \mid \varphi \wedge \varphi$$

où c appartient à $\mathcal{C}$, n est un entier positif et $\sim \in \{<, \leq, =, \geq, >\}$. On note par $v \models \varphi$ lorsque la valuation v vérifie la contrainte φ.
On définit $\top \stackrel{def}{=} c \geq 0$ et $\bot \stackrel{def}{=} c < 0$ (pour $c \in \mathcal{C}$) comme des abbréviations naturelles.

Définition 4.2
Un automate temporisé *(ou TA) est un 6-uplet $\mathcal{A} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathcal{C}, \mathsf{Inv} \rangle$ avec :*
- *$\langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta \rangle$ est un système de transitions (la relation de transition δ est spéciale dans notre cas, on donne la définition ci-dessous) ;*
- *$\mathcal{C}$ est un ensemble fini de variables d'horloge ;*
- *$\mathsf{Inv}\colon \mathsf{Loc} \to \mathsf{Constr}(\mathcal{C})$ définit les invariants dans chaque état ;*
- *$\delta \subseteq \mathsf{Loc} \times (\mathsf{Constr}(\mathcal{C}), \mathsf{Loc}, 2^{\mathcal{C}})^*$ est un ensemble fini de transitions.*

Pour que le système ne comprenne pas d'états bloquants, on demande que pour toute transition $(q, (\Phi_i, q_i, Z_i)_{0 \leq i \leq p}) \in \delta$, on a toujours $(\Phi_p, q_p, Z_p) = (\top, q, \varnothing)$.

La sémantique continue des TA est définie en termes d'un système (infini) de transitions pondéré. Dans notre définition, nous combinons les transitions de délai et celles des actions :

Définition 4.3
Avec un TA $\mathcal{A} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathcal{C}, \mathsf{Inv} \rangle$, on associe un système de transitions pondéré $\mathcal{S} = \langle S, \mathsf{AP}, \mathsf{Lab}', \delta_t \rangle$ défini par :
- *$S = \{(q, v) \in \mathsf{Loc} \times (\mathbb{R}^+)^{\mathcal{C}} \mid v \models \mathsf{Inv}(q)\}$;*
- *$\mathsf{Lab}'((q, v)) = \mathsf{Lab}(q)$;*
- *$\delta_t \subseteq S \times \mathbb{R}^+ \times S$ est t.q. $(s, d, s') \in \delta_t$ ssi, ayant $s = (q, v)$ et $s' = (q', v')$:*
  - *soit $q = q'$ et $v' = v + d$;*
  - *soit il existe une transition $(q, (\Phi_i, q_i, Z_i)_{0 \leq i \leq p}) \in \delta$ et un entier $j \leq p$ tel que $v+d \models \Phi_j$ et $v + d \not\models \Phi_k$ pour tout $k < j$, $q' = q_j$ et $v' = (v + d)[Z \leftarrow 0]$.*

Les éléments de S sont appelés les états *de $\mathcal{A}$.*

On définit une exécution d'un automate temporisé comme une exécution du système de transitions sous-jacent. De plus, on note une position le long d'une exécution r par le couple $(k, d) \in \mathbb{N} \times \mathbb{R}^+$ désignant l'état $(q_k, v_k + d)$.

Remarque 4.1
Cette définition des automates temporisés est légèrement différente de la définition classique. Dans notre syntaxe, une transition choisie est toujours *franchissable. Notre définition est plus adaptée au contexte des jeux quand un choix mène vers une transition à franchir.*

4.2.2 Les CGS temporisées

Dans les CGS temporisées, le choix d'un joueur consiste en une fonction de coups, qui donne pour une durée réelle positive un coup possible, et une date où le joueur souhaite exécuter son action.

Définition 4.4
L'ensemble des contraintes de choix *sur l'ensemble des agents* Agt *et l'ensemble de coups* $\mathbb{M}$ *est défini par :*

$$\mathsf{SymbConstr}(\mathbb{M}) \ni \varphi ::= \top \mid \tau_i \leq \tau_j \mid f_i(\tau_{\min}) = m \mid \varphi \wedge \varphi \mid \neg\varphi$$

avec $m \in \mathbb{M}$, τ_i *est la durée choisie par le joueur* i *et* $\tau_{\min} = \min_{i \in \mathsf{Agt}} \tau_i$.

Définition 4.5
Une structure de jeu concurrent temporisée *(Timed CGS, ou TCGS simplement) est un* 10-*uplet* $\mathcal{T} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathcal{C}, \mathsf{Inv}, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg} \rangle$ *avec*

- $\langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathcal{C}, \mathsf{Inv} \rangle$ *est un TA ;*
- $\mathsf{Agt} = \{a_1, \ldots, a_k\}$ *est un ensemble fini d'agents ;*
- $\mathbb{M}$ *est l'ensemble de coups possibles ;*
- $\mathsf{Mov}\colon \mathsf{Loc} \times \mathsf{Agt} \to \mathcal{P}(\mathbb{M}) \setminus \varnothing$ *indique l'ensemble de choix possibles pour chaque agent dans chaque état ;*
- $\mathsf{Edg}\colon \mathsf{Loc} \to (\mathsf{SymbConstr}(\mathbb{M}) \times \delta)^*$ *est la table de transitions qui satisfait pour tout* $q \in \mathsf{Loc}$ *que* $\mathsf{Edg}(q) = ((\psi_i, tr_i)_{0 \leq i \leq p})$ *alors* $\psi_p = \top$.

Dans ce chapitre, on s'intéresse au cas où les ensembles Loc et $\mathbb{M} \subseteq \mathbb{N}$ sont finis.

Dans une TCGS, les agents ne choisissent pas seulement le délai et l'action, mais aussi les actions qu'ils auraient jouées si la transition était prise avant le délai souhaité. Formellement :

Définition 4.6
Soit $\mathcal{T} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathcal{C}, \mathsf{Inv}, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg} \rangle$ *une TCGS. Soient* $q \in \mathsf{Loc}$, *et* $v \in (\mathbb{R}^+)^{\mathcal{C}}$. *Un* choix complet *d'un joueur* $a \in \mathsf{Agt}$ *depuis l'état* (q, v) *est une paire* (t, f) *avec* $t \in \mathbb{R}^+$ *et* $f\colon \mathbb{R}^+ \to \mathsf{Mov}(q, a)$ *t.q.* $v + t \models \mathsf{Inv}(q)$. *On écrit* $\mathsf{FM}((q, v), a)$ *pour l'ensemble des choix complets du joueur* a *dans l'état* (q, v). *On a*

$$\mathsf{FM}((q, v), a) = \{t \in \mathbb{R}^+ \mid v + t \models \mathsf{Inv}(q)\} \times (\mathsf{Mov}(q, a))^{\mathbb{R}^+}.$$

On utilise la notation $\mathsf{FM}(\mathbb{R}^+, \mathbb{M})$ *pour désigner l'ensemble* $\mathbb{R}^+ \times \mathbb{M}^{\mathbb{R}^+}$ *de tous les choix complets possibles.*

On définit maintenant la sémantique d'une TCGS en l'associant à une CGS infinie.

Définition 4.7
Pour une TCGS $\mathcal{T} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathcal{C}, \mathsf{Inv}, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg} \rangle$, *on associe la CGS infinie* $\mathcal{S} = \langle S, \mathsf{AP}, \mathsf{Lab}', R, \mathsf{Agt}, \mathsf{FM}(\mathbb{R}^+, \mathbb{M}), \mathsf{Mov}', \widetilde{\mathsf{Edg}} \rangle$ *définie par :*

- $\langle S, AP, Lab', R\rangle$ *est le système de transitions pondéré associé avec le TA* $\langle Loc, AP, Lab, \delta, \mathcal{C}, Inv\rangle$;
- *pour tout* $(q, v) \in S$ *et tout* $a \in Agt$, $Mov'((q, v), a) = FM((q, v), a)$;
- $\widetilde{Edg} : S \times (FM(\mathbb{R}^+, \mathbb{M}))^{|Agt|} \to R$ *est la table de transitions. Pour un état* (q, v) *et un choix complet* $(t_l, f_l)_{l\in Agt}$. $\widetilde{Edg}((q, v), (t_l, f_l)_{l\in Agt}) = \delta_i$ *si* $Edg(q) = ((\psi_i, tr_i)_{0\leq i\leq p})$ *et il existe un entier* $j \leq p$ *tel que* $(t_l, f_l)_{l\in Agt} \models \psi_j$ *et pour tout* $k < j$ *on a* $(t_l, f_l)_{l\in Agt} \not\models \psi_k$. *La transition* δ_i *à franchir sera calculée pour* $tr_j = (q, (\Phi_m, q_m, Z_m)_{0\leq m\leq n})$, *est* $((q, v), \tau_{\min}, (q_l, v + \tau_{\min}[Z_l \leftarrow 0]))$ *si* l *est le plus petit entier tel que* $v + \tau_{\min} \models \Phi_l$. *L'état d'arrivée est donc* $(q_l, v + \tau_{\min}[Z_l \leftarrow 0])$.

4.2.3 Les structures de jeux temporisés

Nous venons de définir les TCGS comme étant un modèle général pour manipuler les structures de jeux temporisés. On s'intéresse dans cette partie à présenter d'autres types de structures considérées comme des cas particuliers de notre modèle général, et on présente, ensuite, une traduction vers nos TCGS.

4.2.3.1 Automates à jeux temporisés

Les *automates à jeux temporisés*, ou simplement les TGA (Timed Game Automata), ont été introduits dans [MPS95, dAFH+03]. Ce nouveau formalisme est obtenu en ajoutant la notion des jeux concurrents aux automates temporisés, et est un des premiers à traiter la notion de temps dense dans les jeux. Il s'agit donc d'une structure temporisée avec 2 (ou plusieurs) joueurs, les transitions dans cette structure sont réparties entre ses différents joueurs.

Ci-dessous, nous présentons formellement la définition de ces jeux, ensuite, nous présentons la méthode pour traduire ces modèles à l'aide des TCGS.

Définition 4.8 (Syntaxe)
Un automate à jeux temporisés *est un* 8*-uplet* $\mathcal{T} = \langle Loc, AP, Lab, \delta, \mathcal{C}, Act_1, Act_2, Inv\rangle$[1]

- $\langle Loc, AP, Lab, \delta, \mathcal{C}, Inv\rangle$ *est un automate temporisé dans la définition de [AD94] mais c'est équivalent à notre définition.*
- $Act_1, Act_2 \subseteq \delta$ *deux ensembles disjoints d'actions possibles pour le joueur* 1 *et le joueur* 2, *respectivement.*

La sémantique est la suivante :

Définition 4.9 (Sémantique)
Depuis un état donné, chaque joueur va proposer un délai et une transition à franchir $(d_i, \delta_i)_{i\in\{1,2\}}$, *on cherche la durée minimale* $d_{min} = \min_{i\in\{1,2\}} d_i$ *et l'action du joueur qui a choisi la durée minimale sera tirée. Et donc, l'action* δ_i *où* $d_i = d_{min}$ *sera prise au temps* d_{min} *tout en respectant la sémantique de l'automate temporisé sous-jacent. Dans le cas d'un choix égal pour les durées des deux joueurs, la transition à prendre sera choisie d'une manière non-déterministe parmi les choix des joueurs.*

[1] La définition originale a introduit, dans chaque état, deux invariants, un pour chaque joueur. Dans cette définition, un seul invariant a été utilisé, mais cela ne change pas l'expressivité des modèles puisqu'on peut augmenter le nombre d'états pour obtenir le même résultat.

4.2.3.2 TCGS$_{ex}$

Pour la version non-temporisée des CGS, nous avons montré deux modèles qui se diffèrent par la manière de définir la table de transitions. De même, dans le cas des TCGS, la version de la définition 4.7 a traité le cas où la table de transition est donnée d'une manière symbolique, or il est possible qu'elle soit donnée explicitement, ce qui a été étudié dans [BLMO07]. La définition est donc la suivante :

Définition 4.10 (Syntaxe)
Une TCGS $\mathcal{T}_{ex} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathcal{C}, \mathsf{Inv}, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg} \rangle$ *est un 10-uplet donné par :*

- $\mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathcal{C}, \mathsf{Inv}, \mathsf{Agt}, \mathbb{M}$ *et* Mov *sont définis de la même manière que pour les TCGS.*
- $\mathsf{Edg} : \mathsf{Loc} \times \mathbb{M}^{\mathsf{Agt}} \to \delta$ *est la table de transitions définie explicitement.*

Définition 4.11 (Sémantique)
Pour une TCGS $\mathcal{T}_{ex} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathcal{C}, \mathsf{Inv}, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg} \rangle$*, on associe la CGS infinie* $\mathcal{S} = \langle S, \mathsf{AP}, \mathsf{Lab}', R, \mathsf{Agt}, \mathsf{FM}(\mathbb{R}^+, \mathbb{M}), \mathsf{Mov}', \mathsf{Edg}' \rangle$ *définie par :*

- $\langle S, \mathsf{AP}, \mathsf{Lab}', R \rangle$ *est le TTS associé avec le* $\mathcal{A}$ $\langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathcal{C}, \mathsf{Inv} \rangle$ *;*
- *pour tout* $(q, v) \in S$ *et tout* $a \in \mathsf{Agt}$, $\mathsf{Mov}'((q, v), a) = \mathsf{FM}((q, v), a)$ *;*
- $\mathsf{Edg} : S \times (\mathsf{FM}(\mathbb{R}^+, \mathbb{M}))^{|\mathsf{Agt}|} \to \delta$ *est la* table de transitions explicite. $\mathsf{Edg}'((q, v), ((t_1, f_1), \ldots, (t_k, f_k)))$ *est définie par : soit* $t_{\min} = \min\{t_i \mid i \leq k\}$, $m_i = f_i(t_{\min})$ *pour chaque* $i \leq k$, $(q, f) = \mathsf{Edg}(q, (m_1, \ldots, m_k))$, *et* $f(v + t_{\min}) = (q', Z)$, *on a* $\mathsf{Edg}'((q, v), ((t_1, f_1), \ldots, (t_k, f_k))) = ((q, v), t_{\min}, (q', v + t_{\min}[Z \leftarrow 0]))$.

Exemple 4.1
Considérons le jeu à deux joueurs que montre la figure 4.1. Dans cette figure, les actions qui ne figurent pas sont supposées être des boucles. L'objectif pour le joueur 1 *est d'éviter l'état* q_4 *(cet objectif est exprimé dans la logique* ATL *par la formule* $\langle\langle 1 \rangle\rangle \mathbf{G} \neg q_4$*). Il est facile de se convaincre que le protagoniste (joueur* 1*) dispose d'une stratégie pour gagner le jeu. On décrit une action complète potentielle pour le joueur* 1 *à partir de l'état* $(q_2, 0)$*. Dans cet état, l'objectif du joueur* 1 *est d'atteindre l'état* q_3 *où son objectif sera accompli pour toujours. Mais, ceci n'est possible que dans le cas où la valuation de l'horloge* x *est supérieure à* 5*. Donc, une stratégie possible pour le joueur* 1 *est d'attendre* 5 *unités de temps et jouer ensuite l'action* 1*. Si le joueur* 2 *propose de jouer une durée plus courte, le choix du joueur* 1 *sera de jouer l'action* 2*, pour atteindre l'état* q_1*. Ce choix serait perdant dans le cas où* $x = 2$*. Ce qui nous conduit à l'action complète résultante présentée dans la figure 4.2.*

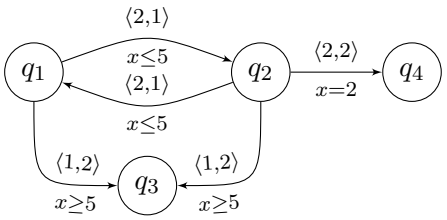


FIG. 4.1 – Exemple d'une TCGS.

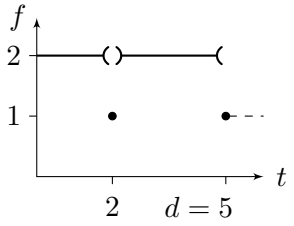


FIG. 4.2 – Un choix complet (d, f).

4.2.3.3 Traduction vers les TCGS

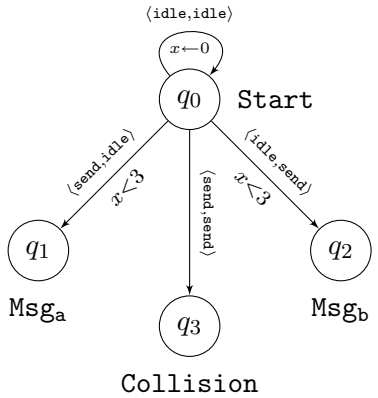
Les TCGS symboliques sont des modèles généraux, il est donc possible de traduire les autres structures vers ce formalisme expressif. Pour traduire les TCGS explicites, la démarche est directe et est similaire à celle dans le cas non-temporisé. Dans le cas des TGA, on tient compte des conditions disponibles pour l'ajouter avec les combinaisons booléennes dans la table de transitions. Ces méthodes sont présentées ci-dessous :

Des TCGS$_{ex}$ vers les TCGS En partant d'une approche similiare à celle présentée dans le chapitre 2, nous présentons la traduction des TCGS$_{ex}$ vers les TCGS, cette traduction est une méthode directe et la complexité est donc linéaire : ayant une TCGS$_{ex}$ $\mathcal{T}_e = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathcal{C}, \mathsf{Inv}, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg}_{ex} \rangle$, on construit la TCGS symbolique $\mathcal{T} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathcal{C}, \mathsf{Inv}, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg} \rangle$ en codant la table de transitions symboliquement de la manière suivante :

$$\mathsf{Edg}(q) = \left(\left(\bigwedge_{A_i \in \mathsf{Agt}} f_i(\tau_{min}) = m_i, \qquad \mathsf{Edg}_{ex}(q, m_1, \ldots, m_k) \right)_{m_i \in \mathsf{Mov}(q, A_i)} \right)$$

Exemple 4.2
Revenons à l'exemple des canaux de communication, nous présentons la TCGS$_{ex}$ qui décrit l'envoi d'un message par l'intermédiaire d'un canal de communication, ce système comprend deux agents qui pourront envoyer le message à tout moment. On note que le canal ne peut pas faire passer plus d'un message à un moment donné.

Notre exemple dans la figure 4.3 est constitué donc de 4 états indiquant les différentes possibilités dans le système. Même s'il ressemble à une CGS non temporisée, on maintient la sémantique temporisée : chaque joueur va choisir le temps où son message sera envoyé, le temps le plus court sera choisi, et la transition sera choisi suivant les deux choix de chaque agent. Convertir cette TCGS$_{ex}$ est une tâche relativement directe, il suffit donc d'adapter la méthode du cas non temporisé, on obtient la table de transitions de la TCGS voulue dans la figure 4.3



$$\left(\begin{array}{ll} ((f_1(\tau_{min}) \stackrel{?}{=} \texttt{send}) \wedge (f_2(\tau_{min}) \stackrel{?}{=} \texttt{idle}), & q_1), \\ ((f_1(\tau_{min}) \stackrel{?}{=} \texttt{idle}) \wedge (f_2(\tau_{min}) \stackrel{?}{=} \texttt{send}), & q_2), \\ ((f_1(\tau_{min}) \stackrel{?}{=} \texttt{send}) \wedge (f_2(\tau_{min}) \stackrel{?}{=} \texttt{send}), & q_3), \\ (\top, & q_0) \end{array} \right)$$

FIG. 4.3 – Une TCGS$_{ex}$ et sa TCGS correspondante

Des TGA vers les TCGS Ici, la traduction est un peu subtile, on note qu'une telle traduction n'est probablement pas possible quand il s'agit de traduire une TGA vers une TCGS_{ex}.

