THESE DE DOCTORAT
DE L'ECOLE NORMALE SUPERIEURE DE CACHAN

Présentée par

Monsieur Régis Gascon

pour obtenir le grade de

DOCTEUR DE L'ECOLE NORMALE SUPERIEURE DE CACHAN

Domaine:

INFORMATIQUE

Sujet de la thèse:

Spécification et vérification de propriétés quantitatives sur des automates à contraintes

Thèse soutenue à Cachan le 22/11/2007 devant le jury composé de

Anca Muscholl	Professeur des universités, Bordeaux I	Présidente
Ahmed Bouajjani	Professeur des universités, Paris VII	Rapporteur
Denis Lugiez	Professeur des universités, Aix-Marseille I	Rapporteur
Bernard Boigelot	Professeur, Université de Liège	Examinateur
Valentin Goranko	Associate Professor, Univ. of the Witwatersrand	Examinateur
Stéphane Demri	Chargé de recherche CNRS, ENS Cachan	Directeur de thèse

Laboratoire Spécification et Vérification
ENS CACHAN & CNRS
61 avecnue du Président Wilson
94235 CACHAN CEDEX (France)

Remerciements

Je remercie les rapporteurs de mon manuscrit Ahmed Bouajjani et Denis Lugiez ainsi que Bernard Boigelot, Valentin Goranko et Anca Muscholl pour avoir accepté d'être membre de mon jury et être venus assister à ma soutenance. Je remercie aussi Stéphane Demri pour ses conseils et sa disponibilité au cours des trois années passées malgré un emploi du temps parfois chargé. Merci aussi à toutes les autres personnes qui se sont déplacées pour assister à ma soutenance et en particulier aux membres de ma famille : mes parents Gérard et Geneviève, mon frère Sylvain, Coco et Alex. Merci à Dorothée pour m'avoir supporté à distance pendant ces trois années passées. Enfin, merci aussi à Dams qui n'a malheureusement pas pu se déplacer pour cette occasion mais qui je suis sûr m'a soutenu par la pensée.

*****

La préparation d'une thèse c'est un peu comme le grand huit : il y a hauts et des bas, ça va très vite et ça peut parfois donner la nausée. Par conséquent, il y a différente façon de vivre cette épreuve selon que l'on soit amateur de sensation forte où pas. Je connais même des doctorants qui ont malheureusement sauté du wagon en route (au sens figuré bien sûr, il n'y a pas eu de blessés). Pour ma part, je dois avouer que cette période s'est relativement bien déroulée et j'atteste qu'au moment où j'écris ces lignes que je n'ai gardé aucune séquelle physique ou morale ayant un quelconque rapport avec ma thèse. Je voudrais donc profiter de ces remerciements un peu plus informels pour livrer un témoignage de mon expérience au cas où elle puisse être utile à quelqu'un.

Tout d'abord, un élément déterminant dans le bon déroulement de ma thèse a sans aucun doute été la bonne ambiance de mon laboratoire d'acceuil que je tiens à souligner. En témoigne les bon moments que j'ai pu partager avec les membres du labo à Barbizon/Vaulx de Cernay/Deauville une fois par an, en mission ou encore lors de "repas thésards étendus"[1]. Je remercie donc l'ensemble des membres du Laboratoire Spécification et Vérification pour cette ambiance qui favorise à un travail de qualité. Je pense en particulier aux personnes qui ont eu le plaisir (ou la douleur) de partager mon bureau pendant ces années : tout d'abord le noyau dur de "la Renaudeau" 2004-2007 que j'ai lachement quitté en cours de route Sébastien, Pierre-Alain et Fabrice ; ensuite Benjamin et Simon qui sont partis trop tôt ; puis Julien, Nico P. et Chen ; et pour finir Arnaud, Tali et Antoine.

Cependant le mental ne fait pas tout et la préparation d'une thèse requiert aussi une forme physique irréprochable. Si j'ai pu tenir le coup pendant ces trois années c'est aussi grâce aux activités sportives qui ont occupées mon temps libre. En disant cela je ne pense pas qu'au sport à la télé mais aussi aux parties de foot d'un niveau improbable organisée avec les membres du labo : Nico M., Sergiu, Benedikt, Pascal, Arnaud, Thomas, Etienne, Nat, Fanny, Fabrice, Poti, Steve, Mathieu et ses conseils stratégiques et Pierre-Alain[2]. Bravo à ceux qui ont fait de leur mieux pour se hisser au niveau, en particulier P-A et ses deux

[1] Appélation officielle

[2] par ordre (approximatif) de niveau croissant :)

pieds gauches. Merci aussi à l'ensemble des autres personnes avec qui j'ai eu l'occasion de jouer au foot en salle ou en extérieur qu'il serait trop long de tous nommer mais qui se reconnaitront. Je finis ce passage sportif en rendant hommage à l'équipe LSV du tournois de foot du BDS 2007 qui a fait de son mieux pour être le plus digne possible sur le terrain (mais aurait pu mieux faire) : Nico, Benedikt, Arnaud, Thomas, Fabrice, Fanny, Sergiu et moi-même.

Un dernier point important est de savoir décompresser par moment. Pour cela il est indispensable d'avoir des collègues de bureau sur lesquels on peut compter. Je voudrais ainsi remercier plus personnellement ceux avec qui j'ai partagé quelques moments mémorables au bureau et en dehors. Il me semble normal de commencer par mes plus fidèles compagnons de café : Arnaud que je remercie aussi pour les clopes et Pierre-Alain. Je félicite Thomas d'avoir assumer avec brio la lourde tâche de représenter la Belgique aussi bien au niveau intellectuel que sportif, à Cachan comme à New-York (yeah !). J'ai aussi une pensée pour ceux avec que j'ai accompagné en France et un peu partout dans le monde lors de sorties nocturnes qu'il ne serait pas convenable de développer ici. Je vais néanmoins balancer tous les noms : Arnaud, Pierre-Alain, Benedikt, Thomas, Mouchy, Seb, Ghassan, Antoine, Rémi, Simon et Jules.

Enfin, je termine avec 10 des éléments qui m'ont aider à garder le moral et me permettent de citer certaines personnes qu'il me reste à remercier :

1. le café lorsque la machine n'a pas été cassée par Benedikt,
2. la Xblast team : Fabrice, Simon, Arnaud, Pierre-Alain, Antoine, Jean-Loup, Rémi et Florent.
3. le dynamisme communicatif de gens comme Mouchy et Rémi,
4. le stand-up d'Etienne à New-York,
5. les analyses socio-politico-culturelles d'Antoine,
6. la touche de féminité apportée par les doctorantes du LSV : Nat, Houda, Tali et Najla,
7. le "Señor Alvarez" qui se reconnaîtra,
8. le Sudoku-mot,
9. les vacances,
10. la famille.

Voilà, il me reste donc à souhaiter bonne chance aux doctorants actuels (et futurs) du LSV et je leur demande de veiller à protéger l'ambiance bon enfant. Pensez néanmoins à travailler un peu, sinon ça va se voir... Je conclue en remerciant tout ceux que j'aurai malencontreusement oublié et les incite fortement à me contacter pour demander réparation.

Abstract (français)

Mots clés : Vérification formelle, Model-checking, Systèmes infinis, Logiques temporelles, Méthodes à base d'automates, Domaines concrets.

L'utilisation omniprésente des systèmes informatiques dans notre vie quotidienne, dans des contextes parfois critiques, impose de s'assurer au préalable de leur bon fonctionnement. Face a la complexité croissante de ces systèmes, le recours aux méthodes de vérification formelles est une solution pour suppléer les méthodes de simulations et de tests qui ne peuvent être complètement exhaustives du fait de la multitude de scénarios possibles.

Parmi les méthodes de vérification formelles, le model-checking présente l'avantage d'être complètement automatisé. Le principe général de cette approche consiste à développer des algorithmes pour vérifier qu'une spécification exprimée la plupart du temps sous la forme d'une formule logique est satisfaite par une représentation symbolique (modèle) du système.

Les langages historiques de spécification tels que LTL ou CTL* utilisent comme formules atomiques des variables propositionnelles. En conséquence, les propriétés exprimées dans ces logiques portent principalement sur les états de contrôle du modèle. Dans cette thèse, notre but est de vérifier des propriétés plus riches portant sur divers objets (données) manipules par les modèles : des entiers (compteurs), des réels (horloges), des chaînes de caractères (piles, files)... Une particularité de ces objets est qu'ils peuvent prendre une infinité de valeurs et induisent donc des systèmes avec un nombre infini d'états.

Nous proposons la définition d'un cadre général pour l'extension des logiques temporelles classiques avec des contraintes induites par un domaine concret, c'est-à-dire un domaine d'interprétation (infini) pour les variables et un ensemble de relations. De plus, les extensions que nous considérons permettent de comparer la valeur des variables a différents états de l'exécution. Nous établissons des résultats de décidabilité et de complexité pour plusieurs problèmes de model-checking impliquant diverses instances de ces extensions. Nous privilégions pour cela l'approche à base d'automates en combinant des constructions connues pour les logiques propositionelles classiques avec différentes méthodes d'abstraction finie des modèles dont les variables sont interprétées dans des domaines infinis. Nous considérons entre autre divers fragments de logiques temporelles (linéaires et arborescentes) étendues avec des contraintes de Presburger ainsi qu'une variante de LTL avec des contraintes de répétition comportant un mécanisme de stockage sous-jacent.

Abstract (english)

Keywords : Formal verification, Model-checking, Infinite systems, Temporal logics, Automata based approaches, Concrete domains.

The ubiquity of computer systems in everyday life and particularly in catical contexts impose to ensure their good behavior. These systems are more and more complex and the use of formal verification methods is a good way to supply testing. Indeed simulations cannot be exhaustive because of the large amount of possible scenarios.

Model-checking is a technique to verify automatically computer systems. Basically, it consists in developping algorithms to check that a specification usually expressed by some logical formula is satisfied by a symbolic representation (model) of the system. Historical specification language such that LTL or CTL* use propositional variables as atomic formulas. Consequently, these logics allow to state properties only on the control locations of the models. In this thesis, we aim at checking richer properties on the objects (data) that models can handle : intergers (counters), reals (clocks), strings (stacks, queues)... A particularity of this kind of data data is that their interpretation domain is infinite and so the corresponding models have an infinite amount of states.

We introduce a general definition for the extensions of temporal logics with constraints induced by a concrete domain, i.e. an (infinite) interpretation domain and a set of relations. The extensions we consider also allow to compare values of the variables at different states of an execution.

We establish decidability and complexity results for several model-checking problems involving several instances of such extensions. We mainly use automata-based techniques that combine some usual constructions with finite abstraction methods for infinite datas. For instance, we consider several fragments of (linear and branching-time) temporal logics extended with constraints on counters induced by Presburger arithmetic and a variant of LTL with an underlying storing mechanism allowing to express repetition constraints.

Table des Matières

Introduction

Vérification formelle de systèmes informatiques

Aujourd'hui, l'omniprésence des systèmes informatisés dans notre vie quotidienne est incontestable. Les ascenseurs, les automobiles ou les machines à laver sont des exemples parmi la multitude d'objets qui nous entourent dont le fonctionnement dépend d'un programme informatique. Cette utilisation croissante de l'informatique nécessite des méthodes pour assurer le bon comportement de ces systèmes. Cet enjeu est d'autant plus important que sur certains systèmes critiques reposent parfois des intérêts humains ou financiers énormes comme par exemple dans les hôpitaux, les aéroports ou les centrales nucléaires.

Malgré les efforts importants qui sont faits pour valider les systèmes, les exemples récents de problèmes plus ou moins graves liés à l'informatique sont encore nombreux.

– En juillet 2007, une faille de sécurité importante est découverte dans le navigateur internet Firefox. Une mauvaise gestion des paramètres passés dans une adresse de type URI permet l'exécution de code malveillant sur la machine de l'utilisateur. Quelques jours plus tard, Microsoft est mis en cause car on découvre que c'est l'installation du programme Internet Explorer 7 qui modifie la gestion des URI dans le système d'exploitation Windows et rend ainsi Firefox vulnérable. Le problème viendrait donc en fait l'interaction de ces deux programmes.
– La mégapanne du Nord-Est américain du 14 août 2003, dont l'origine n'est pas liée à l'informatique mais à un mauvais entretien des lignes haute-tension, aurait cependant eu des répercussions moins importantes si un bug n'avait pas empêché l'alarme d'être déclenchée pour prévenir les réactions en chaîne provoquées par l'incident. Cette panne a privé 50 millions de personnes d'électricité et provoqué une perte financière estimée à 6 milliards de dollars.
– En 1994, une erreur dans le processeur Pentium découverte pour l'opération de division des nombres flottants oblige la société Intel à échanger un grand nombre de ces processeurs mis sur le marché provoquant des pertes financières considérables.
– Le 15 janvier 1990, un problème sur un des commutateurs qui gèrent les appels longue distance de la société de télécommunication américaine AT&T déclenche une réaction en chaîne qui a pour résultat que 114 machines se mettent à redémarrer toutes les 60 secondes. Un bug dans le programme des machines provoque une erreur à la réception du message généré lorsqu'une des machines voisines récupère elle-même après une erreur. Au bout de neuf heures, le problème sera réglé en réinstallant la précédente version du programme sur les machines en question. Le nombre d'appels qui n'ont pas pu être passés pendant cette période est estimé à 70 millions.
– Nous pouvons aussi citer comme exemples le dysfonctionnement de l'appareil médical Therac-25 (1985-1987), le crash d'Ariane 5 (1996), ou le fameux “bug de l'an 2000”.

Les exemples de bugs dans l'histoire récente sont encore nombreux. Le développement rapide des méthodes de conception et la mise en interaction des systèmes rendent en effet difficile la vérification des systèmes informatiques du fait du nombre important de configurations possibles.

Ainsi, la solution qui consiste à utiliser des méthodes de simulations pour valider les systèmes informatiques ne peut être complètement satisfaisante. La complexité des systèmes maintenant est telle qu'il est impossible de générer tout les comportements possibles. En conséquence, il est hasardeux de considérer qu'un système est totalement fiable lorsqu'il a passé un jeu de simulations avec succès. Ceci est un problème quand on sait que la moindre faille dans certains systèmes critiques peut potentiellement avoir des conséquences gravissimes. Il est donc nécessaire de développer d'autres méthodes pour garantir de manière automatique des propriétés de sécurité.

Approches formelles pour la vérification

La validation de systèmes informatiques a bénéficié ces dernières années du développement de nombreuses techniques de vérifications formelles. Ces méthodes permettent de vérifier automatiquement des propriétés de sécurité sur les systèmes informatiques. Elles sont une alternative intéressante permettant de compléter ou guider les expérimentations réelles telles que les simulations et les tests.

On peut distinguer différents types de méthodes formelles pour la vérification des systèmes informatiques.

- La génération de tests consiste plutôt à essayer de générer un ensemble de tests ciblés pour assurer certaines propriétés. Cependant, cette méthode rencontre les mêmes problèmes que la simulation puisqu'il est difficile de prévoir tous les comportements du système et l'analyse peut donc être incomplète.
- La preuve assistée qui consiste à générer une preuve que le programme vérifie bien les propriétés désirées. La preuve assure donc le bon comportement du programme mais son écriture est souvent manuelle et peut être longue et fastidieuse.
- Le model-checking qui consiste à vérifier automatiquement qu'un ensemble de propriétés de sécurité est vérifié par une représentation symbolique (ou modélisation) du système.

L'approche qui nous intéresse ici est celle du model-checking dont nous développons les principales caractéristiques.

Model-checking

Le model-checking [CGP99] fait partie de ces méthodes formelles qui connaissent un franc succès comme en témoignent le prix Turing remis à Amir Pnueli en 1996 ou le prix Gödel remis à Moshe Vardi et Pierre Wolper en 2000 pour leurs travaux respectifs dans ce domaine. Comme la plupart des méthodes de vérification formelle le principe du model-checking repose sur une abstraction du système informatique par un modèle mathématique et la définition d'un langage formel pour la spécification des propriétés de sécurité à vérifier. Le but est ensuite de définir des méthodes automatiques pour vérifier qu'un modèle satisfait une spécification donnée. Le système est généralement représenté par un graphe orienté appelé *système de transition* avec des sommets représentant les différents états du système et des règles de transitions entre les états qui définissent les comportements possibles du système. Ces systèmes peuvent manipuler selon les cas divers objets tels que des compteurs, des horloges, des piles ou des files pour citer quelques exemples. Le langage servant à la spécification des propriétés à vérifier est en général une formule dans un langage logique tel que les *logiques temporelles* introduites plus bas.

Une des principales difficultés rencontrées par cette approche est liée à l'explosion du nombre d'états des modèles. En particulier, certains formalismes permettent une représentation fini d'un système avec un nombre infini d'états. C'est le cas des systèmes à compteurs qui ont de nombreuses applications en vérification formelle, notamment la vérification de protocoles de diffusion [FL02] ou de programmes avec pointeurs [BFLS06, BBH$^+$06]. Un système à compteurs est une machine avec un nombre fini d'états de contrôle et un nombre fini de variables, appelées compteurs, prenant des valeurs entières. Une des principales problématiques du model-checking consiste donc à définir des formalismes qui peuvent manipuler au mieux ces représentations de systèmes infinis.

Logiques temporelles

Un langage naturel permettant d'exprimer les spécifications pour les problèmes de model-checking sont les logiques temporelles. Les logiques temporelles ont été introduites dans le cadre de la vérification formelle de systèmes réactifs en 1977 par Pnueli [Pnu77]. Depuis, ce type de logiques est devenu un des langages les plus utilisés pour la spécification de propriétés sur les systèmes réactifs. Historiquement, c'est Arthur Prior qui en 1957 introduit une logique construite à partir des modalités suivantes.

- $\mathsf{G}A$ signifiant *"il est toujours vrai dans le futur que A."*
- HA signifiant *"il était toujours vrai dans le passé que A"*.

D'autres modalités ont par la suite été introduites :

- $\mathsf{F}A$ signifiant *"il est vrai (au moins une fois) dans le futur que A"*, ce qui peut aussi être dit *"il n'est pas vrai que dans le futur on a toujours non A"* et s'écrit $\neg\mathsf{G}\neg A$,
- $A\mathsf{U}B$ signifiant *"A est vrai jusqu'à ce que B soit vrai"*, ne peut pas être exprimé en fonction des autres et augmente l'expressivité de la logique,

ainsi que leur symétrique dans le passé F^{-1} et S. Les modalités U et S ont la particularité par rapport aux autres d'avoir deux arguments et ont été introduites plus tard [Kam68].

Ces modalités sont interprétées sur des structures de Kripke qui sont des graphes définissant des transitions entre des états, aussi appelés mondes, et associant à chaque monde un ensemble de propriétés qui sont supposées vraies. Une telle représentation convient à la représentation du comportement d'un programme informatique. Les états du programme sont abstraits par rapport à des ensembles de propriétés partagées qui définissent par exemple l'état des différents registres. La relation de transition définit alors les évolutions possibles à partir de chaque état symbolique. Pnueli montre en 1977, que la vérification de propriétés exprimées dans la logique temporelle peut être automatisée, c'est-à-dire qu'il existe une méthode automatique appelée *algorithme* permettant de résoudre ce problème sur n'importe quelle entrée valide. Un problème vérifiant une telle propriété est dit *décidable*.

Néanmoins, on peut encore se poser d'autres questions par rapport à la sémantique de cette logique. En 1980, Lamport oppose la sémantique de $\mathsf{F}A$ avec celle de $\neg\mathsf{G}\neg A$ en nuançant deux interprétations possibles de l'adjectif "toujours" :

(1) *"il est toujours vrai* ***dans une exécution*** *du système"*,
(2) *"il est toujours vrai* ***dans toutes les exécutions*** *du système"*.

À partir de cette remarques se dégagent deux familles de logiques temporelles : (1) les logiques temporelles linéaires et (2) les logiques temporelles arborescentes. La différence entre les définitions de ces familles de logiques réside dans la capacité ou non de pouvoir exprimer que le système a plusieurs possibilité d'évoluer à l'instant suivant. Les logiques représentatives de ces deux classes sont respectivement LTL qui correspond à la logique initialement considérée par Pnueli dans [Pnu77] et la logique CTL introduite dans [CE81]. L'expressivité de ces logiques est incomparable, ce qui signifie qu'il existe certaines propriétés dans chacune de ces deux logiques qui ne peuvent s'exprimer dans l'autre. Cependant la logique CTL* introduite plus tard dans [EH83] englobe ces deux formalismes.

Enfin, une autre question intéressante est la *complexité* théorique de ces problèmes, c'est-à-dire le temps de calcul ou l'espace mémoire nécessaire pour résoudre ces problèmes. Ces mesures s'expriment en général par rapport à la taille des données de départ du problème. (Pour les définitions générales concernant les différentes classes de complexité ainsi que les résultats fondamentaux le lecteur peut se référer à [Pap94].)

Motivations et contributions

Les logiques temporelles classiques ont pour formules atomiques des variables propositionnelles qui ne permettent d'exprimer que des propriétés qualitatives telles que :

– *"toute requête sera satisfaite un jour"*,
– ou *"on ne peut pas atteindre un mauvais état du système"*.

En règle générale, ces propriétés se réduisent à l'accessibilité d'un état de contrôle dans la représentation symbolique du système. Dans cette thèse, nous introduisons des langages logiques qui permettent d'exprimer des propriétés plus riches que des propriétés sur les seuls états de contrôle de la représentation symbolique du système. L'objectif est de spécifier des propriétés sur les objets manipulés par ces systèmes tels que des nombres entiers (compteurs), des nombres réels (horloges) ou encore des chaînes de caractères (piles, files). Ainsi on voudrait par exemple pouvoir exprimer des propriétés telles que

– *"il existe un moment dans le futur où le compteur x est égal au compteur y"*,
– *"la différence entre les compteurs x et y est toujours inférieure à 10"*,
– *"tout événement A est suivi d'un événement B dans les 10 unités de temps"*
– *"la variable N ne prend jamais deux fois la même valeur le long d'une exécution du programme"*,
– *"la prochaine valeur de w est un sous-mot de sa valeur courante"*.

Nous introduisons une définition générale pour une nouvelle famille de langages qui étendent les logiques temporelles propositionnelles avec des formules atomiques permettant d'exprimer des contraintes sur les objets manipulés par le modèle. Les contraintes sont induites par un *domaine concret* qui est composé d'un ensemble de relations et d'un ensemble d'interprétation pour les variables. Pour reprendre un exemple ci-dessus, la propriété *"il existe un moment dans le futur où le compteur x est égal au compteur y"* peut s'exprimer par

$\mathsf{F}(x = y)$ dans la logique LTL étendue avec le domaine composé du domaine d'interprétation $\mathbb{N}$ pour les variables et de la relation d'égalité. Nous introduisons aussi la possibilité de comparer des valeurs des variables à différents états du modèle. Le dernier exemple illustre ce propos car on compare la valeur courante de w à sa prochaine valeur. Plus généralement, il est possible d'exprimer des contraintes du type *"la valeur de x est inférieure à la valeur de y 5 états plus loin"*.

Nous proposons d'étudier quels sont les langages logiques de cette famille ayant une expressivité suffisante tout en conservant de bonnes propriétés algorithmiques. Nous analysons par la même occasion les différents éléments qui peuvent causer l'indécidabilité des problèmes de vérification pour ces langages. Pour définir la décidabilité et la complexité des problèmes de model-checking liés à ces langages de spécification, nous privilégions des approches à base d'automates. Un *automate* est une structure composée d'états et de transitions qui représente symboliquement une machine destinée à reconnaître un ensemble d'objets ou à effectuer un calcul particulier. Dans le cas de l'approche qui nous intéresse, l'automate sert à reconnaître l'ensemble des modèles qui satisfont une spécification.

Cette méthode de traduction des logiques vers des automates est l'objet de nombreux travaux depuis les résultats de Büchi en 1963 mettant en relation la logique monadique du second ordre à un successeur avec les automates de mots infinis [Büc62]. Les approches qui nous intéressent plus particulièrement sont les construction d'automates reconnaissant des ensembles de modèles pour les formules données dans une logique temporelle telle que la construction de Vardi et Wolper pour LTL [VW86]. Nous définissons plusieurs approches originales qui étendent ces approches afin de traiter les problèmes de model-checking relatifs aux différentes logiques que nous considérons.

Contenu de la thèse

Le reste de ce document est organisé de la façon suivante. Les premiers chapitres de cette thèse contiennent des définitions et résultats utilisés ou étendus par la suite. Le Chapitre 1 définit les logiques temporelles classiques LTL et CTL* ainsi que les résultats fondamentaux concernant des problèmes standards liés à ces logiques. Le Chapitre 2 introduit différentes classes d'automates qui sont des éléments primordiaux dans les approches que nous utilisons par la suite. Enfin le Chapitre 3 introduit la définition générale des extensions des logiques temporelles que nous considérons ainsi que la classe des modèles associés à ces logiques pour le problème du model-checking. Ces logiques étendent les logiques temporelles classiques avec des contraintes sur les variables manipulées par le modèle. À la fin de ce chapitre nous introduisons plusieurs langages de contraintes utilisés dans la suite qui manipulent des compteurs, c'est-à-dire des variables prenant des valeurs entières.

Dans la deuxième partie, nous étudions des extensions de la logique temporelle linéaire LTL avec des langages de contraintes sur les compteurs. Nous définissons une classification de ces différents langages par rapport à plusieurs restrictions : l'ensemble de contraintes utilisées, le nombre de variables utilisées ou la distance maximale entre deux états du modèle pour lesquels on peut comparer les variables. En ce qui concerne le langage utilisé, nous montrons dans le Chapitre 4 que des contraintes qualitatives (sans l'addition) sur les entiers permettent d'étendre LTL en préservant la décidabilité et la complexité théorique. Nous réduisons le problème du model-checking pour la logique LTL étendue avec un large ensemble de contraintes comprenant des comparaisons et des contraintes de périodicité à un problème pour les automates de mots infinis. Cette réduction, qui repose sur une abstraction des modèles de la logique, nous permet de prouver que ce problème est PSPACE-complet.

Par opposition, le Chapitre 5 illustre qu'il est beaucoup plus difficile de retrouver la décidabilité lorsque l'on introduit des contraintes du type $x = y + 1$. Nous étudions alors en détails quelles sont les restrictions nécessaires pour retrouver la décidabilité et définissons des frontières entre décidabilité et indécidabilité pour les fragments obtenus par rapport aux restrictions évoquées plus haut. En ce qui concerne les résultats positifs, nous établissons entre autre que le problème du model-checking pour la logique LTL étendue avec le fragment sans quantificateur de l'arithmétique de Presburger sur les automates à un compteur est un problème PSPACE-complet.

Nous introduisons dans la troisième partie des mécanismes de stockage dans les logiques temporelles. Pour ce faire, nous étendons un peu plus la logique LTL en ajoutant une dose de quantification du premier ordre. Ceci nous permet de stocker la valeur d'une variable afin de la réutiliser quand bon nous semble dans le modèle. Concrètement, les logiques que nous étudions dans le Chapitre 6 permettent d'exprimer des contraintes de répétition de la valeur d'une variable dans le passé ou le futur. Nous définissons plusieurs résultats de décidabilité et complexité pour les problèmes de vérifications liés à ces extensions. Nous utilisons une traduction vers un problème de vérification pour les réseaux de Petri, établissant ainsi un autre type de relation avec les systèmes à compteurs.

Dans la dernière partie de ce document nous nous intéressons aux extensions des logiques arborescentes avec des contraintes. L'objectif est de généraliser dans la mesure du possible les résultats obtenus pour les extensions des logiques temporelles linéaires puisque CTL* englobe LTL. Dans le Chapitre 7, nous définissons une approche générale qui étend les différentes approches que nous avons utilisées dans le cas linéaire. Ainsi, la classe d'automates que nous utilisons permet de manipuler des abstractions de modèles arborescents. Nous utilisons cette approche pour montrer des résultats de décidabilité et complexité pour les extensions de CTL* avec des contraintes lorsque la logique obtenue vérifie une propriété générale d'abstraction des modèles. Enfin, dans le Chapitre 8 nous étendons une partie des résultats du Chapitre 4 en prouvant la décidabilité du model-checking pour un fragment de CTL* étendu avec le même langage de contraintes qualitatives sur les entiers. Nous utilisons cependant une approche différente qui nous permet de construire une représentation des états du modèle qui satisfont la spécification.

Première partie

Préliminaires

Chapitre 1

Logiques temporelles

Sommaire

Nous introduisons dans ce chapitre les principales logiques temporelles que nous considérons dans ce document. Ces formalismes sont à la base des langages de spécification utilisés pour les divers problèmes de model-checking qui nous intéressent par la suite.

Étant donnée une spécification exprimée par une formule logique, de nombreuses questions peuvent se poser. Nous nous intéressons principalement à deux d'entre elles. Comme nous l'avons déjà dit à plusieurs reprises, l'intérêt que nous portons aux logiques temporelles est motivé principalement par l'approche du model-checking qui consiste à déterminer si une représentation symbolique du système (un modèle) satisfait la spécification. Ce problème peut formellement être défini de la façon suivante :

Problème du model-checking
Entrée : Une spécification, dans le cas des logiques temporelles une formule, et un modèle du système informatique.
Question : Le modèle satisfait-il la spécification ?

Cependant, on peut aussi se demander s'il existe un modèle qui satisfait la propriété exprimée par la formule. Ce problème est appelé *problème de la satisfaisabilité* et nous pouvons le définir formellement de la façon suivante.

Problème de la satisfaisabilité
Entrée : Une spécification.
Question : Existe-t-il un modèle qui satisfait cette spécification ?

Nous verrons par la suite que ces deux problèmes sont fortement liés et parfois même équivalents pour certains formalismes logiques. Notons qu'une variante du problème de la satisfaisabilité parfois utilisée dans la littérature est le problème de la *validité* qui consiste à déterminer si pour tout modèle la formule est satisfaite. Ce problème est le dual du problème de la satisfaisabilité car une formule est valide si sa négation n'est pas satisfaisable. Ainsi, ces deux problèmes sont équivalents lorsque le langage est clos par négation.

Dans la suite de ce chapitre, nous considérons un ensemble infini dénombrable de variables propositionnelles PROP $= \{p_0, p_1, p_2, \ldots\}$. Étant donné un ensemble d'éléments quelconques X, nous notons $\mathcal{P}(X)$ l'ensemble des parties de X et $\mathcal{P}^+(X)$ l'ensemble des parties non-vides de X.

1.1 Logiques temporelles linéaires

1.1.1 Définition standard de LTL

La logique temporelle linéaire LTL a été introduite pour spécifier des propriétés sur *une exécution* particulière du système [Pnu77, GPSS80]. Les formules de cette logique sont définies par la grammaire suivante :

$$\phi ::= p \mid \phi \wedge \phi \mid \neg\phi \mid \mathsf{X}\phi \mid \phi \mathsf{U} \phi$$

où $p \in \mathrm{PROP}$ est une variable propositionnelle. Les opérateurs X (*next*) et U (*until*) permettent respectivement d'exprimer qu'une formule est vérifiée à l'instant suivant et qu'une première formule est vérifiée jusqu'à ce qu'une seconde formule soit vérifiée. Nous définissons aussi les opérateurs F et G tels que $\mathsf{F}\phi \equiv \top\mathsf{U}\phi$ et $\mathsf{G}\phi \equiv \neg\mathsf{F}\neg\phi$. Ainsi la formule $\mathsf{F}\phi$ se lit *"il existe une position dans le futur où ϕ est vérifiée"* et $\mathsf{G}\phi$ signifie *"la formule ϕ est vraie à tout instant dans le futur"*.

Le caractère linéaire de LTL est explicité par les modèles de cette logique. Comme on considère une exécution particulière, chaque état du modèle a un unique successeur dans le temps qui correspond au prochain état de l'exécution. Un modèle σ de LTL est donc une séquence infinie de la forme $\mathbb{N} \to \mathcal{P}(\mathrm{PROP})$ qui associe à chaque position de l'exécution un ensemble de propositions. Ces différents ensembles définissent une valeur de vérité pour les variables propositionnelles à chaque position dans le sens où les propositions associées à une position donnée sont considérées comme vraie à cette position de l'exécution. Étant donnés une formule ϕ de LTL, un modèle σ et une position i dans ce modèle, la relation de satisfaction $\sigma, i \models \phi$ est définie par

- $\sigma, i \models p$ ssi $p \in \sigma(i)$ pour toute variable propositionnelle $p \in \mathrm{PROP}$,
- $\sigma, i \models \phi \wedge \phi'$ ssi $\sigma, i \models \phi$ et $\sigma, i \models \phi'$,
- $\sigma, i \models \neg\phi$ ssi $\sigma, i \models \phi$ n'est pas vérifié (dans ce cas on note aussi $\sigma, i \not\models \phi$),
- $\sigma, i \models \mathsf{X}\phi$ ssi $\sigma, i+1 \models \phi$,
- $\sigma, i \models \phi\mathsf{U}\phi'$ ssi il existe $j \geq i$ tel que $\sigma, j \models \phi'$ et pour tout $i \leq k < j$ on a $\sigma, k \models \phi$.

Nous notons $\sigma \models \phi$ lorsque $\sigma, 0 \models \phi$. Par définition, le problème de la satisfaisabilité consiste à déterminer l'existence d'un modèle σ pour une formule ϕ donnée tel que $\sigma \models \phi$. Pour la logique LTL, le problème de la satisfaisabilité est décidable et la complexité de ce problème est définie dans [SC85].

Théorème 1 [SC85] *Le problème de la satisfaisabilité pour* LTL *est* PSPACE*-complet.*

Le problème du model-checking pour LTL utilise comme modèle les structures de Kripke. Nous rappelons qu'une structure de Kripke est un graphe orienté mettant en relation des états (appelés aussi mondes) étiquetés dans notre cas par un ensemble de propositions. Considérons une structure de Kripke $K = \langle N, \rightarrow, \ell \rangle$ tel que N est un ensemble fini de sommets, $\rightarrow \subseteq N \times N$ est un ensemble d'arcs et $\ell : N \rightarrow \mathcal{P}(\mathrm{PROP})$ est une fonction d'étiquetage qui associe à chaque sommet un ensemble de propositions. On distingue généralement un état initial dans K qui est le point de départ des exécutions. Nous supposons que pour tout sommet $q \in N$ il existe un successeur de q dans K, c'est-à-dire un sommet $q' \in N$ tel que $q \rightarrow q'$. Une structure de Kripke vérifiant cette propriété est dite *complète* ou *totale*. Dans une structure de Kripke complète, il existe donc un chemin infini à partir de chaque sommet. Une exécution correspond à un chemin infini dans la structure de Kripke ayant pour origine l'état initial et à chaque position de ce chemin est associé un ensemble de propositions qui est celui de l'état courant dans K. Il est donc évident qu'une exécution correspond à un modèle de LTL. Le problème du model-checking pour la logique LTL consiste à déterminer s'il existe un chemin infini dans la structure K à partir d'un état initial fixé qui satisfait la formule ϕ. Notons que nous considérons une définition existentielle. Comme pour les problèmes de la satisfaisabilté et validité, il existe une définition universelle de ce problème : pour toute exécution dans K la formule est vérifiée. Ces deux définitions sont duales et donc les problèmes correspondants sont équivalents puisque le langage LTL est clos par négation.

Pour LTL, le problème du model-checking se réduit en espace logarithmique au problème de la satisfaisabilité. En effet, on peut facilement coder le comportement d'une exécution dans une structure de Kripke par une formule de la logique LTL. Pour cela, on introduit une variable propositionnelle p_q pour chaque état q de K. À chaque position, une seule de ces variables est vraie ce qui se traduit par

$$\phi_{\mathrm{uni}} = \mathsf{G} \bigvee_{q \in N} (p_q \wedge \bigwedge_{q \in N \text{ et } q \neq q'} \neg p_{q'}).$$

Ceci exprime qu'à chaque position le long d'une exécution, on est dans un unique état de K. La validité d'une exécution est alors exprimée par la formule suivante

$$\phi_K = p_{q_{\mathrm{init}}} \wedge \phi_{\mathrm{uni}} \wedge \mathsf{G}(\bigwedge_{q \in N} p_q \Rightarrow (\bigwedge_{p \in \ell(q)} p \wedge \bigwedge_{p \notin \ell(q)} \neg p \wedge \bigvee_{q \longrightarrow q'} \mathsf{X} q'))$$

où q_{init} est l'état initial. La deuxième partie de la formule exprime les différentes conditions correspondant à la définition d'une exécution dans K. Si l'état courant de l'exécution est l'état q alors

- les propositions associées à cet état dans K sont vérifiées (et les autres non)
- et le prochain état de l'exécution est un successeur de q dans la structure K.

Vérifier que K satisfait ϕ est alors équivalent à vérifier que $\phi_K \wedge \phi$ est satisfaisable. Comme cette transformation se fait en espace logarithmique par rapport à la taille de K, ceci permet d'établir le résultat de complexité suivant.

Théorème 2 [SC85] *Le problème du model-checking pour* LTL *est* PSPACE-*complet.*

La borne inférieure dans [SC85] est obtenue par réduction de l'exécution d'une machine de Turing travaillant dans un espace polynomial par rapport à l'entrée.

Une hypothèse importante dans la sémantique de LTL que nous avons définie ci-dessus impose de considérer des séquences infinies. Cependant on peut aussi se poser la question de savoir si une spécification donnée est satisfaite par une exécution finie (non vide). Il faut alors modifier légèrement la sémantique des opérateurs temporels de LTL afin de tenir compte du fait que le modèle a peut être une fin. La relation de satisfaction doit considérer la taille $|\sigma|$ du modèle.

- $\sigma, i \models \mathsf{X}\phi$ ssi $i < |\sigma| - 1$ et $\sigma, i+1 \models \phi$
- $\sigma, i \models \phi \mathsf{U} \phi'$ ssi il existe $i \leq j < |\sigma|$ tel que $\sigma, j \models \phi'$ et pour tout $i \leq k < j$ on a $\sigma, k \models \phi$.

Les autres cas dans la relation de satisfaction sont inchangés. Notons que dans le cas fini, nous pouvons laisser tomber l'hypothèse que la structure de Kripke est complète pour le problème du model-checking.

Pour LTL, la question de l'existence d'un modèle fini satisfaisant une formule ϕ donnée, ou problème de la satisfaisabilité dans le cas fini, se réduit à la question de l'existence d'un modèle infini pour une autre formule obtenue à partir de la formule ϕ.

Lemme 1 *Les problèmes de la satisfaisabilité et du model-checking dans le cas fini pour* LTL *se réduisent respectivement aux mêmes problèmes dans le cas infini.*

Preuve : La réduction utilise une variable propositionnelle supplémentaire p_s qui est vraie dans tous les états jusqu'à une certaine position. Cette propriété est exprimée par la formule $\phi_{\text{fin}} = p_s \mathsf{U}(\mathsf{G}\neg p_s)$. La position où p_s cesse d'être vraie marque la fin du modèle. Nous transformons ensuite la formule afin d'imposer qu'elle soit satisfaite avant cette position à l'aide d'une fonction f définie inductivement par rapport à la structure de ϕ.

- $f(p) = p$ pour tout $p \in \text{PROP}$,
- f est homomorphique par rapport aux opérateurs booléens,
- $f(\mathsf{X}\phi) = \mathsf{X}p_s \wedge \mathsf{X}f(\phi)$,
- $f(\phi \mathsf{U} \phi') = (p_s \Rightarrow f(\phi))\mathsf{U}(p_s \wedge f(\phi'))$.

Ainsi, on peut facilement vérifier que ϕ est satisfaite par un modèle fini ssi $\phi_{\text{fin}} \wedge f(\phi)$ est satisfaite par un modèle infini. Un modèle qui satisfait ϕ correspond au préfixe d'un modèle qui satisfait $\phi_{\text{fin}} \wedge f(\phi)$ s'arrêtant à la position où la proposition p_s cesse d'être vraie.

Le problème du model-checking dans le cas fini se réduit de la même manière au cas infini, il suffit de rajouter un état supplémentaire q_f à la structure de Kripke K que l'on souhaite vérifiée et une transition entre chaque état de K et q_f. On ajoute aussi une boucle $q_f \rightarrow q_f$ pour que la structure soit complète. Puis, on étend l'étiquetage de manière à ce que la proposition p_s soit vérifiée dans tous les états sauf l'état supplémentaire q_f. La structure K' obtenue est telle qu'une transition vers l'état q_f correspond à la fin d'une exécution de K. Ainsi K vérifie ϕ ssi K' vérifie $\phi_{\text{fin}} \wedge f(\phi)$. □

Pour certaines extensions de LTL que nous considérons par la suite, il est impossible de réduire les problèmes dans le cas fini aux même problèmes dans le cas infini. Lorsque cela sera le cas, nous expliciterons les développements nécessaires pour traiter le cas fini.