La TCGS va avoir le même nombre d'états que le TGA en question, la table de transition dans chaque état est donné par les formules de la forme $\left(\left(\bigwedge_{j\neq i}(\tau_i < \tau_j) \wedge f_i(\tau_{min}) \stackrel{?}{=} m_k, \delta_k\right)_{a_i \in \mathsf{Agt}}\right)$. Et donc une action est prise seulement si le joueur a choisi la durée minimum.

On remarque que cette traduction ne tient pas compte du non-déterminisme présent dans la sémantique des TGA, mais on peut donner la priorité à un des joueurs, ou bien laisser un joueur supplémentaire prendre ce choix. Sauf qu'en utilisant cette dernière approche, il faut apporter une modification légère à la sémantique pour ne pas tenir en compte la durée choisie par le joueur suplémentaire puisqu'il n'est pas possible de forcer ce joueur de prendre une durée suffisemment large pour ne pas être choisie.

Remarque 4.2
Les TCGS ne sont pas déterminés parce qu'elles héritent cette propriété des CGS. Par contre, la propriété d'être non déterminé pour les TGA résulte quand les agents du système choisissent la même date pour jouer leur action. Pour ces raisons, et sans le prouver formellement, on se convainc que les TCGS_{ex} et les modèles classiques des TGA [AMPS98, dAFH+03] sont incomparables de point de vue expressivité.

4.3 Équivalence des régions et Stratégies

4.3.1 Équivalence des régions

On rappelle dans cette section les équivalences usuelles sur les valuations d'horloges et les extensions aux états des TA [AD94]. Pour chaque horloge x, on choisit une valeur $M_x \in \mathbb{N}$ représentant la valeur maximale de l'horloge x. Pour $t \in \mathbb{R}^+$, on désigne par $\lfloor t \rfloor$ la partie entière et par $\text{frac}(t)$ la partie fractionnelle.

Définition 4.12
Deux valuations d'horloge v et v' sont dites équivalentes, qu'on note par $v \approx_T v'$, si et seulement si les conditions suivantes sont satisfaites :

- *pour tout $x \in \mathcal{C}$, soit $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ ou les deux valuations $v(x)$ et $v'(x)$ sont supérieures à M_x ;*
- *pour tout $x, y \in \mathcal{C}$ avec $v(x) \leq M_x$ et $v(y) \leq M_y$, $\text{frac}(v(x)) \leq \text{frac}(v(y))$ si et seulement si $\text{frac}(v'(x)) \leq \text{frac}(v'(y))$;*
- *pour tout $x \in \mathcal{C}$ avec $v(x) \leq M_x$, $\text{frac}(v(x)) = 0$ si et seulement si $\text{frac}(v'(x)) = 0$.*

On étend cette relation d'équivalence $\approx_T$ aux états d'un TA :

$$(q, v) \approx_T (q', v') \quad \textit{ssi} \quad q = q' \text{ et } v \approx_T v'.$$

De la même manière, on l'étend aussi aux exécutions d'un TA :

$$((q_i, v_i))_i \approx_T ((q'_i, v'_i))_i \quad \textit{ssi} \quad \textit{ils possèdent la même longueur } m \text{ et } \forall i.i \leq m\ (q_i, v_i) \approx_T (q'_i, v'_i).$$

Une classe d'équivalence pour $\approx_T$ sur les états est appelée *région*. On note par $[(q, v)]$ la région qui contient (q, v), et $\mathcal{R}_\mathcal{A}$ pour l'ensemble de toutes les régions. Puisque tous les M_x sont fixés, le nombre des régions est fini.

Définition 4.13
Soit $\mathcal{A}$ un TA et soit (q, v) un état du $\mathcal{A}$. Le futur temporisé de (q, v) *est la fonction* $\mathrm{Fut}(q, v) \colon \mathbb{R}^+ \to \mathcal{R}_\mathcal{A}$ *définie par :*

$$\mathrm{Fut}(q, v)(t) = [(q, v + t)].$$

Étant donné un état (q, v) de $\mathcal{A}$, la fonction $\mathrm{Fut}(q, v)$ est constante par partie. En particulier, on peut diviser $\mathbb{R}^+$ en un ensemble fini d'intervalles sur lesquels $\mathrm{Fut}(q, v)$ est constante. Nous avons

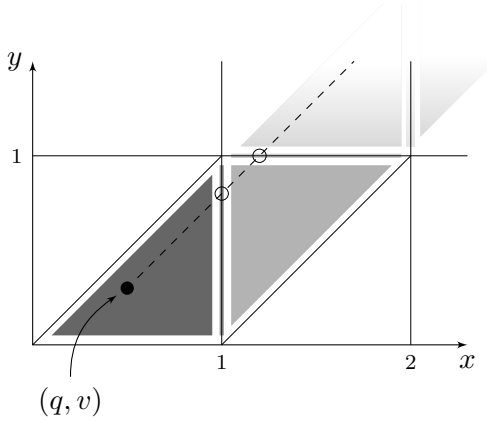
$$\overline{\mathrm{Fut}}(q, v) = I_0\, I_1\, I_2 \cdots I_l$$

pour souligner ceci. Dans cette notation, $(I_i)_i$ est une suite d'intervalles qui partitionne $\mathbb{R}^+$, et dans chacun, $\mathrm{Fut}(q, v)$ est constante. De même, on exige aussi que cette liste soit minimale, *i.e.*, pour tout $j \geq 0$ et tout $t \in I_j$ et $t' \in I_{j+1}$, on a que $\mathrm{Fut}(q, v)(t) \neq \mathrm{Fut}(q, v)(t')$.

Définition 4.14
Étant donnés un TA $\mathcal{A}$ et une région $r \in \mathcal{R}_\mathcal{A}$, on définit le successeur immédiat *de r (noté par* $\mathsf{succ}(r)$*) comme la région r' telle que : $r' \neq r$ et il existe $(q, v) \in r$ et $t \in \mathbb{R}^+$ tel que $(q, v+t) \in r'$ et $\forall 0 < t' < t$ on a $(q, v + t') \in r \cup r'$. On note par* succ^i *la i-ème itération du* succ.

Donc, un successeur immédiat d'une région donnée, indique la "plus proche" région traversée quand le temps s'écoule. succ^i désigne alors la i-ème région traversée en faisant passer le temps.

Exemple 4.3
La figure 4.4 montre une représentation du graphe des régions pour un automate temporisé à deux horloges x et y, la partie en gris foncé désigne la région d'équivalence de l'état (q, v), v étant une valuation qui satisfait la condition suivante $(0 < v(x)) \wedge (v(x) < 1) \wedge (0 < v(y)) \wedge (v(y) < 1) \wedge (\mathrm{frac}(v(y)) < \mathrm{frac}(v(x)))$. *Les parties en gris clair dans cette figure montrent les successeurs de cette région.*
En pointillé, on trouve la ligne indiquant le futur temporisé de l'état (q, v).



La région donnée par la couleur ● est définie par :
– $0 < x < 1$,
– $0 < y < 1$,
– $\mathrm{frac}(y) < \mathrm{frac}(x)$.

FIG. 4.4 – Exemple d'une région et le successeur de l'état (q, v)

4.3.1.1 Isomorphisme et stratégies

On définit maintenant la notion d'isomorphisme entre deux états d'un TA.

Définition 4.15
Soient $\mathcal{A}$ un TA et (q, v), (q', v') deux états de $\mathcal{A}$. On appelle la bijection $\sigma\colon \mathbb{R}^+ \to \mathbb{R}^+$ un isomorphisme *entre (q, v) et (q', v') si elle satisfait les propriétés suivantes :*
- *(strictement croissante) pour tout t_1, $t_2 \in \mathbb{R}^+$, $t_1 < t_2$ ssi $\sigma(t_1) < \sigma(t_2)$;*
- *(préservant le futur) pour tout $t \in \mathbb{R}^+$, $\mathrm{Fut}(q, v)(t) = \mathrm{Fut}(q', v')(\sigma(t))$.*

Dans ce cas, on dit que les deux états (q, v) et (q', v') sont isomorphes.

Donc, deux états sont isomorphes si leurs futurs temporisés sont équivalents.

La figure 4.5 montre un exemple d'un isomorphisme. Clairement, un tel isomorphisme doit

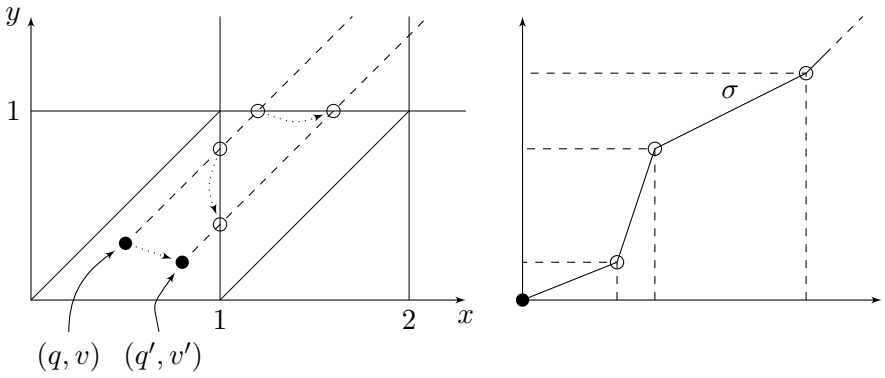


FIG. 4.5 – Exemple d'un isomorphisme entre deux états (q, v) et (q', v').

satisfaire $\sigma(0) = 0$, pour qu'on obtienne nécessairement $(q, v) \approx_T (q', v')$ quand un tel isomorphisme existe. La réciproque est aussi vraie :

Lemme 4.1
Soit $\mathcal{A}$ un TA, et soient (q, v), (q', v') deux états de $\mathcal{A}$. Alors $(q, v) \approx_T (q', v')$ si et seulement si (q, v) et (q', v') sont isomorphes.

Démonstration: On suppose que $(q, v) \approx_T (q', v')$ et on construit un isomorphisme entre ces deux états. Supposons que (q, v) se trouve dans une région où le temps peut s'écouler (le cas où (q, v) se trouve dans une région transitoire est traité de la même manière). Dans ce cas, on a :

$$\overline{\mathrm{Fut}}(q, v) = [0, a_1)\ \{a_1\}\ (a_1, a_2)\ \cdots\ \{a_M\}(a_M, \infty),$$

où a_i sont des éléments de $\mathbb{R}^+$ t.q. $a_1 \neq 0$ et $a_i < a_{i+1}$ pour $1 \leq i < M$. Puisque $(q, v) \approx_T (q', v')$, il existe des entiers $(b_i)_{i \leq M}$ t.q.

$$\overline{\mathrm{Fut}}(q', v') = [0, b_1)\ \{b_1\}\ (b_1, b_2)\ \cdots\ \{b_M\}(b_M, \infty),$$

avec $b_1 \neq 0$ et $b_i < b_{i+1}$ pour $1 \leq i < M$. On obtient un isomorphisme entre (q, v) et (q', v') en définissant des bijections croissantes entre chaque paire des intervalles correspondants. □

Cette notion d'isomorphisme ainsi que celle du lemme 4.1, s'étendent naturellement aux exécutions du TA.

Définition 4.16
Soit $\mathcal{A}$ un TA et soient $\rho = (q_i, v_i)_i$ et $\rho' = (q'_i, v'_i)_i$ deux exécutions du $\mathcal{A}$. On appelle la bijection $\sigma_e \colon \mathbb{N} \times \mathbb{R}^+ \to \mathbb{N} \times \mathbb{R}^+$ un isomorphisme entre ρ et ρ' *s'il existe une suite $(\sigma_i)_i$ de bijections dans $\mathbb{R}^+$ qui satisfait les deux conditions suivantes :*
- *pour tout $(i,t) \in \mathbb{N} \times \mathbb{R}^+$, $\sigma_e(i,t) = (i, \sigma_i(t))$;*
- *pour tout $i \in \mathbb{N}$, σ_i est un isomorphisme entre (q_i, v_i) et (q'_i, v'_i).*

Dans ce cas, on dit que les deux exécutions ρ et ρ' sont isomorphes.

Lemme 4.2
Soit $\mathcal{A}$ un TA, et soient $((q_i, v_i))_i$ et $((q'_i, v'_i))_i$ deux exécutions de $\mathcal{A}$. Alors, $((q_i, v_i))_i \approx_T ((q'_i, v'_i))_i$ si et seulement si $((q_i, v_i))_i$ et $((q'_i, v'_i))_i$ sont isomorphes.

4.3.2 Des stratégies plus simples

Dans cette section, nous montrons que les stratégies gagnantes pour des objectifs "raisonnables" peuvent être simplifiées.

On commence par définir les notions suivantes.

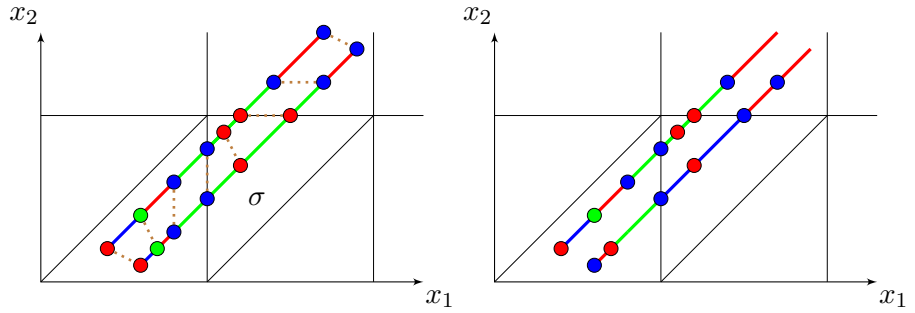
Définition 4.17
On dit qu'un objectif gagnant Ω est définissable par région *si, pour tout chemin $\rho \in \Omega$ et tout $\rho' \approx_T \rho$, on a $\rho' \in \Omega$.*

Cette notion d'un objectif nous paraît raisonnable et c'est elle que nous considérerons dans la suite. De même pour les stratégies, avoir des stratégies simples signifie aussi que celles-là dépendent seulement des régions. Ce qui entraîne deux propriétés : d'un côté, on parlera d'une stratégie *invariante par région* pour signifier que la stratégie ne dépend pas de toute l'histoire d'une exécution, mais seulement de son abstraction dans les régions ; de l'autre côté, on dira qu'une stratégie est *uniforme par région* pour signifier que la valeur renvoyée par la stratégie reste constante dans les régions intermédiaires qui ont été visitées.

Exemple 4.4
La figure 4.6 montre une stratégie invariante par région. Ici, le choix d'un joueur pour un temps donné est illustré par une couleur différente. Dans ce cas, on trouve que pour une stratégie invariante par région la séquence des couleurs maintient le même ordre et le même type d'objet (segments ou points) pour deux trajectoires traversant les mêmes régions. Dans le cas d'une stratégie uniforme par région que montre la figure 4.7, on trouve que la stratégie garde la même couleur dans chaque région tout le long d'une trajectoire.

Définition 4.18
Soit $\mathcal{T}$ une TCGS, and λ_A une stratégie pour une coalition $A \subseteq \mathsf{Agt}$. La stratégie λ_A est invariante par région *si, pour toutes les exécutions finies ρ et ρ' t.q. $\rho \approx_T \rho'$, on a un isomorphisme σ entre $\mathsf{last}(\rho)$ et $\mathsf{last}(\rho')$ t.q., notant $(d, f) = \lambda_A(\rho)$ et $(d', f') = \lambda_A(\rho')$, on a $d' = \sigma(d)$ et $f'(t) = f(\sigma^{-1}(t))$ pour tout $t \in \mathbb{R}^+$.*

En résumant, quand une stratégie λ_A pour une coalition A est invariante par région, alors n'importe quelles deux trajectoires isomorphes produisent le "même" ensemble d'outcomes (à un isomorphisme près).



region-invariant *NON* region-invariant

FIG. 4.6 – Une stratégie invariante par région

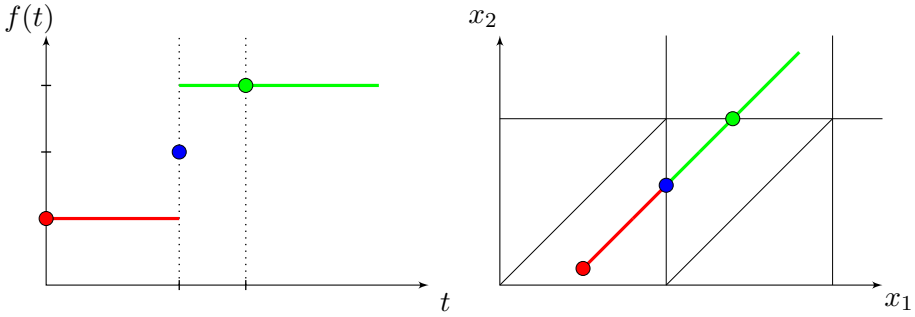


FIG. 4.7 – Une stratégie uniforme par région

Proposition 4.1
Soit $\mathcal{T}$ une TCGS, et soit $A \subseteq \mathsf{Agt}$ une coalition. Pour deux trajectoires finies équivalentes r et r', et une stratégie λ_A pour A, on peut construire une stratégie invariante par région λ'_A t.q., pour tout $\rho' \in \mathsf{Out}(r', \lambda'_A)$, il existe $\rho \in \mathsf{Out}(r, \lambda_A)$ t.q. $\rho \approx_T \rho'$.

Démonstration: On fixe deux trajectoires finies équivalentes r et r', et une stratégie λ_A pour les agents dans A. Pour pouvoir définir λ'_A, on fixe une trajectoire représentative dans chaque classe d'équivalence de $\approx_T$, avec la condition supplémentaire pour que le représentant soit choisi parmi $\mathsf{Out}(r, \lambda_A)$ quand c'est possible, et que r soit la trajectoire représentative de sa classe d'équivalence. Alors, et pour toute trajectoire finie ρ, on écrit $\pi(\rho)$ pour cette trajectoire représentative, et on fixe un isomorphisme σ_ρ entre $\mathsf{last}(\rho)$ et $\mathsf{last}(\pi(\rho))$; alors, si $\lambda_{a_l}(\pi(\rho)) = (d, f)$, on donne $\lambda'_{a_l}(\rho) = (d', f')$ avec $d' = \sigma_\rho^{-1}(d)$ et $f'(t) = f \circ \sigma_\rho(t)$, pour tout $t \in \mathbb{R}^+$. Il est évident donc que cette stratégie est invariante par région.

Maintenant, on montre par induction (sur les outcomes finis) que cette stratégie vérifie nos conditions. Supposons que $\rho' \in \mathsf{Out}^1(r', \lambda'_A)$, et considérons un ensemble de coups $((d'_l, f'_l))_{a_l \in \mathsf{Agt}}$ résultant ρ' depuis r' et t.q. $(d_l, f_l) = \lambda'_{a_l}(r')$ pour chaque $a_l \in A$. Soit $\rho = \pi(r') \cdot \widetilde{\mathsf{Edg}}(\mathsf{last}(\pi(r')), ((\sigma_{r'}(d'_l), f'_l \circ \sigma_{r'}))_{a_l \in \mathsf{Agt}})$ avec $\widetilde{\mathsf{Edg}}$ la table de transitions de la CGS (infinie) associée. Alors ρ est isomorphe à ρ', et il est l'outcome à une étape de λ_A par construction.