1.1.2 Ajout d'opérateurs

Dans certaines définitions classiques de LTL, on introduit des opérateurs temporels supplémentaires référant au passé. L'ensemble des formules est alors défini par :

$$\phi ::= p \mid \phi \wedge \phi \mid \neg\phi \mid \mathsf{X}\phi \mid \phi\mathsf{U}\phi \mid \mathsf{X}^{-1}\phi \mid \phi\mathsf{S}\phi$$

où les opérateurs X^{-1} et S sont respectivement les versions pour le passé de X et U. Donc $\mathsf{X}^{-1}\phi$ se lit *"la formule ϕ est vraie à l'instant précédent"* et $\phi\mathsf{S}\phi'$ *"la formule ϕ est vraie depuis que ϕ' a été vraie"*. On peut aussi définir $\mathsf{F}^{-1}\phi \equiv \top\mathsf{S}\phi$ et $\mathsf{G}^{-1}\phi \equiv \neg\mathsf{F}^{-1}\neg\phi$. Les modèles pour LTL étendu avec les opérateurs passés ne changent pas et les opérateurs supplémentaires sont interprétés de la manière suivante :

- $\sigma, i \models \mathsf{X}^{-1}\phi$ ssi $i > 0$ et $\sigma, i-1 \models \phi$
- $\sigma, i \models \phi\mathsf{S}\phi'$ ssi il existe $0 \geq j \leq i$ tel que $\sigma, j \models \phi'$ et pour tout $j < k \leq i$ on a $\sigma, k \models \phi$.

Il est démontré dans [GPSS80, Gab89] que les opérateurs passés n'ajoutent pas d'expressivité lorsque l'on interprète les formules par rapport à l'origine du modèle. Ces opérateurs peuvent donc être supprimés pour LTL. Dans la suite de ce document, nous traitons la plupart du temps cette extension dans une famille d'extensions de LTL définie ci-dessous avec des opérateurs dont la définition plus générale englobe celle des opérateurs passés. Pour ces raisons et sauf lorsque cela est explicitement mentionné, nous parlons toujours de la version de LTL sans les opérateurs référant au passé.

Nous parlons aussi à plusieurs reprises dans la suite des extensions de LTL avec des opérateurs définissables dans la logique monadique du second ordre (MSO) car plusieurs des résultats que nous énonçons peuvent s'étendre à cette variante. Nous rappelons rapidement la sémantique de MSO sur les mots infinis dans notre cas particulier. Nous avons dans les formules de MSO deux types de variables : un ensemble de variables de position V_p qui prennent leurs valeurs dans $\mathbb{N}$ et un ensemble de variables d'ensembles de positions V_s qui prennent leurs valeurs dans $\mathcal{P}(\mathbb{N})$. Étant donnés un modèle $\sigma : \mathbb{N} \to \mathcal{P}(\mathrm{PROP})$ et une fonction ν associant à chaque position de σ une valeur dans $\mathbb{N}$ aux variables de V_p et un ensemble de valeurs de $\mathbb{N}$ aux variables de V_s on a

- $\sigma, \nu \models_{\mathrm{MSO}} x \leq y$ ssi $\nu(x) \leq \nu(y)$,
- $\sigma, \nu \models_{\mathrm{MSO}} P_p(x)$ ssi $p \in \sigma(\nu(x))$,
- $\sigma, \nu \models_{\mathrm{MSO}} X(x)$ ssi $\nu(x) \in \nu(X)$,
- $\sigma, \nu \models_{\mathrm{MSO}} \exists x\phi$ ssi il existe $i \in \mathbb{N}$ tel que $\sigma, \nu[x \leftarrow i] \models_{\mathrm{MSO}} \phi$,
- $\sigma, \nu \models_{\mathrm{MSO}} \exists X\phi$ ssi il existe $I \subseteq \mathbb{N}$ tel que $\sigma, \nu[X \leftarrow I] \models_{\mathrm{MSO}} \phi$,

où $\nu[x \leftarrow i]$ est la fonction qui associe à x la valeur i et coïncide avec ν pour les autres variables. La fonction $\nu[X \leftarrow I]$ est définie de manière similaire en associant un ensemble I à la variable X. Nous notons $\sigma \models \phi(\nu(X_1), \ldots, \nu(X_n), \nu(x_1), \ldots, \nu(x_{n'}))$ lorsque l'on a $\sigma, \nu \models \phi(X_1, \ldots X_n, x_1, \ldots, x_{n'})$ où $\phi(X_1, \ldots X_n, x_1, \ldots, x_{n'})$ est une formule de MSO ayant pour variables libres $X_1, \ldots X_n$ et $x_1, \ldots, x_{n'}$.

La sémantique associée à un opérateur MSO-définissable M ayant n arguments est exprimée par une formule de MSO $[\![M]\!](X_1, \ldots X_n, x)$ où $[\![M]\!]$ est la modalité correspondant à M, $x \in V_p$ et $X_1, \ldots X_n \in V_s$. Intuitivement, la variable x prend la valeur de la position courante et les ensembles $X_1, \ldots X_n$ correspondent aux ensembles d'états dans lesquelles les formules $\phi_1, \ldots \phi_n$ données en argument sont vérifiées. Ainsi, la sémantique des opérateurs temporels de LTL est MSO-définissable. En fait, LTL a le même pouvoir d'expression que la logique du premier ordre sur les ordres Dedekind-complets [Kam68] qui est incluse dans MSO. Nous posons

- $[\![p]\!](X, x) = P_p(x)$,
- $[\![\neg]\!](X, x) = \neg X(x)$,
- $[\![\wedge]\!](X, X', x) = X(x) \wedge X'(x)$,
- $[\![\mathsf{X}]\!](X, x) = X(x + 1)$,
- $[\![\mathsf{X}^{-1}]\!](X, x) = x > 0 \wedge X(x - 1)$,
- $[\![\mathsf{U}]\!](X, X', x) = \exists z(x \leq z \wedge X'(z) \wedge \forall y(x \leq y < z \Rightarrow X(y)))$,
- $[\![\mathsf{S}]\!](X, X', x) = \exists z(0 \leq z \leq x \wedge X'(z) \wedge \forall y(z < y \leq x \Rightarrow X(y)))$.

L'expression $X(x + 1)$ est une abréviation pour $\exists y((x < y \wedge \forall z(x < z \Rightarrow y \leq z)) \wedge X(y))$. L'ensemble X^σ_ϕ des positions de σ dans lesquelles une formule ϕ de la forme $M(\phi_1, \ldots, \phi_n)$ est vérifiée est définie inductivement par

$$X^\sigma_\phi = \{i \mid \sigma \models_{\mathrm{MSO}} [\![M]\!](X^\sigma_{\phi_1}, \ldots, X^\sigma_{\phi_n}, i)\}.$$

Pour faire le lien avec la définition de la sémantique que nous avons donnée pour LTL, on a $\sigma, i \models \phi$ ssi $i \in X^\sigma_\phi$.

Nous notons d'une manière générique xLTL les logiques obtenues en étendant LTL avec des opérateurs MSO-définissables. D'après les remarques ci-dessus, cette définition englobe LTL avec les opérateurs passés mais aussi l'extension de [Var88] avec des opérateurs de points fixes ou la logique ETL de [Wol83] qui introduit des opérateurs représentés par des automates finis. En effet, une autre manière de représenter les opérateurs MSO-définissables consiste à les définir sous la forme d'automates de mots infinis, ce qui est possible grâce à un résultat de [Büc62]. Ainsi, une formule atomique est vérifiée ssi l'automate correspondant accepte la séquence à partir de la position courante. En fait, le résultat de [Büc62] implique plus généralement que l'on peut construire à partir de toute formule ϕ d'une logique temporelle linéaire dont les opérateurs sont MSO-définissables un automate qui accepte les modèles qui satisfont ϕ. Ce résultat peut être utilisé pour établir la décidabilité du problème de la satisfaisabilité pour les logiques de la famille de xLTL. Nous reviendrons plus tard sur ce résultat ainsi que la complexité de cette procédure (voir Section 2.1).

1.2 Logiques temporelles arborescentes

Les logiques temporelles arborescentes ont été introduites pour pouvoir exprimer les différentes possibilités d'évoluer qu'a une exécution à l'instant suivant. Contrairement à LTL, on peut ainsi considérer les différents états du système qui sont possibles à l'instant suivant. Ceci est possible grâce à l'introduction de quantificateurs sur les exécutions. Les formules de la logique temporelle arborescente CTL* sont définies par la grammaire suivante :

$$\begin{aligned}\phi &::= p \mid \phi \wedge \phi \mid \neg\phi \mid E\psi \\ \psi &::= \phi \mid \phi \wedge \phi \mid \neg\phi \mid \mathsf{X}\psi \mid \psi \mathsf{U} \psi.\end{aligned}$$

où $p \in$ PROP. La différence la plus importante avec la définition de LTL est la présence du quantificateur E qui exprime une quantification existentielle sur les exécutions. Nous définissons son dual le quantificateur universel A tel que $A\psi \equiv \neg E \neg\psi$. Une formule de la forme $E\psi$ signifie donc *"il existe une exécution à partir de maintenant pour laquelle ψ est vérifiée"* alors que $A\psi$ se lit *"pour toutes les exécutions possibles à partir de maintenant ψ est vérifiée"*. Les opérateurs temporels gardent la même signification que dans le cas linéaire. Le fragment existentiel ECTL* (respectivement universel ACTL*) est le fragment obtenu en interdisant l'utilisation du quantificateur universel (respectivement existentiel). Ceci implique que chaque quantificateur existentiel soit dans la portée d'un nombre pair de négation (puisqu'une occurrence de $\neg E\phi$ est équivalente à $A\neg\phi$).

Notons que LTL est un fragment de CTL* puisque une formule de ψ de LTL correspond à une propriété relative à une exécution particulière, c'est-à-dire à la formule $E\psi$ de CTL*. Un autre fragment bien connu de CTL* est la logique CTL [CE81] dont les formules sont définies par :

$$\phi ::= p \mid \phi \wedge \phi \mid \neg\phi \mid E\mathsf{X}\phi \mid E\phi\mathsf{U}\phi.$$

Dans CTL les opérateurs temporels doivent être directement préfixés d'un quantificateur. Historiquement, la logique CTL* a été introduite dans [EH83] afin d'unifier les logiques LTL et CTL dont les pouvoirs d'expression sont incomparables. Certaines propriétés exprimables par une formule de LTL ne peuvent s'exprimer avec une formule de CTL et vice-versa.

Un *arbre* est un graphe particulier qui ne contient aucun cycle et tel que chaque sommet, à l'exception du sommet initial appelé aussi *racine*, a exactement un seul prédécesseur. Un modèle de CTL* est un arbre dont tous les sommets ont au moins un successeur et qui associe à chaque sommet un ensemble de variables propositionnelles. Donc un modèle $T = \langle N, n_0, \rightarrow, \Gamma\rangle$ de CTL* est tel que

- N est un ensemble fini ou infini de sommets,
- $n_0 \in N$ est la racine de l'arbre,
- $\rightarrow \subseteq N \times N$ est un ensemble d'arcs tel que pour tout $n \in N$
  - il existe au moins un sommet $n' \in N$ tel que $n \rightarrow n'$,
  - si $n \neq n_0$ il existe exactement un sommet $n' \in N$ tel que $n' \rightarrow n$,
  - $n \not\rightarrow n_0$ (la racine n'a pas de prédécesseur),
- et $\Gamma : N \rightarrow \mathcal{P}(\text{PROP})$ est une fonction associant à chaque sommet un ensemble de variables propositionnelles.

Un chemin dans T (et dans un graphe en général) est une séquence de sommets $\pi = n_0{\cdot}n_1 \cdots$ telle que $n_i \rightarrow n_{i+1}$ pour tout $0 \leq i < |\pi| - 1$. Nous notons $\pi^i = n_i \cdot n_{i+1} \cdots$ le $i^{\text{ème}}$ suffixe de π et $\pi(i)$ le $i^{\text{ème}}$ sommet de ce chemin. Une *branche* est un chemin maximal dans un l'arbre. Nous remarquons qu'une branche correspond à un modèle pour LTL. Dans la définition de

CTL* ci-dessus, on appelle les formules décrites par ϕ des *formules d'état* et celles décrites par ψ des *formules de chemin*. La différence entre ces deux types de formules est la façon dont leur satisfaction est déterminée. Comme leurs noms l'indiquent, la satisfaction d'une formule d'état est liée à un sommet du modèle alors que la satisfaction d'une formule de chemin dépend d'un chemin dans le graphe. Soit T un modèle de CTL*, n un sommet de T et π une branche de T. La relation de satisfaction pour une formule d'état est définie par analyse de cas de la façon suivante.

- $\langle T, n\rangle \models p$ ssi $p \in \Gamma(n)$ pour toute variable propositionnelle $p \in \mathrm{PROP}$,
- $\langle T, n\rangle \models \neg\phi$ ssi $\langle T, n\rangle \not\models \phi$,
- $\langle T, n\rangle \models \phi \wedge \phi'$ ssi $\langle T, n\rangle \models \phi$ et $\langle T, n\rangle \models \phi'$,
- $\langle T, n\rangle \models E\psi$ ssi il existe un chemin infini $\pi = n_0 \cdot n_1 \cdots$ dans T tel que $n_0 = n$ et $\langle T, \pi\rangle \models \psi$.

La relation de satisfaction pour une formule de chemin est définie par

- $\langle T, \pi\rangle \models \phi$ ssi $\langle T, \pi(0)\rangle \models \phi$;
- $\langle T, \pi\rangle \models \neg\psi$ ssi $\langle T, \pi\rangle \not\models \psi$,
- $\langle T, \pi\rangle \models \psi \wedge \psi'$ ssi $\langle T, \pi\rangle \models \psi$ et $\langle T, \pi\rangle \models \psi'$,
- $\langle T, \pi\rangle \models \mathsf{X}\psi$ ssi $\langle T, \pi^1\rangle \models \psi$,
- $\langle T, \pi\rangle \models \psi \mathsf{U} \psi'$ ssi il existe $i \in \mathbb{N}$ tel que $\langle T, \pi^i\rangle \models \psi'$ et pour tout $0 \leq j < i$ on a $\langle T, \pi^j\rangle \models \psi$.

Un modèle T satisfait une formule de CTL* ssi la racine n_0 de T est telle que $\langle T, n_0\rangle \models \phi$. Dans ce cas nous notons $T \models \phi$. Le problème de la satisfaisabilité pour CTL* est décidable et sa complexité est établie par le résultat suivant.

Théorème 3 [ES84] *Le problème de la satisfaisabilité pour* CTL* *est* 2EXPTIME-*complet.*

Le problème du model-checking pour CTL* est défini par rapport à un *dépliage* d'une structure de Kripke K qui permet de représenter, sous la forme d'un arbre, l'ensemble des exécutions possibles à partir d'un état. Un dépliage T_K de K est une arbre étiqueté par des états de K tel que

- la racine est étiquetée par l'état initial de K et
- pour tout sommet n de T_K étiqueté par un état q de K, on a un arc $n \to n'$ dans T_K tel que n' est étiqueté par q' ssi on a une transition de la forme $q \to q'$ dans K.

On associe à chaque sommet de T_K l'ensemble de propositions qui est associé à l'état de K qui l'étiquette pour obtenir un modèle de CTL*. Ainsi, une structure K satisfait une formule ϕ de CTL* ssi le dépliage T_K est tel que $\langle T, n_0\rangle \models \phi$ où n_0 est la racine de T_K.

Au contraire de LTL il n'existe pas de réduction évidente entre les problèmes de la satisfaisabilité et du model-checking pour CTL*. Le problème du model-checking est décidable mais sa complexité est moins importante que celle du problème de la satisfaisabilité.

Théorème 4 [CES86, EL87] *Le problème du model-checking pour la logique* CTL* *est* PSPACE-*complet.*

Notations : Nous définissons quelques notations et propriétés sur les modèles de CTL* qui nous facilite leur manipulation par la suite. Le degré d'un sommet dans un graphe G est égal au nombre d'arcs ayant pour origine ce sommet dans G (si ce nombre est infini, le degré est ω). Par extension, le degré d'un graphe est égal à la borne supérieure des degrés associés à ses sommets. Nous définissons un *d-graphe* comme étant un graphe dont tous les sommets ont pour degré d.

Pour tout graphe G de degré d, nous supposons que les successeurs de chaque sommet de G sont ordonnés. Ceci se traduit par le fait que nous étiquetons les arcs du graphe avec des éléments de $\{0, \ldots d-1\}$. Pour tout $a \in \{0, \ldots, d-1\}$, nous disons que n' est le a-successeur de n ssi $n \xrightarrow{a} n'$. Pour tout sommet $n \in N$ et $a \in \{0, \ldots, d-1\}$, n a au plus un a-successeur. Cet étiquetage n'est pas indispensable mais nous permet par la suite de décrire un chemin par rapport à son origine et au mot sur l'alphabet $\{0, \ldots, d-1\}$ qui décrit les transitions successives empruntées par ce chemin. Par exemple, lorsque nous parlons du chemin fini décrit par $w = 2 \cdot 0 \cdot 1$ commençant au sommet initial n_0, il s'agit du chemin $n_0 \xrightarrow{2} n_1 \xrightarrow{0} n_2 \xrightarrow{1} n_3$. Notons que si ce chemin existe alors il est unique. Étant donné un chemin π dans G, nous notons w_π le mot tel que pour tout $0 \leq i \leq |\pi| - 1$ on a $\pi(i) \xrightarrow{w(i)} \pi(i+1)$.

Toutes ces définitions s'appliquent bien entendu aux arbres qui sont des graphes particuliers et par conséquent aux modèles de CTL*.

Chapitre 2

Automates

Sommaire

En informatique, un automate est une machine abstraite constituée d'états et de transitions. Les transitions d'un automate ont pour attributs des éléments d'un alphabet fini ce qui permet de traiter différentes informations représentées par des objets construits à partir de cet alphabet. Les principaux éléments qui composent un automate sont

– un ensemble d'*états de contrôle* dans lesquels évolue une exécution,
– on distingue un sous-ensemble d'*états initiaux* parmi les états de contrôle,
– un *alphabet* fini sur lesquels sont construits les objets manipulés par l'automate,
– une *relation de transition* qui définit les mouvements autorisés dans l'automate en fonction de l'état de contrôle dans lequel on se trouve et de la lettre de l'alphabet lue,
– une *condition d'acceptation*.

On représente généralement un automate sous la forme d'un graphe dont les sommets sont les états de contrôle et les transitions sont des arcs étiquetés par des éléments de l'alphabet. Nous donnons une première idée informelle du rôle de ces différents éléments. La définition de la relation de transition d'un automate dépend des objets qu'il manipule, dans les cas qui nous intéressent des mots ou des arbres. Une exécution pour un automate est une séquence, ou un arbre selon les cas, débutant par un état initial et régie par la relation de transition. La condition d'acceptation définit si une exécution est acceptée ou non. Lorsqu'une exécution est acceptée, l'objet défini par rapport aux différentes lettres lues au cours de l'exécution est dit accepté par l'automate. Ainsi, un automate définit un langage composé de l'ensemble des objets qu'il accepte. Nous reprenons ces éléments en détail dans une définition plus formelle selon les différents cas par la suite.

Un problème important pour les automates consiste donc à déterminer si le langage reconnu par un automate donné est vide ou non. Nous donnons une définition générale de ce problème appelé *problème du vide.*

Problème du vide
Entrée : Un automate $\mathcal{A}$.
Question : Le langage reconnu par $\mathcal{A}$ est-il vide ou non ?

Un automate peut aussi être utilisé comme modèle opérationnel. Dans ce cas la définition peut être étendue en ajoutant un ensemble de variables qui sont mises à jour à chaque fois que l'on franchit une transition. C'est le cas de automates à compteurs dont nous parlons dans la dernière section de chapitre.

Nous introduisons maintenant les principales classes d'automates que nous utilisons.

2.1 Automates de mots

Un automate de mots reconnaît des séquences d'éléments appartenant à l'alphabet. Nous définissons un automate de mots $\mathcal{A}$ comme étant un tuple $\langle Q, I, \mathrm{Cond}, \Sigma, \delta \rangle$ tel que

- Q est un ensemble fini d'états de contrôle,
- $I \subseteq Q$ est un ensemble d'états initiaux,
- Σ est un alphabet fini,
- $\delta \subseteq Q \times \Sigma \times Q$ est une relation de transition,
- Cond défini une condition d'acceptation.

Nous verrons dans un instant que Cond peut prendre plusieurs formes : sous-ensemble d'états de contrôle, ensemble de sous-ensembles d'états de contrôle, fonction d'étiquetage des états de contrôle...

Nous notons $q \xrightarrow{a} q'$ lorsque $\langle q, a, q' \rangle \in \delta$ et $q \rightarrow q'$ s'il existe une lettre $a \in \Sigma$ telle que $q \xrightarrow{a} q'$. Une *exécution* pour $\mathcal{A}$ est une séquence d'états de contrôle $q_0 \cdot q_1 \cdot q_2 \cdots$ finie ou infinie telle que pour toute position i de la séquence on a $q_i \rightarrow q_{i+1}$. Nous notons aussi les exécutions $q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} \cdots$ lorsque nous voulons faire apparaître les lettres lues successivement au long de l'exécution. On peut distinguer deux catégories d'automates. L'automate $\mathcal{A}$ est un *automate déterministe* ssi pour tout état $q \in Q$ et toute lettre $a \in \Sigma$ il existe un unique état de contrôle $q' \in Q$ tel que $q \xrightarrow{a} q'$. Ceci signifie qu'il existe un unique déplacement dans chaque état de contrôle lorsque l'on lit une lettre donnée. Les automates *non-déterministes* généralisent cette classe d'automates en autorisant dans la relation de transition plusieurs possibilités d'évoluer à partir du même état et de la même lettre.

La condition d'acceptation sert à définir le langage accepté par l'automate. On dit qu'un mot w est accepté ou reconnu par $\mathcal{A}$ ssi il existe une exécution de $\mathcal{A}$ de la forme $q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} \cdots$ qui vérifie la condition d'acceptation telle que pour tout $0 \leq i < |w|$ on a $a_i = w(i)$. Une autre façon de voir les choses est qu'un mot w est une entrée pour l'automate et qu'un pas dans l'exécution consiste à lire la lettre de la position courante de w, effectuer une transition valide dans l'automate puis passer à la position suivante. Le fait que ce mot soit accepté ou non est toujours lié au fait que l'exécution satisfait la condition d'acceptation.

Dans le cas des automates de mots finis, la condition d'acceptation est définie en général par un sous-ensemble d'états de contrôle finaux et une exécution finie est acceptée ssi elle termine dans un état final. L'ensemble des mots reconnus par $\mathcal{A}$ forme le langage reconnu par $\mathcal{A}$ que nous notons $\mathrm{L}(\mathcal{A})$.

On peut introduire des transitions particulières dans un automate appelées *ε-transitions* de la forme $q \xrightarrow{\varepsilon} q'$. L'élément ε est donc rajouté à l'alphabet mais n'est pas une lettre à proprement dit. Intuitivement, ces transitions autorisent le franchissement d'une transition sans lire de lettre de l'alphabet. Un mot w est accepté par un automate $\mathcal{A}$ avec des ε-transitions ssi il peut être obtenu à partir d'un mot w' en effaçant toutes les occurrences de ε et w' est reconnu strictement par $\mathcal{A}$ en considérant ε comme une lettre normale de l'alphabet.

2.1.1 Automates de mots infinis

Nous avons besoin de définir des conditions d'acceptation spécifiques pour reconnaitre des séquences infinies. Il existe différentes conditions d'acceptation pour les mots infinis. Nous présentons ici celles que nous utiliserons par la suite. Pour plus de détails sur les résultats qui suivent, nous conseillons de se référer à [Tho90].

Une *condition de Büchi* est définie par un ensemble d'états finaux $F \subseteq Q$. Une exécution est acceptante ssi il existe un état de F qui est visité infiniment souvent. Un langage reconnu par un automate de mots avec condition d'acceptation de Büchi est appelé *langage ω-régulier*. L'ensemble des langages ω-réguliers est clos par rapport aux opérations classiques sur les langages.

Théorème 5 *L'ensemble des langages reconnus par un automate de mots avec condition d'acceptation de Büchi est clos par intersection, union et complémentation.*

Soient $\mathcal{A}_1 = \langle Q_1, I_1, F_1, \Sigma_1, \delta_1\rangle$ et $\mathcal{A}_2 = \langle Q_2, I_2, F_2, \Sigma_2, \delta_2\rangle$ deux automates de Büchi. Sans perte de généralité nous supposons que les ensembles d'états Q_1 et Q_2 sont disjoints. L'automate reconnaissant l'union des langages $\mathrm{L}(\mathcal{A}_1)$ et $\mathrm{L}(\mathcal{A}_2)$ est l'automate de Büchi tel que

- l'ensemble des états de contrôle est $Q_1 \uplus Q_2$,
- l'ensemble des états initiaux est $I_1 \uplus I_2$,
- l'ensemble des états finaux est $F_1 \uplus F_2$,
- l'alphabet est l'ensemble $\Sigma_1 \cup \Sigma_2$,
- la relation de transition coïncide avec δ_1 pour les états de Q_1 et δ_2 pour les états de Q_2, c'est à dire que pour tout $i \in \{1, 2\}$ on a

$$\langle q, a, q'\rangle \in \delta \text{ ssi } q, q' \in Q_i, a \in \Sigma_i \text{ et } \langle q, a, q'\rangle \in \delta_i.$$

Pour ce qui est de l'intersection, la construction n'est pas aussi triviale car il nous faut synchroniser les automates et assurer que les états finaux sont visités infiniment souvent dans chacune des deux structures. L'automate reconnaissant l'intersection des langages $\mathrm{L}(\mathcal{A}_1)$ et $\mathrm{L}(\mathcal{A}_2)$ est l'automate de Büchi tel que

- l'ensemble des états de contrôle est $Q_1 \times Q_2 \times \{1, 2\}$,
- l'ensemble des états initiaux est $I_1 \times I_2 \times \{1\}$,
- l'alphabet est l'ensemble $\Sigma_1 \cup \Sigma_2$,
- la relation de transition est définie par $\langle q_1, q_2, i\rangle \xrightarrow{a} \langle q'_1, q'_2, i'\rangle$ ssi
  - $\langle q_1, a, q'_1\rangle \in \delta_1$ et $\langle q_2, a, q'_2\rangle \in \delta_2$,
  - si $q_1 \in F_1$ et $i = 1$ alors $i' = 2$,
    sinon si $q_2 \in F_2$ et $i = 2$ alors $i' = 1$,
    sinon $i = i'$
- l'ensemble des états finaux est $Q \times F_2 \times \{2\}$.

Les deux premiers éléments qui composent un état de contrôle codent l'état courant dans $\mathcal{A}_1$ et $\mathcal{A}_2$. Le principe de base de la relation de transition est qu'un mouvement est autorisé dans l'automate synchronisé ssi il est autorisé dans les deux automates $\mathcal{A}_1$ et $\mathcal{A}_2$. Pour la condition d'acceptation, le troisième élément i qui compose un état de contrôle stocke une information chaque fois que l'on visite un état final d'un des deux automates. Si la dernière composante vaut i et qu'on passe par un état correspondant à un état final pour $\mathcal{A}_i$, c'est-à-dire que la $i^{\text{ème}}$ composante de l'état de contrôle appartient à F_i, alors i change de valeur. Ainsi, entre deux occurrences d'un état de la forme $Q \times F_2 \times \{2\}$ on doit passer par un état de la forme $F_1 \times Q \times \{1\}$ ce qui assure que si l'ensemble des états de $Q \times F_2 \times \{2\}$ est visité infiniment souvent alors il en est de même pour l'ensemble $F_1 \times Q \times \{1\}$. On peut facilement montrer la correction de la construction à partir de cette remarque.

Enfin, la complémentation est une opération bien plus compliquée car il est impossible de définir un algorithme qui étant donné un automate de Büchi non-déterministe construit un automate de Büchi déterministe acceptant le même langage. On peut cependant utiliser la construction définie de Safra [Saf88] pour construire un automate de Büchi reconnaissant le complément du langage accepté par un automate de Büchi donné.

Nous nous intéressons maintenant au problème du vide, qui est décidable pour la classe des automates de Büchi.

Théorème 6 *Le problème du vide pour les automates de mots avec condition d'acceptation de Büchi est* NLOGSPACE*-complet.*

Preuve : Un témoin pour une exécution acceptante est un chemin fini composé d'un sous-chemin entre un état initial et un état final suivi d'une boucle sur l'état final, c'est-à-dire un sous-chemin commençant et terminant dans cet état final. En effet, la boucle assure qu'on peut visiter infiniment souvent l'état final et la première partie du chemin que cet état final peut être atteint a partir d'un état initial. La taille de ce témoin peut être bornée par le nombre d'états de contrôle de l'automate puisque toute boucle n'ayant pas pour origine un état final rallonge inutilement le chemin. La vérification qu'un tel témoin est valide ne nécessite pas de mémoire supplémentaire. Ce problème est donc dans NLOGSPACE.

La NLOGSPACE-dureté est directe car le problème de l'accessibilité dans un graphe est déjà NLOGSPACE-complet. □

Une autre propriété des automates de Büchi est que si le langage reconnu par un automate de cette classe n'est pas vide alors il existe un mot dans ce langage qui est ultimement périodique. Un mot ultimement périodique est un mot de la forme $w_1 \cdot (w_2)^\omega$ terminant par un suffixe répété infiniment souvent.

Théorème 7 *Tout langage non-vide accepté par un automate de mots avec condition d'acceptation de Büchi contient un mot ultimement périodique.*

En effet, considérons un mot accepté par un automate de Büchi $\mathcal{A}$. D'après la condition d'acceptation, il existe dans l'exécution de $\mathcal{A}$ prenant comme entrée ce mot deux positions correspondant à la répétition d'un même état final. On peut répéter infiniment souvent le sous-chemin entre ces deux positions pour obtenir une exécution ultimement périodique acceptante. Le mot correspondant à cette exécution est ultimement périodique.

Une variante de la condition de Büchi est la *condition de Büchi généralisée*. Une telle condition est spécifiée par un ensemble d'ensembles d'états finaux $\{F_1, \ldots, F_n\}$ tel que pour tout $1 \leq i \leq n$ on a $F_i \subseteq Q$. Une exécution est acceptante ssi il existe dans chaque F_i un état visité infiniment souvent. Un automate avec condition de Büchi généralisée peut être transformé en automate avec condition de Büchi standard.

Théorème 8 *Étant donné un automate de mots avec condition d'acceptation de Büchi généralisée, on peut construire en espace logarithmique un automate de mots avec condition d'acceptation de Büchi standard acceptant le même langage.*

En quelques mots, la construction utilise une idée semblable à celle utilisée plus haut dans la synchronisation pour l'intersection d'automates de Büchi afin d'assurer que tous les ensembles sont successivement visités. On introduit un nouvel ensemble d'états de contrôle $Q \times \{1, \ldots, n\}$ où n est le nombre d'ensembles d'états finaux. Un état du nouvel automate est donc de la forme $\langle q, i \rangle$ où q correspond à un état de l'automate généralisé et i code le prochain ensemble d'états finaux que l'on veut visiter. Ainsi, la relation de transition est définie par $\langle q, i \rangle \xrightarrow{a} \langle q', i' \rangle$ ssi

- $q \xrightarrow{a} q'$ dans l'automate de Büchi généralisé
- si $q \in F_i$ alors $i' = i + 1$ si $i < n$ et $i' = 0$ sinon,
  sinon $i' = i$.

L'ensemble des états initiaux est $Q \times \{1\}$ et l'ensemble des états finaux $F_n \times \{n\}$.

Ainsi, les automates avec condition de Büchi généralisée héritent des mêmes propriétés de clôtures que les automates de mots avec condition d'acceptation de Büchi standard et le problème du vide a la même complexité.

Enfin, nous utilisons aussi la condition d'acceptation de parité. Nous utilisons surtout cette condition pour le cas des automates d'arbres dans la Section 2.2 mais il est plus facile de l'introduire d'abord pour les mots. Une telle condition d'acceptation est définie par une fonction qui affecte à chaque état de l'automate un rang appartenant à $\mathbb{N}$. Une exécution est acceptante ssi la priorité minimale affectée aux états visités infiniment souvent est paire.

Pour information, l'ensemble des langages reconnaissables par les les automates de mots avec condition d'acceptation de Büchi est égal à l'ensemble des langages reconnaissables par les les automates de mots avec condition d'acceptation de parité.

Théorème 9 *Les ensembles des langages reconnaissables par les automates de mots avec condition d'acceptation de Büchi, de Büchi généralisée et de parité sont égaux.*

2.1.2 LTL et les automates de Büchi

On peut dire que les différentes approches utilisant une traduction vers des automates pour prouver la satisfaisabilité d'une formule dans une logique temporelle linéaire ont pour origine le résultat de [Büc62] qui établit une relation entre les langages définissables par une formule de MSO et les automates de Büchi.

Théorème 10 [Büc62] *Tout langage de mots définissable par une formule de MSO peut être reconnu par un automate avec condition d'acceptation de Büchi.*

D'après les remarques faites dans la Section 1.1.2, ce résultat implique donc la décidabilité du problème de la satisfaisabilité pour xLTL puisque tester si le langage reconnu par un automate avec condition d'acceptation de Büchi est non vide est un problème décidable. La complexité de cette construction pour l'ensemble des modèles qui satisfont une formule de xLTL est définie dans [GK03].

Théorème 11 [GK03] *Étant donnée une formule de* xLTL*, on peut construire en utilisant un espace polynomial par rapport à* $|\phi|$ *un automate de Büchi* $\mathcal{A}_\phi$ *qui reconnaît l'ensemble des modèles de* ϕ.

En conséquence, la formule ϕ est satisfaisable ssi le langage $\mathrm{L}(\mathcal{A}_\phi)$ est non-vide. Comme le problème du vide pour les automates de Büchi est dans NLOGSPACE ceci implique le résultat de complexité suivant.

Théorème 12 *Le problème de la satisfaisabilité est* PSPACE*-complet pour* xLTL.

En effet, la composition d'un test en NLOGSPACE avec une construction en espace polynomial donne une procédure en espace polynomial non-déterministe d'après [BDG88, Corollaire 3.36]. D'après le théorème de Savitch, cette classe de complexité est égale à PSPACE [Sav70].

Cependant, une construction optimale plus ancienne pour LTL a été introduite par Vardi et Wolper dans [VW86] (voir aussi [VW94]). Nous décrivons ci-dessous cette construction qui est celle que nous étendons dans plusieurs preuves par la suite.

Théorème 13 [VW94] *Étant donnée une formule* ϕ *de* LTL *on peut construire en espace polynomial par rapport à* $|\phi|$ *un automate de Büchi* $\mathcal{A}_\phi$ *tel que* ϕ *est satisfaisable ssi le langage reconnu par* $\mathcal{A}_\phi$ *est non-vide.*

Nous décrivons cette construction pour la version de la logique LTL avec les opérateurs passés (voir [VW94] et [LPZ85]). Soit ϕ une formule de cette logique. La *clôture* de ϕ, notée $cl(\phi)$ est le plus petit ensemble contenant ϕ et vérifiant :

– pour toute formule ϕ_1 appartenant à $cl(\phi)$, la formule $\neg\phi_1$ appartient à $cl(\phi)$ (on identifie $\neg\neg\phi_1$ et ϕ_1),
– pour toute formule de la forme $\phi_1 \wedge \phi_2$ ou $\phi_1 \vee \phi_2$ appartenant à $cl(\phi)$ les formules ϕ_1 et ϕ_2 appartiennent à $cl(\phi)$,
– pour toute formule de la forme $\mathsf{X}\phi_1$ ou $\mathsf{X}^{-1}\phi_1$ appartenant à $cl(\phi)$, la formule ϕ_1 appartient à $cl(\phi)$,
– pour toute formule de la forme $\phi_1\mathsf{U}\phi_2$ appartenant à $cl(\phi)$ les formules ϕ_1, ϕ_2 et $\mathsf{X}(\phi_1\mathsf{U}\phi_2)$ appartiennent à $cl(\phi)$,
– pour toute formule de la forme $\phi_1\mathsf{S}\phi_2$ appartenant à $cl(\phi)$ les formules ϕ_1, ϕ_2 et $\mathsf{X}^{-1}(\phi_1\mathsf{S}\phi_2)$ appartiennent à $cl(\phi)$.

Un *atome* de ϕ est un ensemble X maximalement cohérent d'éléments de $cl(\phi)$ dans le sens où il vérifie les propriétés suivantes.

– pour toute formule ϕ_1 de $cl(\phi)$, soit ϕ_1 ou $\neg\phi_1$ appartient à X mais pas les deux (on identifie toujours $\neg\neg\phi_1$ et ϕ_1),
– la formule $\phi_1 \wedge \phi_2$ appartient à X ssi les formules ϕ_1 et ϕ_2 appartiennent à X,
– la formule $\phi_1 \vee \phi_2$ appartient à X ssi ϕ_1 ou ϕ_2 appartient à X,
– la formule $\phi_1\mathsf{U}\phi_2$ appartient à X ssi la formule ϕ_2 appartient à X ou les formules ϕ_1 et $\mathsf{X}(\phi_1\mathsf{U}\phi_2)$ appartiennent à X,
– si la formule $\phi_1\mathsf{S}\phi_2$ appartient à X alors la formule ϕ_2 ou les formules ϕ_1 et $\mathsf{X}^{-1}(\phi_1\mathsf{S}\phi_2)$ appartiennent à X.

Nous notons $\mathrm{Atom}(\phi)$ l'ensemble des éléments vérifiant ces conditions.

L'automate $\mathcal{A}_\phi$ a pour ensemble d'états de contrôle l'ensemble $Q = \mathrm{Atom}(\phi)$ et l'ensemble des états initiaux est défini par $I \subseteq Q$ tel que pour tout $At \in I$ on a

– $\phi \in At$,
– pour toute formule de la forme $\mathsf{X}^{-1}\phi' \in cl(\phi)$ on a $\neg\mathsf{X}^{-1}\phi' \in At$.

L'alphabet de $\mathcal{A}_\phi$ est $\mathcal{P}(P)$ où $P \subseteq \mathrm{PROP}$ est l'ensemble fini des variables propositionnelles utilisées dans ϕ. La relation de transition est telle que pour tout $At, At' \in Q$ et $X \in \mathcal{P}(P)$ on a $At \xrightarrow{X} At'$ ssi

– pour toute proposition atomique $p \in \mathrm{PROP}$ on a $p \in At$ ssi $p \in X$,
– pour chaque formule $\mathsf{X}^{-1}\psi \in cl(\phi)$ on a $\psi \in At$ ssi $\mathsf{X}^{-1}\psi \in At'$,
– pour chaque formule $\mathsf{X}\psi \in cl(\phi)$ on a $\mathsf{X}\psi \in At$ ssi $\psi \in At'$.

Nous définissons comme condition d'acceptation une condition de Büchi généralisée mais d'après le résultat du Théorème 8 on peut construire en utilisant un espace logarithmique à partir de l'automate obtenu un automate de Büchi standard reconnaissant le même langage. S'il n'existe aucune formule de la forme $\phi_1 \mathsf{U} \phi_2$ dans $cl(\phi)$ alors tous les états sont acceptants, c'est-à-dire que $F = Q$. Sinon, on pose

$$F = \big\{\{At \in Q \mid \phi_1 \mathsf{U} \phi_2 \notin At \text{ ou } \phi_1 \in At\} \mid \phi_1 \mathsf{U} \phi_2 \in cl(\phi)\big\}.$$

Pour plus de détails sur cette construction ainsi que la preuve de sa correction, voir par exemple [VW94]. Intuitivement, cette preuve utilise le fait qu'à chaque position dans une exécution acceptante, toute sous-formule appartenant à l'atome courant est vérifiée. Cette construction permet de retrouver la borne PSPACE par composition avec le test du vide pour un automate de Büchi.

2.2 Automates d'arbres

Nous considérons maintenant les automates d'arbres qui généralisent les automates de mots définis dans la section précédente. Le langage accepté par ces automates n'est plus composé de séquences mais d'arbres.