Pour l'étape inductive, on suppose que $\rho' \in \mathsf{Out}^{n+1}(r', \lambda'_A)$. En particulier, ρ' est un outcome à une étape d'une certaine trajectoire ρ'', qui est considérée comme un outcome à n-étapes

de r'. Par hypothèse d'induction, ρ'' est isomorphe à un outcome (à n-étape) ρ''' de r, qu'on suppose qu'il est le représentant de sa classe d'équivalence. De même, on note par $((d'_l, f'_l))_{a_l \in \mathsf{Agt}}$ l'ensemble des choix qui étendent ρ'' à ρ', on construit ρ en ajoutant la transition donnée par $\widetilde{\mathsf{Edg}}(\mathsf{last}(\rho'''), ((\sigma_{\rho''}(d'_l), f'_l \circ \sigma_{\rho''}))_{a_l \in \mathsf{Agt}})$ à ρ'''. Par construction, la trajectoire résultante est un outcome à une étape de ρ''' par la stratégie λ_A, et est isomorphe à ρ'.

En appliquant le même argument inductivement, on montre le résultat pour tout outcome fini. On traite les outcomes infinis comme des limites de Cauchy des séquences d'outcomes finis. □

Définition 4.19
Soit $\mathcal{T}$ une TCGS, et soit λ_A une stratégie pour une coalition $A \subseteq \mathsf{Agt}$. On appelle la stratégie λ_A uniforme par région *si, pour toute exécution finie ρ avec $\lambda_A(\rho) = (d, f)$, la valeur de f est constante dans les régions, i.e., en écrivant $(q, v) = \mathsf{last}(\rho)$, pour tout $t, t' \in \mathbb{R}^+$, si $(q, v + t) \approx_T (q, v + t')$, alors $f(t) = f(t')$.*

Proposition 4.2
Soient $\mathcal{T}$ une TCGS et $A \subseteq \mathsf{Agt}$ une coalition. Pour deux trajectoires équivalentes r et r', pour toute stratégie λ_A d'une coalition A, on peut construire une stratégie uniforme et invariante par région λ'_A t.q., pour tout $\rho' \in \mathsf{Out}(r', \lambda'_A)$, il existe $\rho \in \mathsf{Out}(r, \lambda_A)$ avec $\rho \approx_T \rho'$.

Démonstration: Soient r et r' deux trajectoires équivalentes finies, et soit λ_A une stratégie pour A. On écrit λ''_A pour la stratégie invariante par région donnée par la proposition 4.1.

On construit la stratégie λ'_A depuis λ''_A comme suit : soient r'' une trajectoire finie, $(d_l, f_l) = \lambda''_{a_l}(r'')$ pour chaque $a_l \in A$, et $d_A = \min\{d_l \mid a_l \in A\}$. Supposons que $\overline{\mathrm{Fut}}(\mathsf{last}(r'')) = I_1 I_2 \dots I_m$, on choisit une valeur t_i dans chaque I_i, avec une condition supplémentaire que $t_i = d_A$ quand $d_A \in I_i$. Pour chaque $a_l \in A$, on définit donc f'_l par $f'_l(t) = f_l(t_i)$ quand $t \in I_i$, et on donne $\lambda'_{a_l}(r'') = (d_A, f'_l)$.

Cette stratégie est clairement uniforme par région. De plus, elle peut être définie pour qu'elle reste invariante par région (puisque λ''_A l'est aussi). On montre maintenant qu'elle satisfait notre condition. La preuve est aussi par induction pour les outcomes finis, et remontée aux trajectoires infinies par les séquences de Cauchy.

Soit $\rho' \in \mathsf{Out}^1(r', \lambda'_A)$, et soit $(d'_l, f'_l)_{a_l \in \mathsf{Agt}}$ un ensemble de choix qui induit cet outcome. Soit $d_{\mathsf{Agt}} = \min\{d'_l \mid a_l \in \mathsf{Agt}\}$. On obtient deux cas à étudier :

1. si le délai minimal a été choisi par la coalition A, alors, la valeur de $f'_l(d_{\mathsf{Agt}})$ correspond à la valeur de $f_l(d_{\mathsf{Agt}})$ donnée par la stratégie λ''_{a_l}, par construction de f'. Donc, $\rho' \in \mathsf{Out}^1(r', \lambda''_A)$, avec le même ensemble de choix. La proposition 4.1 assure l'existence d'un outcome de λ_A depuis r qui est équivalent à ρ', comme souhaité.
2. sinon, d_{Agt} est strictement inférieure à la valeur d_A proposée par la coalition A. Là encore, on a deux cas :
   (a) si $\mathrm{Fut}(r')(d_{\mathsf{Agt}}) = \mathrm{Fut}(r')(d_A)$, on considère alors la transition donnée par les actions $((d_A, f_l))_{a_l \in \mathsf{Agt}}$. Ajouter cette transition à la fin de r' résulte ρ'', qui est donc dans $\mathsf{Out}^1(r', \lambda''_A)$ et est équivalente à ρ'. Encore, par la proposition 4.1, on obtient ρ'' qui est équivalente à l'outcome à une étape de λ_A après r.

(b) sinon, d_{Agt} appartient à un intervalle I_i, et on considère la transition donnée par les actions (t_i, f_l) (utilisée pour la définition de λ'_A) pour les agents $a_l \notin A$ et les choix (d_A, f_l) pour les agents dans A. Ajoutée à la fin de r', on obtient un outcome dans $\mathsf{Out}^1(r', \lambda''_A)$ qui est isomorphe à ρ'. Encore une fois, la proposition 4.1 donne le résultat désiré.

L'étape d'induction est complétée de la même manière. □

On déduit immédiatement le corollaire suivant :

Corollaire 4.1
Soit $\mathcal{T}$ une TCGS, et soient $A \subseteq \mathsf{Agt}$ une coalition et Ω un objectif gagnant invariant par région. Étant données deux trajectoires équivalentes finies r et r', il existe une stratégie gagnante pour A après r par rapport à Ω si et seulement si il existe une stratégie gagnante uniforme et invariante par région *pour A après r' par rapport à Ω.*

4.4 La CGS des régions

Dans cette section, on définit une CGS finie pour une TCGS $\mathcal{T}$, on l'appelle une *CGS des régions* de $\mathcal{T}$. On montre ensuite que cette CGS des régions est bisimilaire-alternante à la TCGS originale $\mathcal{T}$. Cette CGS des régions peut être considérée comme une "version de jeu concurrent" de l'automate région classique.

On commence par rappeler la définition de l'automate des régions [AD94], on donne ensuite la définition formelle des CGS des régions avec l'abstraction temporisée d'un choix complet.

Définition 4.20
*Étant donné un TA $\mathcal{A} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathcal{C}, \mathsf{Inv} \rangle$, on définit l'*automate des régions *par le système de transitions $\mathcal{R}(\mathcal{A}) = \langle \mathsf{S}, \mathsf{AP}, \mathsf{Lab}', R \rangle$ comme suit : premièrement, pour chaque horloge $x \in \mathcal{C}$, on prend M_x la constante maximale avec qui l'horloge x est comparée dans $\mathcal{A}$. Alors :*

- $\mathsf{S} = \mathcal{R}_{\mathcal{A}}$;
- $\mathsf{Lab}'(\langle q, [v] \rangle) = \mathsf{Lab}(q)$.
- $R \subseteq \mathsf{S} \times \mathsf{S}$ *tel que* $(\langle q, [v] \rangle, \langle q', [v'] \rangle) \in R$ *si et seulement s'il existe une transition* $(q, (\Phi_i, q_i, Z_i)_{0 \leq i \leq p}) \in \delta$ *telle que il existe* $j \leq p$ *où* $q' = q_j$ *et il existe* $d > 0$ *avec* $\langle q, [v + d] \rangle \in \mathsf{S}$, $v + d \models \Phi_j$ *et* $v' = v + d[Z_j \leftarrow 0]$.

Définition 4.21
Soit $\mathcal{T} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \mathcal{C}, \mathsf{Inv}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg} \rangle$ une TCGS. Un choix complet discret *d'un joueur $a \in \mathsf{Agt}$ dans une région $[(q, v)]$ est une paire (d, f) avec $d \in \mathbb{N}$ et $f : \mathbb{N} \to \mathsf{Mov}(q, a)$.*

On écrit $\mathsf{FM}([(q, v)], a)$ pour désigner l'ensemble des choix complets d'un joueur a dans la région $[(q, v)]$. On a :

$$\mathsf{FM}([(q, v)], a) = |\overline{\mathrm{Fut}}((q, v))| \times (\mathsf{Mov}(q, a))^{\mathbb{N}}.$$

L'ensemble $\mathsf{FM}(\mathbb{N}, \mathbb{M})$ dénote l'ensemble $\mathbb{N} \times \mathbb{M}^{\mathbb{N}}$ de tous les choix complets discrets possibles.

Le premier élément d'un choix complet spécifie le délai qu'un agent souhaite attendre avant de franchir une transition, en terme de nombre de régions à visiter. La deuxième partie est la fonction de choix dans les régions intermédiaires.

Définition 4.22
Avec une TCGS $\mathcal{T} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \mathcal{C}, \mathsf{Inv}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg} \rangle$*, on associe la CGS (finie)* $\mathcal{R} = \langle S, \mathsf{AP}, \mathsf{Lab}', R, \mathsf{Agt}, \mathsf{FM}(\mathbb{N}, \mathbb{M}), \mathsf{Mov}', \mathsf{Edg}' \rangle$ *définie comme suit :*

- $\langle S, \mathsf{AP}, \mathsf{Lab}', R \rangle$ *est l'automate des régions associé à l'automate temporisé* $\langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \mathcal{C}, \mathsf{Inv}, \delta \rangle$ *;*
- *pour tout* $s \in S$*, pour tout* $a \in \mathsf{Agt}$*,* $\mathsf{Mov}'(s, a) = \mathsf{FM}(s, a)$ *;*
- $\mathsf{Edg}'([q, v], ((d_l, f_l)_{l \in \mathsf{Agt}}))$ *est donnée par : pour chaque* d_l*, on choisit une durée* t_l *telle que* $(q, v + t_l) \in \mathsf{succ}^{d_l}(q, v)$*, on définit* $f'_l(t_i) = f_l(d_i)$ *pour tout* $i \in \mathbb{N}$ *et* $l \in \mathsf{Agt}$*. L'état d'arrivée sera donné par la transition de la CGS associée à* $\mathcal{T}$ $\widetilde{\mathsf{Edg}}((q, v), (t_l, f'_l)_{l \in \mathsf{Agt}}) = ((q, v), (q', v'))$*, on obtient donc* $\mathsf{Edg}'([(q, v)], ((d_l, f_l)_{l \in \mathsf{Agt}})) = ([(q, v)], [(q', v')])$*.*

On appelle cette CGS finie la CGS des régions *associée à* $\mathcal{T}$*.*

Remarque 4.3
Prenons la TCGS $\mathcal{T} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \mathcal{C}, \mathsf{Inv}, \delta, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg} \rangle$ *et sa CGS des régions* $\mathcal{R} = \langle S, \mathsf{AP}, \mathsf{Lab}', R, \mathsf{Agt}, \mathsf{FM}(\mathbb{N}, \mathbb{M}), \mathsf{Mov}', \mathsf{Edg}' \rangle$*, et supposons* D *la constante maximale qui apparaissent dans* $\mathcal{T}$*.*

Puisque le nombre des régions est exponentiel dans le nombre des horloges et dans la constante maximimale des horloges, la taille de l'ensemble d'états dans la CGS des régions l'est aussi d'où $|S| = \mathcal{O}\left(|\mathsf{Loc}| \cdot D \cdot 2^{|\mathcal{C}|}\right)$*. Par contre, la taille de la table de transitions remonte d'une exponentielle supplémentaire puisqu'un coup représente une fonction dont le nombre d'entrées dépend de la constante maximale des horloges qui est représentée en binaire. Ce qui donne une table de taille doublement exponentielle par rapport à la TCGS de départ* $|\mathsf{Edg}|' = \mathcal{O}\left(|S| \left(|\mathbb{M}|^D\right)^{|\mathsf{Agt}|}\right)$*.*

Prenons l'exemple suivant :

Exemple 4.5
Dans la figure 4.8, nous montrons une TCGS (qui décrit un TGA similaire à celui qui figure dans [dAFH+03] avec une différence quand $\tau_1 = \tau_2$ *notre modèle reste déterministe), nous trouvons ensuite la CGS des régions associée.*

La table de transitions dans la TCGS dans l'état q_1 *est donnée par :*

$$\mathsf{Edg}(q_1) = \begin{pmatrix} (\tau_1 < \tau_2 & , (q_1, (x \leq 1, q_2, \varnothing), (\top, q_1, \varnothing)), \\ (\top & , (q_1, (\top, q_1, \{x\})) \end{pmatrix}$$

Puisque la table de transitions dans la TCGS ne dépendent que du temps choisi par les joueurs, la CGS des régions l'est aussi. Dans la figure, les étiquettes des (quelques) transitions (issues de l'état $(q_1, x = 0)$*) décrivent le nombre de régions que chacun des joueurs souhaite parcourir.*

Soit $\mathcal{T}$ une TCGS et soit $\mathcal{R}$ sa CGS des régions. Avec une exécution r de $\mathcal{T}$, on peut naturellement associer une exécution unique de $\mathcal{R}$, qu'on note par $[r]$.

On montre maintenant que cette abstraction est correcte, dans le sens qu'une TCGS $\mathcal{T}$ et son abstraction des régions $\mathcal{R}$ sont bisimilaires-alternantes (voir définition 2.2).

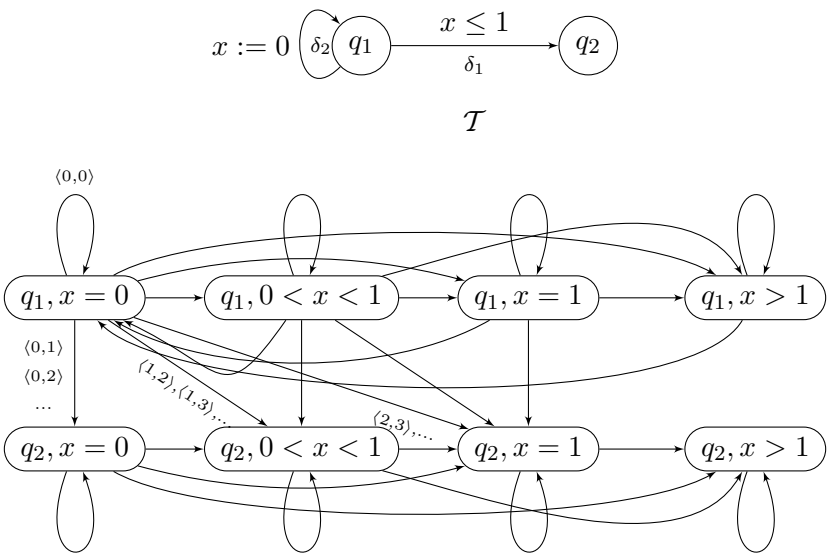


FIG. 4.8 – Une TCGS $\mathcal{T}$ et sa CGS région correspondante $\mathcal{R}$.

Proposition 4.3
Soit $\mathcal{T}$ une TCGS, et soient $\mathcal{R}$ sa CGS des régions et A une coalition. Pour toute trajectoire finie r dans $\mathcal{T}$, et toute stratégie λ_A dans $\mathcal{T}$ pour A, on peut construire une stratégie λ'_A dans $\mathcal{R}$ t.q., pour tout $\rho' \in \mathsf{Out}_{\mathcal{R}}([r], \lambda'_A)$, il existe $\rho \in \mathsf{Out}_{\mathcal{T}}(r, \lambda_A)$ t.q. $[\rho] = \rho'$.

Démonstration: Soit λ_A une stratégie dans $\mathcal{T}$ pour A. Définissons une stratégie λ'_A sur $\mathcal{R}$ pour A. Tout d'abord, on construit une stratégie *invariante et uniforme par région* λ''_A sur $\mathcal{T}$ pour A, suivant la proposition 4.2. Soit r une trajectoire finie de $\mathcal{T}$. Supposant que $\lambda''_A(r) = (d''_l, f''_l)_{a_l \in A}$; Pour chaque $a_l \in A$, on pose $d'_l \in \mathbb{N}$ tel que, pour $(q, v) = \mathsf{last}(r)$, on a $(q, v + d''_l) \in \mathsf{succ}^{d'_l}([\mathsf{last}(r)])$ et on définit f' par : $f'_l(j) = f''_l(t) \quad$ pour $t \in I_j$. On donne donc $\lambda'_A([r])$ comme une abstraction temporisée du choix complet $(d'_l, f'_l)_{a_l \in A}$.

On montre maintenant par induction (sur des outcomes finis) que cette stratégie satisfait les conditions. Si ρ' se réduit à un seul état, le résultat précédent est clairement satisfait.

Pour l'étape inductive, on suppose que $\rho' \in \mathsf{Out}^{n+1}([r], \lambda'_A)$. En particulier, ρ' est un outcome à une étape d'une trajectoire ρ'_n, qui est l'outcome à n étape de $[r]$. Par hypothèse d'induction, il existe un outcome (à n-étape) ρ_n de r tel que $[\rho_n] = \rho'_n$. Soit $((d'_l, f'_l)_{a_l \in \mathsf{Agt}})$ un ensemble de choix qui étend ρ'_n à ρ', en d'autres termes, $\widetilde{\mathsf{Edg}}(\mathsf{last}(\rho'_n), (d'_l, f'_l)_{a_l \in \mathsf{Agt}}) = \mathsf{last}(\rho')$. ρ est donc obtenue dans deux étapes ; premièrement, on construit ρ'' en ajoutant la transition donnée par $\widetilde{\mathsf{Edg}}(\mathsf{last}(\rho_n), ((d''_l, f''_l))_{a_l \in \mathsf{Agt}})$ (avec (d'_l, f'_l) et (d''_l, f''_l) sont liées entre elles par la proposition 4.2 comme mentionné ci-dessus) à la fin de ρ_n. On note que cette transition est correcte puisque l'équivalence d'horloge est une abstraction temporisée de la bisimulation. Par construction, la trajectoire résultante est un outcome à une étape de ρ_n par la stratégie λ''_A et donc elle appartient à $\mathsf{Out}(r, \lambda''_A)$. De plus, elle satisfait $\rho' = [\rho'']$. La Proposition 4.2 indique qu'une exécution finie $\rho \in \mathsf{Out}(r, \lambda_A)$ tel que $\rho \approx_T \rho''$, implique que $[\rho] = \rho'$.

Là encore, on étend facilement ce résultat aux outcomes infinis suivant les séquences de

Cauchy. □

Proposition 4.4
Soit $\mathcal{T}$ une TCGS, et soient $\mathcal{R}$ sa CGS des régions et A une coalition. Pour toute trajectoire finie r dans $\mathcal{R}$, toute trajectoire finie r' dans $\mathcal{T}$ tel que $r = [r']$ et toute stratégie λ_A sur $\mathcal{R}$ pour A, on peut construire une stratégie λ'_A sur $\mathcal{T}$ t.q., pour tout $\rho' \in \mathsf{Out}(r', \lambda'_A)$, il existe $\rho \in \mathsf{Out}([r], \lambda_A)$ t.q. $\rho = [\rho']$.

Démonstration: Soit λ_A une stratégie dans $\mathcal{R}$ pour A. On définit maintenant une stratégie λ'_A sur $\mathcal{T}$ pour A. Soit r une exécution finie dans $\mathcal{R}$. On suppose que $\lambda_A(r) = ((d_l, f_l))_{a_l \in A}$. Pour toute exécution finie r' de $\mathcal{T}$ tel que $r = [r']$ et $\overline{\mathrm{Fut}}(\mathsf{last}(r')) = I_0\, I_1 \cdots I_k$, on définit $\lambda'_A(r')$ par le choix complet $((d'_l, f'_l))_{a_l \in A}$ où, pour chaque $a_l \in A$, et ayant $(q, v) = \mathsf{last}(r')$, on a $(q, v + d'_l) \in \mathsf{succ}^{d_l}(\mathsf{last}(r))$ et

$$f'_l(t) = f_l(j) \quad \text{avec } j \text{ t.q. } t \in I_j,$$

pour tout $t \in \mathbb{R}^+$. La preuve donc suit les mêmes étapes inductives que celles d'avant :
On suppose que $\rho' \in \mathsf{Out}^{n+1}(r', \lambda'_A)$. ρ' est donc un outcome à une étape d'une autre trajectoire ρ'_n qui est aussi un outcome à n étape de la trajectoire r'. Par hypothèse d'induction, il existe une outcome ρ_n de r tel que $\rho_n = [\rho'_n]$. Si $(d'_l, f'_l)_{l \in \mathsf{Agt}}$ est l'ensemble de choix à l'aide duquel on a étendu ρ'_n, on obtient donc ρ en ajoutant à ρ_n la transition $\widetilde{\mathsf{Edg}}(\mathsf{last}(\rho_n), (d_l, f_l)_{l \in \mathsf{Agt}})$ (avec $(d_l, f_l)_{l \in \mathsf{Agt}}$ et $(d'_l, f'_l)_{l \in \mathsf{Agt}}$ sont liés comme on l'a montré ci-dessus). Par construction, la trajectoire résultante est un outcome à une étape de ρ_n par la stratégie λ_A, et donc $\rho = [\rho']$. □

On déduit immédiatement le corollaire suivant :

Corollaire 4.2
Soit $\mathcal{T}$ une TCGS. L'équivalence de région induit une bisimulation alternante entre (la CGS infinie associée à) $\mathcal{T}$ et sa CGS des régions $\mathcal{R}$.

4.5 La logique ATL temporisée

Dans le chapitre 3, nous avons étudié une version de la logique ATL dans le cas temporisé discret, qu'on appelle TATL. Dans la suite, nous présentons une variante avec horloges de formule qui est plus riche, et qui a été définie également dans [HP06]. Mais tout d'abord, on commence par définir l'extension la plus large, la logique TATL*.

Définition 4.23 (syntaxe)
Étant donné un ensemble d'horloges de formules $\mathcal{X}$, la logique TATL* est définie par la grammaire suivante :

$$\begin{aligned} \mathit{TATL}^\star \ni \varphi_s &::= P \mid c \sim n \mid \neg\varphi_s \mid \varphi_s \vee \varphi_s \mid \langle\!\langle A \rangle\!\rangle \varphi_p \\ \varphi_p &::= \varphi_s \mid c.\varphi_p \mid \varphi_p \wedge \varphi_p \mid \varphi_p \vee \varphi_p \mid \varphi_p \,\mathbf{U}\, \varphi_p \mid \varphi_p \,\mathbf{R}\, \varphi_p \end{aligned}$$

avec $P \in \mathit{AP}$, $c \in \mathcal{X}$, $\sim \in \{<, \leq, =, \geq, >\}$, $n \in \mathbb{N}$, et $A \subseteq \mathit{Agt}$. Les formules de la forme φ_s sont appelées des formules d'états, *tandis que les formules de la forme φ_p sont dites des* formules de chemins.

Définition 4.24 (sémantique)
Soit $\mathcal{T} = \langle \mathit{Loc}, \mathit{AP}, \mathit{Lab}, \delta, \mathcal{C}, \mathit{Inv}, \mathit{Agt}, \mathbb{M}, \mathit{Mov}, \mathit{Edg} \rangle$ *une TCGS. Étant donnée une trajectoire infinie* r *dans* $\mathcal{T}$ *et soit* j *une position le long de* r*. Soient* $\varphi \in \mathit{TATL}^\star$*, et* w *une valuation d'horloges de formules qui apparaissent dans* φ*. On donne* r, j *satisfaisant* φ *sous la valuation* w*, qu'on note par* $r, j \models^w_{\mathcal{T}} \varphi$*, est définie récursivement par :*

$$
\begin{array}{rcl}
r, j \models^w_{\mathcal{T}} P & \Leftrightarrow & P \in \mathit{Lab}(r[j]) \\
r, j \models^w_{\mathcal{T}} c \sim n & \Leftrightarrow & w(c) \sim n \\
r, j \models^w_{\mathcal{T}} \neg\varphi_s & \Leftrightarrow & r, j \not\models \varphi_s \\
r, j \models^w_{\mathcal{T}} \varphi_s \vee \psi_s & \Leftrightarrow & r, j \models_w \varphi_s \text{ ou } r, j \models^w_{\mathcal{T}} \psi_s \\
r, j \models^w_{\mathcal{T}} \langle\!\langle A \rangle\!\rangle \varphi_p & \Leftrightarrow & \exists \lambda_A \in \mathsf{Strat}(A) \text{ t.q. } \forall \rho \in \mathit{Out}^\infty(0 \twoheadrightarrow_r j, \lambda_A).\ \rho, 0 \models^w_{\mathcal{T}} \varphi_p \\
r, j \models^w_{\mathcal{T}} c.\varphi_p & \Leftrightarrow & r, j \models^{w[c \leftarrow 0]}_{\mathcal{T}} \varphi_p \\
r, j \models^w_{\mathcal{T}} \varphi_p \wedge \psi_p & \Leftrightarrow & r, j \models^w_{\mathcal{T}} \varphi_p \text{ et } r, j \models^w_{\mathcal{T}} \psi_p \\
r, j \models^w_{\mathcal{T}} \varphi_p \vee \psi_p & \Leftrightarrow & r, j \models^w_{\mathcal{T}} \varphi_p \text{ ou } r, j \models^w_{\mathcal{T}} \psi_p \\
r, j \models^w_{\mathcal{T}} \varphi_p \mathbf{U} \psi_p & \Leftrightarrow & \exists j''.j <_r j'' \text{ et } r, j'' \models^{w''}_{\mathcal{T}} \psi_p \text{ et} \\
 & & \forall j <_r j' <_r j''.\ r, j'' \models^{w'}_{\mathcal{T}} \varphi_p, \\
 & & \text{où } w^{(i)} = w + \mathit{Cost}(j \twoheadrightarrow_r j^{(i)}) \\
r, j \models^w_{\mathcal{T}} \varphi_1 \mathbf{R} \varphi_2 & \Leftrightarrow & r, j \models^w_{\mathcal{T}} \neg(\neg\varphi_1 \mathbf{U} \neg\varphi_2).
\end{array}
$$

On utilise les raccourcis syntaxiques usuels définis dans 1.17, comme $\mathbf{F}\,\varphi$ pour désigner $\top \mathbf{U}\,\varphi$ (un jour φ), $\mathbf{G}\,\varphi$ pour $\bot \mathbf{R}\,\varphi$ (toujours φ). On définit aussi $\overset{\infty}{\mathbf{F}}\,\varphi \overset{\text{def}}{\equiv} \mathbf{G}\,\mathbf{F}\,\varphi$ (infiniment souvent φ) et $\overset{\infty}{\mathbf{G}}\,\varphi \overset{\text{def}}{\equiv} \mathbf{F}\,\mathbf{G}\,\varphi$ (le dual de $\mathbf{G}\,\mathbf{F}\,\varphi$).

Il est évident que la logique TPTL [AH89] est un fragment de la logique TATL*, or le model-checking de TPTL est connu pour être indécidable pour les automates temporisés sous la sémantique continue [AH94]. Ce qui va rendre le model-checking de la logique TATL* indécidable à son tour. Nous nous intéressons donc aux fragments décidables de TATL*. On peut voir donc que la logique ATL* est un fragment (classique) de la logique TATL* quand les horloges n'interviennent pas, et les logiques TATL$_\mathcal{C}$ et ATL sont les fragments de TATL* et ATL*, resp., dans lesquelles chaque formule de chemin est définie par la grammaire suivante :

$$\varphi_p ::= \varphi_s \mathbf{U}\,\varphi_s \mid \varphi_s \mathbf{R}\,\varphi_s \mid c.\varphi_p$$

Évidemment, pour la logique ATL, on n'utilise pas des formules de la forme $c.\varphi_p$. On définit également un nouveau fragment de TATL*, qu'on nomme TALTL, et qui contient les logiques ATL* et TATL$_\mathcal{C}$:

Définition 4.25
On définit la syntaxe de la logique TALTL *par la grammaire suivante :*

$$
\begin{array}{rcl}
\mathit{TALTL} \ni \varphi_s & ::= & p \mid c \sim n \mid \neg\varphi_s \mid \varphi_s \vee \varphi_s \mid \langle\!\langle A \rangle\!\rangle \varphi_p \mid c.\varphi_s \\
\varphi_p & ::= & \varphi_s \mid \varphi_p \wedge \varphi_p \mid \varphi_p \vee \varphi_p \mid \varphi_p \mathbf{U}\,\varphi_p \mid \varphi_p \mathbf{R}\,\varphi_p
\end{array}
$$

Cette logique peut être considérée comme une extension de la logique TCLTL ci-dessous.

Définition 4.26
La logique TCLTL *est définie de la même manière :*

$$\mathit{TALTL} \ni \varphi_s ::= p \mid c \sim n \mid \neg\varphi_s \mid \varphi_s \vee \varphi_s \mid \mathbf{E}\varphi_p \mid \mathbf{A}\varphi_p \mid c.\varphi_s$$
$$\varphi_p ::= \varphi_s \mid \varphi_p \wedge \varphi_p \mid \varphi_p \vee \varphi_p \mid \varphi_p \,\mathbf{U}\, \varphi_p \mid \varphi_p \,\mathbf{R}\, \varphi_p$$

À notre connaissance, la logique TALTL n'a jamais été étudiée avant, même dans le contexte des automates temporisés. La différence avec la logique TATL* se situe au niveau de remise à zéro des horloges : ce n'est possible que dans les formules d'état, et non dans les formules de chemin. À l'aide de cette nouvelle logique, on peut maintenant exprimer des propriétés plus intéressantes des propriétés exprimables avec la logique TATL, et on va voir que l'on peut les vérifier par des algorithmes de model-checking (même si la complexité est grande). D'une manière similaire, on peut étendre la logique temporelle arborescente temporisée avec des propriétés d'équité par exemple (la formule suivante $c.\,\mathbf{A}(\overset{\infty}{\mathbf{F}}\, p \rightarrow \mathbf{F}\,(q \wedge \mathbf{F}\,(q' \wedge c \leq 10)))$ est une formule TCLTL, qui indique que le long d'une exécution équitable, q et ensuite q' vont avoir lieu dans moins de 10 unités de temps) tout en restant décidable.

On commence par montrer que la logique TALTL ne peut pas distinguer deux états région-équivalents dans une TCGS. Ceci nécessite d'étendre notre définition précédente : Étant donnés une TCGS $\mathcal{T} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathcal{C}, \mathsf{Inv}, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg}\rangle$ et un ensemble d'horloges de formule $\mathcal{X}$ (disjoint de $\mathcal{C}$), on définit $\mathcal{T}' = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathcal{C} \cup \mathcal{X}, \mathsf{Inv}, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg}\rangle$. Cette TCGS utilise $\mathcal{X}$, mais ces horloges ne jouent pas de rôle dans la sémantique. Dans cette TCGS étendue, on note par (q, v, w) l'état de la CGS infinie associée à $\mathcal{T}'$, où v est la valuation des horloges dans $\mathcal{C}$ et w est la valuation des horloges dans $\mathcal{X}$. On écrit $\mathsf{proj}((q, v, w)) = (q, v)$, et on étend cette notation pour une relation qui relie les trajectoires dans $\mathcal{T}'$ aux trajectoires correspondantes dans $\mathcal{T}$.

Théorème 4.1

Soient $\mathcal{T}$ une TCGS, et $\mathcal{T}'$ sa TCGS étendue correspondante avec un ensemble d'horloges de formule $\mathcal{C}'$. Soit φ une formule TALTL construite avec des horloges dans $\mathcal{C}'$. Pour deux états région-équivalents (q, v, w) et (q, v', w') de $\mathcal{T}'$, et deux trajectoires région-équivalentes r et r' commençant par (q, v, w) et (q, v', w'), resp., on a

$$\mathsf{proj}(r), 0 \models^w_{\mathcal{T}} \varphi \quad \text{ssi} \quad \mathsf{proj}(r'), 0 \models^{w'}_{\mathcal{T}} \varphi.$$

De plus, si φ est une formule d'état, alors ce résultat reste valable même si on relâche l'hypothèse que r et r' sont région-équivalents.

Démonstration: La preuve se fait par induction sur la structure de φ.

On commence par les formules d'état, et cela ne nécessite pas que r et r' soient région-équivalentes.

- si φ est une proposition atomique, le résultat est une conséquence du fait que les positions initiales de r et r' correspondent aux mêmes états dans $\mathcal{T}$.
- si φ est de la forme $c \sim n$, le résultat est une conséquence du fait que les deux états initiaux se trouvent dans la même région.

– si φ est une conjonction, une disjonction ou une négation de formules d'états, on retrouve le résultat immédiatement de l'hypothèse d'induction.
– si $\varphi = c.\psi$ avec $c \in \mathcal{C}'$, avoir $\mathsf{proj}(r), 0 \models^w_{\mathcal{T}} \varphi$ nous donne que $\mathsf{proj}(r), 0 \models^{w[c\leftarrow 0]}_{\mathcal{T}} \psi$. Par hypothèse d'induction (et puisque ψ est une formule d'état), pour toute trajectoire r' commençant par $(q, v', w'[c \leftarrow 0])$, on a $\mathsf{proj}(r'), 0 \models^{w[c\leftarrow 0]}_{\mathcal{T}} \psi$, et donc on obtient $\mathsf{proj}(r'), 0 \models^w_{\mathcal{T}} \varphi$.
– finalement, on suppose que $\varphi = \langle\!\langle A \rangle\!\rangle \psi$ pour une coalition $A \subseteq \mathsf{Agt}$ et pour une formule de chemin ψ : posons que $\mathsf{proj}(r), i \models^w_{\mathcal{T}} \langle\!\langle A \rangle\!\rangle \psi$, et soit λ_A une stratégie de A qui garantit ψ pour tous ses outcomes. D'après la proposition 4.1, on peut définir une stratégie λ'_A dont les outcomes sont région-équivalents à un outcome de λ_A. Avec l'hypothèse d'induction sur les formules de chemin on termine la démonstration de cette partie de la preuve.

On s'intéresse maintenant au cas où φ est une formule de chemin. On va étudier un seul cas, puisque les cas où φ est une formule d'état ou une combinaison booléenne de formules de chemin sont directs. On suppose donc que $\varphi = \varphi_1 \mathbf{U} \varphi_2$, pour deux formules de chemins φ_1 et φ_2. Si $\mathsf{proj}(r), 0 \models^w_{\mathcal{T}} \varphi_1 \mathbf{U} \varphi_2$, il existe donc une position i_0 t.q. $\mathsf{proj}(r), i_0 \models^{w+\mathsf{Cost}(0 \twoheadrightarrow_{\mathsf{proj}(r)} i_0)}_{\mathcal{T}} \varphi_2$, et t.q. toute position intermédiaire $0 <_r i_1 <_r i_0$ satisfait φ_1 pour la valuation $w + \mathsf{Cost}(0 \twoheadrightarrow_{\mathsf{proj}(r)} i_1)$. Il suffit donc de choisir une position i'_0 le long de r' dont l'état correspond à la même région que l'état à la position i_0 le long r. Il suffit d'appliquer l'hypothèse d'induction à partir de l'état à la position i'_0 et aux états intermédiaires pour avoir finalement que $\mathsf{proj}(r'), 0 \models^{w'}_{\mathcal{T}} \varphi_1 \mathbf{U} \varphi_2$. □

Une conséquence de ce résultat est que si φ est une formule d'état de la logique TALTL, on peut définir que $(q, v) \models^w_{\mathcal{T}} \varphi$ comme étant l'existence d'une trajectoire r dans $\mathcal{T}$ commençant par (q, v) t.q. $r, 0 \models^w_{\mathcal{T}} \varphi$.

Remarque 4.4
On note que le résultat précédent n'est pas valable dans le cas de la logique TATL⋆ : il est facile de construire un exemple de deux trajectoires qui visitent les mêmes séquences de régions étendues, mais où une seule d'entre elles satisfait la formule $c.\mathbf{F}\,(q \wedge c \geq 1)$ *(*c.f. *figure 4.9). L'application du résultat du théorème 4.1 sur les* formules d'état *de TATL⋆ reste une conjecture.*

FIG. 4.9 – Deux trajectoires équivalentes distinguées par la formule $c.\mathbf{F}\,(q \wedge c \geq 1)$.

4.5.1 Model-checking

Nous consacrons cette partie pour décrire un algorithme basé sur les régions pour le model-checking des formules TALTL sur les TCGS. Étant données une TCGS $\mathcal{T}$ et des horloges de formules $\mathcal{C}'$, on considère la TCGS $\mathcal{T}'$ qui étend $\mathcal{T}$ avec les horloges dans $\mathcal{C}'$, et on note par $\mathcal{R}'$ la CGS des régions associée. L'algorithme 3 étiquette cette CGS finie avec les sous-formules d'une

formule d'état de la logique TALTL ψ vérifiée sur $\mathcal{T}$. Pour cela, on remplit récursivement une table booléenne $T([(q,v,w)],\psi)$, avec $[(q,v,w)]$ dans l'ensemble des régions de $\mathcal{R}'$ et ψ une sous-formule d'état de φ. Cet algorithme fait appel à une procédure supplémentaire `ATLstar-labeling`, qui est la procédure classique, de complexité 2EXPTIME, pour le model-checking de ATL* comme définie dans [AHK02] (*cf.* section 2.1.4.3).

Théorème 4.2

Soient $\mathcal{T}$ une TCGS, $\mathcal{C}'$ un ensemble d'horloges de formule, $\mathcal{T}'$ la TCGS étendue avec les horloges de $\mathcal{C}'$, et $\mathcal{R}'$ la CGS des régions étendue qui correspond à $\mathcal{T}'$. Posons $\varphi \in$ TALTL une formule ayant les horloges dans $\mathcal{C}'$. On obtient la table T en appliquant l'algorithme 3 sur $\mathcal{R}'$ et φ. Ayant (q,v,w) un état dans $\mathcal{T}'$, alors $(q,v) \models^w_{\mathcal{T}} \varphi$ ssi $T([(q,v,w)],\varphi) = \top$.

Démonstration: La preuve est par induction sur la structure de la formule.