2.2.1 Quelques mots sur les automates d'arbres

La principale différence entre la définition d'un automate d'arbres et celle d'un automate de mots réside dans la relation de transition. Lorsque l'on reconnaît des mots, chaque paire composée d'un état de contrôle et une lettre de l'alphabet correspond à un mouvement dans l'automate. Ensuite l'exécution continue à partir de la position suivante dans le mot. Dans un automate d'arbres, chaque sommet ayant plusieurs successeurs, on envoie une copie de l'exécution pour chaque noeud fils afin de reconnaître les différents sous-arbres correspondants. La relation de transition est donc de la forme $\delta \subseteq Q \times \Sigma \times \mathcal{P}(Q)$ et associe un mouvement dans l'automate pour chaque fils du sommet courant en fonction de la lettre lue. Par exemple, la transition $\delta(q, a) = \{q_1, q_2\}$ signifie que si l'état courant est q, qu'on lit la lettre a et que le sommet courant à deux fils alors un sommet continue l'exécution dans l'état q_1 et l'autre fait de même mais en passant dans l'état q_2.

Nous ne nous étendons pas plus sur les propriétés générales des automates d'arbres car nous utilisons une classe particulière d'automates d'arbres dans ce document. La définition des automates d'arbres que nous introduisons généralise la définition standard en introduisant *l'alternation* [CKS81]. De plus, certaines définitions sont spécifiques aux automates d'arbres dont nous aurons besoin. Pour plus d'informations sur les automates d'arbres, le lecteur peut se référer à [Tho90] ou [CDG+97]. Nous décrivons avec plus de détails la classe d'automates qui nous intéresse.

2.2.2 Automates d'arbres alternants

La classe des automates d'arbres alternants généralise celle des automates d'arbres en définissant la relation de transition par une formule logique. Soit $\mathrm{FBP}(X)$ l'ensemble des formules Booléennes positives construites sur les éléments de X qui est défini par la grammaire suivante :

$$\alpha ::= \top \mid \bot \mid p \mid \alpha \wedge \alpha \mid \alpha \vee \alpha$$

où $p \in X$. Un ensemble $Y \subseteq X$ satisfait une formule $\alpha \in \mathrm{FBP}(X)$ ssi la valuation qui affecte $\top$ à chaque élément de Y et $\bot$ aux autres éléments de X satisfait α.

Un automate d'arbres alternant d'arité $d+1$ est une structure $\langle Q, I, \mathrm{Cond}, \Sigma, \delta \rangle$ telle que

- Q est un ensemble fini d'états,
- Σ est un alphabet d'entrée,
- $I \subseteq Q$ est un ensemble d'états initiaux,
- Cond spécifie la condition d'acceptation et
- la transition de relation δ est une fonction de la forme $(Q \times \Sigma) \to \mathrm{FBP}(\{0, \ldots, d\} \times Q)$.

La relation de transition associe à une paire de la forme $\langle q, a \rangle \in Q \times \Sigma$ une formule booléenne positive sur les éléments de $\{0, \ldots, d\} \times Q$. L'hypothèse selon laquelle les successeurs de chaque sommet sont ordonnés nous permet d'associer un fils particulier à chaque mouvement dans l'automate. Chaque élément de $\{0, \ldots, d\} \times Q$ spécifie une direction dans laquelle on se déplace dans l'arbre (un successeur) et un état dans lequel passe l'automate. Par exemple, la transition

$$\delta(q, a) = \langle 0, q_1 \rangle \vee (\langle 0, q_2 \rangle \wedge \langle 1, q_3 \rangle)$$

signifie que lorsque l'automate est dans l'état q et que la lettre lue est a alors

- soit on se déplace pour lire le 0-successeur du sommet courant et l'automate entre dans l'état q_1,
- soit on lance une copie de l'automate qui passe dans l'état q_2 et se déplace dans la direction du 0-successeur du sommet courant, et une autre qui passe dans l'état q_3 et se déplace dans la direction du 1-successeur.

La différence avec un automate d'arbres classique est que la relation de transition laisse la possibilité d'envoyer plusieurs copies de l'exécution dans la même direction. Notons que le non-déterminisme correspond à une disjonction dans la relation de transition. En effet, plusieurs relations par rapport à la même paire composée d'un état et d'une lettre, par exemple $\delta(q, a) = \langle 0, q_1 \rangle$ et $\delta(q, a) = \langle 0, q_2 \rangle$, peuvent être rassemblées dans la même transition $\delta(q, a) = \langle 0, q_1 \rangle \vee \langle 0, q_2 \rangle$.

Un automate d'arbres alternant $\mathcal{A} = \langle Q, I, \Sigma, \delta, \mathrm{Cond} \rangle$ *d'arité* $d+1$ prend pour entrée des arbres de degré $d+1$ dont les sommets sont étiquetés par des éléments de Σ. Une exécution de $\mathcal{A}$ sur l'entrée $T = \langle N, \{n_0\}, \delta, \Gamma \rangle$ est un autre arbre $T_r = \langle N_r, \{n_{0,r}\}, \to_r, \Gamma_r \rangle$ dont la racine est étiquetée par un état $q_0 \in I$ et les autres sommets par des éléments de $N \times Q$. Un sommet de T_r étiqueté par une paire de la forme $\langle q, n \rangle$ correspond à une copie de l'automate dans l'état q et lisant la lettre associé au sommet n de T. Chaque sommet de T_r et son ensemble de successeurs doivent satisfaire la relation de transition :

Pour tout sommet n_r de T_r tel que $\Gamma_r(n_r) = \langle q, n \rangle$ et $\delta(q, \Gamma(n)) = \alpha$, l'ensemble $S \subseteq (\{0, \ldots, d\} \times Q)$ défini par

$\langle a, q' \rangle \in S$ ssi il existe un successeur de n_r dans T_r étiqueté par $\langle n', q' \rangle \in S$ tel que n' est le a-successeur de n dans T

satisfait la contrainte α.

Notons qu'une exécution peut avoir des sommets sans successeurs, aussi appelés *feuilles*, puisque si α vaut $\top$ alors rien n'empêche qu'on n'ait aucun successeur. Au contraire il est impossible de construire une exécution en utilisant une règle où α vaut $\bot$.

Une exécution est *acceptante* ssi toutes ses branches infinies vérifient la condition d'acceptation. Un arbre T de degré d est accepté par $\mathcal{A}$ ssi il existe une exécution acceptante de $\mathcal{A}$ sur l'entrée T. L'ensemble des arbres de degré d acceptés par $\mathcal{A}$ forme le *langage accepté* par l'automate $\mathcal{A}$.

Les conditions d'acceptation utilisées sont similaires à celles des automates de mots mais s'appliquent à toutes les branches infinies de l'exécution. Dans le cas d'une condition d'acceptation de Büchi, Cond est un sous-ensemble d'états de l'automate $F \subseteq Q$ et une branche vérifie la condition d'acceptation ssi il existe un état de F visité infiniment souvent. De même, la condition de Büchi généralisée est telle que Cond est un ensemble de sous-ensembles acceptants. Une branche vérifie cette condition ssi tous les sous-ensembles sont visités infiniment souvent.

Théorème 14 *Un automate d'arbres alternant avec condition de Büchi généralisée peut être transformé en temps polynomial en automate d'arbres alternant avec condition de Büchi acceptant le même langage.*

La transformation reprend les mêmes opérations que dans le cas des automates de mots.

Dans le cas d'une condition d'acceptation de parité, nous rappelons que Cond est une fonction Rang : $Q \to \mathbb{N}$ qui associe un rang (ou priorité) à chaque état de l'automate. Une branche infinie est acceptée ssi la priorité minimale de l'ensemble des états visités infiniment souvent est paire. Nous utilisons par la suite le fait qu'une condition de Büchi peut être transformé en condition de parité.

Théorème 15 *Un automate d'arbres alternant avec condition de Büchi peut être transformé en temps linéaire en automate d'arbres alternant avec condition de parité.*

Preuve : On a juste besoin de deux priorités. Les états finaux ont pour priorité 0 et les autres 1. Le reste de l'automate ne change pas. Il est facile de vérifier qu'une branche passe infiniment souvent par un état final ssi la priorité minimale est 0. $\square$

Le principal intérêt de cette transformation est que les langages reconnus par les automates d'arbres alternants avec condition de parité sont clos par complémentation. La construction d'un automate d'arbres alternant avec condition de parité reconnaissant le complément d'un autre automate de la même classe est d'ailleurs peu coûteuse en terme de complexité.

Théorème 16 *Étant donné un automate d'arbres alternant $\mathcal{A}$ avec condition de parité, on peut construire en temps linéaire un autre automate d'arbres alternant $\tilde{\mathcal{A}}$ avec condition de parité tel que pour tout arbre T, T appartient au langage reconnu par $\mathcal{A}$ ssi T n'appartient pas au langage reconnu par $\tilde{\mathcal{A}}$.*

Pour construire $\tilde{\mathcal{A}}$, il suffit de dualiser les formules qui définissent la relation de transition de l'automate et d'ajouter 1 aux priorités affectées à chaque état (pour la preuve de correction de cette construction, voir par exemple [Kir02]). Cette propriété de clôture par complémentation n'est pas vraie pour les automates d'arbres avec condition d'acceptation de Büchi en général. L'ensemble des langages reconnus par les automates d'arbres alternant avec condition de Büchi est donc inclus dans l'ensemble des langages reconnus par les automates d'arbres alternant avec condition de parité (d'après le Théorème 15). Par contre les deux classes d'automates partage la propriété de clôture par union et intersection.

Théorème 17 *L'ensemble des langages reconnus par un automate d'arbres avec condition d'acceptation de Büchi (respectivement parité) est clos par intersection et union.*

En quelques mots, l'intersection de deux automates d'arbres alternants $\mathcal{A}_1 = \langle Q_1, \{q_0^1\}, \mathrm{Cond}_1, \Sigma_1, \delta_1 \rangle$ et $\mathcal{A}_2 = \langle Q_2, \{q_0^2\}, \mathrm{Cond}_2, \Sigma_2, \delta_2 \rangle$ est obtenue en ajoutant un état supplémentaire q_0 défini comme le nouvel (et unique) état initial et en ajoutant une nouvelle règle de transition pour l'état initial qui envoie deux copies de l'exécution vers les états initiaux q_0^1 et q_0^2 des automates $\mathcal{A}_1$ et $\mathcal{A}_2$. Ceci peut se traduit par la règle

$$\delta(q_0, a) = \delta_1(q_0^1, a) \wedge \delta_2(q_0^2, a)$$

pour tout $a \in \Sigma_1 \cup \Sigma_2$. Le reste de la relation de transition et la condition d'acceptation coïncide, pour tout $i \in \{1, 2\}$, avec la définition de $\mathcal{A}_i$ pour chaque état de Q_i (nous supposons sans perte de généralité que $Q_1 \cap Q_2 = \emptyset$). La construction de l'automate qui reconnaît l'union des langages reconnus par $\mathcal{A}_1$ et $\mathcal{A}_2$ est similaire en remplaçant la conjonction par une disjonction dans la règle $\delta(q_0, a)$. Ainsi, à partir de l'état initial on peut choisir de poursuivre dans l'exécution dans l'un ou l'autre des automates.

Pour ce qui est du test du vide, ce problème est décidable pour les automate d'arbres alternant avec condition de parité et la complexité est la suivante.

Théorème 18 *Le problème du vide pour un automate d'arbres alternant $\mathcal{A}$ avec condition de parité est* EXPTIME*-complet.*

Le résultat énoncé dans le Théorème 15 permet de voir que le problème du vide pour les automates d'arbres avec condition de Büchi est aussi dans EXPTIME. De même, ce problème est aussi EXPTIME-dur.

Théorème 19 *Le problème du vide pour un automate d'arbres alternant $\mathcal{A}$ avec condition de Büchi ou de Büchi généralisée est* EXPTIME*-complet.*

2.2.3 Méthodes à base d'automates pour les logiques arborescentes

Il existe moins de méthodes à base d'automates pour résoudre les problèmes liés aux logiques arborescentes. Citons juste les travaux de [Wil99, KVW00] qui montrent qu'on peut construire un automate d'arbres alternant qui reconnaît l'ensemble des modèles d'une formules de CTL ou CTL* donnée. Contrairement à la construction pour LTL à laquelle nous faisons référence à plusieurs reprises, nous ne définissons pas ici la construction pour

CTL* que nous utilisons plus localement dans ce document. Nous reprendrons les principaux éléments de cette construction le moment venu (voir Section 7.3.2). Le résultat qui nous intéresse est le suivant.

Théorème 20 *Étant donnée une formule ϕ de* CTL* *on peut construire en temps exponentiel par rapport à $|\phi|$ un automate d'arbres alternant $\mathcal{A}_\phi$ avec une condition d'acceptation de parité tel que ϕ est satisfaisable ssi le langage reconnu par $\mathcal{A}_\phi$ est non-vide.*

Intuitivement, la construction de cet automate se fait de manière inductive par rapport à la structure de ϕ. Le cas des formules de la forme $E\psi$ utilise la construction définie pour les formules de LTL et la condition de parité est nécessaire pour pouvoir complémenter l'automate obtenu dans le cas où l'on voudrait construire l'automate qui reconnaît les modèles de $A\neg\psi$. La taille de $\mathcal{A}_\phi$ est par conséquent d'ordre exponentiel par rapport à celle de ϕ.

2.3 Automates à compteurs

Une dernière classe d'automates qui tient une place importante dans la suite est la classe des automates à compteurs. Les automates à compteurs sont différents des automates que nous avons considérés jusqu'à présent. En effet, ces automates sont équipés d'un ensemble fini de variables prenant des valeurs entières. En plus des lettres de l'alphabet, on associe aux transitions des tests et mises à jour pour les variables. Un automate à compteur est donc une structure définie par

- un ensemble fini d'états de contrôle Q,
- une ensemble d'états initiaux I,
- un ensemble fini de compteurs C,
- une relation de transition $\delta \in Q \times \mathcal{P}(\text{Tests}) \times \text{Up} \times Q$ telle que Tests est un ensemble fini de tests sur les compteurs et Up est un ensemble fini de fonctions de la forme $f : \mathbb{N}^{|C|} \rightarrow \mathbb{N}^{|C|}$ définissant une mise à jour des compteurs,
- et une condition d'acceptation.

Nous oublions l'alphabet pour le reste de cette section car les propriétés qui nous intéressent portent principalement sur les exécutions et pas les langages. Néanmoins, rien n'empêche d'utiliser un automate à compteurs comme générateur de langage.

Les *configurations* d'un automate à compteurs sont donc des éléments de $Q \times \mathbb{N}^{|C|}$ dont la deuxième composante représente une valuation pour les compteurs. L'ensemble des configurations d'un automate à compteurs est donc infini contrairement aux automates que nous avons vu jusqu'à présent. Nous notons $\langle q, v\rangle \xrightarrow{X,u} \langle q', v'\rangle$ ssi il existe un élément $\langle q, X, u, q'\rangle \in \delta$ tel que la valuation des compteurs v satisfait tous les tests de X et $v' = u(v)$.

Une classe particulière d'automates à compteur à laquelle nous auront à faire référence est la classe des machines de Minsky [Min67]. Cette classe est une classe simple d'automates à compteurs dont les seules opérations permises sur les compteurs sont l'incrémentation et la décrémentation par 1 et les tests à zéro. Par rapport à la définition ci-dessus, une machine de Minsky est un automate à compteurs tel que :

- l'ensemble des tests est un ensemble de tests à zéro de la forme $x =? 0$ pour tout compteur $x \in C$,
- l'ensemble des fonctions de mises à jour est telle que tout $u \in \mathrm{Up}$, $u(x_1, \ldots, x_{|C|}) = (u(x_1), \ldots, u(x_{|C|}))$ vérifie pour tout $x \in C$
  - $u(x) = x$
  - ou $u(x) = x + 1$,
  - ou $u(x) = x - 1$.

Notons que comme un compteur ne peut pas avoir une valeur négative, toute opération de décrémentation implique un test à zéro implicite. On définit aussi un ensemble d'états finaux. Au lieu du problème du vide, on parle pour les machines de Minsky

- du *problème de l'arrêt* qui correspond déterminer l'existence d'une exécution finie terminant dans un état final,
- et du *problème de l'exécution récurrente* qui correspond à l'existence d'une exécution vérifiant une condition de Büchi.

La raison pour laquelle nous introduisons cette classe maintenant est que nous l'utilisons à plusieurs reprises pour prouver des résultats d'indécidabilité. En effet, le problème de l'accessibilité pour les machines de Minsky avec deux compteurs ou plus est indécidable.

Théorème 21
Le problème de l'arrêt pour les machines de Minsky à deux compteurs non-deterministes est indécidable. [Min67]

Le problème de l'exécution récurrente d'une machine de Minsky à deux compteurs non-deterministes est Σ_1^1-complet. [AH94, Lemme 8]

Comme les machines de Minsky à deux compteurs sont des automates à compteurs assez simples ce résultat implique beaucoup de résultats d'indécidabilité pour les problèmes liés à la classe automates à compteurs. Cependant, de nombreuses sous-classes ont été étudiées, pour lesquelles des problèmes de model-checking sont décidables :

1. les réseaux de Petri [Pet81, Jan90, CEP95, DE95],
2. les machines à compteurs dont le nombre d'alternations entre croissance et décroissance des valeurs est borné [Iba78],
3. les systèmes à compteurs plats [CJ98],
4. les systèmes à compteurs plats avec des fonctions de mise à jour affines formant un monoïde fini [Boi98, FL02, BFLP03],
5. les systèmes à compteurs admissibles avec contraintes de Presburger [DFGvD06].

Certains langages de spécification que nous introduisons par la suite ont pour objectif principal de spécifier des propriétés sur les automates à compteurs. Ainsi, les résultats pour les problèmes de model-checking correspondants à ces langages complètent ou étendent les résultats connus.

Chapitre 3

Logiques temporelles sur des domaines concrets

Sommaire

3.1 Définition du langage logique

3.1.1 Extension de logiques sur domaines concrets

Les structures de Kripke abstraient les états du programmes par rapport à un ensemble propositions qui prennent des valeurs binaires. Cette abstraction des systèmes informatiques peut paraître limitée puisque dans la réalité les programmes manipulent des variables qui peuvent représenter des entiers, des réels ou des chaînes de caractères. De nombreuses représentations symboliques ont été introduites pour représenter des systèmes infinis qui manipulent des objets aussi divers que des compteurs, des horloges, des files, des canaux de communication... Il est intéressant de pouvoir spécifier sur ces modèles des contraintes plus riches que la simple accessibilité d'un état de contrôle. Cependant, les variables propositionnelles ne sont pas satisfaisantes lorsque l'on souhaite exprimer des propriétés relatives aux objets manipulés par ces représentations symboliques. C'est pourquoi nous introduisons une famille d'extensions des logiques temporelles sur des domaines concrets qui permettent d'exprimer des propriétés sur des variables qui évoluent dans un ensemble potentiellement infini au cours d'une exécution.

Soit VAR $= \{x_0, x_1, \ldots\}$ un ensemble infini dénombrable de variables. Un *domaine concret* est une paire $\mathcal{D} = \langle D, \mathcal{R} \rangle$ telle que D est un domaine d'interprétation pour les variables et $\mathcal{R}$ est un ensemble dénombrable de relations sur les éléments de D. Nous notons les contraintes de $\mathcal{D}$ sous la forme $\mathrm{R}(x_{j_1}, \ldots, x_{j_k})$ telle que R est un symbole associé à une relation du domaine dont l'arité est k et $x_{j_1}, \ldots, x_{j_k} \in \mathrm{VAR}$. Une *valuation* est une fonction de la forme $v : \mathrm{VAR} \to D$ qui assigne à chaque variable une valeur dans D. Une contrainte $\mathrm{R}(x_{j_1}, \ldots, x_{j_k})$ de $\mathcal{D}$ est satisfaite par une D-valuation ssi $(v(x_{j_1}), \ldots, v(x_{j_k})) \in R$ où R est la relation de $\mathcal{D}$ associée au symbole R. Dans ce cas, nous notons $v \models \mathrm{R}(x_{j_1}, \ldots, x_{j_k})$. De même, nous notons $v \models X$ pour un ensemble de contraintes X ssi v satisfait toutes les contraintes de X. Un domaine concret $\mathcal{D} = \langle D, \mathcal{R} \rangle$ est un *domaine concret trivial* ssi pour toute relation $R \in \mathcal{R}$, soit $R = \emptyset$ ou bien $R = D^k$ où k est l'arité de R. Ceci implique que la satisfaction d'une contrainte dans un domaine concret trivial ne dépend pas de la valuation.

Nous définissons dans ce chapitre une famille d'extensions des logiques temporelles propositionnelles dans lesquelles les propositions sont remplacées par des contraintes induites par un domaine concret. À la différence de ce qui est dit ci-dessus, ces contraintes ne portent pas sur des variables mais sur des termes représentant les valeurs des variables à différents états du modèle. Un *terme* est une expression de la forme $\mathsf{X}\mathsf{X}\ldots\mathsf{X}x$ composée d'une variable préfixée par un certain nombre de symboles X. Nous réutilisons le symbole X de la modalité de LTL car cela ne cause aucune confusion par la suite. Nous définissons l'abréviation $\mathsf{X}^i x$ où $i \geq 0$ pour noter ces termes mais leur codage requiert $\mathcal{O}(i+j)$ bits où j est la taille nécessaire pour coder x. L'interprétation de ces termes sera formellement définie plus loin. Intuitivement, $\mathsf{X}^i x$ représente la valeur de x au $i^{\text{ème}}$ état suivant le long d'une exécution. Par exemple, une contrainte telle que $\mathsf{X}x = x + 1$ exprime qu'à l'instant suivant, la variable x est incrémentée de 1 et $\mathsf{X}^2 x \preceq \mathsf{X}y$ que la valeur de x dans 2 états est un sous-mot de la valeur de y à l'état suivant. Une *contrainte transitionnelle* est une contrainte de la forme $\mathrm{R}(\mathsf{X}^{i_1} x_{j_1}, \ldots, \mathsf{X}^{i_k} x_{j_k})$ telle que $i_1, \ldots, i_k \leq 1$. Seules les valeurs des variables à l'état courant et à l'état suivant sont utilisées dans ce type de contraintes ce qui permet de voir ces contraintes comme une mise à jour des variables.

Bien que déjà présente dans la logique du premier ordre, cette idée d'étendre les logiques temporelles avec des langages de contraintes est inspirée des travaux réalisés dans le domaine des logiques de description (voir par exemple [BH91, Lut04]). Les travaux réalisés dans le cadre de la vérification de données structurées ou semi-structurées introduisent aussi ce type d'extensions afin de comparer les données [BMS$^+$06, Seg06]. En ce qui concerne les logiques temporelles, de nombreuses extensions pouvant être reliées à notre définition ont été introduites [WZ00, BC02, GKK$^+$03], souvent dans le but d'exprimer des relations de comptage entre des variables interprétées par des entiers (voir par exemple [Čer94, BEH95, BH96, CC00, DD07]).

3.1.2 Logiques du temps linéaires

En ce qui concerne les logiques temporelles linéaires, nous définissons CLTL($\mathcal{D}$) comme l'extension de LTL définie par la grammaire suivante :

$$\phi ::= \mathrm{R}(\mathsf{X}^{i_1} x_{j_1}, \ldots, \mathsf{X}^{i_k} x_{j_k}) \mid \phi \wedge \phi \mid \neg\phi \mid \mathsf{X}\phi \mid \phi \mathsf{U} \phi$$

où $x_{j_1}, \ldots, x_{j_k} \in \mathrm{VAR}$ sont des variables et $\mathrm{R}(\mathsf{X}^{i_1} x_{j_1}, \ldots, \mathsf{X}^{i_k} x_{j_k})$ est une contrainte de $\mathcal{D}$ sur des termes. Les opérateurs X et U de la logique sont les opérateurs classiques de LTL.

Nous utilisons aussi les opérateurs $\mathsf{F}\phi$ et $\mathsf{G}\phi$ définis dans LTL respectivement par $\top\mathsf{U}\phi$ et $\neg\mathsf{F}\neg\phi$. Étant donnée une formule ϕ de CLTL($\mathcal{D}$) nous définissons sa longueur temporelle, notée $|\phi|_{\mathsf{X}}$, comme étant le plus grand entier l tel qu'un terme de la forme $\mathsf{X}^l x$ appartient à ϕ. Intuitivement, cette mesure définit la taille d'une fenêtre d'états consécutifs du modèle dans lesquels les différentes valeurs prises par les variables peuvent être comparées.

Les modèles de CLTL($\mathcal{D}$) sont des séquences infinies de valuations de la forme $\sigma : \mathbb{N} \rightarrow (\mathrm{VAR} \rightarrow D)$. La relation de satisfaction pour la logique CLTL($\mathcal{D}$) est définie de la même façon que pour LTL, excepté au niveau atomique.

- $\sigma, i \models \mathrm{R}(\mathsf{X}^{i'_1}x_{j_1}, \ldots, \mathsf{X}^{i'_k}x_{j_k})$ ssi $(\sigma(i+i'_1)(x_{j_1}), \ldots, \sigma(i+i'_k)(x_{j_k})) \in R$,
- $\sigma, i \models \phi \wedge \phi'$ ssi $\sigma, i \models \phi$ et $\sigma, i \models \phi'$
- $\sigma, i \models \neg\phi$ ssi $\sigma, i \not\models \phi$,
- $\sigma, i \models \mathsf{X}\phi$ ssi $\sigma, i+1 \models \phi$,
- $\sigma, i \models \phi\mathsf{U}\phi'$ ssi il existe $i_0 \geq i$ tel que $\sigma, i_0 \models \phi'$ et pour tout $i \leq j < i_0$ on a $\sigma, j \models \phi$.

Nous notons $\sigma \models \phi$ si $\sigma, 0 \models \phi$. Le problème de la satisfaisabilité pour CLTL($\mathcal{D}$) consiste à déterminer, étant donnée une formule ϕ, s'il existe une modèle $\sigma : \mathbb{N} \rightarrow (\mathrm{VAR} \rightarrow D)$ tel que $\sigma \models \phi$. Nous notons $\mathrm{CLTL}^l_k(\mathcal{D})$ le fragment de CLTL($\mathcal{D}$) restreint aux formules ayant au plus k variables et dont la longueur temporelle est inférieure ou égale à l.

Avant de continuer, voici quelques exemples de formules de CLTL($\mathcal{D}$).

- $\mathsf{G}(x = \mathsf{X}x)$ est une formule de la logique $\mathrm{CLTL}(\langle\mathbb{Z}, =\rangle)$ et plus précisément de son fragment $\mathrm{CLTL}^1_1(\langle\mathbb{Z}, =\rangle)$ qui signifie qu'à tout moment la valeur de x à l'instant suivant est égale à sa valeur courante, ce qui signifie que la valeur de x est constante.
- $\mathsf{F}((x < \mathsf{X}y) \wedge (x < \mathsf{XX}y) \wedge (x < \mathsf{XXX}y))$ est une formule du fragment $\mathrm{CLTL}^3_2(\langle\mathbb{Z}, <\rangle)$ de $\mathrm{CLTL}(\langle\mathbb{Z}, <\rangle)$. Elle exprime qu'on atteint dans le futur une position où les 3 prochaines valeurs de y sont strictement supérieures à la valeur courante de x.
- $(\mathit{sent} = \mathsf{X}\mathit{sent})\mathsf{U}(\mathit{sent} \preceq \mathsf{X}\mathit{received})$ est une formule du fragment $\mathrm{CLTL}^1_2(\langle\Sigma^*, \preceq\rangle)$ de $\mathrm{CLTL}(\langle\Sigma^*, \preceq\rangle)$ exprimant que le message envoyé reste le même jusqu'à ce qu'il soit un sous-mot du message reçu à l'instant suivant.
- $\mathit{conc}(a, x, \mathsf{X}x) \Rightarrow \mathsf{X}\mathit{conc}(b, \mathsf{X}x, x)$ est une contrainte de $\mathrm{CLTL}^1_1(\langle\Sigma^*, R_{\mathit{conc}}\rangle)$ telle que pour tout $a \in \Sigma$ et $x, y \in \mathrm{VAR}$ on a $(a, x, y) \in R_{\mathit{conc}}$ ssi $a \cdot x = y$. Cette contrainte signifie que si l'on dépile un a dans la pile x alors à l'instant d'après on empile un b.

La formule $\mathsf{F}((x < \mathsf{X}y) \wedge (x < \mathsf{XX}y) \wedge (x < \mathsf{XXX}y))$ est satisfaite par le modèle suivant (voir les positions en caractères gras) :

$$\sigma = \left(\begin{array}{ccc|cccc|cc} x: & -1 & 4 & \mathbf{2} & 5 & 7 & -1 & 2 & \cdots \\ y: & -2 & 1 & -2 & \mathbf{4} & \mathbf{3} & \mathbf{6} & -1 & \cdots \end{array}\right).$$

Les résultats qui suivent montrent des réductions possibles entre les différents fragments de la forme $\mathrm{CLTL}^l_k(\mathcal{D})$. Ces réductions établissent des correspondances qui facilitent la classification de ces fragments en fonction de la restriction du nombre de variables et de la longueur temporelle des formules (voir la Section 5.1.3). La première réduction permet de réduire la longueur temporelle d'une formule mais augmente le nombre de variables.

Lemme 2 *Pour tous $k, l, k', l' \in \mathbb{N} \setminus \{0\}$ et pour tout domaine concret $\mathcal{D}$, il existe une réduction en temps exponentiel de $\mathrm{CLTL}_k^l(\mathcal{D})$ vers $\mathrm{CLTL}_{k'}^{l'}(\mathcal{D})$ lorsque $k \times l = k' \times l'$ et $k' = k \times m$ pour $m \geq 2$.*

Preuve : L'idée de cette preuve est de coder m états d'un modèle de $\mathrm{CLTL}_k^l(\mathcal{D})$ en un seul état d'un modèle de $\mathrm{CLTL}_{k'}^{l'}(\mathcal{D})$. Par exemple, si $k' = 2k$ alors le modèle de $\mathrm{CLTL}_k^l(\mathcal{D})$ ci-dessous

$$\begin{pmatrix} x_1^0 \\ x_2^0 \\ \dots \\ x_k^0 \end{pmatrix} \begin{pmatrix} x_1^1 \\ x_2^1 \\ \dots \\ x_k^1 \end{pmatrix} \begin{pmatrix} x_1^2 \\ x_2^2 \\ \dots \\ x_k^2 \end{pmatrix} \dots$$

correspond au modèle de $\mathrm{CLTL}_{2k}^{l/2}(\mathcal{D})$ suivant

$$\begin{pmatrix} x_1^0 \\ x_2^0 \\ \dots \\ x_k^0 \\ x_1^1 \\ x_2^1 \\ \dots \\ x_k^1 \end{pmatrix} \begin{pmatrix} x_1^2 \\ x_2^2 \\ \dots \\ x_k^2 \\ x_1^3 \\ x_2^3 \\ \dots \\ x_k^3 \end{pmatrix} \dots$$

Nous définissons une fonction $f : \mathrm{CLTL}_k^l(\mathcal{D}) \times \{0, \dots, m-1\} \rightarrow \mathrm{CLTL}_{k'}^{l'}(\mathcal{D})$ par induction sur la structure de la formule telle que $f(\phi, 0)$ exprime les conditions ci-dessus. Le deuxième argument de la fonction f permet de se repérer par rapport au modèle original et à déterminer lorsqu'il faut passer à l'état suivant dans le modèle d'arrivée. Ceci se produit à chaque fois que l'on a passé m états du modèle original (voir par exemple la définition ci-dessous de $f(\mathsf{X}\phi, pos)$). Le deuxième argument compte donc les positions du modèle original modulo m.

- $f(\mathrm{R}(\mathsf{X}^{a_1} x_{b_1}, \dots, \mathsf{X}^{a_s} x_{b_s}), pos) = \mathrm{R}(\mathsf{X}^{a'_1} x_{b'_1}, \dots, \mathsf{X}^{a'_s} x_{b'_s})$ où pour tout $i \in \{1, \dots, s\}$ on a $a'_i = q_i$ et $b'_i = r_i k + b_i$ où q_i et r_i sont respectivement le quotient et le reste de la division euclidienne de $a_i + pos$ par m, c'est-à-dire qu'on a $a_i + pos = q_i m + r_i$ tel que $0 \leq q_i$ et $0 \leq r_i \leq m - 1$.
  Comme $0 \leq q_i \leq (a_i + pos)/m$ et $0 \leq a_i + pos < l + m$ nous obtenons $0 \leq q_i < (l + m)/m$. Notons que si $k' = km$ et $kl = k'l'$ on a $l' = ml$ ce qui nous permet de déduire que $0 \leq q_i < ((l' + 1)m)/m$. Donc on a $0 \leq a_i \leq l'$. De même $0 \leq r_i \leq m - 1$ et $1 \leq b_i \leq k$ impliquent $1 \leq kr_i + b_i \leq km$ ce qui fait que l'on a bien $1 \leq b'_i \leq k'$.

  Cette opération permet de mettre en relation les termes du modèle original avec le modèle d'arrivée. La division entière permet de savoir combien d'états on saute dans le modèle d'arrivée. Pour un terme $\mathsf{X}^i x_j$, ce nombre d'états dépend bien entendu de i mais aussi de la position *pos* où l'on se trouve dans le modèle original. À partir du reste de cette division, nous calculons alors à quelle variable de l'état d'arrivée correspond le terme.

– f est homomorphique par rapport aux opérateurs Booléens.

– $f(\mathsf{X}\phi, pos) = f(\phi, pos + 1)$ pour tout $pos < m - 1$.
Lorsque $pos < m-1$, le passage à un état suivant dans le modèle original ne correspond pas à un changement d'état du modèle final mais il est nécessaire de prendre en compte ce fait dans la fonction.

– $f(\mathsf{X}\phi, m - 1) = \mathsf{X}f(\phi, 0)$.
Lorsque $pos = m - 1$, le passage à un état suivant dans le modèle original correspond aussi à un changement d'état dans le modèle final.

– $f(\phi \mathsf{U} \psi, pos)$ est égal à

$$f(\psi, pos) \vee (f(\phi, pos) \wedge (f(\psi, pos + 1) \vee \ldots f(\phi, m - 2) \wedge (f(\psi, m - 1) \vee$$
$$((f(\phi, m - 1) \wedge \mathsf{X}(\phi' \mathsf{U} \psi'))) \ldots))$$

tel que

  – $\phi' = \bigwedge_{0 \leq i \leq m-1} f(\phi, i)$,
  – $\psi' = \bigvee_{0 \leq i \leq m-1}((\bigwedge_{0 \leq i' < i} f(\phi, i')) \wedge f(\psi, i))$.

Puisque plusieurs états du modèle original sont codés dans un seul état du modèle d'arrivée, nous ne pouvons pas nous contenter de nous déplacer d'état en état dans le modèle d'arrivée. Il est nécessaire de vérifier que toute séquence de k variables codant un état du modèle original vérifie soit ϕ ou ψ. Ceci explique tout le développement avant la sous-formule $\mathsf{X}(\phi' \mathsf{U} \psi')$.

La taille de la formule obtenue $f(\phi, 0)$ est exponentielle par rapport à la taille de ϕ. Nous pouvons facilement montrer que ϕ est satisfaisable par un modèle de $\mathrm{CLTL}^l_k(\mathcal{D})$ ssi $f(\phi, 0)$ est satisfaisable par un modèle de $\mathrm{CLTL}^{l'}_{k'}(\mathcal{D})$ en utilisant la correspondance entre les modèles définis au début de cette preuve. Pour tout modèle $\sigma_k : \mathbb{N} \to \{x_1, \ldots, x_k\} \to D$ il existe un modèle $\sigma_{k'} : \mathbb{N} \to \{x_1, \ldots, x_{k'}\} \to D$ tel que pour tout $i \in \mathbb{N}$ et $j \in \{1, \ldots, k\}$ on a $\sigma_k(i)(x_j) = \sigma_{k'}(i')(x_{j'})$ tel que $i = i' \times m + r$ et $j' = r \times k + j$ avec $i', j', r \geq 0$. Pour tout modèle $\sigma_{k'}$, nous pouvons aussi définir la construction inverse de σ_k vérifiant la même propriété. Par construction de f, nous avons $\sigma_k, i \models \phi$ ssi $\sigma_{k'}, i' \models f(\phi, pos)$ où $i = i'm + pos$ tel que $i' \geq 0$ et $0 \leq pos \leq m - 1$. □

Ce résultat permet par exemple de réduire le problème de la satisfaisabilté pour n'importe quel fragment de la forme $\mathrm{CLTL}^l_k(\mathcal{D})$ au même problème pour $\mathrm{CLTL}^1_{kl}(\mathcal{D})$. Comme pour tout fragment $\mathrm{CLTL}^l_k(\mathcal{D})$, la réduction vers $\mathrm{CLTL}^1_{kl}(\mathcal{D})$ se fait de manière uniforme, nous avons le corollaire suivant.

Corollaire 1 *Le problème de la satisfaisabilité pour* $\mathrm{CLTL}(\mathcal{D})$ *peut se réduire en temps exponentiel au problème de la satisfaisabilité pour* $\mathrm{CLTL}^1(\mathcal{D})$.

Nous définissons maintenant un réduction qui permet de réduire le nombre de variables de la formule tout en augmentant la longueur temporelle.

Lemme 3 *Pour tous $k, l, k', l' \in \mathbb{N}\backslash\{0\}$ et pour tout domaine concret $\mathcal{D}$ muni d'une relation d'égalité et ayant au moins trois éléments, il existe une réduction en espace logarithmique de $\mathrm{CLTL}^l_k(\mathcal{D})$ vers $\mathrm{CLTL}^{l'}_{k'}(\mathcal{D})$ lorsque $3k \times l \leq k' \times l'$ et $k = k' \times m$ pour $m \geq 2$.*

Preuve : L'idée de cette réduction est de coder un état d'un modèle de $\mathrm{CLTL}^l_k(\mathcal{D})$ par $3m$ états d'un modèle de $\mathrm{CLTL}^{l'}_{k'}(\mathcal{D})$. Pour des raisons techniques liées à la transitivité de la relation d'égalité, seulement un état sur trois dans le modèle de $\mathrm{CLTL}^{l'}_{k'}(\mathcal{D})$ code les valeurs du modèle de $\mathrm{CLTL}^l_k(\mathcal{D})$. Les états intermédiaires sont utilisés pour marquer le début des séquences de $3m$ états consécutifs qui correspondent à un seul état du modèle de $\mathrm{CLTL}^l_k(\mathcal{D})$. Par exemple, le modèle de $\mathrm{CLTL}^l_2(\mathcal{D})$ suivant

$$\begin{pmatrix} x^0_1 \\ x^0_2 \end{pmatrix}\begin{pmatrix} x^1_1 \\ x^1_2 \end{pmatrix}\cdots$$

est codé par le modèle de $\mathrm{CLTL}^{l'}_1(\mathcal{D})$ ci dessous

$$\boxed{x^0_1 = \circ} \neq \circ \neq x^0_2 \neq \circ \neq \circ \neq \boxed{x^1_1 = \circ} \neq \circ \neq x^1_2 \neq \circ \neq \circ \ldots$$

où $\circ$ dénote des valeurs arbitraires qui satisfont les relations avec leurs voisins représentées sur le schéma. Notons que pour que ces relations puissent toujours être satisfaites, il faut que le domaine d'interprétation comporte au moins trois éléments distincts. Un état du modèle de $\mathrm{CLTL}^{l'}_{k'}(\mathcal{D})$ est le début d'une séquence codant un état du modèle de $\mathrm{CLTL}^l_k(\mathcal{D})$ ssi la valeur de la variable x_1 à l'état suivant est identique à celle de l'état courant (le choix de la variable x_1 est arbitraire).

Nous définissons une fonction f qui transforme une formule de $\mathrm{CLTL}^l_k(\mathcal{D})$ en formule de $\mathrm{CLTL}^{l'}_{k'}(\mathcal{D})$ en vérifiant la correspondance décrite ci-dessus et qui force ainsi les variables $x_1, \ldots, x_k$ à être interprétées par blocs de $3m$ états dans la formule résultante.

- $f(\mathrm{R}(\mathsf{X}^{a_1}x_{b_1}, \ldots, \mathsf{X}^{a_s}x_{b_s})) = \mathrm{R}(\mathsf{X}^{a'_1}x_{b'_1}, \ldots, \mathsf{X}^{a'_s}x_{b'_s})$ où pour tout $i \in \{1, \ldots, s\}$ on a $a'_i = 3ma_i + 3q_i$ et $b'_i = r_i + 1$ tel que $q_i > 0$ et $r_i \in \{0, \ldots, k'-1\}$ sont respectivement le quotient et le reste de la division euclidienne de $b_i - 1$ par k', c'est-à-dire que $b_i - 1 = q_ik' + r_i$.
  Cette opération établie la correspondance entre les termes du modèle original avec le modèle d'arrivée. Puisqu'un état du modèle de départ correspond à $3m$ états du modèle d'arrivée, X^i dans le modèle original correspond à X^{3mi} dans le modèle d'arrivée. Ensuite il faut rajouter $3q$ état où q est le quotient de la division de $b - 1$ par k' car le modèle d'arrivée a moins de variables et qu'on se sert de 3 états consécutifs pour coder chaque variable du modèle initial. Nous calculons enfin à quelle variable de l'état d'arrivée correspond le terme à partir du reste de la division.
- f est homomorphique par rapport aux opérateurs Booléens,
- $f(\mathsf{X}\phi) = (x_1 \neq \mathsf{X}x_1)\mathsf{U}((x_1 = \mathsf{X}x_1) \wedge f(\phi))$.
  Pour passer à l'état correspondant à un état suivant du modèle original, il faut se caler sur le prochain état du modèle d'arrivée vérifiant $(x_1 = \mathsf{X}x_1)$.
- $f(\phi\mathsf{U}\psi) = ((x_1 = \mathsf{X}x_1) \Rightarrow f(\phi))\mathsf{U}((x_1 = \mathsf{X}x_1) \wedge f(\psi))$.
  Seuls les états vérifiant $(x_1 = \mathsf{X}x_1)$ correspondent au début d'un état du modèle original. Il faut que la formule $f(\phi)$ soit vérifiée dans tous les états vérifiant cette propriété

jusqu'à ce que $f(\psi)$ le soit.

La taille de $f(\phi)$ est linéaire par rapport à $|\phi|$.

Soit ϕ_{3m} la formule suivante exprimant que la contrainte $x_1 = \mathsf{X}x_1$ est vraie exactement tous les $3m$ états :

$$x_1 = \mathsf{X}x_1 \wedge \bigwedge_{1 \leq i \leq 3m-1} \mathsf{X}^i(x_1 \neq \mathsf{X}x_1) \wedge \mathsf{G}(x_1 = \mathsf{X}x_1 \Leftrightarrow \mathsf{X}^{3m}x_1 = \mathsf{X}x_1)$$

Un modèle σ satisfait ϕ_{3m} ssi pour tout $i \in \mathbb{N}$ on a $\sigma, i \models x_1 \neq \mathsf{X}x_1 \Leftrightarrow i \equiv_{3m} 0$.

Nous pouvons facilement montrer que la formule ϕ de $\mathrm{CLTL}^l_k(\mathcal{D})$ est satisfaisable ssi la formule $f(\phi) \wedge \phi_{3m}$ est satisfaisable. Pour tout modèle $\sigma_k : \mathbb{N} \to \{x_1, \ldots, x_k\} \to D$ il existe un modèle $\sigma_{k'} : \mathbb{N} \to \{x_1, \ldots, x_{k'}\} \to D$ tel que pour tout $i \in \mathbb{N}$ et $j \in \{1, \ldots, k\}$ on a $\sigma_k(i)(x_j) = \sigma_{k'}(3mi + 3q)(r + 1)$ tel que $j - 1 = qk' + r$ $(q, r \geq 0)$ et $\sigma_{k'} \models \phi_{3m}$. Ceci correspond à la transformation décrite au début de cette preuve, qui est possible car D contient 3 valeurs distinctes. Étant donné un modèle $\sigma_{k'}$ tel que $\sigma_{k'} \models \phi_{3m}$, nous pouvons aussi construire un modèle σ_k vérifiant la même correspondance des valeurs en gardant les valeurs significatives de $\sigma_{k'}$. Par construction de f, nous avons $\sigma_k, i \models \phi$ ssi $\sigma_{k'}, 3im \models f(\phi) \wedge \phi_{3m}$ (par induction sur la structure de ϕ). □

De la même façon que pour le résultat précédent, le Lemme 3 nous permet de définir une réduction uniforme des formule de $\mathrm{CLTL}(\mathcal{D})$ tel que $\mathcal{D}$ est muni d'une relation d'égalité et a au moins trois éléments, vers $\mathrm{CLTL}_1(\mathcal{D})$.

Corollaire 2 *Le problème de la satisfaisabilité pour* $\mathrm{CLTL}(\mathcal{D})$ *tel que* $\mathcal{D}$ *est muni d'une relation d'égalité et contient au moins trois éléments distincts peut se réduire en temps logarithmique au problème de la satisfaisabilité pour* $\mathrm{CLTL}_1(\mathcal{D})$.

3.1.3 Logiques arborescentes

Nous nous intéressons maintenant au cas des logiques arborescentes. L'extension de la logique temporelle $\mathrm{CTL}^\star$ sur un domaine concret $\mathcal{D} = \langle D, \mathcal{R} \rangle$, notée $\mathrm{CCTL}^\star(\mathcal{D})$, est définie de la manière suivante.

$$\begin{aligned} \phi &::= \top \mid E\psi \mid \neg\phi \mid \phi \wedge \phi \\ \psi &::= \phi \mid \mathrm{R}(\mathsf{X}^{i_1}x_{j_1}, \ldots, \mathsf{X}^{i_k}x_{j_k}) \mid \neg\psi \mid \psi \wedge \psi \mid \mathsf{X}\psi \mid \psi \mathsf{U} \psi \end{aligned}$$

où $x_{j_1}, \ldots, x_{j_k} \in \mathrm{VAR}$ et $\mathrm{R} \in \mathcal{R}$. La longueur temporelle d'une formule est définie de la même manière que dans le cas linéaire ainsi que les opérateurs F et G. Le quantificateur E est un quantificateur existentiel sur les exécutions, et nous utilisons son dual $A\psi$ pour noter la quantification universelle qui est équivalente à $\neg E \neg \psi$. Comme dans le cas propositionnel, nous distinguons dans $\mathrm{CCTL}^\star(\mathcal{D})$ les *formules d'état* représentées dans la définition par ϕ et les *formules de chemin* représentées par ψ. Notons que les formules atomiques $\mathrm{R}(\mathsf{X}^{i_1}x_{j_1}, \ldots, \mathsf{X}^{i_k}x_{j_k})$ de la logique sont interprétées par rapport à un chemin à cause de la présence des termes. En effet, lorsqu'on utilise un terme $\mathsf{X}^i x$ il est nécessaire de spécifier si l'on réfère à une valeur future possible pour la variable x dans i états ou bien à toutes les valeurs possibles. Cependant, la logique $\mathrm{CTL}^\star$ peut toujours être définie par rapport à une instance particulière de CCTL$^\star(\mathcal{D})$. En effet CTL* est équivalent à CCTL$^\star(\mathcal{D})$ tel que $\mathcal{D} = \langle \{0, 1\}, [\![]\!] \rangle$ où $[\![]\!]$ est une relation d'interprétation unaire telle que $[\![x]\!] = \top$ ssi $x = 1$. Dans les logiques que nous étudions par la suite, les variables propositionnelles peuvent la plupart du temps être rajoutées en utilisant des variables fraîches sous condition que le domaine soit non-trivial.

Les modèles de CCTL$^\star(\mathcal{D})$ sont des arbres associant à chaque sommet une valuation de la forme VAR $\rightarrow D$. Soit $T = \langle N, \{n_0\}, \rightarrow, \Gamma \rangle$ tel que N est un ensemble fini ou infini de sommets, $n_0 \in N$ est la racine du modèle, $\rightarrow \subseteq N \times N$ est un ensemble d'arcs tel que pour tout $n \in N$ il existe un sommet $n' \in N$ tel que $n \rightarrow n'$, et $\Gamma : N \rightarrow (\text{VAR} \rightarrow D)$ est une fonction associant à chaque sommet une valuation. Les modèles de CCTL$^\star(\mathcal{D})$ héritent des définitions que nous avons énoncées pour CTL* (voir la Section 1.2), en particulier :

- le degré d'un sommet est le nombre d'arêtes ayant pour origine ce sommet et le degré d'un graphe est la borne supérieure de l'ensemble des degrés de ces noeuds,
- un *d-arbre* est un arbre dont tous les sommets ont pour degré d,
- les successeurs d'un noeud sont ordonnés.

Nous rappelons qu'un chemin dans T est une séquence $\pi = n_0 \cdot n_1 \cdots$ telle que pour tout $0 \leq i < |\pi|$ on a $n_i \rightarrow n_{i+1}$. Notons la correspondance entre un chemin dans T et un modèle dans le cas linéaire. Nous notons aussi $\pi^i = n_i \cdot n_{i+1} \cdots$ le $i^{\text{ème}}$ suffixe de π et $\pi(i)$ le $i^{\text{ème}}$ sommet. La relation de satisfaction pour une formule d'état est définie par :

- $\langle T, n \rangle \models \top$ pour tout n,
- $\langle T, n \rangle \models \neg\phi$ ssi $\langle T, n \rangle \not\models \phi$,
- $\langle T, n \rangle \models \phi \wedge \phi'$ ssi $\langle T, n \rangle \models \phi$ et $\langle T, n \rangle \models \phi'$,
- $\langle T, n \rangle \models E\ \psi$ ssi il existe un chemin infini $\pi = n_0 \cdot n_1 \cdots$ dans T tel que $n_0 = n$ et $\langle T, \pi \rangle \models \psi$,

et la relation de satisfaction pour une formule de chemin par

- $\langle T, \pi \rangle \models \phi$ ssi $\langle T, \pi(0) \rangle \models \phi$,
- $\langle T, \pi \rangle \models \mathrm{R}(\mathsf{X}^{i_1} x_{j_1}, \ldots, \mathsf{X}^{i_k} x_{j_k})$ ssi $(\Gamma(\pi(i_1))(x_{j_1}), \ldots, \Gamma(\pi(i_k))(x_{j_k})) \in R$,
- $\langle T, \pi \rangle \models \neg\psi$ ssi $\langle T, \pi \rangle \not\models \psi$,
- $\langle T, \pi \rangle \models \psi \wedge \psi'$ ssi $\langle T, \pi \rangle \models \psi$ et $\langle T, n \rangle \models \psi'$,
- $\langle T, \pi \rangle \models \mathsf{X}\psi$ ssi $\langle T, \pi^1 \rangle \models \psi$,
- $\langle T, \pi \rangle \models \psi \mathsf{U} \psi'$ ssi il existe $i \in \mathbb{N}$ tel que $\langle T, \pi^i \rangle \models \psi'$ et pour tout $0 \leq j < i$ on a $\langle T, \pi^j \rangle \models \psi$.

Une formule ϕ de CCTL$^\star(\mathcal{D})$ est satisfaisable ssi il existe un modèle $T = \langle N, \{n_0\}, \rightarrow, \Gamma \rangle$ tel que $\langle T, n_0 \rangle \models \phi$. Le problème de la satisfaisabilité consiste à déterminer si une formule donnée est satisfaisable.

Nous démontrons pour conclure cette section quelques propriétés sur les formules et modèles de CCTL$^\star(\mathcal{D})$ qui nous permettront par la suite de faire des restrictions sans perte de généralité. Une formule de CCTL$^\star(\mathcal{D})$ est sous forme positive si elle peut être définie par rapport à la grammaire suivante :

$$\phi ::= \top \mid E\psi \mid A\psi \mid \phi \wedge \phi \mid \phi \vee \phi$$
$$\psi ::= \phi \mid \mathrm{R}(\mathsf{X}^{i_1}x_{j_1}, \ldots, \mathsf{X}^{i_k}x_{j_k}) \mid \neg\mathrm{R}(\mathsf{X}^{i_1}x_{i_1}, \ldots, \mathsf{X}^{i_k}x_{j_k}) \mid \psi \wedge \psi \mid \psi \vee \psi \mid \mathsf{X}\psi \mid \psi\mathsf{U}\psi \mid \psi\tilde{\mathsf{U}}\psi$$

où $x_{j_1}, \ldots, x_{j_k} \in$ VAR, $\mathrm{R}(\mathsf{X}^{i_1}x_{j_1}, \ldots, \mathsf{X}^{i_k}x_{j_k})$ est une contrainte de $\mathcal{D}$ et $\tilde{\mathsf{U}}$ est le dual de l'opérateur U, c'est-à-dire que $\phi\tilde{\mathsf{U}}\phi' \equiv \neg(\neg\phi\mathsf{U}\neg\phi')$. N'importe quelle formule de CCTL$^\star(\mathcal{D})$ peut s'écrire sous forme positive.

Lemme 4 *Pour toute formule ϕ de* CCTL$^\star(\mathcal{D})$*, il existe une formule ϕ' de* CCTL$^\star(\mathcal{D})$ *sous forme positive telle que ϕ est satisfaisable ssi ϕ' est satisfaisable.*

Preuve : Étant donnée une formule ϕ de CCTL$^\star(\mathcal{D})$, nous pouvons définir une normalisation des formules qui pousse les négations au niveau atomique. La fonction f est définie par induction sur la structure de ϕ.

- f est égal l'identité pour les cas $\mathrm{R}(\mathsf{X}^{i_1}x_{j_1}, \ldots, \mathsf{X}^{i_k}x_{j_k})$ et $\neg\mathrm{R}(\mathsf{X}^{i_1}x_{j_1}, \ldots, \mathsf{X}^{i_k}x_{j_k})$,
- $f(\neg\neg\phi) = f(\phi)$,
- $f(\neg(\phi \wedge \phi')) = f(\neg\phi) \vee f(\neg\phi')$,
- $f(\neg(\phi \vee \phi')) = f(\neg\phi) \wedge f(\neg\phi')$,
- $f(\neg E\phi) = Af(\neg\phi)$,
- $f(\neg A\phi) = Ef(\neg\phi)$,
- $f(\neg\mathsf{X}\phi) = \mathsf{X}f(\neg\phi)$,
- $f(\neg(\phi\mathsf{U}\phi')) = f(\neg\phi)\tilde{\mathsf{U}}f(\neg\phi')$,
- $f(\neg(\phi\tilde{\mathsf{U}}\phi')) = f(\neg\phi)\mathsf{U}f(\neg\phi')$,
- pour les cas restant, la fonction est homomorphique.

Il est évident que le calcul de $f(\phi)$ termine puisque les appels récursifs dans chaque cas se font sur des sous-formules et que nous identifions $\neg\neg\phi$ avec ϕ. Par construction, les seules sous-formules qui peuvent être négatives sont les contraintes atomiques et donc $f(\phi)$ est sous forme positive. Enfin, il est facile de montrer par induction que ϕ est satisfaisable ssi $f(\phi)$ est satisfaisable car toutes les règles de définition de f respectent des équivalences logiques. □

Nous remarquons que la mise sous forme positive se fait en temps linéaire et que la taille de ϕ' est du même ordre que celle de ϕ. On a besoin d'un seul parcours de la formule pour la réécrire sous forme positive en poussant les négations à l'intérieur des sous-formules. Nous notons $E_\sharp(\phi)$ le nombre de quantificateurs existentiels d'une formule sous forme positive ϕ.

Nous définissons maintenant des propriétés sur l'ensemble des modèles d'une formule de CCTL$^\star(\mathcal{D})$. Le résultat suivant énonce le même genre de propriété de modèle réduit pour CCTL$^\star(\mathcal{D})$ que dans le cas propositionnel (voir [ES84]).

Lemme 5 *Une formule ϕ de* CCTL$^\star(\mathcal{D})$ *est satisfaisable ssi il existe un arbre T de degré $E_\sharp(\phi) + 1$ ayant pour racine n_0 tel que $\langle T, n_0\rangle \models \phi$.*

Preuve : Soit ϕ une formule de CCTL$^\star(\mathcal{D})$ et $d = E_\sharp(\phi)+1$. Nous notons $\{E\psi_1, \ldots, E\psi_d\}$ l'ensemble des sous-formules existentielles de ϕ. Supposons qu'il existe un modèle $T = \langle N, \{n_0\}, \rightarrow, \Gamma\rangle$ qui satisfait ϕ. Nous définissons la construction d'un nouveau modèle T' qui préserve l'ensemble des formules d'états vérifiées à chaque état en copiant les chemins qui servent à satisfaire les différentes formules de cet ensemble. Ainsi, pour chaque sommet n' de T' correspondant à un sommet n de T et chaque formule existentielle $E\psi_i$ telle que $\langle T, n\rangle \models E\psi_i$, il existe un chemin distinct dans T' ayant pour origine n' qui satisfait $E\psi_i$. L'arbre $T' = \langle N', \{n'_0\}, \rightarrow', \Gamma'\rangle$ est défini de la façon suivante :

- L'ensemble des sommets est tel que $N' \subseteq N \times \{1, \ldots, d\}^*$. Chaque noeud de T' est noté sous la forme d'une paire $\langle n, w\rangle$ telle que la première composante représente le sommet de T auquel il correspond et la deuxième décrit le chemin depuis la racine de T' jusqu'à ce sommet. Ce chemin est unique car un arbre n'a pas de cycle et nous sert à différencier les différentes copies éventuelles d'un même sommet du modèle T original. Nous appelons la longueur de ce chemin le niveau du noeud.
- La racine est $n'_0 = \langle n_0, \emptyset\rangle$.
- La fonction d'étiquetage vérifie $\Gamma'(\langle n, w\rangle) = \Gamma(n)$ pour tout $\langle n, w\rangle \in N'$.
- Nous définissons la relation $\rightarrow'$ par induction sur le niveau des sommets.
  Supposons qu'il existe un chemin entre n'_0 et un noeud $\langle n, w\rangle$ et posons $\{E\psi_1^n, \ldots, E\psi_k^n\}$ l'ensemble des formules existentielles satisfaites par n dans T. Pour chaque formule $E\psi_j$ appartenant à cet ensemble, il existe un chemin infini $\pi = n_0^j \cdot n_1^j \cdot n_2^j \cdots$ dans T tel que $n_0^j = n$ et $\langle G, \pi\rangle \models E\psi_j$. Nous ajoutons une copie de ce chemin dans T' en définissant les arcs suivants
  - $\langle n, w\rangle \xrightarrow{j}' \langle n_1^j, w \cdot j\rangle$
  - $\langle n_l^j, w \cdot (0)^l\rangle \xrightarrow{0}' \langle n_{l+1}^j, w \cdot (0)^{l+1}\rangle$ pour tout $l > 0$, où $(0)^l$ représente le mot composé de l occurrences du caractère 0.

  Si l'ensemble de formules existentielles est vide alors nous copions n'importe lequel des chemins au départ du noeud correspondant dans T' afin de préserver le caractère complet de l'ensemble des arcs du modèle.

Cette définition implique que pour chaque sommet, le chemin qui suit tout le temps la direction 0 est défini et correspond au suffixe d'une copie d'un chemin déjà défini ayant pour origine un sommet de niveau inférieur. Cependant, la construction au niveau i n'ajoute jamais de successeur qui n'est pas un 0-successeur à un sommet de niveau supérieur à i. Les autres successeurs sont définis lorsque l'on traite le sommet en question. Chaque sommet de T' a donc au plus d successeurs. Nous pouvons alors facilement montrer que $\langle T', n'_0\rangle \models \phi$ par induction sur la structure de la formule ϕ. Cette propriété est directe considérant que l'ensemble des formules existentielles satisfaites par chaque sommet est préservée.

L'implication inverse est triviale. □

Ce résultat peut être légèrement modifié de la façon suivante, afin d'obtenir une classe de modèles encore plus restreinte.

Corollaire 3 *Une formule ϕ de* CCTL$^\star(\mathcal{D})$ *est satisfaisable ssi elle est satisfaite par un* $(E_\sharp(\phi)+1)$*-arbre.*

Preuve : Dans la construction précédente le nombre de successeurs de chaque sommet dépend du nombre de formules existentielles satisfaites dans le modèle initial. Nous pouvons facilement compléter cette construction en rajoutant des copies d'un chemin déjà issu de la construction afin que chaque sommet ait un nombre de successeurs égal à $E_\sharp(\phi) + 1$. Comme ces copies ne modifient pas l'ensemble de formules existentielles satisfaites à chaque position, la propriété est toujours vérifiée. □

3.2 Automates à contraintes et model-checking

3.2.1 Définition des automates à contraintes

Nous définissons le problème du model-checking pour les logiques temporelles étendues sur des domaines concrets par rapport à une classe particulière d'automates à contraintes. Pour tout domaine concret $\mathcal{D}$, un *$\mathcal{D}$-automate* est un automate de Büchi avec un ensemble de variables associé (comme pour les automates à compteur) dont les transitions sont étiquetées par des combinaisons Booléennes de contraintes transitionnelles définies par rapport à $\mathcal{D}$. Les transitions impliquent des contraintes sur les valeurs des variables à l'état courant et à l'état suivant, ce qui permet à la fois d'exprimer des tests comme par exemple $x = 0$ où des mises à jour telles que $\mathsf{X}x = x + 1$. Formellement, un $\mathcal{D}$-automate $\mathcal{A}$ avec k variables est une structure $\langle Q, \delta, I, F\rangle$ telle que :

- Q est un ensemble fini d'états de contrôle,
- $I \subseteq Q$ est un ensemble d'états initiaux,
- $F \subseteq Q$ est un ensemble d'états finaux,
- $\delta \subseteq Q \times 1SC_k(\mathcal{D}) \times Q$ est une relation de transition finie, où $1SC_k(\mathcal{D})$ représente l'ensemble des formules obtenues par combinaisons Booléennes de contraintes transitionnelles définies par rapport à $\mathcal{D}$ avec les variables $\{x_1, \ldots, x_k\}$.

Nous utilisons la notation $q \xrightarrow{\alpha} q'$ lorsque $\langle q, \alpha, q'\rangle \in \delta$. Notons qu'en général, les $\mathcal{D}$-automates que nous considérons par la suite n'ont pas d'alphabet. Ils servent plus de modèles opérationnels que d'accepteurs de langages.

Une *configuration* de $\mathcal{A}$ est une paire $\langle q, v\rangle \in Q \times D^k$ où v est une valuation pour les variables $\{x_1, \ldots, x_k\}$. La sémantique sur un pas est définie par $\langle q, v\rangle \to \langle q', v'\rangle$ ssi il existe une transition $\langle q, \alpha, q'\rangle \in \delta$ dans $\mathcal{A}$ telle que la valuation qui pour tout $i \in \{1, \ldots, k\}$ associe à x_i la valeur $v(x_i)$ et à $\mathsf{X}x$ la valeur de $v'(x_i)$ satisfait la contrainte α. Nous notons $\langle q, v\rangle \xrightarrow{\alpha} \langle q', v'\rangle$ lorsque nous avons besoin de préciser la contrainte sur la transition empruntée.

3.2.2 Exécutions linéaires d'un $\mathcal{D}$-automate

Une exécution linéaire de $\mathcal{A}$, que nous appelons parfois chemin dans $\mathcal{A}$, est une séquence finie ou infinie π d'éléments de $Q \times D^k$ telle que pour tout $0 \leq i < |\pi| - 1$ on a $\pi(i) \to \pi(i+1)$. Nous notons $\pi(i)$ le $i^{\text{ème}}$ élément de π et π^i le suffixe commençant à la position i. Pour des raisons de présentation, nous explicitons parfois les transitions entre les configurations successives de π en notant π de la façon suivante $\pi(0) \xrightarrow{\alpha_0} \pi(1) \xrightarrow{\alpha_1} \cdots$.

Nous notons $\rightarrow^*$ la cloture transitive de $\rightarrow$ telle que $\langle q, v\rangle \rightarrow^* \langle q', v'\rangle$ ssi il existe un chemin fini de $\langle q, v\rangle$ à $\langle q', v'\rangle$. Une exécution acceptante pour $\mathcal{A}$ est une exécution infinie π telle que $\pi(0) \in I \times D^k$ et l'ensemble $\{i \in \mathbb{N} : \pi(i) \in F \times D^k\}$ est infini. La condition d'acceptation est donc une condition de Büchi standard qui ne tient pas compte de la valeur des variables. Un modèle $\sigma : \mathbb{N} \rightarrow (\mathrm{VAR} \rightarrow D)$ réalise une exécution infinie $\pi = \pi(0) \xrightarrow{\alpha_0} \pi(1) \xrightarrow{\alpha_1} \cdots$ de $\mathcal{A}$ ssi pour tout $i \in \mathbb{N}$ la configuration $\pi(i)$ est de la forme $\langle q_i, \sigma(i)\rangle$ pour un état $q_i \in Q$.

Le problème du model-checking pour $\mathrm{CLTL}(\mathcal{D})$ prend comme entrées une formule ϕ de $\mathrm{CLTL}(\mathcal{D})$ et un $\mathcal{D}$-automate et consiste à déterminer s'il existe un modèle σ qui réalise une exécution acceptante de $\mathcal{A}$ et satisfait ϕ. Si la réponse est positive, nous notons alors $\mathcal{A} \models \phi$. Pour la restriction du problème au fragment $\mathrm{CLTL}^l_k(\mathcal{D})$, la formule ϕ appartient à $\mathrm{CLTL}^l_k(\mathcal{D})$ et l'automate $\mathcal{A}$ est un $\mathcal{D}$-automate avec k variables. Sans perte de généralité, nous supposons aussi que ϕ et $\mathcal{A}$ sont construits sur le même ensemble de variables.

Notons que le problème de la satisfaisabilité se réduit facilement au problème du model-checking. En effet, considérons le $\mathcal{D}$-automate $\mathcal{A}_\top$ dont l'ensemble des états est un singleton $\{q_0\}$ et la relation de transition est définie par la seule règle $q_0 \xrightarrow{\top} q_0$. Le langage reconnu par cet automate correspond à l'ensemble des modèles de $\mathrm{CLTL}(\mathcal{D})$ car toute séquence d'éléments de $Q \times D^{\mathrm{VAR}}$ est acceptée. Par conséquent, il est facile de montrer que $\mathcal{A}_\top \models \phi$ ssi ϕ est satisfaisable. La réduction inverse est aussi possible en utilisant la méthode de [SC85]. Le comportement d'un $\mathcal{D}$-automate dans le cas linéaire peut en effet être codé par une formule de $\mathrm{CLTL}(\mathcal{D})$. La preuve de cette affirmation est similaire à la réduction pour LTL (voir Section 1.1.1). Nous avons donc le résultat suivant.

Théorème 22 *Le problème de la satisfaisabilité de* $\mathrm{CLTL}(\mathcal{D})$ *et le problème du model-checking de* $\mathrm{CLTL}(\mathcal{D})$ *sur les $\mathcal{D}$-automates sont inter-reductibles en espace logarithmique.*

3.2.3 Exécutions arborescentes d'un $\mathcal{D}$-automate

Dans le cas des logiques arborescentes, une exécution d'un $\mathcal{D}$-automate $\mathcal{A}$ est représentée sous la forme d'un dépliage dont les sommets sont étiquetés par des configurations de $\mathcal{A}$ appartenant à l'ensemble $Q \times D^k$. De plus, l'étiquetage respecte la relation de transition. Formellement, une exécution d'un $\mathcal{D}$-automate $\mathcal{A} = \langle Q, \delta, I, F\rangle$ avec k variables est un arbre $\langle N, \{n_0\}, \rightarrow, \Gamma\rangle$ avec une fonction d'étiquetage de la forme $\Gamma : N \rightarrow Q \times D^k$ tel que :

- $\Gamma(n_0) = \langle q_0, \vec{0}\rangle$ où $q_0 \in I$ et $\vec{0}$ est une valuation initiale,
- Pour tout sommet $n \in N$ tel que $\Gamma(n) = \langle q, v\rangle$ et toute transition $\langle q, v\rangle \xrightarrow{\alpha} \langle q', v'\rangle$ possible dans $\mathcal{A}$ il existe un successeur de n étiqueté par $\langle q', v'\rangle$.

Une telle exécution est acceptante ssi chaque branche infinie visite infiniment souvent un état acceptant, ce qui correspond à une condition de Büchi standard pour les automates d'arbres. Notons que chaque branche $\pi = n_0 \cdot n_1 \cdots$ ayant pour état initial la racine correspond à une exécution linéaire. Un $\mathcal{D}$-automate $\mathcal{A}$ satisfait une formule ϕ de $\mathrm{CCTL}^\star(\mathcal{D})$ ssi il existe une exécution $T = \langle N, \{n_0\}, \rightarrow, \Gamma\rangle$ de cet automate telle que $\langle T', n_0\rangle \models \phi$ où $T' = \langle N, \{n_0\}, \rightarrow, \Gamma'\rangle$ est l'arbre correspondant à T dont la fonction d'étiquetage $\Gamma' : N \rightarrow D^k$ est la restriction de Γ aux valuations affectées à chaque sommet. Dans ce cas, nous notons $\mathcal{A} \models \phi$. Plus généralement, nous notons $[\![\phi]\!]_\mathcal{A}$ l'ensemble des états de $\mathcal{A}$ qui satisfont ϕ. Nous avons $\langle q, v\rangle \in [\![\phi]\!]_{\mathcal{A}}$ ssi il existe une exécution acceptante pour $\mathcal{A}$ qui satisfait ϕ lorsque l'on fixe la racine comme étant étiquetée par $\langle q, v\rangle$ (voir la construction du dépliage ci-dessus). Donc on a $\mathcal{A} \models \phi$ ssi il existe $q_0 \in I$ tel que $\langle q_0, \vec{0}\rangle \in [\![\phi]\!]_{\mathcal{A}}$.

Le problème du model-checking pour $\mathrm{CCTL}^\star(\mathcal{D})$ prend comme entrée une formule ϕ de $\mathrm{CCTL}^\star(\mathcal{D})$ et un $\mathcal{D}$-automate $\mathcal{A}$ et consiste à vérifier si il existe une exécution de $\mathcal{A}$ qui satisfait ϕ. Notons que ce problème ne se réduit pas au problème de la satisfaisabilité. En effet, si nous considérons en particulier une exécution de l'automate $\mathcal{A}_\top$ défini précédemment, nous remarquons que l'ensemble des successeurs de l'état initial étiqueté par $\langle q_0, \vec{0}\rangle$ est infini puisque les variables ne sont pas contraintes par la transition $q_0 \xrightarrow{\top} q_0$. Nous n'avons aucun moyen d'exprimer ce genre de comportement dans la logique $\mathrm{CCTL}^\star(\mathcal{D})$.

3.3 Domaines induits par l'arithmétique de Presburger

3.3.1 Spécification de propriétés sur les systèmes à compteurs

Dans la suite nous considérons plusieurs extensions de logiques temporelles utilisant des domaines concrets particuliers dont les contraintes sont induites par l'arithmétique de Presburger, c'est-à-dire la théorie du premier ordre sur les entiers sans la multiplication [Pre29]. Les relations d'un domaine concret définissent un langage de contraintes qui peut aussi être défini par une grammaire dans certains cas. Pour tout langage de contraintes L inclus dans l'arithmétique de Presburger, nous notons simplement $\mathrm{CLTL}(L)$, $\mathrm{CCTL}^\star(L)$ ou $\mathrm{CCTL}(L)$ l'extension d'une logique temporelle sur le domaine concret induit par ce langage. Le domaine d'interprétation pour les variables n'est pas explicité dans cette notation. Cependant, il est facile de mettre en relation cette définition avec la définition précédente des logiques sur domaines concrets : le domaine d'interprétation est $\mathbb{Z}$ et l'ensemble infini des relations est donné par le sous-ensemble de l'arithmétique de Presburger correspondant au langage.

Étendre des logiques temporelles avec de telles contraintes permet d'exprimer des propriétés sur des automates à compteur qui sont contenus dans la classe des automates avec contraintes de Presburger. Ceci est particulièrement intéressant car ce modèle à de nombreuses applications en vérification. Il permet entre autre la représentation symbolique de nombreux systèmes ayant un nombre infinis d'états tels que les protocoles de diffusion [FL02] ou les programmes manipulant des variables de pointeurs [BFLS06, BBH+06]. Même la restriction de ce modèle à un seul compteur a des applications dans la vérification de protocoles cryptographiques [LLT05] ou la validation de documents XML [CR04]. Cependant de nombreux problèmes de model-checking pour les automates à compteur, tel que la simple atteignabilité d'un état de contrôle, sont indécidables. C'est déjà le cas pour les machines de Minsky à deux compteurs [Min67]. Il existe néanmoins des classes restreintes d'automates à compteurs dont certains problèmes de model-checking sont décidables (voir la liste non exhaustive de la Section 2.3).

De nombreux travaux établissent des résultats de complexité pour LTL étendu avec des contraintes de Presburger [BEH95, CC00, DD07, Dem04]. D'autres extensions pouvant être associées à cette famille de logiques sont définies dans [BEH95, CC00, ID01]. Il existe moins de résultats connus sur les extensions arborescentes de ce type de logiques. Dans [DFGvD06] le model-checking de $\mathrm{CTL}^\star$ avec des contraintes de Presburger sur des classes restreintes d'automates avec contraintes de Presburger est considéré. Dans [Čer94] c'est le langage logique qui est restreint au fragment existentiel de CTL*. L'extension de logiques temporelles avec des contraintes de Presburger nous permet à la fois de traiter des problèmes liés aux langages logiques qui sont plus larges que la simple atteignabilité d'un état de contrôle, mais aussi d'exprimer des contraintes sur le comportement des compteurs. Bien entendu, l'extension d'une logique telle que LTL avec la totalité de l'arithmétique de Presburger est déjà indécidable puisqu'on peut coder l'exécution d'une machine de Minsky dans la logique. En effet, nous avons vu que dans les exécutions linéaires d'un $\mathcal{D}$-automate peuvent se coder par une formule de CLTL($\mathcal{D}$). Lorsque cet automate permet d'exprimer les contraintes $x = y+1$ et $x = 0$, il peut simuler une machine de Minsky. Notre objectif consiste donc à déterminer des fragments décidables de logiques temporelles avec contraintes de Presburger. Les principales restrictions que nous considérons portent sur l'ensemble de contraintes utilisées ou les ressources syntaxiques des formules telles que le nombre de variables ou la longueur temporelle. Nous présentons ci-dessous les fragments de l'arithmétique de Presburger considérés par la suite.

3.3.2 Fragments étudiés

Nous séparons en deux catégories les fragments de l'arithmétique de Presburger que nous considérons : les *fragments qualitatifs* et les *fragments quantitatifs*. Nous appelons contrainte qualitative une contrainte qui n'implique pas de relation stricte entre les variables telles que $x < y + 1$ ou $x \equiv_k y + 1$, par opposition aux contraintes quantitatives telles que $x = y + 1$. Pour préciser davantage, si l'on fixe une valeur pour x dans les deux premiers exemples les solutions possibles pour y sont multiples, ce qui n'est pas le cas pour $x = y + 1$. Nous faisons une exception pour la relation d'égalité qui appartient à tous les fragments que nous considérons. Un fragment est qualitatif lorsqu'il ne contient pas de contraintes quantitatives et quantitatif autrement. Cette classification des fragments nous permet de montrer dans quelle mesure la présence de contraintes quantitatives dans un fragment de l'arithmétique de Presburger cause plus facilement l'indécidabilité des logiques temporelles étendues avec ce fragment. Intuitivement, la possibilité de pouvoir exprimer qu'une variable est incrémentée à l'état suivant grâce à une contrainte de la forme $\mathsf{X}x = x + 1$ simplifie considérablement le codage d'une machine de Minsky. Nous définissons ci-dessous ces différents fragments par rapport à cette classification. Pour le reste de cette section, nous considérons un ensemble infini dénombrable de variables $\mathrm{VAR} = \{x_0, x_1, \ldots\}$.

Fragments qualitatifs

Nous commençons par introduire les différents fragments qualitatifs auxquels nous faisons référence. Le langage de contraintes IPC* est le plus gros fragment qualitatif de l'arithmétique de Presburger que nous considérons. L'ensemble des contraintes de IPC* est défini par la grammaire suivante :

$$\alpha ::= \theta \mid x < y \mid \alpha \wedge \alpha \mid \neg\alpha$$

$$\begin{array}{rl}\theta ::= & x \equiv_k [c_1, c_2] \mid x \equiv_k y + [c_1, c_2] \mid x = y \mid x < d \mid x = d \mid \\ & \theta \wedge \theta \mid \neg\theta \mid \exists x\ \theta\end{array}$$

où $x, y \in \mathrm{VAR}$, $k \in \mathbb{N} \setminus \{0\}$, $c_1, c_2 \in \mathbb{N}$ et $d \in \mathbb{Z}$. Le symbole $\sim$ est utilisé pour représenter $=$ ou $<$. Nous utilisons par la suite les abréviations $x \equiv_k c$ pour $x \equiv_k [c, c]$ et $x \equiv_k y + c$ pour $x \equiv_k y + [c, c]$. Nous définissons aussi les restrictions suivantes de ce langage :

- WIPC* (fragment faible) est la restriction de IPC* aux contraintes définies par

$$\alpha_w ::= x \sim y \mid x \sim d \mid x \equiv_k c \mid \alpha \wedge \alpha \mid \neg\alpha$$

  où $x, y \in \mathrm{VAR}$, $\sim \in \{<, =\}$, $d \in \mathbb{Z}$ et $k, c \in \mathbb{N}$.
- IPC^{++} est la restriction de IPC* aux contraintes représentées par θ.
- Z^c est la restriction de IPC* aux contraintes de la forme $x \sim y$ et $x \sim d$ pour $x, y \in \mathrm{VAR}$ et $d \in \mathbb{Z}$.
- Z est la restriction de IPC* aux seules contraintes $x \sim y$ pour $x, y \in \mathrm{VAR}$.

La définition de la sémantique est standard par rapport à l'arithmétique de Presburger. Une contrainte α de IPC* est satisfaite par une valuation $v : \mathrm{VAR} \to \mathbb{Z}$, noté $v \models \alpha$ ssi

- $v \models x \sim y$ ssi $v(x) \sim v(y)$;
- $v \models x \sim d$ ssi $v(x) \sim d$;
- $v \models x \equiv_k [c_1, c_2]$ ssi il existe $c_1 \leq c \leq c_2$ et $z \in \mathbb{Z}$ tels que $v(x) - c = kz$;
- $v \models x \equiv_k y + [c_1, c_2]$ ssi il existe $c_1 \leq c \leq c_2$ et $z \in \mathbb{Z}$ tels que $v(x) - v(y) - c = kz$;
- $v \models \alpha \wedge \alpha'$ ssi $v \models \alpha$ et $v \models \alpha'$;
- $v \models \neg\alpha$ ssi $v \not\models \alpha$;
- $v \models \exists x\ \alpha$ ssi il existe $d \in \mathbb{Z}$ tel que $v[x \leftarrow d] \models \alpha$
  où $v[x \leftarrow z]$ est la valuation qui coïncide avec v sur toute les variables différentes de x est assigne la valeur d à x, c'est-à-dire que $v[x \leftarrow d](x') = v(x')$ si $x \neq x'$ et $v[x \leftarrow d](x) = d$.

Étant donné un ensemble X de contraintes de IPC*, nous notons $v \models X$ lorsque $v \models \alpha$ pour tout $\alpha \in X$. Une contrainte α est satisfaisable ssi il existe une valuation v telle que $v \models \alpha$. Deux contraintes sont dites équivalentes ssi elles sont satisfaites exactement par les mêmes valuations.

Lemme 6 **(I)** *Le problème de satisfaisabilité de* IPC* *est* PSPACE-*complet.*

(II) *Pour toute contrainte de* IPC* *il existe une contrainte équivalente de* IPC* *sans quantificateur.*

Preuve : **(I)** Le problème de satisfaisabilité pour IPC^{++} est PSPACE-complet [Dem04] et celui de Z est NLOGSPACE-complet. Comme les contraintes de IPC* sont des combinaisons Booléennes de contraintes de IPC^{++} et de Z, on peut prouver que le problème de satisfaisabilité de IPC* est dans PSPACE en étendant la preuve de [Dem04, Théorème 3] qui définit une procédure similaire à celle du model-checking de la logique du premier ordre [CM77]. La PSPACE-dureté est une conséquence de la PSPACE-dureté de IPC^{++} qui est un fragment de $\mathrm{IPC}^{\star}$.

(II) IPC^{++} admet l'élimination des quantificateurs [Dem04] et donc $\mathrm{IPC}^{\star}$ aussi puisque Z n'a pas de quantificateur. □

Nous discutons à la fin de cette section de l'expressivité et de la concision de $\mathrm{IPC}^{\star}$. Nous montrons d'abord que, bien que $\mathrm{WIPC}^{\star}$ soit un fragment strict de $\mathrm{IPC}^{\star}$, ce langage est aussi expressif que $\mathrm{IPC}^{\star}$.