- si φ est une proposition atomique, on a $T([(q,v,w)],\varphi) = \top$ ssi $\varphi \in \mathsf{Lab}(q)$, i.e., ssi $(q,v) \models^w_{\mathcal{T}} \varphi$.
- si φ est de la forme $c \sim n$, alors $T([(q,v,w)],\varphi) = \top$ ssi $w(c) \sim n$, qui est équivalent à avoir $(q,v) \models^w_{\mathcal{T}} \varphi$.
- si φ est une négation, une disjonction ou une conjonction de sous-formules d'état, le résultat est une conséquence immédiate de l'hypothèse d'induction.
- si $\varphi = c.\varphi'$, on a $T([(q,v,w)],\varphi) = \top$ ssi $T([(q,v,w[c \leftarrow 0])],\varphi') = \top$. Par hypothèse d'induction, c'est équivalent à avoir $(q,v) \models^{w[c\leftarrow 0]}_{\mathcal{T}} \varphi'$, qui est équivalent à son tour à $(q,v) \models^w_{\mathcal{T}} c.\varphi$.
- le cas où $\varphi = \langle\langle A\rangle\rangle \varphi_p$ est le cas le plus intéressant. Premièrement, on suppose que $(q,v) \models^w_{\mathcal{T}} \langle\langle A\rangle\rangle \varphi_p$, et on choisit une stratégie λ_A t.q. tout outcome $\rho \in \mathsf{Out}_{\mathcal{T}}((q,v),\lambda_A)$ satisfait la formule de chemin φ_p. Cette stratégie peut être considérée comme une stratégie dans $\mathcal{T}'$, on peut donc appliquer le résultat de la proposition 4.3 pour $\mathcal{T}'$ et $\mathcal{R}'$. Cela conduit à une stratégie λ'_A dans $\mathcal{R}'$ dont tous les outcomes depuis $[(q,v,w)]$ sont la projection dans $\mathcal{R}'$ d'un outcome de λ_A dans $\mathcal{T}'$ depuis (q,v,w).
  Soit $\rho \in \mathsf{Out}_{\mathcal{R}'}([(q,v,w)],\lambda'_A)$, et $\rho' \in \mathsf{Out}_{\mathcal{T}'}((q,v,w),\lambda_A)$ t.q. $\rho = [\rho']$. Soit $\hat{\varphi_p}$ une formule LTL construite à partir de φ_p en remplaçant chaque formule d'état ψ par la nouvelle proposition atomique correspondante p_ψ.
  Puisque ρ' satisfait φ_p dans $\mathcal{T}'$, sa projection dans $\mathcal{R}'$ satisfait $\hat{\varphi_p}$, alors $\rho \models \hat{\varphi_p}$ dans $\mathcal{R}'$. Ce qui donne que $[(q,v,w)] \models_{\mathcal{R}'} \langle\langle A\rangle\rangle \hat{\varphi_p}$. Par la correction de l'algorithme du model-checking de la logique ATL*, on peut conclure que $T([(q,v,w)],\varphi) = \top$.

  Réciproquement, supposant que $T([(q,v,w)],\varphi) = \top$. Il existe donc une stratégie λ'_A dans $\mathcal{R}'$ t.q. tous les outcomes de λ'_A depuis $[(q,v,w)]$ satisfont $\hat{\varphi_p}$. De la même manière que dans la preuve de la proposition 4.4, on peut transformer la stratégie λ'_A en une stratégie λ_A dans $\mathcal{T}'$ pour que les outcomes de λ_A depuis (q,v,w) suivent les mêmes régions qu'un outcome de λ'_A depuis $[(q,v,w)]$. Par conséquence, tous les outcomes de λ_A depuis (q,v,w) satisfont φ_p, pour avoir finalement $(q,v) \models^w_{\mathcal{T}} \varphi$.

□

Dans l'algorithme 3, la procédure `replace`$(\psi, \psi' \rightarrow p_{\psi'})$ renvoie la formule ψ où chaque

Procedure `labeling`($\mathcal{R}'$, ψ) **begin**
  foreach $[(q,v,w)] \in \mathcal{R}'$ **do** $T([(q,v,w)],\psi) = \bot$;
  switch ψ **do**
    case p :
      foreach $[(q,v,w)] \in \mathcal{R}'$ **do**
        if $p \in \mathit{Lab}(q)$ **then** $T([(q,v,w)],\psi) = \top$;
    case $c \sim n$:
      foreach $[(q,v,w)] \in \mathcal{R}'$ **do**
        if $w(c) \sim n$ **then** $T([(q,v,w)],\psi) = \top$;
    case $\neg\psi_1$:
      `labeling`($\mathcal{R}$',ψ_1) ;
      foreach $[(q,v,w)] \in \mathcal{R}'$ **do**
        if $T([(q,v,w)],\psi_1) = \bot$ **then** $T([(q,v,w)],\psi) = \top$;
    case $\psi_1 \vee \psi_2$:
      `labeling`($\mathcal{R}$',ψ_1) ;
      `labeling`($\mathcal{R}$',ψ_2) ;
      foreach $[(q,v,w)] \in \mathcal{R}'$ **do**
        if $(T([(q,v,w)],\psi_1) = \top$ **or** $T([(q,v,w)],\psi_1) = \top)$ **then**
          $T([(q,v,w)],\psi) = \top$;
    case $\psi_1 \wedge \psi_2$:
      `labeling`($\mathcal{R}$',ψ_1) ;
      `labeling`($\mathcal{R}$',ψ_2) ;
      foreach $[(q,v,w)] \in \mathcal{R}'$ **do**
        if $(T([(q,v,w)],\psi_1) = \top$ **and** $T([(q,v,w)],\psi_1) = \top)$ **then**
          $T([(q,v,w)],\psi) = \top$;
    case $c.\psi_1$:
      `labeling`($\mathcal{R}$',ψ_1) ;
      foreach $[(q,v,w)] \in \mathcal{R}'$ **do**
        if $T([(q,v,w[c \leftarrow 0])],\psi_1) = \top$ **then** $T([(q,v,w)],\psi) = \top$;
    case $\langle\!\langle A \rangle\!\rangle \psi_p$:
      $\hat{\psi}_p = \psi_p$;
      $\hat{\mathcal{R}}' = \mathcal{R}'$;
      foreach $\psi' \in$ **State-Subf**(ψ_p) **do**
        `labeling`($\mathcal{R}'$,ψ') ;
        $\hat{\psi}_p =$ `replace`($\hat{\psi}_p$, $\psi' \rightarrow p_{\psi'}$) ;
        $\hat{\mathcal{R}}' =$ `decorate`($\hat{\mathcal{R}}'$,$p_{\psi'}$) ;
      T'=`ATLstar-labeling`($\hat{\mathcal{R}}'$, $\langle\!\langle A \rangle\!\rangle \hat{\psi}_p$) ;
      foreach $[(q,v,w)] \in \mathcal{R}'$ **do**
        if $(T'([(q,v,w)], \langle\!\langle A \rangle\!\rangle \hat{\psi}_p) = \top)$ **then**
          $T([(q,v,w)], \langle\!\langle A \rangle\!\rangle \psi_p) = \top$;
end

Algorithme 3 : Étiquetage de $\mathcal{R}'$ avec la formule d'état ψ

occurence de ψ' dans les sous-formules de ψ sera remplacée par la proposition atomique $p_{\psi'}$. D'autre part, la procédure `decorate`$(\mathcal{R}, p_\psi)$ va renvoyer une CGS où on a étiqueté les états par la proposition atomique p_ψ.

Pour le model-checking de la logique $\mathsf{TATL}_\mathcal{C}$, on peut obtenir un algorithme plus efficace en remplaçant la procédure `ATLstar-labeling` par la procédure `ATL-labeling` pour les formules ATL qui est dans PTIME. On obtient donc le corollaire suivant :

Corollaire 4.3
Le model-checking d'une formule TALTL sur les TCGS est décidable et il est 2EXPTIME*-complet. Le model-checking d'une formule* $\mathsf{TATL}_\mathcal{C}$ *sur les TCGS se fait dans* EXPSPACE *et il est* EXPTIME*-dur.*

Démonstration: L'algorithme de model-checking de la logique $\mathsf{ATL}^\star$ est 2EXPTIME dans la taille de la formule, et il est PTIME dans la taille du modèle. Notre algorithme est donc 2EXPTIME dans $|\varphi|$, et dans 2EXPTIME dans $|\mathcal{T}|$, d'où la complexité globale de 2EXPTIME. La dureté 2EXPTIME pour la logique TALTL est une conséquence d'avoir le model-checking de la logique $\mathsf{ATL}^\star$ 2EXPTIME-dur sur les CGS.

Puisque le model-checking de la logique ATL est dans PTIME dans la taille de la formule et du modèle, cette procédure fait appel au calcul du prédécesseur contrôllable (l'opérateur $CPre$) un nombre polynomial de fois. Dans notre cas la table de transitions est doublement exponentielle, on doit pouvoir calculer le $CPre$ dans un espace exponentielle en utilisant un compteur (de taille exponentielle simple) pour parcourir la table de transitions. On adapte l'algorithme pour avoir une complexité dans EXPSPACE dans $|\mathcal{T}|$ et $|\varphi|$, et donc la complexité globale est dans EXPSPACE. On peut montrer la dureté dans EXPTIME en codant notre formalisme par les jeux countdown [JLS07] (*cf.* section 3.2). □

4.6 Traitement des stratégies Zeno

Dans la section précédente, nous avons montré la décidabilité du problème du model-checking de la logique TALTL sans imposer des contraintes aux stratégies des joueurs. Par exemple, ayant la TCGS dans la figure 4.8, l'adversaire (joueur 2) dispose d'une stratégie qui empêche le premier joueur de gagner et qui consiste jouer la transition δ_2 à la durée zéro. Un joueur pourrait atteindre un objectif de sûreté en bloquant le temps. Dans cette section, on va montrer comment on peut interdire ce genre de comportement non réalisable physiquement, on expliquera les méthodes pour forcer les agents à jouer d'une manière "juste", en éliminant les stratégies qui font converger le temps (appelées des *stratégies Zeno*). Pour aboutir à cette fin, on adopte le cadre qui a été introduit dans [dAFH$^+$03]. On note que des approches similaires ont été utilisées dans [GSSAL94] pour manipuler les exécutions Zeno des *automates temporisés avec Entrée/Sortie*.

Formellement, une exécution infinie $\rho = ((s_i, d_i, s_{i+1}))_{i\in\mathbb{N}}$ est Zeno si $\mathsf{Cost}(\rho) < \infty$. Un tel comportement n'est pas réalisable par une machine, et donc ne devrait pas être utilisé par les joueurs comme un moyen pour gagner les jeux temporisés : par exemple, dans la TCGS de la figure 4.8, la transition δ_1 est franchissable seulement après l'écoulement de moins de 1 unité de temps, le jeu est perdant pour le protagoniste (joueur 1), la stratégie de l'adversaire pour gagner est de choisir de jouer son action à temps nul (et donc l'horloge reste bloquée à jamais). Cette stratégie n'est clairement pas réalisable physiquement.

On adapte notre définition d'une stratégie gagnante par la définition suivante :

Définition 4.27
Soit Ω un objectif gagnant. Une stratégie λ_A pour une coalition A est gagnante non-Zeno *après un préfixe ρ_0 si tous les outcomes infinis non-Zeno de λ_A après ρ_0 vérifient l'objectif gagnant donné, et le long de tous les outcomes Zeno (s'il en existe), les joueurs de A en sont responsables un nombre fini de fois.*

En suivant [dAFH+03], on commence par détecter les outcomes Zeno. Alors, ayant une TCGS $\mathcal{T}$, on ajoute à cette TCGS une horloge supplémentaire $z \notin \mathcal{C}$ qui est mise à zéro à chaque fois qu'elle dépasse 1 unité de temps. Formellement, on réalise ceci par : Soit $\mathcal{T} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathcal{C}, \mathsf{Inv}, \mathsf{Agt}, \mathsf{Mov}, \mathsf{Edg} \rangle$ une TCGS, on définit $\mathcal{T}_z = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta_z, \mathcal{C}_z, \mathsf{Inv}, \mathsf{Agt}, \mathsf{Mov}, \mathsf{Edg}' \rangle$ où $\mathcal{C}_z = \mathcal{C} \cup \{z\}$ et δ_z est définie à partir de δ par : pour tout $(q, (\Phi_i, q_i, Z_i)_{0 \leq i \leq p}) \in \delta$, on a $(q, (\Phi'_i, q'_i, Z'_i)_{0 \leq i \leq 2p}) \in \delta_z$ qui est définie par :

$$(\Phi'_i, q'_i, Z'_i) = \begin{cases} (\Phi_{i/2} \wedge z < 1, q_{i/2}, Z_{i/2}) & \text{si } i \text{ est paire} \\ (\Phi_{(i-1)/2} \wedge z \geq 1, q_{(i-1)/2}, Z_{(i-1)/2} \cup \{z\}) & \text{si } i \text{ est impaire} \\ (\top, q, \varnothing) & \text{si } i = 2p. \end{cases}$$

La table de transitions Edg' est donc adaptée à δ_z. Étant donnée une exécution finie r de $\mathcal{T}'$, on a clairement r une exécution non-Zeno si et seulement si l'horloge z est mise à zéro infiniment souvent.

Étant donnée une exécution Zeno, il est possible qu'une partie seulement des agents en soit responsable. Un joueur est responsable du comportement Zeno d'une exécution s'il a "choisi" un nombre infini de fois le plus petit délai. Une coalition est responsable du comportement Zeno si au moins un de ses membres l'est. On peut enregistrer cette information en *décorant* la CGS infinie associée à la TCGS $\mathcal{T}$ avec des "blâmes" reprécisés avec des propositions atomiques. On définit l'ensemble des propositions atomiques $\mathsf{AP}' = \mathsf{AP} \times \{\mathsf{tick}, \overline{\mathsf{tick}}\} \times \{\mathsf{bl}_A, \mathsf{bl}_{\bar{A}}, \mathsf{bl}_{\mathsf{Agt}}\}$. Les deux premiers symboles (tick et $\overline{\mathsf{tick}}$) sont utilisés quand l'horloge z dépasse (ou non) 1, alors que les trois derniers symboles associent un blâme à la coalition A, à l'adversaire $\bar{A}$, ou les deux coalitions. Cet ensemble est celui de la CGS étendue $\mathcal{E}_A$ dont la définition est la suivante :

Définition 4.28
Soient $\mathcal{T} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \mathcal{C}_z, \mathsf{Inv}, \delta, \mathsf{Agt}, \mathsf{Mov}, \mathsf{Edg} \rangle$ une TCGS (contenant l'horloge z des "ticks"), $\mathcal{S} = \langle S, \mathsf{AP}', \mathsf{Lab}', R, \mathsf{Agt}, \mathsf{Mov}', \widetilde{\mathsf{Edg}} \rangle$ sa CGS infinie associée, et $A \subseteq \mathsf{Agt}$ une coalition. On définit la CGS étendue $\mathcal{E}_A = \langle S^{\mathcal{E}}, \mathsf{AP}', \mathsf{Lab}^{\mathcal{E}}, R^{\mathcal{E}}, \mathsf{Agt}, \mathsf{Mov}^{\mathcal{E}}, \mathsf{Edg}^{\mathcal{E}} \rangle$ par :

- $S^{\mathcal{E}} \subseteq S \times \{\mathsf{tick}, \overline{\mathsf{tick}}\} \times \{\mathsf{bl}_A, \mathsf{bl}_{\bar{A}}, \mathsf{bl}_{\mathsf{Agt}}\}$ *est l'extension de S avec les informations supplémentaires pour avoir la trace des agents dont le choix a été considéré ;*
- $\mathsf{Lab}^{\mathcal{E}}((s,t,b)) = \mathsf{Lab}'(s) \cup \{t, b\}$,
- $R^{\mathcal{E}} \subseteq S^{\mathcal{E}} \times \mathbb{R}^+ \times S^{\mathcal{E}}$ *contient deux genres de transitions :*
  - *pour chaque $(s,t,b) \in S^{\mathcal{E}}$, avec $s = (q,v)$, et $d \in \mathbb{R}^+$ t.q. $v(z) + d < 1$, alors pour chaque $(s,d,s') \in R$ et $b' \in \{\mathsf{bl}_A, \mathsf{bl}_{\bar{A}}, \mathsf{bl}_{\mathsf{Agt}}\}$, la transition $((s,t,b), d, (s', \overline{\mathsf{tick}}, b'))$ est dans $R^{\mathcal{E}}$;*
  - *pour chaque $(s,t,b) \in S^{\mathcal{E}}$, avec $s = (q,v)$, et $d \in \mathbb{R}^+$ t.q. $v(z) + d \geq 1$, alors pour chaque $(s,d,s') \in R$ et $b' \in \{\mathsf{bl}_A, \mathsf{bl}_{\bar{A}}, \mathsf{bl}_{\mathsf{Agt}}\}$, la transition $((s,t,b), d, (s', \mathsf{tick}, b'))$ est dans $R^{\mathcal{E}}$;*
- $\mathsf{Edg}^{\mathcal{E}}$ *est définie à partir de $\widetilde{\mathsf{Edg}}$ par : étant donné l'ensemble de choix complet $((d_l, f_l)_{a_l \in \mathsf{Agt}})$, ayant $d_0 = \min\{d_l \mid a_l \in \mathsf{Agt}\}$, et on donne bl par :*

– bl_A *si* $\exists a_l \in A.\ d_l = d_0$ *et* $\forall a_l \in \bar{A}.\ d_l > d_0$;
– $\mathsf{bl}_{\bar{A}}$ *si* $\exists a_l \in \bar{A}.\ d_l = d_0$ *et* $\forall a_l \in A.\ d_l > d_0$;
– $\mathsf{bl}_{\mathsf{Agt}}$ *si* $\exists a_l \in A.\ d_l = d_0$ *et* $\exists a_l \in \bar{A}.\ d_l = d_0$;
Alors, si $\widetilde{\mathsf{Edg}}(s, ((d_1, f_1), \ldots, (d_k, f_k))) = (s, d_0, s')$, *avec* $s = (q, v)$, *on donne*

$$\mathsf{Edg}^{\mathcal{E}}((s,t,b), ((d_1,f_1),\ldots,(d_k,f_k))) = ((s,t,b), d_0, (s', \overline{\mathsf{tick}}, \mathsf{bl}))$$

si $v(z) + d_0 < 1$, *et*

$$\mathsf{Edg}^{\mathcal{E}}((s,t,b), ((d_1,f_1),...,(d_k,f_k))) = ((s,t,b), d_0, (s', \mathsf{tick}, \mathsf{bl}))$$

sinon.
On note par $\varpi \colon \mathsf{S}^{\mathcal{E}} \to \mathsf{S}$ *la fonction de projection définie par* $\varpi(s,t,b) = s$, *et on étend cette notation aux trajectoires.*

On note que cette CGS infinie *ne* correspond *pas* à la CGS infinie associée à la TCGS puisqu'en général, il n'est pas possible de décorer une TCGS avec des informations sur les joueurs à blâmer.

L'équivalence par rapport aux horloges est étendue naturellement aux états de $\mathcal{E}_A$. On dit que $(s,t,b) \approx_T (s',t',b')$ si et seulement si $s \approx_T s'$, $t = t'$ et $b = b'$. On garde la terminologie d'une *région* pour la classe d'équivalence de l'état étendu (s,z,t,b) pour $\approx_T$. Cette relation d'équivalence est aussi étendue aux exécutions de $\mathcal{E}_A$. Les définitions des stratégies région-invariantes et région-uniformes sont aussi naturellement étendues dans le même contexte.

Maintenant, on adapte les résultats de la section 4.3 à ce nouveau cadre. La proposition 4.5 et les preuves sont des adaptations directes de la proposition 4.1. L'adaptation de la proposition 4.2 est légèrement plus compliquée, puisque les blâmes changent en transformant la stratégie en une stratégie région uniforme.

Proposition 4.5
Soient $\mathcal{T}$ *une TCGS,* $A \subseteq \mathsf{Agt}$ *une coalition et* $\mathcal{E}_A$ *sa CGS étendue associée. Pour deux trajectoires finies équivalentes* r *et* r' *(de* $\mathcal{E}_A$*), et pour toute stratégie* λ_A *pour la coalition* A, *on peut construire une stratégie région-invariante* λ'_A *t.q., pour tout* $\rho' \in \mathsf{Out}(r', \lambda'_A)$, *il existe* $\rho \in \mathsf{Out}(r, \lambda_A)$ *t.q.* $\rho \approx_T \rho'$.