Lemme 7 *Pour toute contrainte α de* $\mathrm{IPC}^{\star}$*, il existe une contrainte équivalente α' appartenant à* $\mathrm{WIPC}^{\star}$.

Preuve : La construction de la formule α est possible grâce aux observations suivantes

- D'après le Lemme 6 **(II)**, $\mathrm{IPC}^{\star}$ admet l'élimination des quantificateurs.
- Toute contrainte $x \equiv_k [c_1, c_2]$ est équivalente à

$$\bigvee_{c_1 \le c \le c_2} x \equiv_k c.$$

- Toute contrainte $x \equiv_k y \ + \ [c_1, c_2]$ est équivalente à

$$\bigvee_{c'_1 \equiv_k c'_2 + c' \ \text{t.q.} \ c_1 \le c' \le c_2} x \equiv_k c'_1 \wedge y \equiv_k c'_2.$$

□

Notons cependant que la taille de la contrainte α' est exponentielle par rapport à $|\phi|$.

Enfin, l'ajout dans $\mathrm{IPC}^{\star}$ de contraintes de la forme $ax + by \equiv_k c$ où $a, b, c \in \mathbb{Z}$ rend le langage plus concis mais n'ajoute aucune expressivité. En effet, soit S l'ensemble des paires $\langle c_x, c_y \rangle \in \{0, \ldots, k-1\}^2$ telles que $c_x + c_y \equiv_k c$. Nous pouvons d'abord développer la contrainte $ax + by \equiv_k c$ en utilisant l'équivalence logique suivante :

$$ax + by \equiv_k c \Leftrightarrow \bigvee_{\langle c_x, c_y \rangle \in S} (ax \equiv_k c_x \wedge by \equiv_k c_y).$$

Les contraintes de la forme $ax \equiv_k c$ peuvent ensuite être traduite dans $\mathrm{IPC}^{\star}$ en résolvant l'équation diophantienne correspondante. En effet, une contrainte de la forme $ax \equiv_k c$ se réduit à

- $\perp$ si $\mathrm{pgcd}(a, k)$ ne divise pas c,
- $x \equiv_{k'} c'$ avec $k' \times \mathrm{pgcd}(a, k) = k$ pour un c' qui peut être calculé en temps polynomial par rapport à la taille de a, k et c_x (grâce à l'algorithme d'Euclide) sinon.

Le même genre de remarque s'applique dans le cas général des contraintes de la forme $\Sigma a_i x_i \equiv_k c$.

Fragments quantitatifs

Nous introduisons maintenant des fragments contenant des contraintes quantitatives. Contrairement aux fragments qualitatifs, nous commençons par présenter le noyau commun de ces fragments pour ensuite parler des extensions. Ce choix est fait afin de mettre en évidence les contraintes qui sont la cause principale de l'indécidabilité des extensions de logiques temporelles sur des fragments quantitatifs. Le plus petit fragment quantitatif dont nous parlons est le langage de contraintes de différences DL dont les contraintes sont définies par

$$\alpha ::= x \sim y + d \mid x \sim d \mid \alpha \wedge \alpha \mid \neg\alpha$$

où $x, y \in V$, $d \in \mathbb{Z}$ et $\sim \in \{<, >, \leq, \geq, =\}$.

Nous considérons aussi l'extension DL$^+$ de cette logique avec des contraintes de périodicités de la forme $x \equiv_k c$ et $x \equiv_k y + c$ où $k, c \in \mathbb{N}$.

Enfin, QFP est le fragment sans quantificateur de l'arithmétique de Presburger que l'on peut définir par la grammaire :

$$\alpha ::= \sum_{i \in I} a_i x_i \sim d \mid \sum_{i \in I} a_i x_i \equiv_k c \mid \alpha \wedge \alpha \mid \neg\alpha$$

où $a_i \in \mathbb{Z}$ et I est un ensemble fini d'indices. Nous avons donc l'inclusion suivante entre ces différents langages : DL $\subseteq$ DL$^+$ $\subseteq$ QFP. Notons que comme l'arithmétique de Presburger admet l'élimination des quantificateurs, le fragment QFP est aussi expressif que l'arithmétique de Presburger complet (mais moins concis).

Deuxième partie

LTL étendu avec des contraintes sur les entiers

Chapitre 4

Vérification de contraintes qualitatives

Sommaire

4.1 La logique CLTL(IPC*)

4.1.1 Propriétés du langage logique

Dans ce chapitre, nous considérons la logique CLTL(IPC*) qui étend LTL avec des contraintes induites par le langage de contraintes IPC* sur des termes. Par exemple, $x < \mathsf{X}^2 y$ et $\mathsf{X}x \equiv_2 y + 1$ sont des contraintes atomiques de CLTL(IPC*). Comme le langage IPC* est obtenu en ajoutant à IPC^{++} des contraintes de la forme $x < y$, il hérite donc de la clôture par combinaisons Booléennes et de la quantification du premier ordre. Notons qu'aucune contrainte de la forme $x < y$ n'est dans la portée d'un quantificateur. Dans le cas contraire, nous pourrions exprimer la relation de successeur d'un entier. En conséquence, le langage de contraintes ne contiendrait alors plus uniquement des contraintes qualitatives et la logique serait indécidable (voir Chapitre 5).

Le langage de contrainte IPC* contient des contraintes de comparaison qui ont de nombreuses applications dans les formalismes logiques. Ainsi, la logique CLTL(IPC*) permet de traiter des abstractions par rapport à des relations de congruence modulo un entier comme dans [CGL94, MOS05] ou des formalismes introduisant des contraintes de calendrier tels que [LM01] et des formalismes de raisonnement dans les bases de données [BBFS98]. Par définition, CLTL(IPC*) ne contient pas de variables propositionnelles mais il est facile de coder une variable propositionnelle p en introduisant des contraintes atomiques $x_p = 0$ où x_p est une variable fraîche.

Il est déjà connu que les problèmes de satisfaisabilité et model-checking pour la logique CLTL(IPC^{++}) sont dans PSPACE [Dem04] de même que pour la logique CLTL(Z) [DD07]. Bien que les preuves de ces deux résultats utilisent une réduction vers le problème du vide pour un automate de Büchi, celles-ci sont de différentes natures. Dans [Dem04] la borne de complexité est obtenue à l'aide d'une propriété de modèle fini alors que [DD07] utilise une approximation de la classe des modèles symboliques car certaines formules de CLTL(Z) ont un ensemble de modèles qui n'est pas ω-régulier. Les résultats de ce chapitre généralisent et unifient ces résultats. Nous montrons que l'ajout des contraintes de la forme $x < y$ à IPC^{++} conduit à de nombreuses complications techniques mais pas à l'indécidabilité. Nous améliorons la méthode introduite pour CLTL(Z) en ajoutant des contraintes de la forme $x \leq d$ où $d \in \mathbb{Z}$ et des contraintes de périodicité. Dans [DD07], il est prouvé que la logique CLTL(Z^{c}) est dans EXPSPACE en utilisant une traduction vers CLTL(Z) qui augmente exponentiellement la taille de la formule lorsqu'on considère un codage binaire des constantes. Nous raffinons donc aussi ce résultat. Le traitement optimal des constantes est la principale contribution technique de la méthode que nous développons ici. Cette méthode permet aussi de montrer la PSPACE-complétude pour CLTL(WIPC*) ou lorsque l'on étend CLTL(IPC*) avec des contraintes de la forme $ax + by \equiv_k c$, malgré les différences de concision de ces langages.

Ce chapitre est une version étendue de l'article [DG05].

4.1.2 IPC*-automates

Le problème du model-checking pour CLTL(IPC*) utilise comme modèle opérationnel la classe des IPC*-automates dont nous développons quelques caractéristiques ici. Notons que les problèmes du model-checking et de la satisfaisabilité pour CLTL(IPC*) sont équivalents. Chacun peut se réduire à l'autre en espace logarithmique en utilisant le même genre de technique que dans [SC85] (voir Section 1.1.1). Dans la suite de ce chapitre, nous parlons surtout du problème de la satisfaisabilité mais nous gardons à l'esprit que les résultats s'appliquent aussi au problème du model-checking.

Les IPC*-automates peuvent être vus comme une abstraction des machines à compteurs classiques où les incrémentations et décrémentations sont abstraits par des opérations modulo un entier. Cette abstraction permet par exemple de modéliser des programmes écrits dans certains langages de programmation usuels où les opérations sont définies modulo une puissance de deux [MOS05]. Ainsi, une opération telle que $x = y + 1$ est représentée par la formule de CLTL(IPC*) $x \equiv_{2^k} y + 1 \ \wedge \ y < x$ où k est en général égal à 32 ou 64. Une telle abstraction est une sous-approximation mais permet de vérifier des propriétés de sûreté par rapport au modèle original. Par exemple, l'évitabilité d'un état de contrôle dans la sous-approximation assure la même propriété pour le modèle initial. La Figure 4.1 représente un IPC*-automate qui est l'abstraction du contrôleur de cabine téléphonique de [CC00, Exemple 1]. La variable x représente le nombre de pièces insérées et y le temps de communication total. L'incrémentation de la variable z est abstraite par la formule $\mathsf{X}z \equiv_{2^{32}} z + 1 \wedge \mathsf{X}z > z$ et la formule $\phi_{=}$ est une abréviation pour $\mathsf{X}x = x \wedge \mathsf{X}y = y$. Les messages ne sont pas présents dans cette figure car ils ne sont pas importants dans le cadre des IPC*-automates.

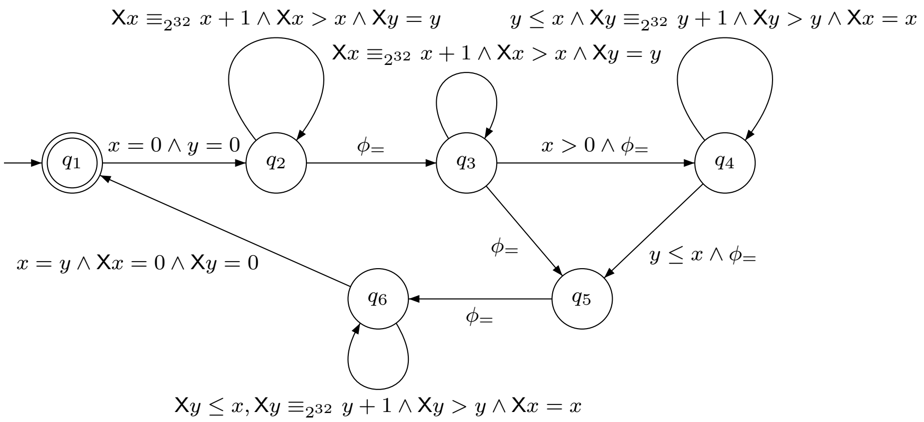


Figure 4.1: Un IPC*-automate

Plusieurs classes de modèles introduites dans la littératures peuvent être comparées à la classe IPC*-automates. Le problème de l'équivalence pour les "*Extended Single-String Automata*" [LM01] peut se réduire au model-checking des IPC*-automates. Le problème du model-checking de CLTL(IPC*) englobe aussi le problème du model-checking pour le fragment linéaire de l'extension de CTL* introduite dans [Čer94]. Le fragment linéaire de cette logique correspond à CLTL$^1(\mathbb{Z}^c)$ avec des formules atomiques supplémentaires référant aux états de contrôle du modèle. Les modèles considérés sont des automates relationnels manipulant des entiers, introduits sous le nom de "*Integral Relational Automata*" (IRA). Un IRA est une représentation symbolique d'un programme avec un nombre de variables fini qui prennent leurs valeurs dans $\mathbb{Z}$. Nous notons OP l'ensemble des opérations et tests de ce modèle qui est composé de :

- comparaisons de la forme $x < y$, $x < d$, $d < y$,
- affectations de la forme $x \leftarrow y$, $x \leftarrow d$,
- instructions d'entrée $?x$,
- instructions de sortie $!x$ or $!d$,
- instruction factice NOP,

où x et y sont des variables et $d \in \mathbb{Z}$ est une constante. Plus formellement, un IRA peut être défini comme un automate $\mathcal{A} = \langle Q, \delta, op, g\rangle$ tel que

- Q est un ensemble fini d'états de contrôle,
- $\delta \subseteq Q \times Q$,
- $op : Q \rightarrow \mathrm{OP}$,
- $g : \delta \rightarrow \{+, -\}$.

Nous notons $Var(\mathcal{A})$ et $Cons(\mathcal{A})$ les ensembles des variables et constantes apparaissant dans les étiquettes définies par la fonction op. Une configuration de $\mathcal{A}$ est une paire $\langle n, v\rangle$ où $n \in Q$ et v est une valuation de la forme $Var(\mathcal{A}) \uplus Cons(\mathcal{A}) \rightarrow \mathbb{Z}$ telle que $v(d) = d$ pour tout $d \in Cons(\mathcal{A})$. Le graphe $\langle S, \rightarrow\rangle$ des configurations de $\mathcal{A}$ est tel que l'ensemble des sommets S est égal à l'ensemble des configurations de $\mathcal{A}$ et on a un arc $\langle n, v\rangle \rightarrow \langle n', v'\rangle$ ssi il existe une transition $t = \langle n, n'\rangle \in \delta$ telle que v et v' vérifient les conditions suivantes imposées par $op(n)$:

- si $op(n) = ?x$ alors pour tout $y \in Var(\mathcal{A}) \setminus \{x\}$ on a $v'(y) = v(y)$,
- si $op(n) = !x$ ou NOP alors $v' = v$,
- si $op(n) = x \leftarrow a$ alors $v' = v[x \leftarrow v(a)]$,
- si $op(n) = a < b$ alors $v = v'$ et
  soit $g(t) = +$ et $v(a) < v(b)$,
  ou $g(t) = -$ et $v(a) \geq v(b)$,

où $a, b \in Var(\mathcal{A}) \uplus Cons(\mathcal{A})$, et $v[x \leftarrow a]$ est la valuation telle que $v[x \leftarrow z](y) = z$ si $x = y$ et $v[x \leftarrow z](y) = v(y)$ dans les autres cas. Notons que l'égalité entre deux éléments peut être testée en effectuant deux tests consécutifs $a < b$ et $b < a$.

La classe des IRA est incluse dans celle des IPC*-automates. Cette inclusion est bien entendu stricte puisque les IPC*-automates permettent par exemple d'exprimer des contraintes de périodicité. Ci-dessous, nous montrons qu'à partir d'un IRA on peut construire un IPC*-automate équivalent dans le sens où il existe un isomorphisme entre les graphes des configurations des deux automates.

Lemme 8 *Pour tout IRA $\mathcal{A}$ on peut construire un* IPC**-automate $\mathcal{A}'$ équivalent à $\mathcal{A}$.*

Preuve : À partir d'un IRA $\mathcal{A} = \langle Q, \delta, op, g \rangle$, nous construisons l'IPC*-automate $\mathcal{A}' = \langle Q', Q'_0, \delta', F' \rangle$ défini de la façon suivante :

1. $Q' = Q'_0 = F' = Q$. Sans perte de généralité, nous identifions Q avec un ensemble fini de constantes de $\mathbb{Z} \setminus Cons(\mathcal{A})$.
2. Pour chaque transition $t = \langle n, n' \rangle$ dans $\mathcal{A}$, on a un transition $n \xrightarrow{\phi_t} n'$ dans δ', où ϕ_t est une conjonction de contraintes définie par :
   - $\mathsf{X}ic = n'$ est une sous-contrainte de ϕ_t tel que ic est une variable introduite pour mémoriser la valeur du compteur d'instruction,
   - si $op(n) = ?x$ alors $\bigwedge_{y \in \mathrm{VAR}(\mathcal{A}) \setminus \{x\}} y = \mathsf{X}y$ est une contrainte de ϕ_t,
   - si $op(n)$ est une instruction de sortie ou NOP, alors $\bigwedge_{y \in \mathrm{VAR}(\mathcal{A})} y = \mathsf{X}y$ est une contrainte de ϕ_t,
   - si $op(n) = x \leftarrow a$ alors $\bigwedge_{y \in \mathrm{VAR}(\mathcal{A}) \setminus \{x\}} y = \mathsf{X}y \wedge \mathsf{X}x = a$ est une contrainte de ϕ_t,
   - si $op(n) = a < b$ alors
     si $g(t) = +$ alors $a < b \wedge \bigwedge_{y \in \mathrm{VAR}(\mathcal{A})} y = \mathsf{X}y$ est une contrainte de ϕ_t,
     si $g(t) = -$ alors $a \geq b \wedge \bigwedge_{y \in \mathrm{VAR}(\mathcal{A})} y = \mathsf{X}y$ est une contrainte de ϕ_t.

Toutes ces conditions, exceptée la première, expriment simplement les contraintes imposées par les transitions d'un IRA. Les graphes de configurations de $\mathcal{A}$ et $\mathcal{A}'$ sont reliés par l'isomorphisme suivant. Il existe une transition $\langle n, v \rangle \rightarrow \langle n', v' \rangle$ dans le graphe des configurations de $\mathcal{A}$ ssi $\langle n, v_{ic} \rangle \rightarrow \langle n', v'_{ic} \rangle$ appartient au graphe de configurations de $\mathcal{A}'$ où $v_{ic} : Var(\mathcal{A}) \cup Cons(\mathcal{A}) \cup \{ic\} \rightarrow \mathbb{Z}$ est une extension conservative de $v : Var(\mathcal{A}) \cup Cons(\mathcal{A}) \rightarrow \mathbb{Z}$ et $v_{ic}(ic) = n$. La valuation v'_{ic} est définie de la même façon par rapport à v'. $\square$

Dans cette preuve, la variable ic est introduite afin de réduire le model-checking du fragment linéaire de la logique définie dans [Čer94] au problème du model-checking de CLTL(IPC*). En effet, les formules atomiques de [Čer94] peuvent référer à un état de contrôle. Les résultats qui suivent nous permettent donc d'établir la borne PSPACE pour le model-checking du fragment linéaire de [Čer94] sur les IRA. Notons enfin que les IPC*-automates sont plus concis que les IRA puisqu'ils autorisent des conjonctions de plusieurs contraintes sur les transitions. Ainsi, on peut par exemple effectuer un test et une mise à jour sur la même transition dans les IPC*-automates. Cette opération ne peut se faire qu'en deux étapes dans les IRA dont la sémantique est plus proche de celle des programmes.

4.2 Représentation symbolique des modèles

Dans cette section, nous définissons une représentation symbolique des modèles d'une formule de CLTL(IPC*) qui est la base de l'approche à base d'automate présentée ensuite.

4.2.1 Valuations symboliques

Étant donné un ensemble fini X de contraintes atomiques de CLTL(IPC*), comme par exemple l'ensemble des contraintes atomiques d'une formule, nous utilisons les notations suivantes :

- l est le plus grand entier tel qu'un terme de la forme $\mathsf{X}^l x$ apparaît dans X. Si l'on considère l'ensemble des contraintes atomiques d'une formule ϕ, il s'agit de $|\phi|_{\mathsf{X}}$.
- V est l'ensemble fini des variables apparaissant dans X.
- Term est l'ensemble des termes $\{\mathsf{X}^i x \mid x \in V \text{ et } i \in \{0, \ldots, l\}\}$. Nous identifions parfois cet ensemble avec $V \times \{0, \ldots, l\}$.
- K est le plus petit commun multiple (ppcm) des entiers $k_1, \ldots, k_n$ tels que des contraintes de périodicité utilisant les relations $\equiv_{k_1}, \ldots, \equiv_{k_n}$ appartiennent à X.
- C est l'ensemble fini des constantes d apparaissant dans les contraintes de X de la forme $x \sim d$.
- m le plus petit élément de C et M le plus grand.
- C' est l'ensemble de constantes $\{m, m+1, \ldots, M\}$.

Nous supposons sans perte de généralité que ces éléments sont définis pour tout ensemble fini de contraintes atomiques de CLTL(IPC*). Notons que la taille de K est de l'ordre de $\mathcal{O}(|k_1| + \cdots + |k_n|)$ et que la cardinalité de C est polynomiale en $|X|$. La taille de C' est de l'ordre de $\mathcal{O}(2^{|m|+|M|})$ et chacun de ces éléments peut être codé sur $\mathcal{O}(|m| + |M|)$ bits.

Un ensemble maximalement cohérent X de contraintes atomiques par rapport aux ensembles Term et C est un ensemble de contraintes utilisant uniquement les termes de Term et les constantes de C tel qu'il existe une valuation $v : \text{Term} \rightarrow \mathbb{Z}$ satisfaisant X et toute extension X' incluant strictement X n'est pas satisfaisable. Nous définissons une abstraction des valuations Term $\rightarrow \mathbb{Z}$ sous la forme de trois ensembles disjoints de contraintes vérifiant les conditions énoncées ci-dessous. En quelques mots, deux ensembles de contraintes fournissent des informations complémentaires permettant d'évaluer les contraintes de comparaisons induites par le langage de contraintes Z^c et un troisième ensemble les informations permettant d'évaluer les contraintes de périodicité.

Étant donné un ensemble fini X de contraintes atomiques de CLTL(IPC*), une valuation symbolique *sv* est un triplet $\langle Y_1, Y_2, Y_3 \rangle$ tel que

- Y_1 est un ensemble maximalement cohérent de contraintes de CLTL(Z^c) par rapport à Term et C.
- Y_2 est une ensemble de contraintes de la forme $\mathsf{X}^i x = d$ où $\mathsf{X}^i x \in$ Term et $d \in C' \setminus C$. Pour tout $\mathsf{X}^i x \in$ Term on a $\mathsf{X}^i x = d \in Y_2$ pour un unique $d \in C' \setminus C$ ssi pour tout $d' \in C$, $\mathsf{X}^i x = d' \notin Y_1$ et $\{m < x, x < M\} \subseteq Y_1$. Ceci implique que chaque terme $\mathsf{X}^i x \in$ Term apparaît au plus une fois dans Y_2.
- Y_3 est un ensemble de contraintes de la forme $\mathsf{X}^i x \equiv_K c$ où $\mathsf{X}^i x \in$ Term et $c \in \{0, \ldots, K-1\}$ tel que tout terme $\mathsf{X}^i x \in$ Term apparaît exactement une fois dans Y_3.

Cette abstraction est comparable à l'abstraction des régions pour les automates temporisés [AD94]. Nous notons $\mathrm{SV}(X)$ l'ensemble des valuations symboliques construite par rapport aux *ressources syntaxiques* de X, c'est-à-dire l, K, V et C. Une conséquence de cette définition est qu'aucune contrainte appartenant à une valuation symbolique $sv = \langle Y_1, Y_2, Y_3 \rangle$ n'apparaît dans plus d'un des ensembles qui composent sv. Pour cette raison, nous notons par la suite $\alpha \in sv$ à la place de $\alpha \in Y_1 \uplus Y_2 \uplus Y_3$. Une valuation symbolique est satisfaisable ssi il existe une valuation v : Term $\rightarrow \mathbb{Z}$ telle que $v \models Y_1 \uplus Y_2 \uplus Y_3$.

Lemme 9 *Soit X un ensemble fini de contraintes de* CLTL(IPC*) *et $sv = \langle Y_1, Y_2, Y_3 \rangle$ un triplet tel que Y_1 est un ensemble de contraintes de* CLTL(Z^c) *construit avec les éléments de* Term *et C, Y_2 est un ensemble de contraintes de* CLTL(Z^c) *construit avec les éléments de* Term *et $C' \setminus C$, Y_3 est un ensemble de contraintes de la forme $\mathsf{X}^i x \equiv_K c$. On peut vérifier que sv est une valuation symbolique satisfaisable en temps polynomial par rapport à la somme des tailles respectives de X et sv.*

Preuve : Les conditions de non redondance des informations imposées par la définition des valuations symboliques peuvent facilement être vérifiées. Nous supposons donc pour la suite que chaque terme apparaît au plus une fois dans Y_2 et exactement une fois dans Y_3. On peut aussi facilement vérifier qu'aucune contrainte de l'ensemble Y_2 n'est pas en contradiction avec les contraintes de Y_1.

La cohérence maximale de Y_1 se vérifie en temps polynomial en utilisant une méthode similaire à [Čer94, Lemme 5.5]. L'ensemble de contraintes Y_1 est maximalement cohérent par rapport à Term et C ssi le graphe associé $G_{Y_1} = \langle \mathrm{Term} \uplus C, \xrightarrow{=}, \xrightarrow{<} \rangle$ tel que $n \xrightarrow{\sim} n'$ est un arc de G_{Y_1} ssi $n \sim n'$ est une contrainte de Y_1 satisfait les conditions suivantes.

(MC1) Pour tout n, n', il existe $\sim \in \{<, =\}$ tel que $n \xrightarrow{\sim} n'$ ou $n' \xrightarrow{\sim} n$.

(MC2) $\xrightarrow{=}$ est une relation de congruence compatible avec $\xrightarrow{<}$.

(MC3) Il n'existe pas de chemin $n_0 \overset{\sim_0}{\rightarrow} n_1 \overset{\sim_1}{\rightarrow} \ldots \xrightarrow{\sim_{|\pi|-1}} n_{|\pi|}$ tel que $n_0 = n_{|\pi|}$ et $<$ appartient à $\{\sim_0, \sim_1, \ldots, \sim_{|\pi|-1}\}$.

(MC4) Pour tout $d_1, d_2 \in C$, $d_1 \sim d_2$ implique $d_1 \overset{\sim}{\rightarrow} d_2$.

(MC5) Pour tout d_1, d_2 tel que $d_1 \leq d_2$, il n'existe pas de chemin $n_0 \overset{\sim_0}{\rightarrow} n_1 \overset{\sim_1}{\rightarrow} \ldots \xrightarrow{\sim_{|\pi|-1}} n_{|\pi|}$ où $n_0 = d_1$ et $n_{|\pi|} = d_2$ tel que le nombre d'occurrences de "$<$" dans $\sim_0, \cdots, \sim_{|\pi|-1}$ est strictement supérieur à $d_2 - d_1$.

Nous devons aussi vérifier que l'ensemble des contraintes de périodicité est compatible avec les deux autres ensembles.

(MC6) Pour toute contrainte $\mathsf{X}^i x \overset{=}{\rightarrow} \mathsf{X}^j y$ de Y_1 telle que $x, y \in V$, les contraintes $\mathsf{X}^i x \equiv_K c$ et $\mathsf{X}^j y \equiv_K c'$ de Y_3 sont telles que $c = c'$. De même pour toute contrainte $\mathsf{X}^i x = d$ dans $Y_1 \cup Y_2$, la contrainte $\mathsf{X}^i x \equiv_K d'$ de Y_3 est telle que $d \equiv_K d'$. □

Comme l'illustre le résultat ci-dessous, les valuations symboliques de $\mathrm{SV}(X)$ contiennent toutes les informations nécessaires pour évaluer les contraintes construites par rapport aux ressources syntaxiques de X. Une contrainte de CLTL(IPC*) est construite par rapport aux ressources syntaxiques de X ssi

- les termes utilisés sont dans Term,
- les constantes utilisées sont dans C,
- les relations de congruences $\equiv_k$ utilisées sont telles que k divise K.

Lemme 10 *Soit X un ensemble fini de contraintes atomiques de* CLTL(IPC*).

(I) *Pour toute valuation v :* Term $\rightarrow \mathbb{Z}$ *il existe une unique valuation symbolique* $\langle Y_1, Y_2, Y_3\rangle \in \mathrm{SV}(X)$ *notée $sv(v)$ telle que $v \models Y_1 \cup Y_2 \cup Y_3$.*

(II) *Pour tout v, v' tel que $sv(v) = sv(v') \in \mathrm{SV}(X)$ et pour toute contrainte atomique α de* CLTL(IPC*) *construite par rapport aux ressources syntaxiques de X, on a $v \models \alpha$ ssi $v' \models \alpha$.*

Preuve : **(I)** Étant donnée une valuation symbolique sv, nous définissons Val_{sv} comme étant l'ensemble des éléments $\langle z_1, \ldots, z_n\rangle$ de $\mathbb{Z}^{|V|}$ (vus comme des fonctions de la forme $V \rightarrow \mathbb{Z}$) tel que $\langle z_1, \ldots, z_n\rangle \models sv$. On peut facilement montrer que $\{\mathrm{Val}_{sv} : sv \in \mathrm{SV}(X)$ et $\mathrm{Val}_{sv} \neq \emptyset\}$ est une partition de $\mathbb{Z}^{|V|}$.

(II) Considérons deux valuations v et v' telles que $sv(v) = sv(v')$. Nous procédons par induction sur la structure de α. Il est suffisant de montrer l'implication $v \models \alpha \Rightarrow v' \models \alpha$, l'autre sens est identique.

- Supposons que α soit de la forme $\mathsf{X}^i x \sim \mathsf{X}^j y$. Si $v \models \alpha$ alors on a $v(\mathsf{X}^i x) \sim v(\mathsf{X}^j y)$. Par définition, une valuation symbolique est maximalement cohérente donc il existe une contrainte de comparaison entre $\mathsf{X}^i x$ et $\mathsf{X}^j y$ dans $sv(v)$. Puisque l'on a $v(\mathsf{X}^i x) \sim v(\mathsf{X}^j y)$ et $v \models sv(v)$, la contrainte α appartient à $sv(v)$. Comme $sv(v) = sv(v')$, on a aussi $v'(\mathsf{X}^i x) \sim v'(\mathsf{X}^j y)$ et donc $v' \models \alpha$.

- Si α est de la forme $\mathsf{X}^i x \sim d$, la preuve est similaire. Puisque la contrainte α est construite par rapport aux ressources syntaxiques de X, la constante d appartient à C on a encore $\alpha \in sv(v)$.
- Si α est de la forme $\mathsf{X}^i x \equiv_k [c_1, c_2]$ et $v \models \alpha$ alors on a $v(\mathsf{X}^i x) \equiv_k c$ pour un $c \in [c_1, c_2]$. Par définition, il existe une contrainte de la forme $\mathsf{X}^i x \equiv_K c'$ dans $sv(v)$ et donc on a aussi $v(\mathsf{X}^i x) \equiv_K c'$. Comme α est construite par rapport aux ressources syntaxiques de X, l'entier k divise K. Il est donc facile de prouver que $c \equiv_k c'$ en utilisant le fait qu'il existe $i, j \in \mathbb{Z}$ tels que $c + ik = c' + jK$ et que k divise K. Comme $sv(v) = sv(v')$, on a aussi $v'(\mathsf{X}^i x) \equiv_K c'$ et puisque $c \equiv_k c'$ et k divise K, on peut facilement en déduire que $v' \models \alpha$.
- Si α est de la forme $\mathsf{X}^i x \equiv_k \mathsf{X}^j y + [c_1, c_2]$ et $v \models \alpha$ alors on a $v(\mathsf{X}^i x) \equiv_k v(\mathsf{X}^j y) + c$ pour un $c \in [c_1, c_2]$. La preuve est semblable au cas précédent. Par définition, il existe une contrainte de la forme $\mathsf{X}^i x \equiv_K c_x$ dans $sv(v)$ et une autre de la forme $\mathsf{X}^j y \equiv_K c_y$. Nous pouvons en déduire que $\mathsf{X}^i x \equiv_K \mathsf{X}^j y + (c_x - c_y)$ et comme k divise K on a $(c_x - c_y) \equiv_k c$. Il est alors facile de conclure de la même manière que dans le cas précédent.

Par induction nous supposons que pour toutes valuations v et v' telles que $sv(v) = sv(v')$, si $v \models \alpha$ alors $v' \models \alpha$.

- Si α est de la forme et $\exists x\ \alpha'$ alors $v \models \alpha$ implique qu'il existe $d \in \mathbb{Z}$ tel que $v[x \leftarrow d] \models \alpha'$ où $v[x \leftarrow d]$ est la valuation qui assigne la valeur d à x et coïncide avec v sur tous les autres éléments. Par définition des valuations symboliques, nous obtenons que $sv(v[x \leftarrow d]) = sv(v'[x \leftarrow d])$. En effet, comme la valeur de x est la seule à changer par rapport à v, toute contrainte de $sv(v)$ qui n'implique pas x appartient aussi à $sv(v[x \leftarrow d])$ et $sv(v'[x \leftarrow d])$. De plus, comme la valeur x prend la valeur d dans les deux valuations, on peut facilement voir que toute contrainte impliquant x dans $sv(v[x \leftarrow d])$ appartient aussi à $sv'(v[x \leftarrow d])$.
  Ainsi, en utilisant l'hypothèse d'induction $v[x \leftarrow d] \models \alpha'$ et $sv(v[x \leftarrow d]) = sv(v'[x \leftarrow d])$ impliquent que $v'[x \leftarrow d] \models \alpha'$. Donc $v' \models \exists x\ \alpha'$.
- Enfin les cas des combinaisons booléennes peuvent facilement être traités par induction. $\square$

Étant données une valuation symbolique $sv \in \mathrm{SV}(X)$ et une contrainte α de CLTL(IPC*) construite par rapport aux ressources syntaxiques de X, nous notons $sv \models_{\mathrm{symb}} \alpha$ ssi pour toute valuation v telle que $sv(v) = sv$ on a $v \models \alpha$.

Lemme 11 *Étant données une valuation symbolique $sv \in \mathrm{SV}(X)$ et une contrainte α de* CLTL(IPC*) *construite par rapport aux ressources syntaxiques de X, tester si $sv \models_{\mathrm{symb}} \alpha$ est un problème* PSPACE*-complet.*

Preuve. Nous montrons la PSPACE-dureté par réduction du problème QBF. En fait, ce problème revient à tester qu'une contrainte de IPC* est satisfaite par une valuation symbolique. Ceci est suffisant car CLTL(IPC*) étend IPC*. Soit $\phi = Q_1 p_1 Q_2 p_2 \ldots Q_n p_n \phi'$ une formule sous forme prénexe telle que ϕ' est une formule sous forme normale conjonctive ayant pour variables propositionnelles $\{p_1, \ldots, p_n\}$ et $\{Q_1, \ldots, Q_n\} \subseteq \{\forall, \exists\}$. Nous traduisons ϕ en une contrainte de IPC* à l'aide de la fonction f telle que :

- $f(\exists\ p_i\ \phi_i) = \exists\ x_i\ (x_i = 0 \vee x_i = 1) \wedge f(\phi_i)$,
- $t(\forall\ p_i\ \phi_i) = \forall\ x_i\ (x_i = 0 \vee x_i = 1) \Rightarrow f(\phi_i)$,
- f est homomorphique par rapport aux opérateurs Booléens,
- $f(p_i) = (x_i = 1)$.

Il est facile de voir que ϕ est valide ssi pour toute valuation symbolique sv on a $sv \models_{\text{symb}} f(\phi)$. Ce dernier problème est équivalent à tester si une valuation symbolique particulière satisfait $f(\phi)$ puisque cette formule n'a pas de variable libre.

Pour la borne supérieure, nous utilisons un algorithme analogue à celui du model-checking de la logique du premier ordre [CM77]. Soit $\mathrm{MC}(sv, \alpha, \alpha')$ la fonction prenant en argument

- une valuation symbolique $sv \in \mathrm{SV}(X)$,
- une contrainte α de IPC* construite par rapport aux ressources syntaxiques de X,
- une sous-contrainte α' de α,

et renvoyant "vrai" ssi $sv \models_{\text{symb}} \alpha'$. Notons qu'une variable de sv qui n'est pas libre dans α' n'influence pas la relation de satisfaction $sv \models_{\text{symb}} \alpha'$. La fonction MC est définie par analyse de cas selon la structure de la contrainte α'. Soit $sv = \langle Y_1, Y_2, Y_3 \rangle$ une valuation symbolique de $\mathrm{SV}(X)$, α une contrainte atomique de CLTL(IPC*) construite par rapport aux ressources syntaxiques de X et α' une sous-contrainte de α.

- $\mathrm{MC}(sv, \alpha, a = b)$ pour $a, b \in \mathrm{Term} \uplus C$ renvoie "vrai" ssi
  - a ou b est un terme et $a = b$ appartient à sv,
  - a et b sont des constantes égales.
- $\mathrm{MC}(sv, \alpha, a < b)$ pour $a, b \in \mathrm{Term} \uplus C$ renvoie "vrai" ssi
  - a et b sont des termes et $a < b$ appartient à sv,
  - a est un terme et b une constante et soit $a < b$ appartient à sv, soit il existe une constante $d < b$ telle que $a \sim d$ appartient à sv pour un $\sim \in \{<, =\}$,
  - a est une constante et b un terme et soit $a < b$ appartient à sv, soit il existe une constante $a < d$ telle que $d \sim b$ appartient à sv pour un $\sim \in \{<, =\}$,
  - a et b sont des constantes telles que $a < b$.
- $\mathrm{MC}(sv, \alpha, \mathsf{X}^i x \equiv_k [c_1, c_2])$ renvoie "vrai" ssi il existe une contrainte $\mathsf{X}^i x \equiv_K c$ dans sv telle que $c \equiv_k c'$ pour $c' \in [c_1, c_2]$. Cette vérification peut se faire en espace polynomial puisque la taille de K est polynomiale par rapport à $|\phi|$ et k divise K. Le cas $\mathsf{X}^i x \equiv_k \mathsf{X}^j y + [c_1, c_2]$ est similaire.
- $\mathrm{MC}(sv, \alpha, \alpha'_1 \wedge \alpha'_2)$ renvoie "vrai" ssi $\mathrm{MC}(sv, \alpha, \alpha'_1)$ et $\mathrm{MC}(sv, \alpha, \alpha'_2)$ renvoient "vrai".
- $\mathrm{MC}(sv, \alpha, \alpha'_1 \vee \alpha'_2)$ renvoie "vrai" ssi $\mathrm{MC}(sv, \alpha, \alpha'_1)$ ou $\mathrm{MC}(sv, \alpha, \alpha'_2)$ renvoie "vrai".
- $\mathrm{MC}(sv, \alpha, \exists x\ \alpha')$, renvoie "vrai" ssi il existe une valuation symbolique $sv' = \langle Y'_1, Y'_2, Y'_3 \rangle$ dans $\mathrm{SV}(X)$ telle que $\mathrm{MC}(sv', \alpha, \alpha')$ renvoie vrai et sv' vérifie : pour toute contrainte

$\alpha \in sv$ qui n'implique pas la variable x on a aussi $\alpha \in sv'$ (dans le même ensemble du triplet). Même si le nombre de valuations symboliques est exponentiel, il est possible de les énumérer en utilisant un espace polynomial pour tester l'existence d'une telle valuation symbolique. En effet, la taille de chaque valuation symbolique est polynomiale par rapport à $|X|$.

Il est facile de vérifier que ces opérations peuvent se faire en espace polynomial. Certaines informations sont données ci-dessus et les cas restants soit reviennent à tester l'appartenance d'une contrainte dans un ensemble, soit peuvent être prouvés par induction. $\square$

4.2.2 Modèles symboliques

Nous utilisons maintenant la représentation symbolique des valuations définie ci-dessus pour abstraire les modèles de CLTL(IPC*) par rapport à une formule donnée. Étant donnée une formule ϕ de CLTL(IPC*), nous notons SV(ϕ) l'ensemble des valuations symboliques construites par rapport à l'ensemble des contraintes atomiques de ϕ. Une paire de valuations symboliques $\langle\langle Y_1, Y_2, Y_3\rangle, \langle Y_1', Y_2', Y_3'\rangle\rangle$ est *cohérente sur un pas* ssi

1. pour tout $1 \leq j, j' \leq l$, on a $\mathsf{X}^j x_i \sim \mathsf{X}^{j'} x_{i'} \in Y_1$ ssi $\mathsf{X}^{j-1} x_i \sim \mathsf{X}^{j'-1} x_{i'} \in Y_1'$,
2. pour tout $1 \leq j \leq l$, on a $\mathsf{X}^j x_i \sim d \in Y_1 \cup Y_2$ ssi $\mathsf{X}^{j-1} x_i \sim d \in Y_1' \cup Y_2'$,
3. pour tout $1 \leq j \leq l$, on a $\mathsf{X}^j x_i \equiv_K c \in Y_3$ ssi $\mathsf{X}^{j-1} x_i \equiv_K c \in Y_3'$.

Une séquence infinie $\rho : \mathbb{N} \to \mathrm{SV}(\phi)$ de valuations symboliques par rapport à ϕ est *cohérente pas à pas* ssi pour tout $i \in \mathbb{N}$ la paire $\langle \rho(i), \rho(i+1)\rangle$ est cohérente sur un pas. Nous appelons une telle séquence un *modèle symbolique* de ϕ. Un modèle symbolique ρ est satisfait par un modèle σ de CLTL(IPC*) ssi pour tout $i \in \mathbb{N}$ et $\alpha \in \rho(i)$ on a $\sigma, i \models \alpha$. Nous notons alors $\sigma \models \rho$.

La cohérence pas à pas des modèles symboliques permet de les représenter sous la forme d'un graphe. Nous étendons la construction du graphe définie dans la preuve du Lemme 9. Étant donné un modèle symbolique ρ, nous définissons le graphe

$$G_\rho = \langle (V \cup C') \times \mathbb{N}, \xrightarrow{=}, \xrightarrow{<}, mod\rangle$$

où $mod : (V \cup C') \times \mathbb{N} \to \{0, \ldots, K-1\}$ est défini de la façon suivante :

$$\begin{array}{lll}
\langle x, i\rangle \xrightarrow{\sim} \langle y, j\rangle & \text{ssi on a } i \leq j \text{ et } x \sim \mathsf{X}^{j-i} y \in \rho(i) \\
& \text{ou } i > j \text{ et } \mathsf{X}^{i-j} x \sim y \in \rho(j), \\
\langle x, i\rangle \xrightarrow{=} \langle d, j\rangle & \text{ssi } x = d \ \in \rho(i), \\
\langle d, i\rangle \xrightarrow{=} \langle x, j\rangle & \text{ssi } x = d \ \in \rho(j), \\
\langle x, i\rangle \xrightarrow{<} \langle d, j\rangle & \text{ssi il existe } d' \sim d \text{ tel que } x \sim' d' \ \in \rho \text{ et } {<} \in \{\sim, \sim'\}, \\
\langle d, i\rangle \xrightarrow{<} \langle x, j\rangle & \text{ssi il existe } d \sim d' \text{ tel que } d' \sim' x \ \in \rho \text{ et } {<} \in \{\sim, \sim'\},
\end{array}$$

$$\begin{aligned}
\langle d_1, i\rangle &\xrightarrow{\sim} \langle d_2, j\rangle \quad \text{ssi } d_1 \sim d_2,\\
mod(\langle x, i\rangle) &= c \quad \text{ssi } x \equiv_K c \in \rho(i),\\
mod(\langle d, i\rangle) &= c \quad \text{ssi } d \equiv_K c,
\end{aligned}$$

pour tout $x, y \in V$, $d_1, d_2 \in C'$ et $i, j \in \mathbb{N}$ tel que $|i-j| \leq l$. Dans cette construction, les variables et les constantes sont traitées de la même façon. Nous insistons sur le fait qu'il existe un arc entre deux sommets $\langle a, i\rangle$ et $\langle b, j\rangle$ ssi $|i-j| \leq l$. En fait, chaque représentation locale du graphe, c'est-à-dire toute partie du graphe correspondant à une fenêtre de l états consécutifs, correspond à la représentation d'une valuation symbolique. En conséquence, cette construction assure que les conditions **(MC1)** à **(MC6)** dans la preuve du Lemme 9 sont vérifiées "localement". La dernière condition peut maintenant s'exprimer en fonction de la représentation graphique, ce qui n'était pas le cas dans la Section 4.2.1 puisque G_{Y_1} ne tenait compte que des contraintes de Y_1.

(MC6) Pour tout n, n' de G_ρ tel que $n \xrightarrow{=} n'$, on a $mod(n) = mod(n')$.

Dans cette représentation graphique, nous disons qu'un sommet n *représente la constante* d ssi il est de la forme $\langle d, i\rangle$ pour un $i \in \mathbb{N}$. Le *niveau* d'un sommet $n = \langle a, i\rangle$, noté lev(n), est l'entier i. Notons qu'il y a une certaine forme de redondance dans G_ρ concernant les comparaisons avec les sommets représentant les constantes. Néanmoins, ceci sera utile par la suite pour établir des relations plus étroites entre ρ et G_ρ.

Nous illustrons cette construction avec un exemple. Soit $V = \{x\}$, $C = \{2, 4\}$, $K = 2$ et $l = 1$. Nous considérons la séquence $\rho = sv^0 \cdot (sv^1 \cdot sv^2)^\omega$ telle que

- $sv^0 = \langle Y_1^0, Y_2^0, Y_3^0\rangle$ où
  - $Y_1^0 = \{x = x, \mathsf{X}x = \mathsf{X}x, x < \mathsf{X}x, 2 < x, x < 4, 2 < \mathsf{X}x, \mathsf{X}x = 4\}$,
  - $Y_2^0 = \{x = 3\}$,
  - $Y_3^0 = \{x \equiv_2 1, \mathsf{X}x \equiv_2 0\}$,
- $sv^1 = \langle Y_1^1, Y_2^1, Y_3^1\rangle$ où
  - $Y_1^1 = \{x = x, \mathsf{X}x = \mathsf{X}x, x < \mathsf{X}x, 2 < x, x = 4, 2 < \mathsf{X}x, 4 < \mathsf{X}x\}$,
  - $Y_2^1 = \emptyset$,
  - $Y_3^1 = \{x \equiv_2 0, \mathsf{X}x \equiv_2 1\}$,
- $sv^2 = \langle Y_1^2, Y_2^2, Y_3^2\rangle$ où
  - $Y_1^2 = \{x = x, \mathsf{X}x = \mathsf{X}x, x = \mathsf{X}x, 2 < x, 4 < x, 2 < \mathsf{X}x, 4 < \mathsf{X}x\}$,
  - $Y_2^2 = \emptyset$,
  - $Y_3^2 = \{x \equiv_2 1, \mathsf{X}x \equiv_2 1\}$.

Le graphe G_ρ est représenté dans la Figure 4.2. Afin de simplifier la représentation, nous omettons certains arcs qui peuvent être obtenus par transitivité ou en utilisant les propriétés de congruence de $\xrightarrow{=}$. La fonction *mod* est directement codée dans les étiquettes des sommets.

Un chemin dans G_ρ est une séquence (finie ou infinie) de la forme $n_0 \xrightarrow{\sim_0} n_1 \xrightarrow{\sim_1} n_2 \xrightarrow{\sim_2} \ldots$ La longueur $|\pi|$ d'un chemin π est égale à son nombre d'arcs. Soit $\pi = n_0 \xrightarrow{\sim_0} n_1 \xrightarrow{\sim_1} \ldots \xrightarrow{\sim_{|\pi|-1}} n_{|\pi|}$ un chemin fini. Le chemin π est un cycle ssi $n_0 = n_{|\pi|}$. La longueur stricte de π, notée

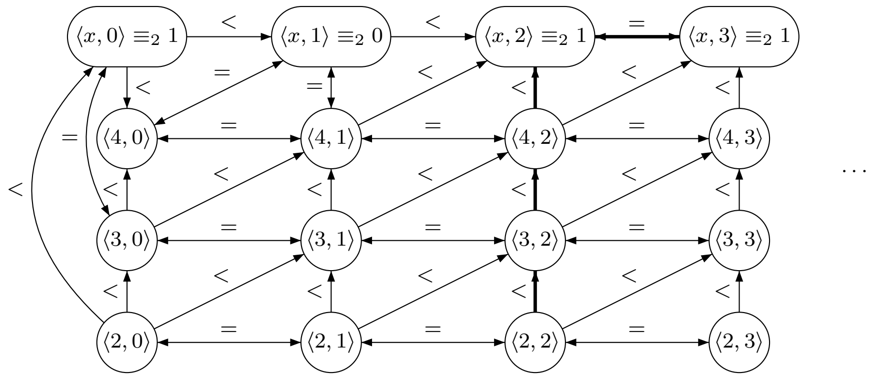


Figure 4.2: Exemple de représentation graphique G_ρ

$slen(\pi)$, est égale au nombre d'indices $i \in \{0, \ldots, |\pi| - 1\}$ tels que $\sim_i$ est égal à $<$ et π est un chemin strict ssi $slen(\pi) > 0$. La longueur stricte entre deux sommets $slen(n_1, n_2)$ est la borne supérieure de l'ensemble des longueurs strictes associées aux chemins finis allant de n_1 à n_2. Par convention, nous posons $slen(n_1, n_2) = -\infty$ s'il n'existe aucun chemin de n_1 vers n_2. Par exemple, dans la Figure 4.2 la longueur stricte $slen(\langle 2,2\rangle, \langle x,3\rangle)$ est égale à 3 (voir le chemin en gras).

La cohérence pas à pas de ρ implique des contraintes globales sur l'ensemble de sa représentation graphique. Nous entendons par contrainte globale une contrainte sur l'ensemble du graphe et pas seulement au niveau local correspondant à la représentation d'une valuation symbolique ou de deux valuations symboliques consécutives.

Lemme 12 *Pour tout modèle symbolique ρ, le graphe G_ρ vérifie les propriétés suivantes :*

(I) *G_ρ n'a aucun cycle strict.*

(II) *Pour tout chemin fini π dont l'origine $\langle d, i\rangle$ représente une constante et la destination est un sommet n de niveau j, si π est strict alors $\langle d, j\rangle \xrightarrow{<} n$ sinon $\langle d, j\rangle \xrightarrow{=} n$.*

(III) *Pour tout un chemin fini π dont l'origine est un sommet n de niveau j et la destination un sommet $\langle d, i\rangle$ représentant une constante, si π est strict alors $n \xrightarrow{<} \langle d, j\rangle$ sinon $n \xrightarrow{=} \langle d, j\rangle$.*

(IV) *Pour tout chemin π ayant pour origine n et destination n' qui ne contient que des arcs de la forme $\xrightarrow{=}$, on a $mod(n) = mod(n')$.*

Preuve. **(I)** Nous montrons que tout chemin $\pi = n_0 \xrightarrow{\sim_0} n_1 \xrightarrow{\sim_1} \ldots \xrightarrow{\sim_{|\pi|-1}} n_{|\pi|}$ tel que $n_0 = n_{|\pi|}$ vérifie forcément $slen(\pi) = 0$. Nous procédons par induction sur la longueur du chemin. Le cas où $|\pi| = 1$ est direct car la représentation graphique locale de la valuation symbolique $\rho(\mathrm{lev}(n_0))$ est maximalement cohérente et donc on a $n_0 \xrightarrow{=} n_0$ (d'après la condition **(MC3)**). Nous supposons maintenant que π est un cycle tel que $|\pi| > 1$ et que la propriété est vraie pour tout chemin de longueur inférieure à $|\pi|$. Soit n_{lmax} le sommet ayant le plus grand niveau dans π. Nous supposons que $0 < \mathrm{lmax} < |\pi|$ puisque l'on peut choisir le début du cycle sans perte de généralité. Comme $\mathrm{lev}(n_{\mathrm{lmax}})$ est maximal, on doit avoir $|\mathrm{lev}(n_{\mathrm{lmax}-1}) - \mathrm{lev}(n_{\mathrm{lmax}+1})| \leq l$ et donc la représentation de $\rho(\min\{\mathrm{lev}(n_{\mathrm{lmax}-1}), \mathrm{lev}(n_{\mathrm{lmax}+1})\})$ contient les sommets $n_{\mathrm{lmax}-1}$, n_{lmax} et $n_{\mathrm{lmax}+1}$. Par construction, il existe donc des arcs entre ces trois sommets. Puisque la représentation de $\rho(\min\{\mathrm{lev}(n_{\mathrm{lmax}-1}), \mathrm{lev}(n_{\mathrm{lmax}+1})\})$ satisfait **(MC2)**, on a $n_{\mathrm{lmax}-1} \xrightarrow{\leq} n_{\mathrm{lmax}+1}$ ssi $n_{\mathrm{lmax}-1} \xrightarrow{\leq} n_{\mathrm{lmax}}$ ou $n_{\mathrm{lmax}} \xrightarrow{\leq} n_{\mathrm{lmax}+1}$. Ceci implique que $slen(\pi) = 0$ ssi $slen(\pi') = 0$ où π' est obtenu à partir de π en remplaçant le sous-chemin entre $n_{\mathrm{lmax}-1}$ et $n_{\mathrm{lmax}+1}$ par l'arc entre $n_{\mathrm{lmax}-1}$ et $n_{\mathrm{lmax}+1}$. Comme la longueur de π' est strictement inférieure à celle de π, nous pouvons conclure en utilisant l'hypothèse d'induction.

(II) Nous démontrons que pour tout chemin $\pi = n_0 \xrightarrow{\sim_0} n_1 \xrightarrow{\sim_1} \dots \xrightarrow{\sim_{|\pi|-1}} n_{|\pi|}$ tel que $n_0 = \langle d, i\rangle$ on a $\langle d, \mathrm{lev}(n_{|\pi|})\rangle \xrightarrow{\leq} n_{|\pi|}$ si π est strict et $\langle d, \mathrm{lev}(n_{|\pi|})\rangle \xrightarrow{=} n_{|\pi|}$ sinon. Nous procédons par induction sur la longueur de π. Le cas de base où $|\pi| = 1$ est direct car la représentation locale de $\rho(\min\{\mathrm{lev}(n_0), \mathrm{lev}(n_1)\})$ contient tous les sommets concernés. Nous considérons maintenant un chemin $\pi = n_0 \xrightarrow{\sim_0} n_1 \xrightarrow{\sim_1} n_2 \xrightarrow{\sim_2} \dots \xrightarrow{\sim_{|\pi|-1}} n_{|\pi|}$ avec $n_0 = \langle d, i\rangle$ et $|\pi| > 1$ et nous supposons que $\langle d, \mathrm{lev}(n_{|\pi|-1})\rangle \xrightarrow{\leq} n_{|\pi|-1}$ si le sous-chemin de π se terminant au sommet $n_{|\pi|-1}$ est strict et $\langle d, \mathrm{lev}(n_{|\pi|-1})\rangle \xrightarrow{=} n_{|\pi|-1}$ sinon. Comme les sommets $n_{|\pi|-1}$ et $n_{|\pi|}$ sont reliés par un arc, ils appartiennent tous deux à la représentation graphique de la valuation symbolique $\rho(\min\{\mathrm{lev}(n_{|\pi|-1}), \mathrm{lev}(n_{|\pi|})\})$. Cette représentation locale satisfait **(MC2)** et donc $\langle d, \mathrm{lev}(n_{|\pi|-1})\rangle \xrightarrow{\leq} n_{|\pi|}$ ssi $\langle d, \mathrm{lev}(n_{|\pi|-1})\rangle \xrightarrow{\leq} n_{|\pi|-1}$ ou $n_{|\pi|-1} \xrightarrow{\leq} n_{|\pi|}$. En conséquence $\langle d, \mathrm{lev}(n_{|\pi|-1})\rangle \xrightarrow{\leq} n_{|\pi|}$ si π est strict. Puisque $\langle d, \mathrm{lev}(n_{|\pi|-1})\rangle \xrightarrow{=} \langle d, \mathrm{lev}(n_{|\pi|})\rangle$, on a donc bien $\langle d, \mathrm{lev}(n_{|\pi|})\rangle \xrightarrow{\leq} n_{|\pi|}$ si π est strict (et $\langle d, \mathrm{lev}(n_{|\pi|})\rangle \xrightarrow{=} n_{|\pi|}$ sinon) en utilisant la propriété **(MC2)**.

(III) La preuve de ce cas est similaire au précédent.

(IV) Considérons un chemin de la forme $\pi = n_0 \xrightarrow{=} n_1 \xrightarrow{=} \dots \xrightarrow{=} n_{|\pi|}$. Nous procédons par induction sur la taille de ce chemin. Si $|\pi| = 1$ alors la représentation locale de la valuation symbolique $\rho(\min\{\mathrm{lev}(n_0), \mathrm{lev}(n_{|\pi|})\})$ contient les deux sommets et nous pouvons conclure en utilisant **(MC6)**. Sinon, supposons que $|\pi| > 1$ et $mod(n_0) = mod(n_{|\pi|-1})$ par hypothèse d'induction. Comme il existe un arc $n_{|\pi|-1} \xrightarrow{=} n_{|\pi|}$, la représentation graphique correspondant à la valuation symbolique $\rho(\min\{\mathrm{lev}(n_{|\pi|-1}), \mathrm{lev}(n_{|\pi|-1})\})$ contient les deux sommets. Puisque cette représentation locale satisfait **(MC6)** on a bien par transitivité $mod(n_{|\pi|}) = mod(n_{|\pi|-1}) = mod(n_0)$. □

Corollaire 4 *Soit ρ un modèle symbolique et G_ρ sa représentation graphique. Pour toute paire de sommets $\langle d_1, i\rangle$ et $\langle d_2, j\rangle$ de G_ρ représentant des constantes telles que $d_1 \leq d_2$, on a $slen(\langle d_1, i\rangle, \langle d_2, j\rangle) = d_2 - d_1$.*

Preuve. Soient $\langle d_1, i\rangle$ et $\langle d_2, j\rangle$ deux sommets de G_ρ représentant respectivement les constantes d_1 et d_2 telles que $d_1 \leq d_2$. Nous traitons le cas $i \leq j$ (le cas $j > i$ est symétrique). Il est évident que $slen(\langle d_1, i\rangle, \langle d_2, j\rangle) \geq d_2 - d_1$ puisque par construction le chemin suivant existe et a pour longueur stricte $d_2 - d_1$:

$$\langle d_1, i\rangle \xrightarrow{=} \langle d_1, i+1\rangle \xrightarrow{=} \langle d_1, i+2\rangle \xrightarrow{=} \cdots \xrightarrow{=} \langle d_1, j\rangle \xrightarrow{\leq} \langle d_1+1, j\rangle \xrightarrow{\leq} \langle d_1+2, j\rangle \xrightarrow{\leq} \cdots \xrightarrow{\leq} \langle d_2, j\rangle.$$

Nous montrons maintenant qu'il n'existe pas de chemin π entre $\langle d_1, i\rangle$ et $\langle d_2, j\rangle$ tel que $slen(\pi) > d_2 - d_1$. Nous procédons par l'absurde et supposons qu'un tel chemin π existe. Considérons la restriction de la clôture transitive de $\stackrel{=}{\rightarrow}$ aux sommets de π. D'après la définition de la longueur stricte, π a exactement $slen(\pi) + 1$ classes d'équivalences par rapport à cette relation puisque le nombre de classes d'équivalence est lié au nombre d'arc strict $\stackrel{<}{\rightarrow}$ du chemin. Soit $X_0, \ldots, X_{slen(\pi)}$ une énumération des classes d'équivalence du chemin π. D'après le Lemme 12 **(II)** et **(III)**, chaque sommet n de π vérifie $\langle d_1, \mathrm{lev}(n)\rangle \stackrel{\sim}{\rightarrow} n \stackrel{\sim'}{\rightarrow} \langle d_2, \mathrm{lev}(n)\rangle$ pour $\sim, \sim' \in \{<, =\}$. De plus, comme C' contient toutes les constantes entre m et M et les représentations locales des valuations symboliques sont maximalement cohérentes, pour tout $i \in \{0, \ldots, slen(\pi)\}$ il existe une constante $d'_i \in C'$ telle que $d_1 \leq d'_i \leq d_2$ et chaque sommet n appartenant à X_i vérifie $n \stackrel{=}{\rightarrow} \langle d'_i, \mathrm{lev}(n)\rangle$. D'après le Lemme 12 **(I)**, chaque entier d_i doit être distinct sinon il existe un cycle stricte puisque les différentes classes d'équivalence sont reliées localement par des arcs stricts. Nous avons donc $slen(\pi) + 1$ classes d'équivalence non vides, chacune associée à un entier distinct compris entre d_1 et d_2. Ceci nous donne une contradiction puisque l'ensemble $\{d_1, \ldots, d_2\}$ a pour cardinalité $d_2 - d_1 + 1$ et nous avons besoin de $slen(\pi) + 1 > d_2 - d_1 + 1$ représentants distincts. $\square$

Pour terminer cette section nous établissons la relation entre les modèles symboliques construits par rapport à une formule ϕ de CLTL(IPC*) et les modèles concrets de ϕ. Pour cela, nous étendons la relation de satisfaction symbolique aux modèles symboliques. Cette relation que nous notons aussi $\models_{\mathrm{symb}}$ est définie de la même façon que la relation de satisfaction de CLTL(IPC*) excepté au niveau atomique : on a $\rho, i \models_{\mathrm{symb}} \alpha$ ssi $\rho(i) \models_{\mathrm{symb}} \alpha$. Le résultat suivant est le pivot de l'approche que nous définissons par la suite.

Lemme 13 *Une formule ϕ de* CLTL(IPC*) *est satisfaisable ssi il existe un modèle symbolique ρ et une séquence de valuations σ tels que $\rho \models \phi$ et $\sigma \models \rho$.*

Preuve : Soit σ un modèle qui satisfait ϕ. Nous considérons le modèle symbolique ρ tel que pour tout $i \in \mathbb{N}$ on a $\rho(i) = sv(v_i)$ où $v_i : \mathrm{Term} \rightarrow \mathbb{Z}$ est la valuation telle que $v_i(\mathsf{X}^j x) = \sigma(i + j)(x)$ pour tout $j \in \{0, \ldots, |\phi|_{\mathsf{X}}\}$. D'après cette construction, la séquence ρ est cohérente pas à pas et $\sigma \models \rho$. En utilisant le Lemme 10 **(II)**, nous obtenons que pour toute valuation v telle que $sv(v) = \rho(i)$ on a pour toute contrainte atomique α de ϕ, si $\sigma, i \models \alpha$ alors $v \models \alpha$. Ceci implique que pour tout $i \in \mathbb{N}$ si $\sigma, i \models \alpha$ alors $\rho, i \models_{\mathrm{symb}} \alpha$. Il est alors facile de conclure que $\rho \models_{\mathrm{symb}} \phi$ par induction sur la structure de ϕ, puisque la relation $\models_{\mathrm{symb}}$ n'est différente de $\models$ qu'au niveau atomique.

Réciproquement, supposons qu'il existe un modèle symbolique ρ et une séquence de valuations σ tels que $\rho \models_{\mathrm{symb}} \phi$ et $\sigma \models \rho$. Puisque pour tout $i \in \mathbb{N}$ on a $\sigma, i \models \rho(i)$, la valuation symbolique $\rho(i)$ est égale à $sv(v_i)$ où $v_i : \mathrm{Term} \rightarrow \mathbb{Z}$ est la valuation définie à partir de σ telle que $v_i(\mathsf{X}^j x) = \sigma(i + j)(x)$ pour tout $j \in \{0, \ldots, |\phi|_{\mathsf{X}}\}$. D'après la définition de $\models_{\mathrm{symb}}$, pour toute contrainte atomique α de ϕ, si $\rho(i) \models_{\mathrm{symb}} \alpha$ alors $v_i \models \alpha$. Ceci implique que si $\rho(i) \models_{\mathrm{symb}} \alpha$ alors $\sigma, i \models \alpha$. Nous pouvons alors déduire que comme $\rho \models_{\mathrm{symb}} \phi$, on a $\sigma \models \phi$ puisque les deux relations de satisfaction ne diffèrent qu'au niveau atomique. $\square$

4.3 Modèles symboliques satisfaisables

D'après le Lemme 13, nous avons donc besoin de caractériser l'ensemble des modèles symboliques qui sont satisfaisables. Jusqu'à présent nous avons établi des propriétés sur le graphe G_ρ uniquement. Nous définissons maintenant des conditions sur G_ρ qui sont équivalentes à l'existence d'un modèle qui satisfait ρ. Une solution pour un graphe de la forme G_ρ obtenu à partir d'un modèle symbolique ρ est une fonction d'étiquetage $lab : ((V \uplus C') \times \mathbb{N}) \to \mathbb{Z}$ vérifiant

- pour tout arc $n_1 \xrightarrow{\sim} n_2$, $lab(n_1) \sim lab(n_2)$,
- pour tout sommet n, $lab(n) \equiv_K mod(n)$.

Une telle solution est dite stricte ssi pour tout sommet $\langle d, i \rangle$ de G_ρ représentant une constante on a $lab(\langle d, i \rangle) = d$.

Lemme 14 *Un modèle symbolique ρ est satisfaisable ssi G_ρ a une solution.*

Preuve : Soit σ un modèle satisfaisant ρ et $lab : ((V \cup C')) \times \mathbb{N} \to \mathbb{Z}$ la fonction d'étiquetage définie par :

$$lab(\langle x, i \rangle) = \sigma(i)(x) \quad \text{et} \quad lab(\langle d, i \rangle) = d \quad \text{pour tout } x \in V, d \in C' \text{ et } i \in \mathbb{N}.$$

Il est facile de montrer que lab est une solution stricte de G_ρ. En effet, on a la suite d'implications suivante :

$\langle x, i \rangle \xrightarrow{\sim} \langle y, j \rangle$ et $i \leq j$,
$\Rightarrow$ $x \sim \mathsf{X}^{j-i} y \in \rho(i)$ (d'après la définition de G_ρ),
$\Rightarrow$ $\sigma, i \models x \sim \mathsf{X}^{j-i} y$ (car σ satisfait ρ),
$\Rightarrow$ $\sigma(i)(x) \sim \sigma(j)(y)$ (par définition de $\models$),
$\Rightarrow$ $lab(\langle x, i \rangle) \sim lab(\langle y, j \rangle)$ (par définition de lab).

Donc $\langle x, i \rangle \xrightarrow{\sim} \langle y, j \rangle$ et $i \leq j$ impliquent $lab(\langle x, i \rangle) \sim lab(\langle y, j \rangle)$. Les autres cas concernant les contraintes imposées par les arcs $\xrightarrow{\leq}$ et $\xrightarrow{=}$ peuvent être traités de la même manière. Nous montrons la satisfaction des contraintes de périodicité en utilisant le même type développement :

$mod(\langle x, i \rangle) = c$,
$\Rightarrow$ $x \equiv_K c \in \rho(i)$ (par définition de G_ρ),
$\Rightarrow$ $\sigma, i \models x \equiv_K c$ (car $\sigma \models \rho$),
$\Rightarrow$ $\sigma(i)(x) \equiv_K c$ (définition de $\models$),
$\Rightarrow$ $lab(\langle x, i \rangle) \equiv_K c$ (définition de lab).

Réciproquement, soit lab une solution de G_ρ. Nous construisons d'abord une solution stricte lab' à partir de lab. Le principe de la construction de lab' consiste à séparer les valeurs supérieures à M en blocs de K valeurs consécutives de manière à ce que si $lab(n) - lab(\langle M, \mathrm{lev}(n)\rangle) = b > 0$ alors $lab'(n)$ prend sa valeur dans le $b^{\text{ème}}$ bloc.

$$\overbrace{M}^{\text{bloc }0}\ \overbrace{M+1\cdots M+K}^{\text{bloc }1}\ \overbrace{M+K+1\cdots M+2K}^{\text{block }2}\ldots\overbrace{M+(\gamma-1)K+1\cdots M+\gamma K}^{\text{bloc }\gamma}\ldots$$

La contrainte définie par la fonction mod impose alors une valeur unique pour $lab'(n)$ dans ce bloc. Nous procédons de la même manière pour les valeurs inférieures à m. Ceci nous donne la construction suivante de lab'.

- Pour tout sommet $n = \langle x, i\rangle$ tel que $\langle x, i\rangle \xrightarrow{\leq} \langle m, i\rangle$, on pose $lab'(\langle x, i\rangle) = a$ tel que
  1. $a \equiv_K mod(\langle x, i\rangle)$,
  2. $m-(lab(\langle m, i\rangle)-lab(\langle x, i\rangle))\times K \leq a \leq m-((lab(\langle m, i\rangle)-lab(\langle x, i\rangle)-1)\times K-1$.
- Pour tout sommet $\langle x, i\rangle$ tel que $\langle M, i\rangle \xrightarrow{\leq} \langle x, i\rangle$, on pose $lab'(\langle x, i\rangle) = a$ tel que
  1. $a \equiv_K mod(\langle x, i\rangle)$,
  2. $M+((lab(\langle x, i\rangle)-lab(\langle M, i\rangle)-1)\times K+1 \leq a \leq M+(lab(\langle x, i\rangle)-lab(\langle M, i\rangle))\times K$.

  Dans les deux cas ci-dessus, nous rappelons que la valeur a est unique puisqu'elle appartient à un intervalle de longueur K (c.f. 2) et que la contrainte de périodicité impose une valeur unique dans cet intervalle (c.f. 1).
- Pour tout $\langle x, i\rangle$ tel que $\langle x, i\rangle \xrightarrow{=} \langle d, i\rangle$ pour $d \in C'$, on pose $lab'(\langle x, i\rangle) = d$.
- Pour tout sommet $\langle d, i\rangle$ représentant une constante, on pose $lab'(\langle d, i\rangle) = d$.

Il est facile de montrer que lab' est une solution stricte de G_ρ. Par exemple, nous traitons le cas $n \xrightarrow{\leq} n'$ où $\mathrm{lev}(n) \leq \mathrm{lev}(n')$ avec $\langle M, \mathrm{lev}(n)\rangle \xrightarrow{\leq} n$ et $\langle M, \mathrm{lev}(n')\rangle \xrightarrow{\leq} n'$. Les autres cas sont semblables. Puisque lab est une solution de G_ρ, on a $lab(\langle M, \mathrm{lev}(n)\rangle) < lab(n) < lab(n')$. Donc d'après la construction de lab', la valeur de $lab'(n)$ se situe dans le bloc $b = lab(n) - lab(\langle M, \mathrm{lev}(n)\rangle)$ et la valeur de $lab'(n')$ se situe dans le bloc $b' = lab(n') - lab(\langle M, \mathrm{lev}(n')\rangle)$. Comme $b < b'$ on a donc bien $lab'(n) < lab'(n')$. Les contraintes imposées par la fonction mod sont vérifiées par construction.

Nous montrons enfin que le modèle σ défini par $\sigma(i)(x) = lab'(\langle x, i\rangle)$ pour tout $x \in V$ et $i \in \mathbb{N}$ satisfait ρ. Pour cela il suffit de dérouler les implications de la première partie de cette preuve dans l'autre sens. Par exemple, pour le cas où $\mathsf{X}^j x \sim d \in \rho(i)$ on a

$\mathsf{X}^j x \sim d \in \rho(i)$,
$\Rightarrow$ $\langle x, i+j\rangle \xrightarrow{\sim} \langle d, i+j\rangle$ dans G_ρ (par construction),
$\Rightarrow$ $lab'(\langle x, i+j\rangle) \sim lab'(\langle d, i+j\rangle)$ (car lab' est une solution de G_ρ),
$\Rightarrow$ $\sigma(i+j)(x) \sim d$ (par définition de σ et puisque lab' est une solution stricte),
$\Rightarrow$ $\sigma, i \models \mathsf{X}^j x \sim \mathsf{X}^{j'} y$ (par définition de $\models$).

L'avant-dernière étape illustre la nécessité que lab' soit une solution stricte. $\square$

Le Lemme 14 établit la correspondance entre un modèle symbolique ρ et sa représentation graphique G_ρ. Nous pouvons maintenant caractériser les modèles symboliques satisfaisables en définissant une condition caractérisant les graphes G_ρ qui admettent une solution.

Lemme 15 *Soit ρ un modèle symbolique. Il existe une solution pour sa représentation graphique G_ρ ssi pour toute paire de sommets n_1, n_2 de G_ρ, on a $slen(n_1, n_2) < \omega$.*

Rappelons que par construction de G_ρ, pour tous sommets $\langle d_1, i\rangle$ et $\langle d_2, j\rangle$ représentant des constantes telles que $d_1 \leq d_2$ on a $slen(\langle d_1, i\rangle, \langle d_2, j\rangle) = d_2 - d_1$ (voir le Corollaire 4). C'est pour cette raison que nous n'avons besoin d'aucune condition particulière pour les sommets du graphe représentant les constantes.

Preuve : Si G_ρ admet une solution lab, il est alors évident que pour toute paire de sommets n_1, n_2 de G_ρ on a $slen(n_1, n_2) < lab(n_2) - lab(n_1)$.

Réciproquement, si pour toute paire de sommets n_1, n_2 de G_ρ on a $slen(n_1, n_2) < \omega$ alors nous définissons la fonction d'étiquetage $lab : (V \cup C') \times \mathbb{N} \to \mathbb{Z}$ telle que :

- Pour tout $d \in C'$ on a $lab(\langle d, i\rangle) = d$,
- Pour tout $\langle x, i\rangle \in V \times \mathbb{N}$,
  - si $\langle x, i\rangle \xrightarrow{=} \langle d, i\rangle$ alors $lab(\langle x, i\rangle) = d$
  - si $\langle x, i\rangle \xrightarrow{\leq} \langle m, i\rangle$ alors $lab(\langle x, i\rangle) = a$ tel que
    1. $a \equiv_K mod(\langle x, i\rangle)$.
    2. $m - slen(\langle x, i\rangle, \langle m, i\rangle) \times K \leq a \leq m - (slen(\langle x, i\rangle, \langle m, i\rangle) - 1) \times K - 1$.
  - si $\langle M, i\rangle \xrightarrow{\leq} \langle x, i\rangle$ alors $lab(\langle x, i\rangle) = a$ tel que
    1. $a \equiv_K mod(\langle x, i\rangle)$.
    2. $M + (slen(\langle M, i\rangle, \langle x, i\rangle) - 1) \times K + 1 \leq a \leq M + slen(\langle M, i\rangle, \langle x, i\rangle) \times K$.

Notons que, comme dans la preuve du Lemme 14, la valeur de a est unique dans l'intervalle de taille K à cause de la contrainte de périodicité.

Nous démontrons maintenant que lab' est une solution stricte G_ρ. Nous procédons par analyse de cas :

- Supposons que $n \xrightarrow{=} n'$. Nous traitons le cas $\mathrm{lev}(n) < \mathrm{lev}(n')$ (l'autre cas est symétrique). Nous rappelons que comme il existe un arc entre n et n' on a $\mathrm{lev}(n') - \mathrm{lev}(n) \leq l$ et donc les sommets n et n' appartiennent tous deux à la représentation locale de la valuation symbolique $\rho(\mathrm{lev}(n))$.

  Cas 1 : $\langle M, \mathrm{lev}(n)\rangle \xrightarrow{\leq} n$ et $\langle M, \mathrm{lev}(n')\rangle \xrightarrow{\leq} n'$.
  Puisque $n \xrightarrow{=} n'$ et $\langle M, \mathrm{lev}(n)\rangle \xrightarrow{=} \langle M, \mathrm{lev}(n')\rangle$ on a

  $$slen(\langle M, \mathrm{lev}(n)\rangle, n) = slen(\langle M, \mathrm{lev}(n)\rangle, n').$$

  Donc $lab(n)$ et $lab(n')$ appartiennent au même bloc de taille K de valeurs supérieures à M. De plus comme la représentation locale de la valuation symbolique $\rho(\mathrm{lev}(n))$ vérifie

(MC6), on a $mod(n) = mod(n')$. Par définition de lab on a donc bien $lab(n) = lab(n')$.

Cas 2 : Le cas $n \xrightarrow{\leq} \langle m, \text{lev}(n)\rangle$ et $n' \xrightarrow{\leq} \langle m, \text{lev}(n')\rangle$ est similaire au Cas 1, en considérant les blocs inférieurs à m.

Cas 3 : Nous considérons maintenant le cas tel que $\langle M, \text{lev}(n)\rangle \xrightarrow{\leq} n$ et $n' \xrightarrow{\sim} \langle M, \text{lev}(n')\rangle$ et nous démontrons par l'absurde que ce cas est impossible. Si nous supposons le contraire alors comme $\rho(\text{lev}(n))$ vérifie **(MC2)**, on a $n \xrightarrow{\sim} \langle M, \text{lev}(n')\rangle$ puisqu'il existe un arc $n \xrightarrow{=} n'$. Il existe donc un chemin

$$\langle M, \text{lev}(n)\rangle \xrightarrow{\leq} n \xrightarrow{\sim} \langle M, \text{lev}(n')\rangle \xrightarrow{=} \langle M, \text{lev}(n)\rangle,$$

qui contredit **(MC3)**.

Cas 4 : Le cas $n \xrightarrow{\lesssim} \langle m, \text{lev}(n)\rangle$ et $n' \xrightarrow{\sim} \langle m, \text{lev}(n')\rangle$ est similaire au Cas 3.

Cas 5 : Le cas $n \xrightarrow{=} \langle d, \text{lev}(n)\rangle$ et $n' \xrightarrow{=} \langle d', \text{lev}(n')\rangle$ avec $d < d'$ est aussi impossible. Sinon, par construction de la représentation de $\rho(\text{lev}(n))$ nous avons un arc de la forme $\langle d, \text{lev}(n)\rangle \xrightarrow{\lesssim} \langle d', \text{lev}(n')\rangle$. Nous pouvons donc construire un cycle strict qui contredit le fait que $\rho(\text{lev}(n))$ vérifie **(MC3)**.

– Le cas $n \xrightarrow{\leq} n'$ peut être traité de manière identique. □

Nous avons donc une caractérisation exacte des modèles symboliques qui sont satisfaisables. Néanmoins, cette caractérisation n'est pas vraiment satisfaisante car nous désirons une condition qui puisse être vérifiée à l'aide d'un automate. Ceci n'est pas évident puisque l'ensemble des modèles symboliques satisfaisable n'est pas ω-régulier, en conséquence d'un résultat de [DD07] pour le fragment CLTL(Z) (voir Lemme 17). Cependant, nous pouvons définir une condition d'approximation ω-régulière telle que tout modèle symbolique ρ *ultimement périodique* est satisfaisable ssi G_ρ vérifie cette condition. Une séquence infinie est ultimement périodique ssi elle est de la forme $\delta \cdot \gamma^\omega$ où δ et γ sont deux séquences finies.

Un *chemin infini en avant* dans G_ρ est un chemin π tel que pour tout $i \in \mathbb{N}$ on a $\pi(i) \xrightarrow{\sim} \pi(i+1)$ et $\text{lev}(\pi(i)) < \text{lev}(\pi(i+1))$. Un *chemin infini en arrière* est défini de la même manière avec pour tout $i \in \mathbb{N}$, $\pi(i) \xleftarrow{\sim} \pi(i+1)$. Un chemin est infiniment souvent strict s'il contient un infinité d'arcs stricts.

Un graphe G_ρ satisfait la condition ($\mathcal{C}$) ssi **il n'existe pas** de paire de sommets n_1, n_2 de G_ρ tels que $|\text{lev}(n_1) - \text{lev}(n_2)| \leq l$ vérifiant toutes les conditions suivantes.

(AP1) Il existe un chemin infini en avant π_{av} partant de n_1.

(AP2) Il existe un chemin infini en arrière π_{ar} partant de n_2.

(AP3) Soit π_{av} ou π_{ar} est infiniment souvent strict.

(AP4) Pour tout $i, j \in \mathbb{N}$ tel que $|\text{lev}(\pi_{\text{av}}(i)) - \text{lev}(\pi_{\text{ar}}(j))| \leq l$ on a $\pi_{\text{av}}(i) \xrightarrow{\leq} \pi_{\text{ar}}(j)$ dans G_ρ.

Un exemple de graphe qui ne satisfait pas la condition ($\mathcal{C}$) est donné dans la Figure 4.3 (certains arcs sont omis).

Si un modèle symbolique ρ est satisfaisable, alors sa représentation graphique G_ρ doit satisfaire ($\mathcal{C}$). Sinon, on a $slen(n_1, n_2) = \omega$ ce qui entraîne une contradiction d'après le

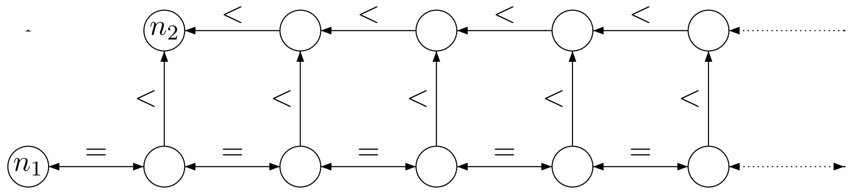


Figure 4.3: Graphe ne vérifiant pas ($\mathcal{C}$)

Lemme 15. L'implication inverse n'est pas vraie en générale. La Figure 4.4 est un exemple de graphe qui satisfait ($\mathcal{C}$) et n'admet pas de solution car $slen(\langle x, 0\rangle, \langle z, 0\rangle) = \omega$. Néanmoins,

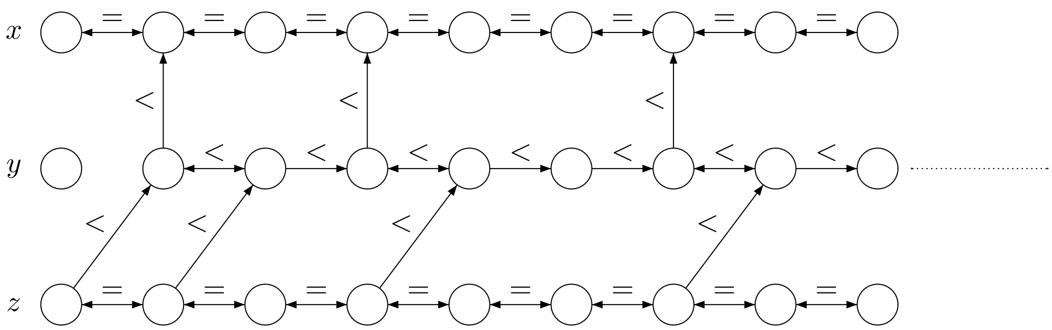


Figure 4.4: Graphe non satisfaisable vérifiant ($\mathcal{C}$)

lorsque le modèle symbolique est ultimement périodique cette condition est suffisante.