Proposition 4.6
Soient $\mathcal{T}$ *une TCGS,* $A \subseteq \mathsf{Agt}$ *une coalition et* $\mathcal{E}_A$ *sa CGS étendue associée. Pour toutes trajectoires* r *et* r' *(de* $\mathcal{E}_A$*) t.q.* $\varpi(r) \approx_T \varpi(r')$, *pour toute stratégie* λ_A *de la coalition* A, *on peut construire une stratégie région-uniforme et région-invariante* λ'_A *t.q., pour tout* $\rho' \in \mathsf{Out}(r', \lambda'_A)$, *il existe* $\rho \in \mathsf{Out}(r, \lambda_A)$ *avec* $\varpi(\rho) \approx_T \varpi(\rho')$. *De plus, pour tout* $i \in \mathbb{N}$, *ayant* $\rho_i = (s_i, b_i, t_i)$ *et* $\rho'_i = (s'_i, b'_i, t'_i)$, *on a :*
– $t'_i = \mathsf{tick}$ *si et seulement si* $t_i = \mathsf{tick}$,
– *si* $b'_i = \mathsf{bl}_{\bar{A}}$ *alors* $b_i \in \{\mathsf{bl}_{\bar{A}}, \mathsf{bl}_{\mathsf{Agt}}\}$, *et* $b_i = b'_i$ *sinon.*

Démonstration: Pour montrer cette proposition, on adapte la preuve de la proposition 4.2. La stratégie λ'_A est la même que celle qu'on a construite dans la preuve de la proposition 4.2. Soit $\rho' \in \mathsf{Out}(r', \lambda'_A)$, et $((d'_l, f'_l))_{a_l \in \mathsf{Agt}}$ un ensemble de coups qui donne cet outcome. Soit $d = \min\{d'_l \mid a_l \in \mathsf{Agt}\}$. En considérant les cas (1) et (2b) de la preuve de la proposition 4.2, on peut

se convaincre que les outcomes obtenus ρ et ρ' sont effectivement région-équivalents (dans $\mathcal{E}_A$), en particulier, ils satisfont les propriétés désirées.

Le seul point difficile est le cas (2a). Effectivement, dans ce cas, le blâme sur une coalition change : puisque d_{Agt} (le temps minimum de tous les agents) est strictement inférieur à d_A (le temps minimum pour la coalition A), ρ' arrive dans un état $(s', t', \mathsf{bl}_{\bar{A}})$; pourtant, puisqu'on considère un délai de d_A unités de temps pour définir ρ, on a que ρ arrive dans un état $(s, t, \mathsf{bl}_{\mathsf{Agt}})$. □

On obtient un corollaire similaire au corollaire 4.1 :

Corollaire 4.4
Soient $\mathcal{T}$ une TCGS, $A \subseteq \mathsf{Agt}$ une coalition et Ω un objectif gagnant région-invariant. Soit r et r' deux trajectoires isomorphes. Il existe une stratégie gagnante non-Zeno pour A après r par rapport à Ω si et seulement s'il existe une stratégie gagnante région-uniforme et région-invariante *non-Zeno pour A après r' par rapport à Ω.*

Démonstration: Soient λ_A une stratégie gagnante non-Zeno pour A, et λ'_A la stratégie obtenue par la proposition précédente. On suppose que l'outcome ρ' de λ'_A après r' n'est pas gagnant. Si ρ' est non-Zeno, puisqu'il est région-équivalent à un outcome (gagnant) de λ_A, on a $\rho' \in \Omega$. Donc, ρ' est Zeno, et les agents dans A sont blâmés infiniment souvent. Ce qui implique que ρ' visite infiniment souvent les états étiquetés par bl_A ou $\mathsf{bl}_{\mathsf{Agt}}$. Par la proposition 4.6, l'outcome correspondant de λ_A est aussi Zeno et visite infiniment souvent les états étiquetés par bl_A ou $\mathsf{bl}_{\mathsf{Agt}}$, ce qui contredit l'hypothèse d'avoir λ_A comme une stratégie gagnante. □

On définit maintenant la sémantique "non-Zeno" de la logique $\mathsf{TATL}^\star$.

Définition 4.29
Soient $\mathcal{T} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathcal{C}, \mathsf{Inv}, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg} \rangle$ une TCGS, et, pour chaque coalition $A \subseteq \mathsf{Agt}$, $\mathcal{E}_A = \langle S, \mathsf{AP}, \mathsf{Lab}', R, \mathsf{Agt}, \mathsf{FM}(\mathbb{R}^+, \mathbb{M}), \mathsf{Mov}', \mathsf{Edg}' \rangle$ la CGS infinie construite suivant la définition 4.28.

Soient $s = (q, v)$ un état de la TCGS, et $w \colon \mathcal{C}' \to \mathbb{R}^+$ la valuation des horloges de formule. Soit $\varphi \in \mathsf{TATL}$. On définit $s \models_z^w {}_{\mathcal{T}}\varphi$ de la même manière que $s \models_{\mathcal{T}}^w \varphi$, à l'exception de $\varphi = \langle\!\langle A \rangle\!\rangle \psi_p$:

$$s \models_z^w {}_{\mathcal{T}} \langle\!\langle A \rangle\!\rangle \psi_p \;\Leftrightarrow\; (s, \overline{\mathsf{tick}}, \mathsf{bl}_A) \models_{\mathcal{E}_A}^w \langle\!\langle A \rangle\!\rangle ((\overset{\infty}{\mathbf{F}}\, \neg \mathsf{bl}_{\bar{A}} \rightarrow \overset{\infty}{\mathbf{F}}\, \mathsf{tick}) \wedge (\overset{\infty}{\mathbf{F}}\, \mathsf{tick} \rightarrow \psi_p))$$

Cette sémantique détecte les comportements Zeno qui doivent être perdants : les outcomes non-Zeno (qui satisfont $\overset{\infty}{\mathbf{F}}\, \mathsf{tick}$) doivent satisfaire la formule de chemin ψ_p, alors que les outcomes Zeno (où la formule $\overset{\infty}{\mathbf{F}}\, \mathsf{tick}$ n'est pas satisfaite) vont satisfaire $\overset{\infty}{\mathbf{G}}\, \mathsf{bl}_{\bar{A}}$, qui se traduisent par une responsabilité sur les adversaires qui contribuent à converger le temps.

Même s'il n'y a pas une bisimulation alternante entre une TCGS étendue et la CGS région étendue, on peut montrer que l'équivalence de région préserve les formules TALTL sous la restriction "non-Zeno".

Théorème 4.3

Soient $\mathcal{T}$ une TCGS, et $\mathcal{T}'$ sa TCGS étendue avec les horloges de formule $\mathcal{C}'$. Soit φ une formule TALTL construite à partir des horloges dans $\mathcal{C}'$. Pour deux états région-équivalents (q, v, w) et (q, v', w') de $\mathcal{T}'$, et toutes trajectoires région-équivalentes r et r' commençant par (q, v, w) et (q, v', w'), resp., on a

$$\mathsf{proj}(r), (q, v) \models_z^w \varphi \quad \text{ssi} \quad \mathsf{proj}(r'), (q, v') \models_z^{w'} \varphi.$$

De plus, si φ est une formule d'état, alors ce résultat reste vrai même si on ne tient plus compte de la condition de la région-équivalence entre r et r'.

Démonstration: La même preuve inductive du théorème 4.1 reste valable à l'exception du cas où $\varphi = \langle\langle A \rangle\rangle \psi_p$ pour une formule de chemin ψ_p. Ce cas est traité d'une manière similaire à la preuve du corollaire 4.4. □

On modifie maintenant l'algorithme 3 pour traiter les stratégies non-Zeno. Ceci nécessite la décoration de la CGS région avec des ticks et des blâmes.

Définition 4.30
Soient $\mathcal{T} = \langle \mathsf{Loc}, \mathsf{AP}, \mathsf{Lab}, \delta, \mathcal{C}, \mathsf{Inv}, \mathsf{Agt}, \mathbb{M}, \mathsf{Mov}, \mathsf{Edg} \rangle$ une TCGS, $\mathcal{T}'$ une TCGS étendue avec les horloges dans $\mathcal{C}'$ et l'horloge de tick *z, A une coalition, $\mathcal{E}_A = \langle \mathsf{Loc}^{\mathcal{E}}, AP', \mathsf{Lab}^{\mathcal{E}}, R^{\mathcal{E}}, \mathsf{Agt}, \mathsf{FM}(\mathbb{R}^+, \mathbb{M}), \mathsf{Mov}^{\mathcal{E}}, \mathsf{Edg}^{\mathcal{E}} \rangle$ la CGS étendue associée à $\mathcal{T}'$ et A, et $\mathcal{R} = \langle \mathsf{Loc}^{\mathcal{R}}, \mathsf{AP}, \mathsf{Lab}^{\mathcal{R}}, R^{\mathcal{R}}, \mathsf{Agt}, \mathsf{FM}(\mathbb{N}, \mathbb{M}), \mathsf{Mov}^{\mathcal{R}}, \mathsf{Edg}^{\mathcal{R}} \rangle$ la CGS région associée à $\mathcal{T}'$. La* CGS région étendue *associée à $\mathcal{T}$ et A, notée par $\mathcal{R}_A^{\mathcal{E}} = \langle \mathsf{Loc}', \mathsf{AP}', \mathsf{Lab}', R', \mathsf{Agt}, \mathsf{FM}(\mathbb{N}, \mathbb{M}), \mathsf{Mov}', \mathsf{Edg}' \rangle$, est définie par :*

- $\mathsf{Loc}' = \mathsf{Loc}^{\mathcal{R}} \times \{\mathsf{tick}, \overline{\mathsf{tick}}\} \times \{\mathsf{bl}_A, \mathsf{bl}_{\bar{A}}, \mathsf{bl}_{\mathsf{Agt}}\}$;
- $\mathsf{Lab}'(([(q, v)], t, b)) = (\mathsf{Lab}^{\mathcal{R}}([(q, v)]), t, b)$;
- $R' \subseteq \mathsf{Loc}' \times \mathsf{Loc}'$ *est définie par :* $(([s_1], t_1, b_1), ([s_2], t_2, b_2)) \in R'$ *si et seulement s'il existe* $s_1' \in [s_1]$, $s_2' \in [s_2]$, *et* $d \in \mathbb{R}^+$ *t.q.* $((s_1', t_1, b_1), d, (s_2', t_2, b_2)) \in R^{\mathcal{E}}$;
- Edg' *est définie à partir de* $\mathsf{Edg}^{\mathcal{E}}$ *par : étant donné un ensemble d'abstractions temporisées de choix complets* $((d_l, f_l))_{a_l \in \mathsf{Agt}}$, *ayant* $d_0 = \min\{d_l \mid a_l \in \mathsf{Agt}\}$, *on donne* bl *par :*
  - bl_A *si* $\exists a_i \in A.\ d_i = d_0$ *et* $\forall a_i \in \bar{A}.\ d_i > d_0$;
  - $\mathsf{bl}_{\bar{A}}$ *si* $\exists a_i \in \bar{A}.\ d_i = d_0$ *et* $\forall a_i \in A.\ d_i > d_0$;
  - $\mathsf{bl}_{\mathsf{Agt}}$ *si* $\exists a_i \in A.\ d_i = d_0$ *et* $\exists a_i \in \bar{A}.\ d_i = d_0$;

  Alors si $\mathsf{Edg}^{\mathcal{E}}((s, t, b), ((d_1, f_1), ..., (d_k, f_k))) = ((s, t, b), d, (s', t', b'))$, *on donne*

$$\mathsf{Edg}'(([s], t, b), ((d_1, f_1), ..., (d_k, f_k))) = (([s], t, b), ([s'], t', \mathsf{bl})).$$

Là encore, on doit noter que les blâmes dans $\mathcal{R}_A^{\mathcal{E}}$ ne correspondent pas toujours aux blâmes dans $\mathcal{E}_A$: *e.g.*, quand tous les agents jouent dans la même région, alors la région d'équivalence est trop petite pour pouvoir décider qui blâmer. Dans ce cas, la CGS région étendue blâme tous les joueurs. Pour cette raison précise, la bisimulation de jeu est absente entre la CGS infinie étendue et la CGS région étendue. Il reste possible d'adapter l'algorithme 3 pour faire le model-checking de la logique TALTL sous un contexte non-Zeno. L'algorithme 4 montre (une partie de) la procédure en question.

On montre maintenant la correction de cet algorithme :

Procedure nonzeno-labeling($\mathcal{R}$, ψ) **begin**
  foreach $[(q, v, w)] \in \mathcal{R}$ **do**
    $T([(q, v, w)], \psi) = \bot$;
  switch ψ **do**
    ...
    case $\langle\langle A \rangle\rangle \psi_p$:
      $\hat{\psi}_p = \psi_p$;
      $\hat{\mathcal{R}}' = \mathcal{R}'$;
      foreach $\psi' \in$ **State-Subf**(ψ_p) **do**
        nonzeno-labeling($\mathcal{R}$,ψ') ;
        $\hat{\psi}_p$ = replace($\hat{\psi}_p$, $\psi' \rightarrow p_{\psi'}$) ;
        $\hat{\mathcal{R}}'$ = decorate($\hat{\mathcal{R}}'$,$p_{\psi'}$) ;
      $\mathcal{R}_A^{\mathcal{E}}$ = build-model ($\hat{\mathcal{R}}'$,A) ;
      non-Zeno(ψ) = $\langle\langle A \rangle\rangle ((\overset{\infty}{\mathbf{F}} \neg \mathsf{bl}_{\bar{A}} \rightarrow \overset{\infty}{\mathbf{F}}\, \mathsf{tick}) \wedge (\overset{\infty}{\mathbf{F}}\, \mathsf{tick} \rightarrow \hat{\psi}_p))$;
      T' = ATLstar-labeling($\mathcal{R}_A^{\mathcal{E}}$,non-Zeno($\psi$)) ;
      foreach $[(q, v, w)] \in \mathcal{R}'$ **do**
        if $(T([(q, v, w), \overline{\mathsf{tick}}, bl_A)], \textit{non-Zeno}(\psi)) = \top)$ **then**
          $T([(q, v, w)], \psi) = \top$;
end

Algorithme 4 : Étiquetage de $\mathcal{R}$ avec des formules d'états ψ non-Zeno

Théorème 4.4

Soient $\mathcal{T}$ une TCGS, et $\mathcal{T}'$ la TCGS étendue correspondante avec l'ensemble des horloges de formule $\mathcal{C}'$. Soit φ une formule TALTL construite à partir des horloges dans $\mathcal{C}'$. Soit T la table obtenue après l'exécution de l'algorithme 4 sur $\mathcal{R}$ et φ. Soit (q, v, w) un état dans $\mathcal{T}'$. Alors, $(q, v) \models_z^w \varphi$ ssi $T([(q, v, w)], \varphi) = \top$.

Démonstration: D'une manière similaire à la preuve du théorème 4.2, la preuve est par induction sur la structure de la formule. Le seul cas intéressant est quand $\varphi = \langle\langle A \rangle\rangle \psi_p$. Dans cette preuve, on note par non-Zeno(ψ_p) la formule de chemin $((\overset{\infty}{\mathbf{F}} \neg \mathsf{bl}_{\bar{A}} \rightarrow \overset{\infty}{\mathbf{F}}\, \mathsf{tick}) \wedge (\overset{\infty}{\mathbf{F}}\, \mathsf{tick} \rightarrow \psi_p))$, et par $\hat{\psi}_p$ la formule LTL obtenue en remplaçant les formules d'état de ψ_p par leur proposition atomique correspondante. Soit $\mathcal{E}_A$ et $\mathcal{R}_A^{\mathcal{E}}$ la CGS étendue et la CGS région étendue pour $\mathcal{T}'$ et A.

On suppose que $(q, v) \models_z^w \varphi$. Respecter la sémantique non-Zeno, nous indique que $((q, v, w), \overline{\mathsf{tick}}, \mathsf{bl}_A) \models_z^w \mathcal{E}_A \varphi$. En choisissant une stratégie λ_A telle que, pour tout outcome $\rho \in \mathsf{Out}_{\mathcal{E}_A}\left(((q, v, w), \overline{\mathsf{tick}}, \mathsf{bl}_A), \lambda_A\right)$ satisfait non-Zeno(ψ_p). On montre que $T([(q, v, w)], \varphi) = \top$ en présentant une stratégie gagnante λ'_A pour A dans la CGS région étendue $\mathcal{R}_A^{\mathcal{E}}$ pour l'objectif non-Zeno($\hat{\psi}_p$). On construit λ'_A en deux étapes. La première étape consiste à construire la stratégie λ''_A définie dans le corollaire 4.4. On sait que pour cette stratégie tous les outcomes infinis $\rho'' \in \mathsf{Out}(((q, v, w), \overline{\mathsf{tick}}, \mathsf{bl}_A), \lambda''_A)$ vérifient non-Zeno(ψ_p). Et puisque λ''_A est région-uniforme et région-invariante, en suivant la preuve de la proposition 4.3, on peut construire la stratégie λ'_A sur $\mathcal{R}_A^{\mathcal{E}}$ à partir de λ''_A.

Pour une contradiction, supposant que λ'_A n'est pas gagnante, *i.e.*, il existe donc une exécution $\rho' \in \mathsf{Out}\left((((q,v,w),\overline{\mathsf{tick}},\mathsf{bl}_A)),\lambda'_A\right)$ qui ne satisfait pas $\mathsf{non\text{-}Zeno}(\hat{\psi_p})$. Soit ρ un outcome de λ''_A t.q. $[\rho]=\rho'$. Par construction de λ'_A, le seul cas possible est quand ρ' est Zeno et que la coalition A est à blâmer (sinon c'est que l'exécution est non-Zeno et qu'elle ne réussit pas à satisfaire $\hat{\varphi_p}$, mais ρ pourrait être aussi non-Zeno et ne satisfaisant pas φ_p, ce qui contredit que λ''_A est gagnante). Ceci indique que l'exécution ρ' contient un nombre fini de ticks, et un nombre fini de $\mathsf{bl}_{\bar{A}}$, mais un nombre infini de bl_A ou un nombre infini de $\mathsf{bl}_{\mathsf{Agt}}$. Dans le premier cas, ρ pourrait être aussi Zeno, et pourrait contenir un nombre infini de bl_A, ce qui contredit que λ''_A est gagnante. Dans le cas où le nombre de $\mathsf{bl}_{\mathsf{Agt}}$ est infini, on déduit que A et $\bar{A}$ ont été choisis infiniment souvent pour jouer dans la même région. Il est donc possible de construire un outcome $\tilde{\rho}$, avec $[\tilde{\rho}]=\rho'$, et le long duquel $\bar{A}$ choisit toujours le même délai que A. Puisque λ''_A est région-invariante, $\tilde{\rho}$ reste un outcome de λ''_A, mais $\tilde{\rho}$ n'est pas gagnant, ce qui révèle une contradiction. Ce qui prouve que λ'_A est gagnante. Et donc la procédure `ATLstar-labeling` donne la valeur $\top$ à $T([(q,v,w)],\varphi)$.

Réciproquement, supposant que $T([(q,v,w)],\varphi)=\top$. Alors, il existe une stratégie λ_A dans $\mathcal{R}^{\mathcal{E}}_A$ t.q. tous les outcomes de λ_A depuis $[((q,v,w),t,b)]$ satisfont $\mathsf{non\text{-}Zeno}(\hat{\psi_p})$. De la même manière que la preuve de la proposition 4.4, on peut transformer la stratégie λ_A en une stratégie λ'_A dans $\mathcal{E}_A$. Dans ce cas, étant donné $\rho' \in \mathsf{Out}(((q,v',w'),\overline{\mathsf{tick}},\mathsf{bl}_A),\lambda'_A)$, il existe $\rho \in \mathsf{Out}([((q,v,w),t,b)],\lambda_A)$ t.q. $\pi(\rho)=[\pi(\rho')]$, et avec ρ' visitant un état étiqueté par $\mathsf{bl}_{\bar{A}}$ ou $\mathsf{bl}_{\mathsf{Agt}}$ à chaque fois que ρ visite $\mathsf{bl}_{\bar{A}}$, et donc ρ' satisfait $\mathsf{non\text{-}Zeno}(\psi_p)$, et $(q,v) \models^{w}_{z} \mathcal{T}\varphi$. □

Théorème 4.5

Sous la sémantique non-Zeno, le problème du model-checking de la logique TALTL sur les TCGS est décidable et est complet pour 2EXPTIME.
Le model-checking de la logique $\mathsf{TATL}_{\mathcal{C}}$ sur les TCGS se fait dans EXPSPACE et il est EXPTIME-dur.