Lemme 16 *Un modèle symbolique ultimement périodique ρ est satisfaisable ssi sa représentation graphique G_ρ satisfait ($\mathcal{C}$).*

Preuve : Cette preuve est essentiellement la même que pour le cas de CLTL(Z). Les principales différences avec [DD07, Lemme 6.2] résident dans la présence de contraintes de périodicité et de constantes. Cependant, les résultats précédents nous permettent de traiter uniformément les variables et les constantes et d'ignorer la fonction *mod*.

Nous considérons un modèle symbolique construit sur des ressources syntaxiques V, C', l et K définies à partir de l'ensemble des contraintes atomiques d'une formule de CLTL(IPC*). Nous rappelons que si deux sommets n_1, n_2 violent la condition ($\mathcal{C}$) alors $slen(n_1, n_2) = \omega$. Donc si ρ est satisfaisable alors G_ρ satisfait ($\mathcal{C}$) d'après les Lemmes 14 et 15.

Réciproquement, supposons que le modèle symbolique ρ soit de la forme $\delta \cdot \gamma^\omega$ et que G_ρ n'admette pas de solution. Si G_ρ n'a pas de solution, il existe d'après le Lemme 15 deux sommets n_1 et n_2 tels que $slen(n_1, n_2) = \omega$. Nous utilisons cette propriété pour montrer que G_ρ ne satisfait pas ($\mathcal{C}$).

Nous notons lev_0 le plus petit niveau tel que

- $lev_0 \equiv_{|\gamma|} |\delta|$ (lev_0 est égal à $|\delta|$ modulo $|\gamma|$),
- $lev_0 \geq \text{lev}(n_1)$ et $lev_0 \geq \text{lev}(n_2)$,

et nous posons $lev_1 = lev_0 + l|\gamma| \times |V \cup C'|$. Comme $slen(n_1, n_2) = \omega$, il existe un chemin π allant de n_1 vers n_2 tel que

$$slen(\pi) > lev_1|V \cup C'| + |\gamma| \times |V \cup C'|.$$

Nous pouvons supposer sans perte de généralité que π n'a pas de cycles puisque d'après le Lemme 12 **(I)** G_ρ n'a pas de cycles stricts et les cycles composés de relations d'égalité uniquement n'affectent pas la longueur stricte d'un chemin. En conséquence chaque sommet de π est visité une seule fois. Comme il n'existe que $lev_1|V \cup C'|$ sommets distincts de niveau inférieur ou égal à lev_1 et que par définition π contient au moins $slen(\pi)$ arcs stricts, il existe donc au moins $|\gamma| \times |V \cup C'| + 1$ arcs stricts entre des sommets de niveau supérieur ou égal à lev_1.

Nous disons que deux sommets $\langle x, i\rangle$ et $\langle x', i'\rangle$ de niveau supérieur à $|\delta|$ (c'est-à-dire dans la partie ultimement périodique) sont *isomorphes* ssi

- $x = x'$,
- $i > |\delta|$ et $i' > |\delta|$,
- $i \equiv_{|\gamma|} i'$.

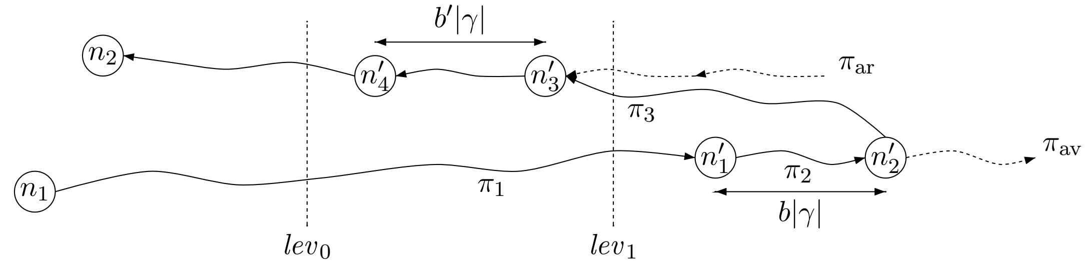
D'après la périodicité, pour tout $j > |\delta|$ on a $\rho(j) = \rho(j + |\gamma|)$. Donc il existe un arc entre deux sommets du graphe ssi il existe le même type d'arc entre leurs images par le même isomorphisme. Par extension, il existe un chemin entre deux sommets du graphe ssi il existe un chemin entre leurs images par le même isomorphisme. De plus, il n'existe pas d'ensemble X de sommets de niveau supérieur à lev_1 tous non-isomorphes tel que $|X| > |\gamma||V \cup C'|$. Comme le nombre d'arêtes strictes de π dans la partie du graphe supérieure à lev_1 est strictement plus grand que $|\gamma||V \cup C'|$, le chemin π visite deux sommets isomorphes n'_1 et n'_2 tels que le sous-chemin de π entre ces deux sommets est strict. Nous supposons sans perte de généralité que n'_1 est visité en premier par π.

Nous pouvons construire à partir de π un chemin π' visitant successivement n_1, n'_1, n'_2 et n_2 qui se décompose en plusieurs parties :

1. un chemin en avant π_1 de n_1 vers n'_1,
2. un chemin strict en avant π_2 de n'_1 vers n'_2
   ou un chemin strict en arrière de n'_2 vers n'_1 selon les cas,
3. un chemin en arrière π_3 entre n_2 et n'_2.

Cette construction repose sur l'observation suivante. Supposons qu'un chemin π ait pour origine un sommet n et destination un sommet n' telle que $\mathrm{lev}(n) < \mathrm{lev}(n')$. On peut construire un chemin en avant entre n et n' en procédant de la façon suivante. Soit $i \in \mathbb{N}$ une position du chemin tel que $\mathrm{lev}(\pi(i+1)) \leq \mathrm{lev}(\pi(i))$, c'est-à-dire que l'arc pris n'est pas en avant. Considérons le plus petit indice $j > i$ tel que $\mathrm{lev}(\pi(j)) > \mathrm{lev}(\pi(i))$, s'il existe. Le sommet $\pi(j)$ doit vérifier $\mathrm{lev}(\pi(j)) - \mathrm{lev}(\pi(i)) \leq l$ car par construction du graphe, un arc traverse au plus l niveaux. Ceci signifie par définition de G_ρ qu'il existe un arc entre $\pi(i)$ et $\pi(j)$. Si $\pi(j) \xrightarrow{<} \pi(i)$ alors on a un cycle strict, ce qui n'est pas possible car le graphe vérifie **(MC3)** localement. Donc les seuls cas possibles sont $\pi(i) \xrightarrow{=} \pi(j)$ ou $\pi(i) \xrightarrow{<} \pi(j)$. Notons qu'on a $\pi(i) \xrightarrow{<} \pi(j)$ ssi il existe un arc strict dans le sous-chemin entre $\pi(i)$ et $\pi(j)$. On peut donc "court-circuiter" le sous-chemin entre $\pi(i)$ et $\pi(j)$ en prenant l'arc en avant entre ces deux sommets. En itérant cette méthode de court-circuit sur le chemin π, nous obtenons un chemin en avant. Le même genre d'argument nous sert à construire un chemin en arrière lorsque $\mathrm{lev}(n') < \mathrm{lev}(n)$. Les chemins π_1 et π_3 peuvent être construit en utilisant cette méthode de court-circuit.

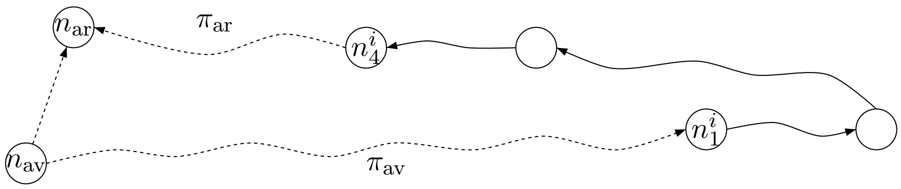
Nous distinguons maintenant deux cas. Si le niveau de n'_1 est inférieur à celui de n'_2 alors la figure suivante schématise la situation.



Le chemin π_2 construit en utilisant la méthode de court-circuit est un chemin en avant strict entre n'_1 et n'_2. Comme ces sommets sont isomorphes, nous pouvons montrer qu'il existe un chemin en avant infiniment souvent strict à partir de n'_1. Posons $n'_1 = \langle x, a\rangle$ et $n'_2 = \langle x, a + b|\gamma|\rangle$ pour $x, x' \in V \cup C'$ et $a, b \in \mathbb{N}$. Le sommet $n'_2 = \langle x, a + b|\gamma|\rangle$ est aussi isomorphe à $\langle x, a + 2b|\gamma|\rangle$ par le même isomorphisme et donc le chemin en avant strict qui existe entre n'_1 et n'_2 existe aussi entre n'_2 et $\langle x, a+2b\rangle$. Cet argument peut être répété pour tout couple de sommets de la forme $\langle x, a+ib\rangle$ et $\langle x, a+(i+1)b\rangle$ avec $i \in \mathbb{N}$ afin de construire un chemin en avant infiniment souvent strict. En conséquence il existe un chemin en avant infiniment souvent strict π_{av} à partir de n_1.

De plus, notons que le chemin π_3 traverse tous les niveaux entre *lev₀* et *lev₁*. Comme un arc traverse au maximum l niveaux, π_3 visite au moins $|\gamma||V \cup C'| + 1$ sommets et donc deux de ces sommets sont forcément isomorphes. Nous notons n'_3 et n'_4 ces sommets tels que le niveau de n'_3 est inférieur à celui de n'_4. On a donc un chemin en arrière entre n'_3 et n'_4. En utilisant les mêmes arguments que pour le chemin entre n'_1 et n'_2, nous pouvons en déduire qu'il existe un chemin en arrière infini π_{ar} à partir de n_2 (pas forcément strict).

Il reste à montrer que pour tous sommets n_{av} de π_{av} et n_{ar} de π_{ar} tels que $|\mathrm{lev}(n_{\mathrm{av}}) - \mathrm{lev}(n_{\mathrm{ar}})| \le l$ on a $n_{\mathrm{av}} \xrightarrow{\le} n_{\mathrm{ar}}$. Nous posons $n'_1 = \langle x, a\rangle$, $n'_2 = \langle x, a + b|\gamma|\rangle$, $n'_4 = \langle y, a'\rangle$ et $n'_3 = \langle y, a' + b'|\gamma|\rangle$ en respectant les isomorphismes respectifs entre ces sommets. Pour tout $i \in \mathbb{N}$, les sommets $n^i_1 = \langle x, a + ibb'|\gamma|\rangle$ et $n^i_4 = \langle y, a' + ibb'|\gamma|\rangle$ sont isomorphes et sont des images de n'_1 et n'_4 sous le même isomorphisme. D'après les étapes précédentes de notre construction, pour tout $i \in \mathbb{N}$ le sommet n^i_1 appartient à π_{av} et n^i_4 à π_{ar}. Soient n_{av} et n_{ar} des sommets respectifs de π_{av} et π_{ar} tels que $|\mathrm{lev}(n_{\mathrm{av}}) - \mathrm{lev}(n_{\mathrm{ar}})| \le l$ et considérons deux sommets n^i_1 et n^i_4 définis par le même isomorphisme dont les niveaux sont minimaux tout en étant supérieurs aux niveaux de n_{av} et n_{ar}. D'après les observations précédentes, il existe un chemin strict entre n^i_1 et n^i_4 car il existe le même type de chemin entre n'_1 et n'_4. Ceci implique l'existence d'un chemin strict entre n_{av} et n_{ar} car n^i_1 appartient à π_{av} et n^i_4 à π_{ar}. Puisque $|\mathrm{lev}(n_{\mathrm{av}}) - \mathrm{lev}(n_{\mathrm{ar}})| \le l$ il existe un arc entre les deux sommets. Cet arc est $n_{\mathrm{av}} \xrightarrow{\le} n_{\mathrm{ar}}$ sinon nous avons un cycle strict ce qui est impossible d'après le Lemme 12 (I). Le dessin suivant schématise cette partie de la preuve.



Si l'on suppose maintenant que le niveau de n_1' est inférieur à celui de n_2' alors le chemin π_2 est en arrière. Ceci nous permet de déduire l'existence d'un chemin en arrière infiniment souvent strict. Nous pouvons alors utiliser les mêmes arguments que dans le cas précédent pour déduire un chemin infini en avant à partir de deux sommets isomorphes visités par le chemin π_1 entre les niveaux *lev0* et *lev1*, puis pour montrer que ces chemins vérifient la propriété (AP4) énoncée dans la définition de $(\mathcal{C})$. □

Pour conclure cette section, nous revenons sur l'affirmation faite plus tôt concernant la non-ω-régularité de l'ensemble des modèles symboliques satisfaisables. Le résultat suivant est une variante de [DD07, Corollaire 6.5].

Lemme 17 *Il existe des ensembles X de contraintes atomiques de* CLTL(IPC*) *tels que l'ensemble des modèles symboliques satisfaisables construit par rapport à X n'est pas ω-régulier.*

Preuve : La preuve s'inspire de [DD07, Corollaire 6.5]. Nous notons L l'ensemble des modèles symboliques $\mathbb{N} \rightarrow \mathrm{SV}(X)$ construits par rapport à un ensemble X de contraintes atomiques de CLTL(IPC*) ayant les ressources syntaxiques suivantes :

- $l = 1$,
- $V = \{x, y, z\}$,
- $C = \emptyset$,
- la valeur de K n'est pas importante ici.

Nous notons $\mathrm{L}_{\mathrm{sat}} \subseteq \mathrm{L}$ l'ensemble des modèles symboliques satisfaisables et $\mathrm{L}_{\mathcal{C}} \subseteq \mathrm{L}$ l'ensemble des modèles symboliques qui satisfont $(\mathcal{C})$. Nous anticipons la construction de l'automate de Büchi $\mathcal{A}_{\mathrm{sat}}$ de la Section 4.4 et admettons que $\mathrm{L}_{\mathcal{C}}$ est un langage ω-régulier. De même il est évident que L est un langage ω-régulier puisque les éléments de ce langage sont justes des séquences de valuations symboliques cohérentes pas à pas (voir la construction de $\mathcal{A}_{\mathrm{pap}}$).

Nous procédons par l'absurde. Supposons que $\mathrm{L}_{\mathrm{sat}}$ soit ω-régulier. Comme les langages ω-réguliers sont clos par complémentation et intersection, le langage $\mathrm{L}' = (\mathrm{L} \setminus \mathrm{L}_{\mathrm{sat}}) \cap \mathrm{L}_{\mathcal{C}}$ est ω-régulier. Le modèle symbolique représenté dans la Figure 4.4 satisfait $(\mathcal{C})$ et n'est pas satisfaisable, il appartient donc à L'. Puisque L' est un langage ω-régulier non-vide, il existe un mot ultimement périodique ρ qui appartient à L'. Or, si ρ appartient à $\mathrm{L}_{\mathcal{C}}$ et ρ est ultimement périodique alors d'après le Lemme 16 ce modèle symbolique est satisfaisable. Ceci contredit la définition de L' car on doit avoir $\rho \notin \mathrm{L}_{\mathrm{sat}}$. □

4.4 Construction de l'automate et complexité

À partir des résultats précédents et en suivant l'approche définie dans [VW94], nous montrons maintenant comment, étant donnée une formule ϕ de CLTL(IPC*), on peut construire un automate de Büchi $\mathcal{A}_\phi$ tel que ϕ est satisfaisable ssi le langage reconnu par $\mathcal{A}_\phi$ est non-vide. De plus, nous montrons que tester si le langage reconnu par $\mathcal{A}_\phi$ est non-vide peut se faire en espace polynomial par rapport à la taille de ϕ. Nous considérons une formule ϕ de CLTL(IPC*) ainsi que les différents éléments associés à l'ensemble de ses contraintes atomiques définis dans la Section 4.2.1 (l, V, C, m, etc...).

À la différence de la construction pour LTL, le langage reconnu par $\mathcal{A}_\phi$ n'est pas un ensemble de modèles mais un ensemble de modèles symboliques. D'après les Lemmes 13 et 16, l'automate $\mathcal{A}_\phi$ peut être défini comme l'intersection de trois automates de Büchi $\mathcal{A}_{\mathrm{pap}} \cap \mathcal{A}_{\mathrm{symb}} \cap \mathcal{A}_{\mathrm{sat}}$ tels que

- L($\mathcal{A}_{\mathrm{pap}}$) est l'ensemble des séquences de valuations symboliques cohérentes pas à pas,
- L($\mathcal{A}_{\mathrm{symb}}$) est l'ensemble des séquences de valuations symboliques ρ qui satisfont symboliquement ϕ, c'est-à-dire $\rho \models_{\mathrm{symb}} \phi$,
- L($\mathcal{A}_{\mathrm{sat}}$) est l'ensemble des séquences de valuations symboliques ρ vérifiant ($\mathcal{C}$).

Nous décrivons ci-dessous la construction de ces différents automates. Tous ont pour alphabet l'ensemble fini SV(ϕ) dont la taille est exponentielle par rapport à $|\phi|$. L'automate $\mathcal{A}_{\mathrm{symb}}$ est obtenu en adaptant la construction de [VW94] avec la relation de satisfaction symbolique. Nous définissons la clôture habituelle de ϕ notée $cl(\phi)$ en considérant chaque formule atomique comme une proposition. Nous rappelons que $cl(\phi)$ est le plus petit ensemble contenant ϕ est vérifiant les conditions suivantes :

- $\phi_1 \in cl(\phi)$ ssi $\neg\phi_1 \in cl(\phi)$ (pour toute formule ϕ_1, on identifie ϕ_1 et $\neg\neg\phi_1$),
- si $\phi_1 \wedge \phi_2 \in cl(\phi)$ ou $\phi_1 \vee \phi_2 \in cl(\phi)$ alors $\phi_1 \in cl(\phi)$ et $\phi_2 \in cl(\phi)$,
- si $\mathsf{X}\phi_1 \in cl(\phi)$ alors $\phi_1 \in cl(\phi)$,
- si $\phi_1 \mathsf{U} \phi_2 \in cl(\phi)$ alors $\phi_1 \in cl(\phi)$, $\phi_2 \in cl(\phi)$ et $\mathsf{X}(\phi_1 \mathsf{U} \phi_2) \in cl(\phi)$.

Un atome de ϕ est un sous-ensemble maximalement cohérent de $cl(\phi)$. L'automate de Büchi généralisé $\mathcal{A}_{\mathrm{symb}} = \langle Q, I, F, \delta \rangle$ est défini de la façon suivante.

- Q est l'ensemble des atomes de ϕ et $I = \{At \in Q \mid \phi \in At\}$,
- $At \xrightarrow{sv} At'$ est une transition de δ ssi

  (contraintes atomiques) pour toute formule atomique α dans At on a $sv \models_{\mathrm{symb}} \alpha$,
  (cohérence sur un pas) pour chaque formule $\mathsf{X}\psi \in cl(\phi)$ on a $\mathsf{X}\psi \in At$ ssi $\psi \in At'$,
- Soit $\{\phi_1 \mathsf{U} \psi_1, \dots, \phi_r \mathsf{U} \psi_r\}$ l'ensemble des formules until de $cl(\phi)$. Si cet ensemble est vide, tous les états sont acceptants. Sinon, on pose $F = \{F_1, \dots, F_r\}$ tel que pour tout $i \in \{1, \dots, r\}$ on a $F_i = \{At \in Q \mid \phi_i \mathsf{U} \psi_i \notin At \text{ ou } \psi_i \in At\}$.

D'après le Lemme 11, les conditions sur les contraintes atomiques peuvent être vérifiées en espace polynomial, donc la construction $\mathcal{A}_{\mathrm{symb}}$ peut se faire en espace polynomial.

L'automate $\mathcal{A}_{\text{pap}} = \langle Q, I, F, \rightarrow \rangle$ est un automate de Büchi tel que $Q = I = F = \text{SV}(\phi)$ et la relation de transition est définie par la règle :

$$sv_1 \xrightarrow{sv} sv_2 \text{ ssi la paire } \langle sv_1, sv_2 \rangle \text{ est cohérente sur un pas et } sv = sv_1.$$

Nous rappelons que l'on peut tester qu'un triplet donné d'ensembles de contraintes de CLTL(IPC*) est une valuation symbolique et qu'une paire de valuations symboliques est cohérente sur un pas en temps polynomial. Donc, la relation de transition de $\mathcal{A}_{\text{pap}}$ est calculable en temps polynomial.

Il nous reste à définir la construction de $\mathcal{A}_{\text{sat}}$ qui reconnaît l'ensemble des séquences de valuations symboliques qui satisfont $(\mathcal{C})$. Nous nous inspirons pour cette construction de [DD07]. Au lieu de construire directement $\mathcal{A}_{\text{sat}}$, il est plus facile de passer par la construction de l'automate de Büchi généralisé $\tilde{\mathcal{A}}_{\mathcal{C}}$ qui reconnaît le complément du langage $\text{L}(\mathcal{A}_{\text{sat}})$. L'automate $\tilde{\mathcal{A}}_{\mathcal{C}}$ est presque le même que l'automate $\mathcal{B}$ de [DD07, Section 6], mais nous avons besoin d'étendre l'alphabet et de tenir compte des relations entre les sommets du graphe représentant des constantes de C' car certaines comparaisons avec ces constantes ne sont pas explicitées dans les valuations symboliques. En effet, nous n'avons pas toutes les informations concernant les constantes $C' \setminus C$ dans la représentation symbolique des valuations. Mis à part ce point, les variables et les constantes ont un traitement uniforme dans la construction de $\tilde{\mathcal{A}}_{\mathcal{C}}$. Formellement, l'automate $\tilde{\mathcal{A}}_{\mathcal{C}} = \langle Q, I, F, \rightarrow \rangle$ est défini de la façon suivante.

- $Q = \{q_0\} \uplus \{(V \cup C') \times \{0, \ldots, l\} \times (V \cup C') \times \{0, \ldots, l\} \times \{\text{av}, \text{ar}\} \times \{0, 1\}\}$ où $l = |\phi|_{\mathsf{X}}$,
- $I = \{q_0\}$,
- La relation de transition vérifie les propriétés ci-dessous.
  (a) $q_0 \xrightarrow{sv} q_0$ pour tout $sv \in \text{SV}(\phi)$.
  (b) $q_0 \xrightarrow{sv} \langle a, i, b, j, \text{av}, 0 \rangle$ et $q_0 \xrightarrow{sv} \langle a, i, b, j, \text{ar}, 0 \rangle$ pour tout $a, b \in V \cup C'$, $i, j \in \{0, \ldots, l\}$ et $sv \in \text{SV}(\phi)$.
  (c) $\langle a, i, b, j, p, bin \rangle \xrightarrow{sv} \langle a, i-1, b, j-1, p, bin \rangle$ pour tout $p \in \{\text{av}, \text{ar}\}$, $bin \in \{0, 1\}$, $i, j \geq 1$ et $sv \in \text{SV}(\phi)$.
  (d) $\langle a, 0, b, j, \text{av}, bin \rangle \xrightarrow{sv} \langle a', i'-1, b, j-1, \text{av}, bin' \rangle$ tel que $i' > 0$, $j > 0$ et

    si $a, a' \in V$ alors,
    1. $a < \mathsf{X}^{i'} a' \in sv$ et $bin' = 1$
       ou $a = \mathsf{X}^{i'} a' \in sv$ et $bin' = 0$,
    2. si $b \in V$, $\mathsf{X}^{i'} a' < \mathsf{X}^j b \in sv$,
       si $b \in C'$, il existe d tel que $d \sim b$, $\mathsf{X}^{i'} a' \sim' d \in sv$ et $<\in \{\sim, \sim'\}$,

    si $a \in V$ et $a' \in C'$ alors,
    1. $a = a' \in sv$ et $bin' = 0$
       ou il existe d tel que $d \sim a'$, $a \sim' d \in sv$, $<\in \{\sim, \sim'\}$ et $bin' = 1$,

2. si $b \in V$, il existe $a' \sim d$ tel que $d \sim' \mathsf{X}^j b \in sv$ et $< \in \{\sim, \sim'\}$,
   si $b \in C'$ alors $a' < b$,

si $a \in C'$ et $a' \in V$ alors,

1. $a = \mathsf{X}^{i'} a' \in sv$ et $bin' = 0$
   ou il existe d tel que $a \sim d$, $d \sim' \mathsf{X}^{i'} a \in sv$, $< \in \{\sim, \sim'\}$ et $bin' = 1$,
2. si $b \in V$, $\mathsf{X}^{i'} a' < \mathsf{X}^j b \in sv$,
   si $b \in C'$, il existe d tel que $d \sim b\ \mathsf{X}^{i'} a' \sim' d \in sv$ et $< \in \{\sim, \sim'\}$,

si $a, a' \in C'$ alors,

1. $a = a'$ et $bin' = 0$
   ou $a < a'$ et $bin' = 1$
2. si $b \in V$, il existe d tel que $a' \sim d$, $d \sim' \mathsf{X}^j b \in sv$ et $< \in \{\sim, \sim'\}$,
   si $b \in C'$ alors $a' < b$,

$\langle a, i, b, 0, \mathrm{av}, bin \rangle \xrightarrow{sv} \langle a, i-1, b, j'-1, \mathrm{av}, bin' \rangle$ si $i > 0$, $j' > 0$ et les conditions du cas précédent sont vérifiées lorsqu'on opère les substitutions suivantes

$$a \leftarrow b, \qquad a' \leftarrow b', \qquad b \leftarrow a.$$

(e) $\langle a, 0, b, 0, \mathrm{av}, bin \rangle \xrightarrow{sv} \langle a', i', b', j', \mathrm{av}, bin' \rangle$ si la combinaison naturellement définie des contraintes des cas précédents est vérifiée.

(f) Les conditions sont similaires lorsque le chemin infiniment souvent strict est π_{ar}, c'est-à-dire pour les états de la forme $\langle a, i, b, j, \mathrm{ar}, bin \rangle$.

– F est l'ensemble des états de la forme $\langle a, i, b, j, p, 1 \rangle$ pour tout $a, b \in V \cup C'$, $i, j \in \{0, \ldots, l\}$ et $p \in \{\mathrm{av}, \mathrm{ar}\}$.

Nous expliquons en quelques mots comment procède l'automate $\tilde{\mathcal{A}}_{\mathcal{C}}$. Lors d'une exécution, on boucle d'abord sur l'état initial (point (a)) afin de passer les valuations symboliques qui se trouvent avant les sommets du graphe qui contredisent $(\mathcal{C})$. Puis, on devine de façon non déterministe deux sommets n_1 et n_2 ainsi que lequel des chemins π_{av} et π_{ar} est infiniment souvent strict (point (b)). Ensuite, l'automate vérifie que ces sommets satisfont toutes les conditions **(AP1)** à **(AP4)** (points (c) à (f) et condition d'acceptation). Le point (c) sert juste à attendre que le sommet courant de π_{av} ou π_{ar} soit de niveau 0 dans la valuation symbolique courante. Les différents cas du point (d) reprennent les conditions correspondant à l'existence d'arcs dans le graphe afin de poursuivre le chemin π_{av} tout en testant que le nouveau sommet courant vérifie bien **(AP4)**. Enfin, la condition d'acceptation garantit que des arcs étiquetés par $<$ sont rencontrés infiniment souvent par le chemin supposé infiniment souvent strict. Nous stockons toutes ces informations dans les états de l'automate $\tilde{\mathcal{A}}_{\mathcal{C}}$. Par exemple, si l'automate est dans l'état $\langle a, i, b, j, \mathrm{av}, 0 \rangle$, cela signifie que :

– la position du sommet courant du chemin en avant dans la valuation symbolique courante est $\langle a, i \rangle$,
– la position du sommet courant du chemin en arrière dans la valuation symbolique courante est $\langle b, j \rangle$,

– le chemin en avant est infiniment souvent strict.

La dernière composante (qui vaut 0 dans notre exemple) sert à marquer les moments où le chemin infiniment souvent strict passe par un arc strict (voir le point (d)). Cette valeur vaut 1 lorsque le chemin franchit un arc étiqueté par $<$. C'est pour cela que la condition d'acceptation impose que l'on visite infiniment souvent un état dont la dernière composante est égale à 1.

Lemme 18 *Une formule ϕ de* CLTL(IPC*) *est satisfaisable ssi* $\mathrm{L}(\mathcal{A}_\phi)$ *est non-vide.*

Si la formule ϕ est satisfaite par un modèle symbolique ρ alors cette séquence est reconnue par $\mathcal{A}_\phi$, d'après le Lemme 13 et la construction de $\mathcal{A}_\phi$ en tant que l'intersection de $\mathcal{A}_{\text{pap}}$, $\mathcal{A}_{\text{symb}}$ et $\mathcal{A}_{\text{sat}}$. L'implication inverse est moins triviale. Par propriété des automates de Büchi si le langage $\mathrm{L}(\mathcal{A}_\phi)$ est non-vide alors $\mathcal{A}_\phi$ accepte une séquence ultimement périodique ρ et donc nous pouvons utiliser le Lemme 16 pour prouver que ρ est satisfaisable. En effet, par construction de $\mathcal{A}_\phi$ on a $\rho \in \mathrm{L}(\mathcal{A}_{\text{sat}})$ et donc ρ vérifie $(\mathcal{C})$. De même, si ρ est reconnu par $\mathcal{A}_\phi$ alors ρ satisfait symboliquement ϕ car $\rho \in \mathrm{L}(\mathcal{A}_{\text{symb}})$. Donc ϕ est satisfaisable d'après le Lemme 13. Puisque nous pouvons construire $\mathcal{A}_\phi$ à partir d'une formule ϕ, le problème de la satisfaisabilité pour CLTL(IPC*) est décidable. Nous avons aussi tous les éléments pour établir la complexité de ce problème.

Théorème 23 *Le problème de la satisfaisabilité pour* CLTL(IPC*) *est* PSPACE-*complet.*

Preuve : La PSPACE-dureté est une conséquence de la PSPACE-dureté de LTL [SC85] puisque LTL est inclus dans CLTL(IPC*). Pour obtenir la borne supérieure, nous analysons la complexité de la construction de $\mathcal{A}_\phi$. La taille des automates $\mathcal{A}_{\text{symb}}$, $\mathcal{A}_{\text{pap}}$ et $\tilde{\mathcal{A}}_{\mathcal{C}}$ est exponentielle en $|\phi|$ mais ces automates peuvent être construits en espace polynomial par rapport à $|\phi|$.

L'automate $\mathcal{A}_{\text{sat}}$ peut être obtenu à partir de $\tilde{\mathcal{A}}_{\mathcal{C}}$ en utilisant la construction de Safra permettant de complémenter un automate de Büchi. Le nombre d'états $P(|\phi|)$ de $\tilde{\mathcal{A}}_{\mathcal{C}}$ est polynomial par rapport à $|\phi|$. On peut donc construire un automate de Street non déterministe acceptant le complément de $\mathrm{L}(\tilde{\mathcal{A}}_{\mathcal{C}})$ et ayant un nombre d'états de l'ordre de $\mathcal{O}(2^{P(|\phi|)\times log(|\phi|)})$ en espace polynomial [Saf88]. Cet automate peut être transformé en automate de Büchi équivalent $\mathcal{A}_{\text{sat}}$ ayant un nombre d'états du même ordre. Ceci nous permet de construire l'automate $\mathcal{A}_{\text{sat}}$ en espace polynomial en $|\phi|$.

Par conséquent, la construction de l'automate $\mathcal{A}_\phi$ qui est l'intersection de $\mathcal{A}_{\text{pap}}$, $\mathcal{A}_{\text{symb}}$ et $\mathcal{A}_{\text{sat}}$ peut se faire en espace polynomial par rapport à $|\phi|$. Puisque le test du vide dans un automate de Büchi est un problème NLOGSPACE-complet, nous obtenons une procédure en espace polynomial non-déterministe par composition avec la construction de l'automate (voir [BDG88, Corollaire 3.36]). D'après le théorème de Savitch, cette procédure est aussi dans PSPACE. □

Puisque le problème de la satisfaisabilité est équivalent au problème du model-checking pour CLTL(IPC*), nous avons aussi le corollaire suivant.

Corollaire 5 *Le problème du model-checking pour* CLTL(IPC*) *est* PSPACE-*complet.*

Notons que la méthode d'abstraction que nous avons utilisée permet d'établir la même borne de complexité pour les problèmes de la satisfaisabilité et du model-checking de la logique CLTL(WIPC*) ou de l'extension de CLTL(IPC*) avec des contraintes de la forme $ax + by \equiv_k c$. En effet, la même représentation symbolique des valuations vérifie le résultat du Lemme 10 pour les contraintes atomiques de ces logiques respectives ce qui permet d'adapter le résultat du Lemme 13 établissant la relation entre les modèles symboliques et les modèles concrets. La taille des valuations symboliques est toujours polynomiale par rapport à la taille de la formule donnée et malgré les différences de concision entre ces langages la construction de $\mathcal{A}_\phi$ reste dans PSPACE.

De plus tous les opérateurs temporels de CLTL(IPC*) peuvent être définis dans la logique monadique du second ordre (MSO). En utilisant [GK03], nous pouvons donc étendre ce résultat de PSPACE-complétude à n'importe quelle extension CxLTL(IPC*) obtenue en ajoutant à CLTL(IPC*) un ensemble fini d'opérateurs MSO-définissables. Nous devons juste modifier $\mathcal{A}_{\text{symb}}$ en adaptant la construction de l'automate de Büchi qui reconnaît les modèles d'une formule appartenant à l'extension de LTL avec les mêmes opérateurs MSO-définissables.

Théorème 24 *Les problèmes de la satisfaisabilité et du model-checking pour* CxLTL(IPC*) *sont* PSPACE*-complets.*

Enfin, d'après la correspondance entre les IPC*-automates et les IRA établie par le Lemme 8, un autre corollaire de ces résultats est que le model-checking du fragment linéaire de la logique de [Čer94] sur les IRA est dans PSPACE. La borne de complexité n'est pas établie par les résultats de [Čer94] pour ce fragment.

Corollaire 6 *Le problème du model-checking du fragment linéaire de l'extension de* CTL* *introduite dans [Čer94] sur les IRA est dans* PSPACE.

Chapitre 5

Vérification de contraintes quantitatives

Sommaire

5.1 LTL avec contraintes quantitatives

5.1.1 Présentation de la logique

Dans ce chapitre nous nous intéressons à un nouveau fragment de LTL étendu avec un ensemble de contraintes de Presburger contenant des contraintes quantitatives de la forme $x = y + d$ où x et y sont des variables et d une constante. Nous étudions principalement la logique CLTL(DL) dont les formules atomiques sont des contraintes différentielles induites par le fragment DL de l'arithmétique de Presburger sans quantificateurs (voir par exemple [SB05]). Le fragment de l'arithmétique de Presburger dans CLTL(DL) est identique à celui de la logique $\mathcal{L}_p$ de [CC00] qui est un fragment de $\mathrm{CLTL}^1(\mathrm{DL})$. Contrairement à cette logique, il est possible dans CLTL(DL) d'exprimer des contraintes entre deux états non consécutifs du modèle. Les résultats de ce chapitre illustrent que ce langage est suffisamment expressif pour que des fragments fortement restreints de la logique soient indécidables. Le langage de contraintes DL étant lui même assez restrictif, il peut être vu comme une sorte de noyau qui cause l'indécidabilité.

Comme CLTL(DL) est assez expressif pour simuler les machines de Minsky [Min67], nous nous intéressons à des fragments de la logique dont le nombre de variables et la longueur temporelle des formules sont bornés à l'avance. Cependant nous ne faisons aucune restriction sur l'utilisation des opérateurs temporels, contrairement à certaines approches telles que [BEH95, CC00] concernant des fragments de Presburger LTL. Restreindre le nombre de variables des formules est une méthode souvent utilisée pour définir des fragments décidables de logiques indécidables ou des classes d'automates équipés de compteurs ou d'horloges dont le problème d'accessibilité d'un état de contrôle est décidable (voir par exemple [Hal95, ISD$^+$00, FS00, OW04, LW05]). Cette restriction permet aussi de mieux cerner les écarts de complexité dans le cas de problèmes décidables [DS02, LMS04]. Dans cette section notre principal objectif est d'identifier les différents fragments décidables et indécidables par rapport aux critères de restrictions ci-dessus, raffinant les résultats de [BEM97, CC00, DD07, DG05].

Nous définissons une classification complète des différents fragments restreignant le nombre de compteurs et la longueur temporelle des formules. Nous notons déjà que le problème de la satisfaisabilité pour CLTL(QFP) et ses fragments appartient à Σ_1^1 dans la hiérarchie analytique, puisque une formule de CLTL(QFP) peut se traduire en une formule arithmétique du premier ordre (car LTL est inclus dans la logique du premier ordre) et que la recherche d'un modèle se traduit par un ensemble de quantificateurs existentiels.[1]C'est aussi le cas du problème du model-checking qui se réduit au problème de la satisfaisabilité en utilisant les techniques de [SC85]. Les deux principaux résultat d'indécidabilité que nous montrons sont les suivants :

– les problèmes de la satisfaisabilité et du model-checking pour le fragment $\mathrm{CLTL}_2^1(\mathrm{DL})$ sont Σ_1^1-complets,
– les problèmes de la satisfaisabilité et du model-checking pour le fragment $\mathrm{CLTL}_1^2(\mathrm{DL})$ sont Σ_1^1-complets.

Ces résultats fixent la limite entre la décidabilité et l'indécidabilité pour cette famille de fragments et raffinent les résultats d'indécidabilité connus des fragments $\mathrm{CLTL}_3^1(\mathrm{DL})$ [CC00] et $\mathrm{CLTL}_2(\mathrm{DL})$ [DD07]. Pour compléter ces résultats nous prouvons aussi la PSPACE-complétude du fragment restant avec une variable et une longueur temporelle bornée à un. Afin de capturer le plus large fragment décidable possible, nous établissons en fait cette borne de complexité pour une variante de $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$ contenant des variables propositionnelles en plus des contraintes de différence et de périodicité au niveau atomique. L'ajout de variables propositionnelles nous permet par la même occasion de réduire le problème du model-checking au problème de la satisfaisabilité et donc de montrer la même borne de complexité pour le problème du model-checking de cette logique. Ces résultats sont démontrés

[1]Dans la hiérarchie analytique, Σ_1^1 correspond à la classe des formules de la forme $\exists X_1 \ldots \exists X_n \phi$ telles que ϕ est une formule de l'arithmétique du second ordre sans quantificateur d'ensemble. Le problème de la satisfaisabilité pour les formules de cette logique est indécidable.

en adaptant à nouveau la méthode par automate de [VW94]. Ainsi, cette méthode peut être généralisée à plusieurs extensions de logiques temporelles sur des domaines concrets générant des classes de modèles ω-régulières du fait de la séparation entre la partie logique temporelle et la partie domaine concret qui composent CLTL(DL$^+$). L'automate que nous construisons appartient à une classe particulière d'automates à compteurs dont nous montrons que le problème du vide est NLOGSPACE-complet.

Ces résultats contrastent avec la haute borne de complexité établie dans [CC00] pour le fragment plat de CLTL^1_ω(DL). En effet, la décidabilité de ce fragment qui restreint l'utilisation de l'opérateur U est obtenue par réduction à l'arithmétique de Presburger. De plus, nous obtenons comme corollaire que le model-checking des automates temporisés discrets à une horloge de [DPK03] est dans PSPACE, ce qui raffine considérablement les résultats d'indécidabilité de [DPK03, Sect. 6]. Enfin, ce résultat de décidabilité est optimal par rapport aux restrictions que nous considérons puisque la satisfaisabilité du fragment CLTL^1_1(QFP) est un problème indécidable. Cependant, nous montrons en utilisant le même genre de technique que pour prouver la décidabilité de CLTL^1_1(DL$^+$) que le model-checking de CLTL(QFP) est un problème PSPACE-complet lorsque l'on restreint les modèles à la classe des automates à un compteur dont les mises à jour sont dans $\mathbb{Z}$.