Démonstration: On explique seulement l'algorithme EXPTIME pour le model-checking de $\mathsf{TATL}_{\mathcal{C}}$. En remarquant que $\langle\!\langle A \rangle\!\rangle\, \mathsf{non\text{-}Zeno}(\psi_p)$ est une formule FairATL avec des conditions d'équité, elle peut donc être vérifiée dans un temps PTIME dans la taille de la formule et du modèle [AHK02]. Dans notre cas, le modèle est exponentiel dans $|\varphi|$ et $|\mathcal{T}|$. □

Conclusion

Nous avons introduit les TCGS, des structures de jeu avec une représentation continue du temps. Pour ces structures, nous avons montré que des stratégies uniformes et invariantes par régions sont suffisantes pour gagner des objectifs définissables par région. Nous avons ensuite montré que la complexité du model-checking de la logique $\mathsf{TATL}_{\mathcal{C}}$ se fait dans EXPSPACE. Nous avons défini une logique plus riche qui étend $\mathsf{TATL}_{\mathcal{C}}$, nommée TALTL. À l'aide de cette nouvelle logique, on peut spécifier des propriétés comprenant du temps et des conditions d'équité tout en bénificiant d'un model-checking décidable.

La table 4.1 récapitule ces résultats ($|\mathcal{T}|$ désigne la taille de la TCGS et $|\varphi|$ dénote la taille de la formule).

	Comportement asymptotique	Classe de complexité	
ATL*	$2^{2^{\mathcal{O}(\lvert\varphi\rvert)}} \cdot 2^{2^{\mathcal{O}(\lvert\mathcal{T}\rvert)}}$	2EXPTIME-complet	
TATL$_\mathcal{C}$	$2^{\mathcal{O}(\lvert\varphi\rvert)\cdot\mathcal{O}(\lvert\mathcal{T}\rvert)}$	EXPTIME-dur	EXPSPACE
TALTL	$2^{2^{\mathcal{O}(\lvert\varphi\rvert)}} \cdot 2^{2^{\mathcal{O}(\lvert\mathcal{T}\rvert)}}$	2EXPTIME-complet	

TAB. 4.1 – La complexité du model-checking des différentes logiques sur les TCGS.

Nous avons également étudié le model-checking des logiques TATL$_\mathcal{C}$ et TALTL dans le même cadre non-Zeno dans [HP06, dAFH$^+$03] et avons montré que la complexité reste inchangée.

Pour en savoir plus ...

Le modèle des TCGS est une extension des automates temporisés avec l'aspect de jeux. Or on peut trouver plusieurs extensions d'automates temporisés. On peut citer par exemple les automates temporisés probabilistes [JLS07, Jen96], où on trouve une distribution de probabilité sur les transitions. Une autre extension, les automates temporisés pondérés, ajoute du poids sur les transitions qui peuvent désigner un coût à payer qui dépend du temps passé dans les états [ALTP01, BFH$^+$01]. D'autres extensions qui décorent les TGA avec des prix ont été étudiées dans les papiers [BLM08, BLMR06].

Conclusion et perspectives

Contribution de la thèse

Nous nous sommes intéressés à l'étude de la logique temporelle du temps alternant (ATL). Nous avons étudié différents aspects de cette logique.

Nous avons étudié le model-checking de cette logique dans le cas non-temporisé et nous avons montré que ce problème pour la logique ATL dépend du type de structure de jeux considéré. En effet, sur les trois différentes variantes des structures de jeu, les CGS bénéficient d'un algorithme polynomial si la table de transitions est donnée explicitement, dans le cas contraire (c.à.d. si on donne la table d'une manière symbolique) la complexité augmente pour devenir Δ_3^{P}. La complexité de la dernière structure étudiée, les ATS est Δ_2^{P}, cette classe est située entre les deux autres classes de complexité selon la hiérarchie polynomiale.
Nous avons aussi étudié des questions d'expressivité, et nous avons montré qu'il était nécessaire de considérer l'opérateur "Release" dans la définition de ATL.

Ensuite, nous avons proposé une extension des CGS, en ajoutant du temps discret. Le temps, dans ces nouveaux modèles —nommés TDCGS— s'écoule dans les transitions, une durée entière positive indique la durée dans chaque transition, et le nouvel état du système est contrôlé par le choix de tous les joueurs. Une version temporisée de la logique ATL a été présentée pour spécifier des propriétés quantitatives. Dans le cas général, nous avons montré que la complexité du problème du model-checking de la logique TATL sur les TDCGS est EXPTIME-complet. Cette complexité est fortement liée aux contraintes de temps utilisées dans les formules. Effectivement, si on ne considère que le fragment de la logique sans les contraintes d'égalité, c.à.d. des formules de la forme $\langle\langle A\rangle\rangle\, \varphi_1\, \mathbf{U}_{>d}\, \varphi_2$ ou $\langle\langle A\rangle\rangle\, \varphi_1\, \mathbf{U}_{<d}\, \varphi_2$, la complexité du model-checking se réduit considérablement pour avoir une complexité polynomiale.
Étendre les TDCGS avec des intervalles d'entiers sur les transitions (au lieu d'avoir une durée unique) est un modèle intéressant (nommé CGS avec durées), et surtout cela ne change pas la complexité théorique du problème du model-checking. On obtient donc un algorithme EXPTIME qui traite le cas général, et un algorithme polynomial quand on interdit les contraintes d'égalité dans la logique.

Finalement, nous avons introduit des modèles temporisés sur les jeux, nommés TCGS. L'avantage de ces modèles est qu'ils sont "plus concurrents" à ce qui a été déjà introduit dans la littérature : une transition résulte du choix d'un coup pour chaque agent. Un choix d'un joueur n'est pas seulement une action et une durée où il souhaite exécuter son action, mais aussi les actions si son choix n'a pas été pris. Nous avons montré que des algorithmes basés sur les régions sont suffisants pour faire du model-checking des propriétés $\mathsf{TATL}_{\mathcal{C}}$, une logique

temporelle temporisée, mais à la différence de TATL, elle traite des propriétés sur des systèmes de jeux à temps dense à l'aide des horloges.

Ensuite, nous avons introduit une extension de cette logique, la logique TALTL qui est plus expressive que la logique $\mathsf{TATL}_{\mathcal{C}}$ puisqu'elle peut spécifier des propriétés temporisées avec des propriétés d'équité. Nous avons montré que le model-checking de cette logique sur les TCGS reste décidable et est 2EXPTIME-complet.

Perspectives

Les directions qui sont envisageables pour continuer ces travaux peuvent être divisées en deux parties, une concernant des études théoriques, et l'autre s'intéressant à l'implémentation :

Études théoriques. Plusieurs pistes sont possibles pour étendre le modèle de la TCGS. Celles-ci peuvent comprendre :

- La synthèse de la stratégie n'a pas fait partie de la thèse. Puisque nous connaissons déjà que les stratégies à base de régions sont suffisantes, nous pouvons étudier la synthèse de stratégie en se basant sur les travaux [HRS05].
- Nous souhaitons étudier de la même manière le sujet de la robustesse [Pur98, DDMR04] où on peut trouver des stratégies dans les jeux temporisés où mesurer le temps ne se fait pas avec une précision infinie.
- L'étude de la possibilité d'étendre la logique $\mathsf{TATL}^\star$ et ses fragments pour avoir une logique préservant le contexte des stratégies d'une manière similaire à ce qui a été étudié pour les logiques sur les jeux dans le cadre non temporisé [CHP07, Pin07, BDLM08].

Implémentation. Le model-checking de la logique TATL sur les TCGS est basé sur le graphe de régions. Malheureusement, en pratique, la construction du graphe des régions est trop coûteux pour l'implémenter dans un outil. Il serait donc nécessaire, pour avoir un outil, de manipuler des zones et des DBM de la même façon que pour les automates temporisés [Dil90, DT98].

Annexe A Les classes de complexité [Pap94]

Les complexités des algorithmes du model-checking que nous avons étudiées ne sont pas toujours dans les classes de complexité classiques. Dans la suite de cette partie, on définit les classes de complexité de la hiérarchie polynomiale ainsi que la classe de complexité AC^0.

A.1 La hiérarchie polynomiale

La hiérarchie polynomiale contient des classes de complexité qui sont contenues dans la classe PSPACE et qui contiennent la classe PTIME. Pour décrire ces classes, on introduit la notion des machines de Turing avec *oracle*.

Définition A.1
Une machine de Turing avec oracle A, *notée* M^A, *est une machine de Turing multi-rubans avec un ruban spécial qu'on appelle "ruban de l'oracle" et trois états spéciaux* q_{oracle}, *l'état de l'oracle,* q_{oui} *et* q_{non}, *les états de réponse. Le calcul dans les machines de Turing avec oracle procède de la même façon qu'une machine de Turing classique, sauf quand il s'agit d'une transition issue de l'état de l'oracle. De ce dernier état, notre machine* M^A *passe soit à l'état* q_{oui}, *soit à l'état* q_{non} *si le mot contenu dans le ruban de l'oracle appartient au langage de* A *ou non. La complexité en temps est calculée de la même manière qu'une machine de Turing classique ; chaque appel à l'oracle est compté comme une seule étape.*

Définition A.2
La hiérarchie polynomiale est l'ensemble de classes de complexité définie par les relations suivantes :

- *Premièrement,* $\Delta_0^{\mathsf{P}} \stackrel{def}{=} \Sigma_0^{\mathsf{P}} \stackrel{def}{=} \Pi_0^{\mathsf{P}} \stackrel{def}{=} \mathsf{PTIME}$
- *Pour tout* $i \geq 0$, *on a :*
  - $\Delta_{i+1}^{\mathsf{P}} \stackrel{def}{=} \mathsf{PTIME}^{\Sigma_i^{\mathsf{P}}}$
  - $\Sigma_{i+1}^{\mathsf{P}} \stackrel{def}{=} \mathsf{NP}^{\Sigma_i^{\mathsf{P}}}$
  - $\Pi_{i+1}^{\mathsf{P}} \stackrel{def}{=} \mathsf{coNP}^{\Sigma_i^{\mathsf{P}}}$

D'après cette définition, on peut conclure que $\Delta_1^{\mathsf{P}} = \mathsf{PTIME}$, $\Sigma_1^{\mathsf{P}} = \mathsf{NP}$ *et* $\Pi_1^{\mathsf{P}} = \mathsf{coNP}$. *On a donc les inclusions suivantes :*

$$\Delta_i^{\mathsf{P}} \subseteq \Pi_i^{\mathsf{P}} \cap \Sigma_i^{\mathsf{P}} \qquad \Sigma_i^{\mathsf{P}} \subseteq \Delta_{i+1}^{\mathsf{P}} \qquad \Pi_i^{\mathsf{P}} \subseteq \Delta_{i+1}^{\mathsf{P}}$$

La figure A.1 illustre une représentation de cette hiérarchie.

A.1.1 Quelques problèmes caractérisant la hiérarchie polynomiale

On présente quelques problèmes qui sont connus d'être complets dans des classes de complexité de la hiérarchie polynomiale.

Définition A.3
Le problème EQSAT$_i$ (Satisfaisabilité quantifiée avec i alternations de quantificateurs, commençant par une quantification existentielle) : étant donnée une expression booléenne φ avec des variables booléennes réparties dans i ensembles $X_1, \ldots, X_i$, est-ce que pour toutes les affections partielles des variables X_1 il existe une affectation partielle de variables dans X_2 telle que toute affectation partielle de variables dans X_3, etc. , telle que φ est satisfaite par l'affectation totale ? Une instance de EQSAT$_i$ est représentée par :

$$\exists X_1 . \forall X_2 . \exists X_3 . \ldots Q X_i . \varphi$$

avec Q est le quantificateur $\exists$ si i est impair et $\forall$ sinon.

Le problème dual (noté AQSAT$_i$) est défini par la formule $\forall X_1 . \exists X_2 . \forall X_3 . \ldots Q X_i . \varphi$.

Un algorithme Σ_i^{P} pour résoudre les problèmes EQSAT$_i$ utilise une machine de Turing NP avec un oracle Π_{i-1}^{P} pour résoudre la formule AQSAT$_{i-1}$ $\psi = \forall X_2 . \exists X_3 . \ldots Q X_i . \varphi$. On montre que cet algorithme est optimal :

Théorème 1.1 [Pap94]

Pour tout $i \geq 1$, le problème EQSAT$_i$ est complet pour Σ_i^{P}.

Il est donc évident que le problème dual AQSAT$_i$ est complet pour Π_i^{P}.

Le problème SNSAT$_k$ suivant est lié à la classe de complexité Δ_k^{P} :

Définition A.4
Le problème SNSAT$_k$ est défini par k ensembles de m familles de variables $X^l = \{X_1^l, \ldots, X_m^l\}$ avec $X_i^l = \left\{x_{i,1}^l, \ldots, x_{i,n}^l\right\}$, m variables z_i, m formules booléennes φ_i, les variables qui apparaissent dans φ_i sont dans $X_i^1 \cup \cdots \cup X_i^k \cup \{z_1, \ldots, z_{i-1}\}$.

La réponse est déterminée par la valeur de z_m définie par :

$$\begin{cases} z_1 \stackrel{def}{=} \exists X_1^1 . \forall X_1^2 \ldots Q X_1^k . \varphi_1(X_1^1, \ldots, X_1^k) \\ z_2 \stackrel{def}{=} \exists X_2^1 . \forall X_2^2 \ldots Q X_2^k . \varphi_2(z_1, X_2^1, \ldots, X_2^k) \\ \quad \ldots \\ z_m \stackrel{def}{=} \exists X_m^1 . \forall X_m^2 \ldots Q X_m^k . \varphi_m(z_1, ..., z_{m-1}, X_m^1, \ldots, X_m^k) \end{cases}$$

Pour pouvoir calculer la sortie z_m, on a besoin de faire m appels à un problème EQSAT$_k$. Ce qui nous donne :

Théorème 1.2

Pour tout $k \geq 1$, le problème SNSAT$_k$ est $\Delta_{k+1}^{\mathsf{P}}$-complet.

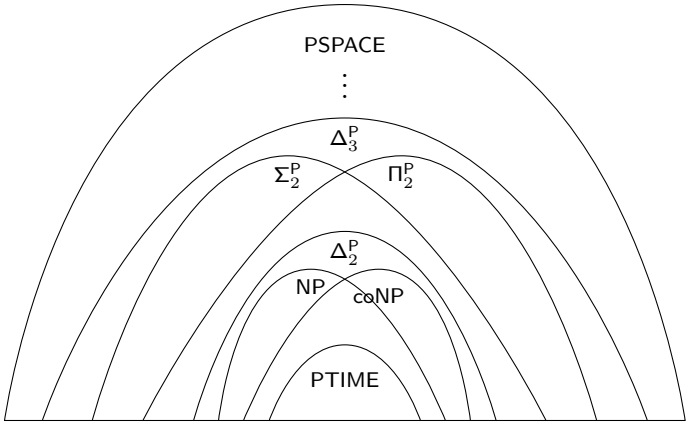


FIG. A.1 – La hiérarchie polynomiale

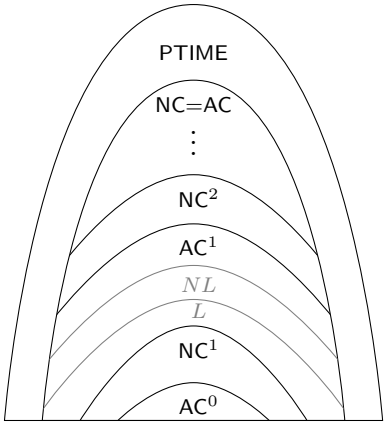


FIG. A.2 – La hiérarchie AC

A.2 La hiérarchie AC

Définition A.5
On appelle un circuit avec un nombre borné d'entrées *s'il est construit avec des portes logiques dont le nombre d'entrées est borné. Les portes qui sont autorisées sont les portes ET, les portes OU et les portes de négation NON (qui possèdent une entrée unique). Si le nombre d'entrées pour ces portes est non borné on appelle ce circuit un* circuit avec un nombre non borné d'entrées.

Exemple A.1
Dans la figure A.3, on cherche à construire la conjonction de n bits avec un circuit ayant un nombre d'entrées borné. On remarque que la profondeur de ce circuit est de $\log_2(n)$. Si on utilise un circuit avec un nombre d'entrées borné, on obtient le circuit de la figure A.4, et la profondeur se réduit à 1 avec la seule porte logique utilisée.

On peut maintenant définir la hiérarchie AC :

Définition A.6
La hiérarchie AC *caractérise les problèmes qui peuvent être résolus à l'aide de circuits avec un nombre d'entrées non borné.* —i.e. *dont les composantes sont des portes OU (ou ET) qui possèdent un nombre non borné d'entrées, et le résultat sur la sortie se fait en calculant la disjonction (ou la conjonction) de toutes les entrées en une seule étape.* — *Pour $i \geq 0$, on définit AC^i comme la classe des langages qui sont décidés par des familles uniformes de circuits (des circuits qui sont calculés par une machine de Turing bornée par une espace $\log n$ pour une entrée de taille n) de taille polynomiale ayant un nombre d'entrées non borné, et avec une profondeur $\mathcal{O}(\log^i n)$ (la profondeur est donc constante pour $i = 0$). AC est définie comme l'union de toutes les classes AC^i.*

On a les inclusions suivantes :

$$\mathsf{AC}^0 \subseteq \mathsf{LOGSPACE} \subseteq \mathsf{NLOGSPACE} \subseteq \mathsf{AC}^1$$

On représente la hiérarchie AC dans la figure A.2.

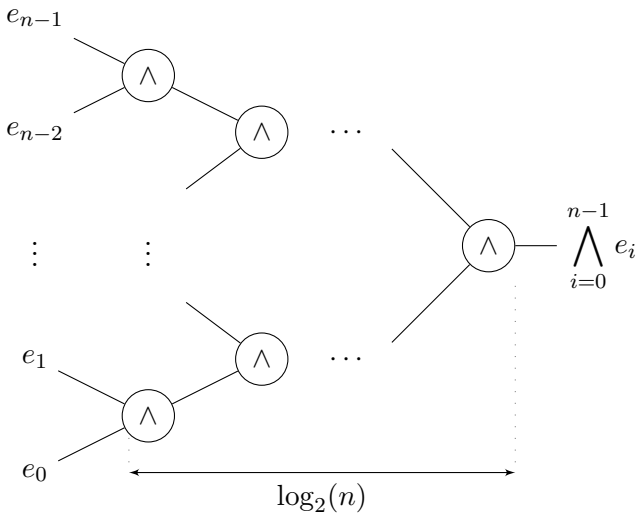


FIG. A.3 – Circuit logique avec un nombre d'entrées borné

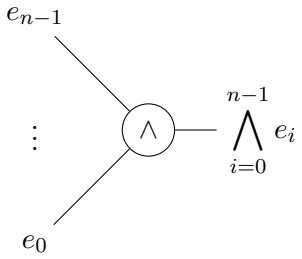


FIG. A.4 – Circuit logique avec un nombre d'entrées non borné

Exemple A.2 ([Vol99])
Calculer la somme de deux nombres de n-bits est dans AC^0. *On suppose qu'on souhaite construire un circuit pour calculer la somme de deux entrées binaires de n-bits* $a = a_{n-1} \cdots a_1 a_0$ *et* $b = b_{n-1} \cdots b_1 b_0$. *Il faut calculer la retenue. On sait qu'une retenue est générée en une position i si et seulement si les deux bits des entrées* a_i *et* b_i *sont* 1. *Ceci conduit aux définitions suivantes : Pour tout* $0 \leq i < n$, *on prend :*

$g_i = a_i \wedge b_i$ *(la position i génère une retenue)*
$p_i = a_i \vee b_i$ *(la position i peut propager une retenue)*

Maintenant, une retenue passe à une position i si et seulement s'il y a une position $j < i$ *qui génère une retenue, et si toutes les positions intermédiaires la propagent. Formellement :*

$$c_i = \bigvee_{j=0}^{i-1} \left(g_j \wedge \bigwedge_{k=j+1}^{i-1} p_k \right) \qquad \text{avec } 1 \leq i \leq n$$

À l'aide de la retenue calculée, on peut maintenant calculer les bits de la sortie : $s_0 = a_0 \oplus b_0$, $s_i = a_i \oplus b_i \oplus c_i$ *pour* $i \geq 1$. *On peut observer que ces formules nous laissent calculer les* c_i *en parallèle puisqu'elles ne dépendent que des entrées a et b. Et quand les bits des retenues seront calculés, les bits de la sortie s pourront être calculés également en parallèle. On peut donc calculer tous les* p_i *et les* g_i *en une unité de temps, et puis* c_i *dans une deuxième étape de telle manière que la somme pourrait être calculée dans un temps constant.*

Bibliographie

[ACD93] Rajeev Alur, Costas Courcoubetis, and David L. Dill. Model-checking in dense real-time. *Information & Computation*, 104(1) :2–34, 1993.

[AD90] R. Alur and D. Dill. Automata for modeling real-time systems. In *ICALP'90*, volume 443 of *LNCS*, pages 322–335. Springer, 1990.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2) :183–235, 1994.

[AH89] Rajeev Alur and Thomas A. Henzinger. A really temporal logic. In *Journal of the ACM*, pages 164–169. IEEE Computer Society Press, 1989.

[AH92] R. Alur and T.A. Henzinger. Logics and models of Real-Time : a survey. In *Proc. REX Workshop Real-Time : Theory in Practice*, volume 600 of *LNCS*, pages 74–106. Springer, 1992.

[AH93] R. Alur and T.A. Henzinger. Real-time logics : complexity and expressiveness. *Information and computation(Print)*, 104(1) :35–77, 1993.

[AH94] R. Alur and T.A. Henzinger. A really temporal logic. *J. ACM*, 41(1) :181–204, 1994.

[AHK97] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS'97)*, pages 100–109. IEEE Comp. Soc. Press, 1997.

[AHK98] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. In Willem-Paul de Roever, Hans Langmaack, and Amir Pnueli, editors, *Revised Lectures of the 1st International Symposium on Compositionality : The Significant Difference (COMPOS'97)*, volume 1536 of *LNCS*, pages 23–60. Springer, 1998.

[AHK02] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5) :672–713, 2002.

[AHKV98] Rajeev Alur, Thomas A. Henzinger, Orna Kupferman, and Moshe Y. Vardi. Alternating refinement relations. In Davide Sangiorgi and Robert de Simone, editors, *Proceedings of the 9th International Conference on Concurrency Theory (CONCUR'98)*, volume 1466 of *LNCS*, pages 163–178. Springer, 1998.

[AI01] Micah Adler and Neil Immerman. An $n!$ lower bound on formula size. In *Proceedings of the 16th Annual Symposium on Logic in Computer Science (LICS'01)*, pages 197–206. IEEE Comp. Soc. Press, 2001.

[ALTP01] Rajeev Alur, Salvatore La Torre, and George J. Pappas. Optimal paths in weighted timed automata. In *HSCC '01 : Proceedings of the 4th International Workshop on Hybrid Systems*, pages 49–62, London, UK, 2001. Springer-Verlag.

[AM99] E. Asarin and O. Maler. As soon as possible : Time optimal control for timed automata. In *HSCC'99*, volume 1569 of *LNCS*, pages 19–30. Springer, 1999.

[AMPS98] Eugene Asarin, Oded Maler, Amir Pnueli, and Joseph Sifakis. Controller synthesis for timed automata. In *Proc IFAC Symp. on System Structure and Control*. Elsevier, 1998.

[ATM03] Rajeev Alur, Salvatore La Torre, and P. Madhusudan. Playing games with boxes and diamonds. In Roberto M. Amadio and Denis Lugiez, editors, *CONCUR*, volume 2761 of *Lecture Notes in Computer Science*, pages 127–141. Springer, 2003.

[BCM05] Patricia Bouyer, Fabrice Chevalier, and Nicolas Markey. On the expressiveness of TPTL and MTL. In R. Ramanujam and Sandeep Sen, editors, *Proceedings of the 25th Conference on Fundations of Software Technology and Theoretical Computer Science (FSTTCS'05)*, volume 3821 of *Lecture Notes in Computer Science*, pages 432–443, Hyderabad, India, December 2005. Springer.

[BDLM08] Thomas Brihaye, Arnaud Da Costa, François Laroussinie, and Nicolas Markey. ATL with strategy contexts and bounded memory. Research Report LSV-08-14, Laboratoire Spécification et Vérification, ENS Cachan, France, April 2008. 22 pages.

[BDM+98] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. Kronos : A model-checking tool for real-time systems. In A. J. Hu and M. Y. Vardi, editors, *Proc. 10th International Conference on Computer Aided Verification, Vancouver, Canada*, volume 1427, pages 546–550. Springer-Verlag, 1998.

[BFH+01] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim Guldstrand Larsen, Paul Pettersson, Judi Romijn, and Frits W. Vaandrager. Minimum-cost reachability for priced timed automata. In *HSCC '01 : Proceedings of the 4th International Workshop on Hybrid Systems*, pages 147–161, London, UK, 2001. Springer-Verlag.

[BGMR08] Thomas Brihaye, Mohamed Ghannem, Nicolas Markey, and Lionel Rieg. Good friends are hard to find ! In Stéphane Demri and Christian S. Jensen, editors, *Proceedings of the 15th International Symposium on Temporal Representation and Reasoning (TIME'08)*, pages 32–40, Montréal, Canada, June 2008. IEEE Computer Society Press.

[BLM08] Patricia Bouyer, Kim G. Larsen, and Nicolas Markey. Model checking one-clock priced timed automata. *Logical Methods in Computer Science*, 4(2 :9), June 2008.

[BLMO07] Thomas Brihaye, François Laroussinie, Nicolas Markey, and Ghassan Oreiby. Timed concurrent game structures. In Luís Caires and Vasco T. Vasconcelos, editors, *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR'07)*, volume 4703 of *Lecture Notes in Computer Science*, pages 445–459, Lisbon, Portugal, September 2007. Springer.

[BLMR06] Patricia Bouyer, Kim G. Larsen, Nicolas Markey, and Jacob Illum Rasmussen. Almost optimal strategies in one-clock priced timed automata. In Naveen Garg

and S. Arun-Kumar, editors, *Proceedings of the 26th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'06)*, volume 4337 of *Lecture Notes in Computer Science*, pages 345–356, Kolkata, India, December 2006. Springer.

[CC95] Sérgio Campos and Edmund M. Clarke. Real-time symbolic model checking for discrete time models. In *Theories and Experiences for Real-Time System Development*, volume 2 of *AMAST Series in Computing*, pages 129–145. World Scientific, 1995.

[CCGR99] A. Cimatti, E.M. Clarke, F. Giunchiglia, and M. Roveri. NuSMV : A new symbolic model verifier. In Nicolas Halbwachs and Doron A. Peled, editors, *CAV'99*, volume 1633 of *LNCS*, pages 495–499. Springer, July 1999.

[CDF+05] Franck Cassez, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. Efficient on-the-fly algorithms for the analysis of timed games. In Martín Abadi and Luca de Alfaro, editors, *Proc. 16th Intl Conf. on Concurrency Theory (CONCUR'05)*, volume 3653 of *LNCS*, pages 66–80. Springer, August 2005.

[CE81] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronous skeletons using branching-time temporal logic. In Dexter Kozen, editor, *Proceedings of the 3rd Workshop on Logics of Programs (LOP'81)*, volume 131 of *LNCS*, pages 52–71. Springer, 1981.

[CES86] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2) :244–263, 1986.

[CHJ05] Krishnendu Chatterjee, Thomas A. Henzinger, and Marcin Jurdziński. Mean-payoff parity games. In *Proceedings, Twentieth Annual IEEE Symposium on Logic in Computer Science, LICS 2005*, pages 178–187. IEEE Computer Society Press, 2005.

[CHP07] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Strategy logic. In Luís Caires and Vasco Thudichum Vasconcelos, editors, *CONCUR*, volume 4703 of *Lecture Notes in Computer Science*, pages 59–73. Springer, 2007.

[dAFH+03] Luca de Alfaro, Marco Faella, Thomas A. Henzinger, Rupak Majumdar, and Mariëlle Stoelinga. The element of surprise in timed games. In Roberto Amadio and Denis Lugiez, editors, *Proc. 14th Intl Conf. on Concurrency Theory (CONCUR'03)*, volume 2761 of *LNCS*, pages 142–156. Springer, August 2003.

[dAFS04] L. de Alfaro, M. Faella, and M. Stoelinga. Linear and branching metrics for quantitative transition systems. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP'04)*, volume 3142 of *Lecture Notes in Computer Science*, pages 1150–1162, Turku, Finland, July 2004. Springer.

[dAHM00] L. de Alfaro, T.A. Henzinger, and FYC Mang. The Control of Synchronous Systems. *Proceedings of the 11th International Conference on Concurrency Theory (CONCUR 2000), Lecture Notes in Computer Science*, pages 458–473, 2000.

[dAHM01] Luca de Alfaro, Thomas A. Henzinger, and Rupak Majumdar. From verification to control : dynamic programs for omega-regular objectives. In *Proceedings of the*

16th Annual IEEE Symposium on Logic in Computer Science (LICS'01), pages 279–290. IEEE Comp. Soc. Press, 2001.

[DDMR04] M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robustness and implementability of timed automata. In *FORMATS-FTRTFT'04*, volume 3253 of *LNCS*, pages 118–133. Springer, 2004.

[Dil90] D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Proceedings of the international workshop on Automatic verification methods for finite state systems*, pages 197–212, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[DOTY96] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In R. Alur, Thomas A. Henzinger, and Eduardo D. Sontag, editors, *Hybrid Systems III : Verification and Control*, volume 1066 of *LNCS*, pages 208–219. Springer, 1996.

[DT98] Conrado Daws and Stavros Tripakis. Model checking of real-time reachability properties using abstractions. In *TACAS '98 : Proceedings of the 4th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, pages 313–329, London, UK, 1998. Springer-Verlag.

[EH85] E. A. Emerson and Joseph Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *J. Comput. Syst. Sci.*, 30(1) :1–24, 1985.

[EH86] E. Allen Emerson and Joseph Y. Halpern. "sometimes" and "not never" revisited : on branching versus linear time temporal logic. *J. ACM*, 33(1) :151–178, 1986.

[Eme90] E. Allen Emerson. Temporal and modal logic. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 16, pages 995–1072. Elsevier, 1990.

[EMSS92] E. Allen Emerson, Aloysius Ka-Lau Mok, A. Prasad Sistla, and Jayashankar Srinivasan. Quantitative temporal reasoning. *Real-Time Systems*, 4 :331–352, 1992.

[FGK08] Diana Fischer, Erich Gradel, and Lukasz Kaiser. Model checking games for the quantitative mu-calculus. In Susanne Albers and Pascal Weil, editors, *Proceedings of the 25th Annual Symposium on Theoretical Aspects of Computer Science (STACS'08)*, pages 301–312, February 2008.

[GJ04] Valentin Goranko and Wojciech Jamroga. Comparing semantics of logics for multi-agent systems. *Synthese*, 139(2), 2004.

[GSSAL94] R. Gawlick, R. Segala, J.F. Søgaard-Andersen, and N.A. Lynch. Liveness in timed and untimed systems. In *ICALP'94*, volume 820 of *LNCS*, pages 166–177. Springer, 1994.

[GvD06] Valentin Goranko and Govert van Drimmelen. Complete axiomatization and decidability of alternating-time temporal logic. *Theoretical Computer Science*, 353(1-3) :93–117, March 2006.

[HNSY94] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real time systems. *Inf. & Comp.*, 111(2) :193–244, 1994.

[HP06] T.A. Henzinger and V. Prabhu. Timed alternating-time temporal logic. In *FORMATS'06*, volume 4202 of *LNCS*, pages 1–17. Springer, 2006.

[HRS02] Aidan Harding, Mark Ryan, and Pierre-Yves Schobbens. Approximating ATL* in ATL. In Agostino Cortesi, editor, *Revised Papers of the 3rd International Workshop on Verification, Model Checking, and Abstract Interpretation (VMCAI'02)*, volume 2294 of *LNCS*, pages 289–301. Springer, 2002.

[HRS05] A. Harding, M. Ryan, and P.-Y. Schobbens. A new algorithm for strategy synthesis in LTL games. In *TACAS'05*, volume 3440 of *LNCS*, pages 477–492. Springer, 2005.

[JÅ06] Wojciech Jamroga and Thomas Ågotnes. What agents can achieve under incomplete information. In *AAMAS'06 : Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 232–234, New York, NY, USA, 2006. ACM.

[JD05] Wojciech Jamroga and Jürgen Dix. Do agents make model checking explode (computationally) ? In Michal Pechoucek, Paolo Petta, and László Zsolt Varga, editors, *Proceedings of the 4th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS'05)*, volume 3690 of *LNCS*. Springer, 2005.

[Jen96] H. Jensen. Model checking probabilistic real time systems. In *7th Nordic Workshop on Programming Theory*, pages 247–261. Chalmers University of Technolog, 1996.

[JLS07] Marcin Jurdzinski, François Laroussinie, and Jeremy Sproston. Model checking probabilistic timed automata with one or two clocks. In Orna Grumberg and Michael Huth, editors, *TACAS*, volume 4424 of *Lecture Notes in Computer Science*, pages 170–184. Springer, 2007.

[Jur98] Marcin Jurdziński. Deciding the winner in parity games is in $\mathsf{UP} \cap \mathsf{coUP}$. *Inf. Process. Lett.*, 68(3) :119–124, 1998.

[LMO06] François Laroussinie, Nicolas Markey, and Ghassan Oreiby. Model checking timed ATL for durational concurrent game structures. In Eugène Asarin and Patricia Bouyer, editors, *Proceedings of the 4th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'06)*, volume 4202 of *Lecture Notes in Computer Science*, pages 245–259, Paris, France, September 2006. Springer.

[LMO07] François Laroussinie, Nicolas Markey, and Ghassan Oreiby. On the expressiveness and complexity of ATL. In Helmut Seidl, editor, *Proceedings of the 10th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'07)*, volume 4423 of *Lecture Notes in Computer Science*, pages 243–257, Braga, Portugal, March 2007. Springer.

[LMO08] François Laroussinie, Nicolas Markey, and Ghassan Oreiby. On the expressiveness and complexity of ATL. *Logical Methods in Computer Science*, 4(2 :7), May 2008.

[LMS01] François Laroussinie, Nicolas Markey, and Philippe Schnoebelen. Model checking CTL^+ and FCTL is hard. In Furio Honsell and Marino Miculan, editors, *Proceedings of the 4th International Conference on Foundations of Software Science and Computation Structure (FoSSaCS'01)*, volume 2030 of *LNCS*, pages 318–331. Springer, 2001.

[LMS06] François Laroussinie, Nicolas Markey, and Philippe Schnoebelen. Efficient timed model checking for discrete-time systems. *Theoretical Computer Science*, 353(1-3) :249–271, 2006.

[LPY97] K.G. Larsen, P. Pettersson, and W. Yi. Uppaal : Status & developments. In *CAV'97*, volume 1254 of *LNCS*, pages 456–459. Springer, 1997.

[MPS95] Oded Maler, Amir Pnueli, and Joseph Sifakis. On the synthesis of discrete controllers for timed systems. In *Proc. 12th Annual Symp. on Theoretical Aspects of Computer Science (STACS'95)*, volume 900 of *LNCS*, pages 229–242. Springer, 1995.

[MS04] Nicolas Markey and Philippe Schnoebelen. Symbolic model checking of simply-timed systems. In Yassine Lakhnech and Sergio Yovine, editors, *Proc. Joint Conf. Formal Modelling and Analysis of Timed Systems (FORMATS'04) and Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'04)*, volume 3253 of *Lecture Notes in Computer Science*, pages 102–117, Grenoble, France, September 2004. Springer.

[Pap94] Christos Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Pin07] Sophie Pinchinat. A generic constructive solution for concurrent games with expressive constraints on strategies. In Kedar S. Namjoshi, Tomohiro Yoneda, Teruo Higashino, and Yoshio Okamura, editors, *ATVA*, volume 4762 of *Lecture Notes in Computer Science*, pages 253–267. Springer, 2007.

[Pnu77] Amir Pnueli. 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.

[Pur98] A. Puri. Dynamical properties of timed automata. In *FTRTFT'98*, volume 1486 of *LNCS*, pages 210–227. Springer, September 1998.

[QS82] Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In Marangiola Dezani-Ciancaglini and Ugo Montanari, editors, *Proceedings of the 5th International Symposium on Programming (SOP'82)*, volume 137 of *LNCS*, pages 337–351. Springer, 1982.

[Sch04] Pierre-Yves Schobbens. Alternating-time logic with imperfect recall. In *Proceedings of the 1st Workshop on Logic and Communication in Multi-Agent Systems (LCMAS'03)*, volume 85 of *ENTCS*. Elsevier, 2004.

[Sch08] Sven Schewe. ATL* satisfiability is 2EXPTIME-complete. In Luca Aceto, Ivan Damgaard, Ann Leslie Goldberg, Magnús M. Halldorsson, Anna Ingolfsdottir, and Igor Walkiewicz, editors, *35th International Colloquium on Automata, Languages and Programming (ICALP 2008)*, volume 5125 of *Lecture Notes in Computer Science*, Reykjavik, Iceland, July 2008. Springer.

[Vol99] Heribert Vollmer. *Introduction to Circuit Complexity : A Uniform Approach*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999.

[VS85] M Y Vardi and L Stockmeyer. Improved upper and lower bounds for modal logics of programs. In *STOC '85 : Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 240–251, New York, NY, USA, 1985. ACM.

[Wal89] M. Mitchell Waldrop. Phobos at mars : A dramatic view—and then failure. *Science*, 245(4922) :1044–1045, 8 September 1989.

[War93] Main Commission Aircraft Accident Investigation Warsaw. Report on the accident to Airbus A320-211 aircraft in Warsaw, 14 September 1993.

[Wil99] Thomas Wilke. CTL^+ is exponentially more succinct than CTL. In C. Pandu Rangan, Vijayshankar Raman, and R. Ramanujam, editors, *Proceedings of the 19th Conferentce on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'99)*, volume 1738 of *Lecture Notes in Computer Science*, pages 110–121. Springer-Verlag, December 1999.

[WLWW06] Dirk Walther, Carsten Lutz, Frank Wolter, and Michael Wooldridge. ATL satisfiability is indeed EXPTIME-complete. *Journal of Logic and Computation*, 16(6) :765–787, 2006.

[ZP96] Uri Zwick and Mike S. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1–2) :343–359, 1996.