Bien que l'intérêt des résultats évoqués ci-dessus soit principalement théorique, nous insistons sur le fait que les automates à un compteur ont diverses applications telle que la vérification de protocoles cryptographiques [LLT05], la validation de flux XML [CR04] ou la définition de langages hors contexte [BB90]. L'intérêt de ce modèle est que, contrairement aux machines ayant plusieurs compteurs, les automates à un compteur ont plusieurs propriétés de comportement décidables (voir par exemple [Kuč00, JKMS04] ou [BHM03]). De plus, cette classe d'automates est équivalente à celle des automates à pile dont l'alphabet est un singleton, ce qui fait que les algorithmes pour les automates à pile peuvent s'appliquer aussi à ce modèle. Les problèmes de model-checking que nous considérons raffinent donc plusieurs résultats connus.

Ainsi le problème du model-checking des automates à un compteur avec le μ-calcul modal est dans PSPACE [Ser06], alors que le model-checking des automates à pile avec le μ-calcul modal est dans EXPTIME [Wal01] de même qu'avec le μ-calcul linéaire [BEM97]. Néanmoins, les formules atomiques dans ces travaux expriment uniquement des propriétés sur les états de contrôle. Dans [FWW97], une extension de CTL* avec des formules atomiques exprimant des propriétés régulières sur le contenu de la pile en plus de propriétés sur les états de contrôle est introduite. Le problème de model-checking pour cette logique est montré dans EXPTIME dans [FWW97] et l'EXPTIME-complétude est prouvée dans [EKS03]. Les contraintes de DL$^+$ sont aussi des contraintes régulières et les bornes de PSPACE-complétude que nous établissons pour nos problèmes de model-checking permettent donc de raffiner ces résultats. Notons néanmoins que dans le cas du model-checking des DL-automates, le modèle est plus expressif que les automates à pile avec une seule lettre puisqu'on autorise des increments modulo et des contraintes plus larges comme $\mathsf{X}x \leq x + 1$. De plus nous rappelons que bien que LTL s'exprime dans le μ-calcul modal, ces deux formalismes n'ont pas la même concision et donc les résultats de complexité ne peuvent pas toujours être transférés directement.

Les résultats présentés ici sont issus de l'article [DG07].

5.1.2 Sous-classes de DL-automates.

La classe des DL-automates englobe plusieurs classes connues d'automates à compteurs. En particulier, les DL-automates peuvent facilement simuler les machines de Minsky non déterministes. Nous faisons référence par la suite à plusieurs sous-classes de DL-automates particulières définies ci-dessous.

Un $\langle k, \mathbb{Z}\rangle$*-automate à compteurs* est un DL-automate tel que pour chacune des transitions $q \xrightarrow{\alpha} q'$ la contrainte α est une conjonction de la forme $\bigwedge_{i\in\{1\ldots k\}} \alpha_{test^i} \wedge \bigwedge_{i\in\{1\ldots k\}} \alpha_{maj^i}$ où pour tout $i \in \{1, \ldots, k\}$

- $\alpha_{test^i} \in \{\top, \neg x_i = 0, x_i = 0, x_i > 0, x_i < 0\}$
- $\alpha_{maj^i} \in \{\mathsf{X}x_i = x_i + u \mid u \in \{u_{\min}, \ldots, u_{\max}\}\}$.

De plus, la valeur initiale de tous les compteurs vaut zéro, ce qui signifie que si $q \xrightarrow{\alpha} q'$ et q est un état initial alors pour tout $i \in \{1, \ldots, k\}$ la contrainte α_{test^i} vaut $x_i = 0$. Pour faciliter la présentation, nous fixons un ordre arbitraire des variables et définissons un codage concis des ensembles de contraintes α_{test^i} et α_{maj^i}.

- Les éléments de $\{\top, \neg x_i = 0, x_i = 0, x_i > 0, x_i < 0\}$ sont codés par $\{\top, \neq, =, >, <\}$.
- Les éléments de $\{\mathsf{X}x_i = x_i+u \mid u \in \{u_{\min}, \ldots, u_{\max}\}\}$ sont codés par $\{u_{\min}, \ldots, u_{\max}\}$.

Ainsi, une transition telle que

$$q \xrightarrow{(\top\wedge x_2=0)\wedge(\mathsf{X}x_1=x_1\wedge\mathsf{X}x_2=x_2-1)} q'$$

sera notée

$$q \xrightarrow{(\top,=),(0,-1)} q'.$$

Un $\langle k, \mathbb{N}\rangle$*-automate* est défini de la même manière excepté que nous considérons uniquement des valeurs non-négatives pour les compteurs. Ceci revient à imposer sur chaque transition que $\bigwedge_{i\in\{1,\ldots,k\}} x_i \geq 0$ soit une partie de la contrainte. Lorsque le contexte fait explicitement référence aux $\mathbb{N}$-automates à compteurs par la suite, nous omettons cette contrainte supplémentaire sur les transitions.

Dans le reste de ce chapitre, nous considérons principalement des automates ayant au plus deux compteurs. Pour établir la borne PSPACE du problème de satisfaisabilité de $\mathrm{CLTL}^1_1(\mathrm{DL})$ nous adaptons la méthode par automate de [VW94] en utilisant un $\langle 1, \mathbb{Z}\rangle$-automate à compteur dont les mises à jour sont restreintes à $\{-1, 0, 1\}$. Nous appelons ces automates des $\langle 1, \mathbb{Z}\rangle$*-automates à compteur simples.* Ainsi, les DL-automates sont utilisés à la fois comme modèle opérationnel pour le problème du model-checking et comme générateur de langage dans notre procédure de décision (l'alphabet nécessaire sera introduit plus tard). Notons que l'existence d'une exécution acceptante pour un $\langle 1, \mathbb{Z}\rangle$-automate à compteur est une question plus compliquée que des questions similaires traitées dans [JKMS04, LLT05] puisque nous avons une condition d'acceptation de Büchi, le compteur prend ses valeurs dans $\mathbb{Z}$ et les tests autorisés sont des tests à zero et des tests de signe. Nous montrons dans la Section 5.4 que ce problème est NLOGSPACE-complet pour les $\langle 1, \mathbb{Z}\rangle$-automate simples. Nous procédons en analysant la forme des exécutions acceptantes. Ce résultat complète les résultats connus à propos des automates de Büchi et des variantes d'automates à un compteur [VW94, LLT05].

De plus, les $\langle 1, \mathbb{Z} \rangle$-automates forment une sous-classe stricte des DL-automates. En effet, la définition des DL-automates autorise des contraintes telles que $\mathsf{X}x > x$ or $\mathsf{X}x < x + d$ sur les transitions, qui ne peuvent être exprimées dans un $\langle 1, \mathbb{Z} \rangle$-automate. En ce qui concerne les $\langle 2, \mathbb{N} \rangle$-automates, le problème de l'existence d'une exécution acceptante est Σ_1^1-dur puisque on peut y réduire facilement le problème de l'exécution récurrente d'une machine de Minsky non-deterministe. Ce dernier problème est Σ_1^1-complet d'après le résultat de [AH94, Lemme 8].

5.1.3 Fragments indécidables de la logique

L'indécidabilité du problème de la satisfaisabilité pour le fragment $\mathrm{CLTL}_3^1(\mathrm{DL})$ est montrée dans [CC00] par réduction du problème de l'arrêt d'une machine de Minsky. Deux variables sont nécessaires pour coder les compteurs et une troisième pour l'état de contrôle. Cette réduction n'utilise que des contraintes de la forme $x = y + 1$ et $x = d$. Au contraire, le fragment plat de $\mathrm{CLTL}^1(\mathrm{DL})$ est décidable. Cependant, ce fragment restreint l'utilisation de l'opérateur U aux occurrences telles que la sous-formule de gauche est une combinaison booléenne de contraintes atomiques. De plus, la procédure de décision définie dans [CC00] a une complexité au moins aussi difficile que le problème de satisfaisabilité pour l'arithmétique de Presburger (2EXPTIME). Puisque DL est muni d'une relation d'égalité et d'un domaine d'interprétation infini, le Corollaire 2 peut s'appliquer et nous pouvons raffiner ces résultats.

Théorème 25 *Le problème de la satisfaisabilité pour* $\mathrm{CLTL}_1^\omega(\mathrm{DL})$ *est* Σ_1^1*-complet.*

Le résultat ci-dessus montre que borner le nombre de variables ne suffit donc pas pour définir un fragment décidable de CLTL(DL). Néanmoins, la réduction utilisée ne marche plus lorsque l'on borne la longueur temporelle des formules (voir preuve du Lemme 3). Le résultat suivant raffine la limite d'indécidabilité et montre qu'une seule variable et une longueur temporelle bornée à deux sont suffisantes pour prouver l'indécidabilité.

Théorème 26 *Le problème de la satisfaisabilité pour* $\mathrm{CLTL}_1^2(\mathrm{DL})$ *est* Σ_1^1*-complet.*

Preuve : Le problème de la satisfaisabilité de CLTL(DL) est dans Σ_1^1. Nous démontrons donc la Σ_1^1-dureté en réduisant l'existence d'une exécution acceptante pour un $\langle 2, \mathbb{N} \rangle$-automate à compteurs à la satisfaisabilité d'une formule de $\mathrm{CLTL}_1^2(\mathrm{DL})$. Ceci est suffisant puisque les $\langle 2, \mathbb{N} \rangle$-automates peuvent facilement simuler une machine de Minsky non deterministe dont le problème de l'exécution récurrente est Σ_1^1-complet.

La première étape consiste à montrer que pour tout $\langle 2, \mathbb{N} \rangle$-automate à compteurs $\mathcal{A}$ on peut construire en espace logarithmique en $|\mathcal{A}|$ un $\langle 2, \mathbb{N} \rangle$-automate à compteurs $\mathcal{A}'$ tel que $\mathcal{A}$ admet une exécution acceptante ssi $\mathcal{A}'$ admet une exécution acceptante et aucune transition de $\mathcal{A}'$ ne laisse tous les compteurs inchangés. Il nous faut donc éliminer les transitions de $\mathcal{A}'$ de la forme $q \xrightarrow{test^1, test^2, =, =} q'$. Soit $\mathcal{A} = \langle Q, \delta, I, F \rangle$ un $\langle 2, \mathbb{N} \rangle$-automate à compteurs, le $\langle 2, \mathbb{N} \rangle$-automate à compteurs $\mathcal{A}' = \langle Q', \delta', I', F' \rangle$ est tel que :

– $Q' = Q \uplus \{t \in \delta : t = q \xrightarrow{test^1, test^2, =, =} q'\}$,

– $I' = I$ et $F' = F$,

– δ' est défini à partir de δ en replaçant chaque transition de la forme $t = q \xrightarrow{test^1, test^2, =, =} q'$ par les transitions $q \xrightarrow{test^1, test^2, +1, =} t$ et $t \xrightarrow{\top, \top, -1, =} q'$.

Nous montrons maintenant qu'un $\langle 2, \mathbb{N}\rangle$-automate à compteurs $\mathcal{A} = \langle Q, \delta, I, F\rangle$ tel que chaque transition modifie au moins un compteur peut être simulé par une formule de $\mathrm{CLTL}^2_1(\mathrm{DL})$. Nous posons $Q = \{q_1, \ldots, q_n\}$, $I = \{q_{a_1}, \ldots, q_{a_m}\}$ et $F = \{q_{f_1}, \ldots, q_{f_{m'}}\}$.

Une configuration de $\mathcal{A}$ de la forme $\langle q_i, c_1, c_2\rangle$ est codée par une sous-séquence du modèle de $2i$ états consécutifs $c_1 \cdot c_1 + c_2 + 1 \cdots c_1 \cdot c_1 + c_2 + 1$ composée de i répétitions de la paire $c_1, c_1 + c_2 + 1$. Rappelons qu'un modèle de $\mathrm{CLTL}^2_1(\mathrm{DL})$ est une séquence infinie d'entiers car les formules de ce fragment n'ont qu'une seule variable. Une nouvelle configuration est détectée lorsque quatre valeurs consécutives $c \cdot d \cdot c' \cdot d'$ vérifient $c \neq c'$ ou $d \neq d'$, ce qui signifie que la valeur de l'un des compteurs change. Notons aussi que ce codage permet de différencier les positions paires et impaires puisqu'à tout moment on a $c_1 < c_1 + c_2 + 1$. Pour savoir si la valeur du modèle σ à la position i code c_1, il suffit de tester si $\sigma, i \models x < \mathsf{X}x$ où x est l'unique variable de la formule. Les formules suivantes nous permettent d'exprimer les différentes conditions permettant de coder une exécution de $\mathcal{A}$.

– La formule ϕ_{ch} exprime que l'on change de configuration. Ceci se traduit par le fait que l'un des compteurs change.

$$\phi_{ch} = x < \mathsf{X}x \ \wedge \ (x \neq \mathsf{X}^2 x \ \vee \ \mathsf{X}(x \neq \mathsf{X}^2 x))$$

– La formule Suiv_i exprime que l'on se trouve à une position du modèle où l'état de contrôle va changer pour devenir q_i. Ceci se traduit par le fait que l'un des compteurs change et que i répétitions de la même paire de valeurs suivent avant que l'un des compteurs change à nouveau.

$$\mathrm{Suiv}_i \stackrel{\text{def}}{=} \phi_{ch} \ \wedge \ \mathsf{X}^2(\bigwedge_{0 \leq j < i-1} \mathsf{X}^{2j}(x = \mathsf{X}^2 x \ \wedge \ \mathsf{X}(x = \mathsf{X}^2 x)) \ \wedge \ \mathsf{X}^{2(i-1)}\phi_{ch})$$

– La formule suivante code les conditions initiales.

$$\phi_{init} \stackrel{\text{def}}{=} x_1 = 0 \ \wedge \ x_2 = 1 \ \wedge \bigvee_{1 \leq i \leq m} (\bigwedge_{0 \leq j < a_i - 1} \mathsf{X}^{2j}(x = \mathsf{X}^2 x \ \wedge \ \mathsf{X}(x = \mathsf{X}^2 x))$$
$$\wedge \bigvee_{\langle q_{a_i}, \phi, q_{j'}\rangle \in \delta} \mathsf{X}^{2(a_i - 1)}(\phi' \ \wedge \ \mathrm{Suiv}_{j'}))$$

– La formule ϕ_{rec} exprime l'existence d'un élément récurrent de F :

$$\phi_{rec} \stackrel{\text{def}}{=} \bigvee_{1 \leq i \leq m'} \mathsf{GF}\mathrm{Suiv}_{f_i}.$$

– Enfin la formule suivante simule une exécution :

$$\phi_{run} \stackrel{\text{def}}{=} \mathsf{G} \bigwedge_{1 \leq i \leq n} (\mathrm{Suiv}_i \Rightarrow \bigvee_{\langle q_i, \phi, q_j\rangle \in \delta} \mathsf{X}^{2i}(\phi' \ \wedge \ \mathrm{Suiv}_j))$$

où ϕ' est obtenu à partir de ϕ

- en remplaçant $x_1 = 0$ par $x = 0$,
- en remplaçant $x_2 = 0$ par $\mathsf{X}x = x + 1$,
- en remplaçant $\mathsf{X}x_1 = x_1 + d_1$ par $\mathsf{X}^2 x = x + d_1$ pour tout $d_1 \in \{-1, 0, 1\}$,
- en remplaçant, pour tout $d_2 \in \{-1, 0, 1\}$, la contrainte $\mathsf{X}x_2 = x_2 + d_2$ par

$$\bigwedge_{d_1 \in \{-1,0,1\}} (\mathsf{X}^2 x = x + d_1) \Rightarrow \mathsf{X}(\mathsf{X}^2 x = x + (d_1 + d_2)).$$

Il est maintenant facile de montrer que cette construction est telle que $\mathcal{A}$ admet une exécution acceptante ssi $\phi_{init} \wedge \phi_{run} \wedge \phi_{rec}$ est satisfaisable. □

Nous pouvons aussi raffiner le résultat de Σ_1^1-dureté de $\mathrm{CLTL}_3^1(\mathrm{DL})$ établi dans [CC00] en utilisant le Théorème 26 et la réduction énoncée dans le Lemme 2.

Corollaire 7 *Le problème de la satisfaisabilité pour* $\mathrm{CLTL}_2^1(\mathrm{DL})$ *est* Σ_1^1*-complet.*

De plus, le problème de la satisfaisabilité peut facilement se réduire au problème du model-checking. En effet, soit $\mathcal{A}_\top$ le DL-automate avec un seul état q ayant pour relation de transition la seul règle $q \xrightarrow{\top} q$. N'importe quel modèle est donc accepté par $\mathcal{A}_\top$. Il est alors évident qu'une formule ϕ de CLTL(DL) est satisfaisable ssi l'automate $\mathcal{A}_\top$ satisfait ϕ. Nous obtenons donc le corollaire suivant.

Corollaire 8 *Le problème du model-checking pour les fragments* $\mathrm{CLTL}_1^2(\mathrm{DL})$ *et* $\mathrm{CLTL}_2^1(\mathrm{DL})$ *sont* Σ_1^1*-complets.*

La borne supérieure Σ_1^1 est obtenue par réduction du problème du model-checking à la satisfaisabilité de CLTL(DL) en utilisant la méthode de [SC85]. En inspectant les preuves du Théorème 26 et du Lemme 2, nous remarquons que la Σ_1^1-dureté peut aussi être établie pour les problèmes de satisfaisabilité et model-checking de $\mathrm{CLTL}_1^2(\mathrm{DL})$ et $\mathrm{CLTL}_2^1(\mathrm{DL})$ lorsque l'on restreint la logique en remplaçant l'opérateur U par F.

5.2 Extension décidable de $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$

Il nous reste à déterminer le statut du fragment de CLTL(DL) avec une seule variable et une longueur temporelle de un. C'est ce que nous faisons en prouvant dans cette section la PSPACE-complétude du problème de la satisfaisabilité pour ce fragment. En fait nous établissons cette borne pour un langage logique plus riche qui étend $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$ avec des variables propositionnelles. Notons que l'ajout de variables propositionnelles peut être vu comme l'introduction de variables supplémentaires prenant leurs valeurs dans un sous-ensemble fini de $\mathbb{Z}$. L'enrichissement de $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$ avec des variables propositionnelles nous permet du même coup de déduire la PSPACE-complétude du problème de model-checking de $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$ par réduction du problème de satisfaisabilité de $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$ avec variables propositionnelles.

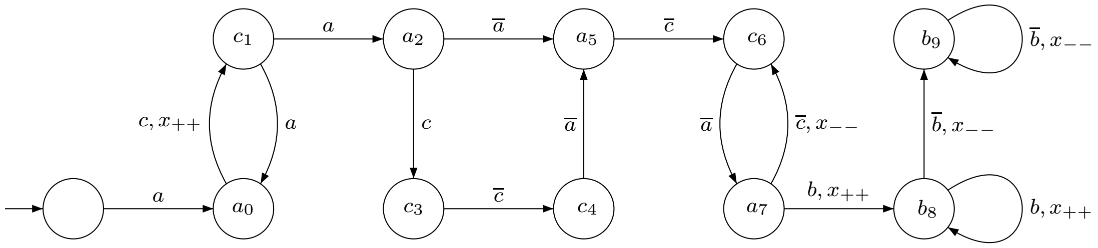


Figure 5.1: Représentation du langage $\{(ac)^n a(\varepsilon|c\overline{c})\overline{a}(\overline{ca})^n b^m \overline{b}^m : n, m \geq 1\}$

En plus de compléter notre classification des fragments de CLTL(DL), ce résultat est intéressant car différents problèmes impliquant des automates à un compteur peuvent se coder dans $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$. Ces problèmes concernent des applications variées telles que la vérification de protocoles cryptographiques [LLT05], la validation de flux XML [CR04] qui sont des représentations de documents XML sous la forme de chaîne de caractères, ou encore dans l'apprentissage automatique de langages [WTT04]. Par exemple, la classe des DL-automates à une variable contient la classe d'automates à un compteur utilisés dans [CR04, Sect.5] pour valider les flux XML par rapport à une spécification donnée. Nous représentons dans la Figure 5.1 l'automate à un compteur reconnaissant le langage $\{(ac)^n a(\varepsilon|c\overline{c})\overline{a}(\overline{ca})^n b^m \overline{b}^m : n, m \geq 1\}$. Dans cette représentation, x est un compteur et l'alphabet est partitionné en un ensemble de balises ouvrantes $\{a, b, c\}$ et un ensemble de balises fermantes $\{\overline{a}, \overline{b}, \overline{c}\}$. L'abréviation x_{++} signifie $\mathsf{X}x = x + 1$, x_{--} signifie $x = \mathsf{X}x - 1$ et les tests à zéro sont omis. Tester si un mot appartient au langage reconnu par cet automate est un problème clé de [CR04, Sect.5] qui peut s'exprimer dans notre formalisme. Le langage $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$ permet aussi d'exprimer des propriétés plus riches telles que des propriétés de sûreté comme $\mathsf{G}(x < 2^m)$ ou de vivacité comme $\mathsf{G}(x \equiv_{2^n} 0 \Rightarrow \mathsf{F}(x \equiv_{3^m} 1))$. Nous rappelons aussi que les DL-automates sont strictement plus expressifs que les automates à un compteur classiques puisque les transitions ne sont pas limitées aux seules opérations d'incrémentation et de décrémentation. Le résultat de cette section raffine aussi la borne de complexité établie dans [FWW97] pour le model-checking de propriétés régulières sur les automates à pile car un automate à un compteur est un cas particulier d'automate à pile dont l'alphabet est un singleton.

5.2.1 Ajout de variables propositionnelles

Soit $\mathrm{PROP} = \{p_1, p_2, \ldots\}$ un ensemble dénombrable infini de variables propositionnelles. Nous définissons la logique $\mathrm{CLTL}(\mathrm{DL}^+, \mathrm{PROP})$ comme l'extension de LTL avec des contraintes atomiques de DL^+ et des propositions. Les modèles de $\mathrm{CLTL}(\mathrm{DL}^+, \mathrm{PROP})$ sont donc des paires de séquences infinies de la forme $\langle \sigma_1, \sigma_2 \rangle$ telles que $\sigma_1 : \mathbb{N} \rightarrow 2^{\mathrm{PROP}}$ est un modèle standard de LTL et $\sigma_2 : \mathbb{N} \rightarrow (\mathrm{VAR} \rightarrow \mathbb{Z})$ est un modèle de $\mathrm{CLTL}(\mathrm{DL}^+)$. La relation de satisfaction pour $\mathrm{CLTL}(\mathrm{DL}^+, \mathrm{PROP})$ est presque identique à celle de $\mathrm{CLTL}(\mathrm{DL}^+)$ excepté au niveau atomique.

- Pour toute variable propositionnelle $p \in \mathrm{PROP}$ on a $\langle \sigma_1, \sigma_2 \rangle, i \models p$ ssi $p \in \sigma_1(i)$.

– Pour toute contrainte atomique α de CLTL(DL$^+$) on a $\langle \sigma_1, \sigma_2 \rangle, i \models \alpha$ ssi $\sigma_2, i \models \alpha$ par rapport à la relation de satisfaction de CLTL(DL$^+$).

Nous ne faisons pas de restriction sur les variables propositionnelles pour les fragments de la forme $\mathrm{CLTL}^l_k(\mathrm{DL}^+, \mathrm{PROP})$. Ainsi, la présence de variables propositionnelles permet de réduire le problème du model-checking de CLTL(DL$^+$) au problème de satisfaisabilité de $\mathrm{CLTL}(\mathrm{DL}^+, \mathrm{PROP})$.

Lemme 19 *Le problème du model-checking pour* CLTL(DL$^+$) *se réduit en espace logarithmique au problème de la satisfaisabilité de* $\mathrm{CLTL}(\mathrm{DL}^+, \mathrm{PROP})$.

Preuve : La preuve de ce résultat est similaire à celle de la réduction du problème du model-checking au problème de la satisfaisabilité pour la logique LTL [SC85]. Soit ϕ une formule de CLTL(DL$^+$) et $\mathcal{A} = \langle Q, I, F, \delta \rangle$ un DL$^+$-automate tel que $\delta \subseteq Q \times 1SC_k(\mathrm{DL}^+) \times Q$. Nous décrivons la construction d'une formule $\phi_{\mathcal{A}}$ de $\mathrm{CLTL}(\mathrm{DL}^+, \mathrm{PROP})$ telle que $\mathcal{A} \models \phi$ ssi la formule $\phi \wedge \phi_{\mathcal{A}}$ est satisfaisable.

Nous introduisons pour chaque état de contrôle $q_i \in Q$ de $\mathcal{A}$ une variable propositionnelle p_i. La formule suivante exprime qu'il n'existe qu'une unique variable propositionnelle vraie à chaque position du modèle, donc que l'automate est dans un unique état de contrôle.

$$\phi_{uni} \stackrel{\text{def}}{=} \bigvee_{i \in \{1,\ldots,|Q|\}} (p_i \wedge \bigwedge_{j \in \{1,\ldots,|Q|\} \setminus \{i\}} \neg p_j).$$

La relation de transition de $\mathcal{A}$ est exprimée par la formule suivante.

$$\phi_\delta \stackrel{\text{def}}{=} \bigwedge_{i \in \{1,\ldots,|Q|\}} (p_i \Rightarrow \bigvee_{\langle q_i, \alpha, q_j \rangle \in \delta} (\alpha \wedge \mathsf{X} p_j)).$$

Enfin, les conditions initiales et d'acceptation sont codées respectivement par

$$\phi_{init} \stackrel{\text{def}}{=} \bigvee_{q_i \in I} p_i,$$

$$\phi_{acc} \stackrel{\text{def}}{=} \mathsf{GF}(\bigvee_{q_i \in F} p_i).$$

Ceci nous donne donc que la formule $\phi_{\mathcal{A}}$ est égale à

$$\phi_{\mathcal{A}} \stackrel{\text{def}}{=} \phi_{init} \wedge \phi_{acc} \wedge \mathsf{G}(\phi_{uni} \wedge \phi_\delta).$$

□

En analysant cette preuve, nous pouvons déduire le même résultat concernant le model-checking du fragment $\mathrm{CLTL}^1_1(\mathrm{DL}^+)$. En effet la formule $\phi_{\mathcal{A}}$ que nous construisons pour simuler le comportement de l'automate $\mathcal{A}$ appartient à $\mathrm{CLTL}^1_1(\mathrm{DL}^+, \mathrm{PROP})$ puisque les gardes d'un DL$^+$-automate à une variable sont des formules atomiques de $\mathrm{CLTL}^1_1(\mathrm{DL}^+)$ qui expriment des contraintes entre la valeur de la variable à l'état courant et sa valeur à l'état suivant. Donc si la formule ϕ à vérifier appartient à $\mathrm{CLTL}^1_1(\mathrm{DL}^+)$, la conjonction $\phi \wedge \phi_{\mathcal{A}}$ est une formule de $\mathrm{CLTL}^1_1(\mathrm{DL}^+, \mathrm{PROP})$.

Corollaire 9 *Le problème du model-checking pour* $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$ *se réduit en espace logarithmique au problème de la satisfaisabilité de* $\mathrm{CLTL}_1^1(\mathrm{DL}^+, \mathrm{PROP})$.

Dans la suite, nous nous intéressons donc à la logique $\mathrm{CLTL}_1^1(\mathrm{DL}^+, \mathrm{PROP})$.

5.2.2 Représentation Symbolique des Modèles

Dans le reste de cette section, nous montrons la décidabilité du problème de la satisfaisabilité pour la logique $\mathrm{CLTL}_1^1(\mathrm{DL}^+, \mathrm{PROP})$. Sans perte de généralité, nous supposons à partir de maintenant que toutes les formules atomiques qui comparent x à $\mathsf{X}x$ sont de la forme $\mathsf{X}x = x + d$ ou $\mathsf{X}x \equiv_k x + c$ avec $d \in \mathbb{Z}$ et $k, c \in \mathbb{N}$.

Afin de construire un automate qui reconnaît des représentations symboliques des modèles d'une formule de $\mathrm{CLTL}_1^1(\mathrm{DL}^+, \mathrm{PROP})$, nous commençons par introduire une représentation symbolique des valuations. Étant donné un ensemble fini X de contraintes atomiques de $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$, nous considérons les ensembles suivants :

- $C_{\mathrm{loc}} = \{d_{\min}, \ldots, d_{-1}, d_0, d_1, \ldots, d_{\max}\}$ est l'ensemble des constantes qui apparaissent dans X dans des contraintes de la forme $x \sim d$ ou $\mathsf{X}x \sim d$. Nous supposons que $d_{\min} < \cdots < d_{-1} < d_0 < d_1 < \cdots < d_{\max}$.
- $C_{\mathrm{step}} = \{e_{\min'}, \ldots, e_{-1}, e_0, e_1, \ldots, e_{\max'}\}$ est l'ensemble des constantes qui apparaissent dans X dans des contraintes de la forme $\mathsf{X}x \sim x + e$. Nous supposons que $e_{\min'} < \cdots < e_{-1} < e_0 < e_1 < \cdots < e_{\max'}$.
- K est le plus petit commun multiple de l'ensemble des entiers k tels qu'une relation de la forme $\equiv_k$ apparaît dans X.

Sans perte de généralité, nous pouvons considérer que $d_0 = e_0 = 0$, $d_{\max}, e_{\max'} \geq 0$ et $d_{\min}, e_{\min'} \leq 0$. Ces différents ensembles constituent les ressources syntaxiques de X. Une contrainte est construite par rapport aux ressources syntaxiques de x ssi

- les contraintes de la forme $x \sim d$ et $\mathsf{X}x \sim d$ sont telles que $d \in C_{\mathrm{loc}}$,
- les contraintes de la forme $\mathsf{X}x \sim x + e$ sont telles que $e \in C_{\mathrm{step}}$
- les contraintes de périodicité utilisent des relations de la forme $\equiv_k$ telles que k divise K.

Nous définissons le même genre d'abstraction que dans le chapitre précédent pour les valuations. Une valuation $\{x, \mathsf{X}x\} \to \mathbb{Z}$, qui peut aussi être vue comme une paire $\langle z_1, z_2 \rangle \in \mathbb{Z}^2$, est abstraite par un ensemble de contraintes $sv = \langle \alpha_x, \theta, \alpha'_x, \theta', \alpha_s \rangle \in \mathrm{C}_x \times \mathrm{Mod}_x \times \mathrm{C}_{\mathsf{X}x} \times \mathrm{Mod}_{\mathsf{X}x} \times \mathrm{C}_{\mathrm{step}}$ tel que

- C_x est composé des contraintes de la forme
  - $d_i < x \wedge x < d_{i+1}$ pour $i \in \{\min, \ldots, \max - 1\}$,
  - $x = d_i$ pour $i \in \{\min, \ldots, \max\}$,
  - $x < d_{\min}$ et $d_{\max} < x$.
- $\mathrm{C}_{\mathsf{X}x}$ est composé des contraintes de la forme
  - $d_i < \mathsf{X}x \wedge \mathsf{X}x < d_{i+1}$ pour $i \in \{\min, \ldots, \max - 1\}$,
  - $\mathsf{X}x = d_i$ pour $i \in \{\min, \ldots, \max\}$,
  - $\mathsf{X}x < d_{\min}$ et $d_{\max} < \mathsf{X}x$.

- Mod_x est composé des contraintes de la forme $x \equiv_K i$ pour $i \in \{0, \ldots, K-1\}$,
- $\mathrm{Mod}_{\mathsf{X}x}$ est composé des contraintes de la forme $\mathsf{X}x \equiv_K i$ pour $i \in \{0, \ldots, K-1\}$,
- $\mathrm{C}_{\mathrm{step}}$ est composé des contraintes de la forme
  - $x + e_i < \mathsf{X}x \wedge \mathsf{X}x < x + e_{i+1}$ pour $i \in \{\min', \ldots, \max' - 1\}$,
  - $\mathsf{X}x = x + e_i$ pour $i \in \{\min', \ldots, \max'\}$,
  - $\mathsf{X}x < x + e_{\min'}$ et $x + e_{\max'} < \mathsf{X}x$.

Nous appelons sv une valuation symbolique et nous réutilisons la notation $\mathrm{SV}(X)$ pour noter l'ensemble des valuations symboliques construites par rapport à X. Par extension, nous notons $\mathrm{SV}(\phi)$ l'ensemble des valuations symboliques construites par rapport à l'ensemble des contraintes atomiques d'une formule ϕ de $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$. La taille de $\mathrm{SV}(\phi)$ est exponentielle par rapport à $|\phi|$ et chaque élément de cet ensemble peut être codé en espace polynomial par rapport à $|\phi|$. Étant données une valuation $v : \{x, \mathsf{X}x\} \to \mathbb{Z}$ et une valuation symbolique sv, nous notons $v \models sv$ ssi v satisfait toutes les contraintes de sv. Le résultat suivant établit la correction de cette abstraction.

Lemme 20 *Soit X un ensemble fini de contraintes atomiques de* $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$.

(I) *Pour toute valuation $v : \{x, \mathsf{X}x\} \to \mathbb{Z}$ il existe une unique valuation symbolique notée $sv(v) \in \mathrm{SV}(X)$ telle que $v \models sv(v)$.*

(II) *Soit $v, v' : \{x, \mathsf{X}x\} \to \mathbb{Z}$ deux valuations telles que $sv(v) = sv(v')$. Pour toute contrainte atomique α de* $\mathrm{CLTL}(\mathrm{DL}^+)$ *construite par rapport aux ressources syntaxiques de X on a $v \models \alpha$ ssi $v' \models \alpha$.*

Preuve : **(I)** La preuve de ce résultat est la même que celle du Lemme 10 **(I)**. Soit $sv \in \mathrm{SV}(X)$ une valuation symbolique et Val_{sv} l'ensemble des valuations $\{x, \mathsf{X}x\} \to \mathbb{Z}$, notées sous la forme de paires $\langle z_x, z_{\mathsf{X}x} \rangle \in \mathbb{Z}^2$, telles que $\langle z_x, z_{\mathsf{X}x} \rangle \models sv$. Il est évident que l'ensemble $\{\mathrm{Val}_{sv} \mid sv \in \mathrm{SV}(X), \mathrm{Val}_{sv} \neq \emptyset\}$ est une partition de l'ensemble $\mathbb{Z}^2$.

(II) Soient v et v' deux valuations telles que $sv(v) = sv(v') = \langle \alpha_x, \theta, \alpha'_x, \theta', \alpha_s \rangle$ et supposons que $v \models \alpha$. Nous procédons par induction sur la structure de α :

- Si α est de la forme $x = d$ alors puisque nous supposons que α est construite par rapport aux ressources syntaxiques de X on a $d \in C_{\mathrm{loc}}$ et donc $x = d$ est une contrainte de C_x par construction de cet ensemble. Par conséquent la contrainte α_x de sv doit être égale à α car c'est la seule contrainte de C_x qui peut vérifier $v \models \alpha$ et $v \models \alpha_x$. Comme $v' \models sv$, on a $v' \models \alpha_x$ et donc $v' \models \alpha$.
- Supposons que α soit de la forme $x < d$. Plusieurs cas se présentent car d'après la construction de C_x plusieurs contraintes α_x permettent d'avoir $v \models \alpha$ et $v \models \alpha_x$:
  - $\alpha_x = (x < d_{\min})$,
  - $\alpha_x = (x = d')$ tel que $d' < d$,
  - $\alpha_x = (d' < x \wedge x < d'')$ tel que $d'' \leq d$.

  Dans n'importe lequel de ces cas l'implication $\alpha_x \Rightarrow (x < d)$ est valide. Donc, comme $v' \models \alpha_x$ nous obtenons $v' \models \alpha$. Le cas $d < x$ est symétrique.
- Pour les cas où α est de la forme $\mathsf{X}x \sim d$ et $\mathsf{X}x \sim x + e$, le traitement est similaire aux deux points précédents en considérant respectivement les ensembles $\mathrm{C}_{\mathsf{X}x}$ et $\mathrm{C}_{\mathrm{step}}$

ainsi que les contraintes correspondantes dans sv.

- Si α est de la forme $x \equiv_k c$ alors k est un diviseur de K car α est construite par rapport aux ressources syntaxiques de X. Nous posons $\theta = (x \equiv_K c')$ dans sv. Comme $v \models \alpha$ et $v \models sv$, ceci implique qu'il existe $i, i' \in \mathbb{Z}$ tels que $v(x) = c + ik$ et $v(x) = c' + i'K$. Comme k divise K, on peut en déduire que $c' \equiv_k c$. Puisque $v' \models \theta$ on a $v'(x) = c' + jK$ pour un $j \in \mathbb{Z}$ qui peut se transformer en $v'(x) = (c + j_1 k) + j_2 k$ $(j_1, j_2 \in \mathbb{Z})$ en considérant que $c' \equiv_k c$ et k divise K. Donc on a bien $v' \models \alpha$.
- Les cas tels que α est de la forme $\mathsf{X}x \equiv_k c$ ou $\mathsf{X}x \equiv_k x + c$ peuvent être traités de la même manière que le cas précédent en considérant respectivement la contrainte θ' de sv et le couple de contraintes θ, θ'.
- Enfin, pour l'étape d'induction, nous supposons que v et v' vérifient la propriété pour deux contraintes α et α'.
  - Si $v \models \alpha \wedge \alpha'$ alors $v \models \alpha$ et $v \models \alpha'$. Par hypothèse d'induction on a donc $v' \models \alpha$ et $v' \models \alpha'$ ce qui implique $v' \models \alpha \wedge \alpha'$.
  - Si $v \models \neg\alpha$ alors $v \not\models \alpha$ et $v \models \alpha'$. Par hypothèse d'induction on a donc $v' \not\models \alpha$ ce qui implique $v' \models \neg\alpha$.

$\square$

Étant donnée une formule atomique α de $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$, nous notons $sv \models_{\mathrm{symb}} \alpha$ ssi pour toute valuation v telle que $sv(v) = sv$ on a $v \models \alpha$. Nous considérons maintenant des séquences de valuations symboliques par rapport à une formule ϕ de $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$. Une telle séquence $\rho : \mathbb{N} \to \mathrm{SV}(\phi)$ est satisfaisable ssi il existe un modèle $\sigma : \mathbb{N} \to \mathbb{Z}$ de $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$ tel que pour tout $i \in \mathbb{N}$ on a $\sigma, i \models \rho(i)$. Dans ce cas, nous notons $\sigma \models \rho$. Un modèle symbolique par rapport à une formule ϕ de $\mathrm{CLTL}_1^1(\mathrm{DL}^+, \mathrm{PROP})$ est une paire $\langle \sigma_1, \rho \rangle$ telle que $\sigma_1 : \mathbb{N} \to 2^{\mathrm{PROP}}$ et $\rho : \mathbb{N} \to \mathrm{SV}(\phi)$. La relation de satisfaction symbolique est étendue à ces modèles symboliques en modifiant la relation de satisfaction de $\mathrm{CLTL}_1^1(\mathrm{DL}^+, \mathrm{PROP})$ pour le cas des contraintes atomiques de $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$: $\langle \sigma, \rho \rangle, i \models_{\mathrm{symb}} \alpha$ ssi $\rho(i) \models_{\mathrm{symb}} \alpha$. Le reste de cette relation est défini de la même manière que pour la relation de satisfaction de $\mathrm{CLTL}_1^1(\mathrm{DL}^+, \mathrm{PROP})$. Nous établissons maintenant la correspondance entre les modèles symboliques et les modèles de $\mathrm{CLTL}_1^1(\mathrm{DL}^+, \mathrm{PROP})$.

Lemme 21 *Une formule ϕ de $\mathrm{CLTL}_1^1(\mathrm{DL}^+, \mathrm{PROP})$ est satisfaisable ssi il existe un modèle symbolique $\langle \sigma_1, \rho \rangle$ tel que $\langle \sigma_1, \rho \rangle \models_{\mathrm{symb}} \phi$ et un modèle σ_2 de $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$ tels que $\sigma_2 \models \rho$.*

Preuve : Si une formule ϕ est satisfaisable alors il existe un modèle $\langle \sigma_1, \sigma_2 \rangle$ tel que $\langle \sigma_1, \sigma_2 \rangle \models \phi$. Soit $\rho : \mathbb{N} \to \mathrm{SV}(\phi)$ la séquence de valuations symboliques telle que pour tout $i \in \mathbb{N}$ on a $\rho(i) = sv(\langle \sigma_2(i), \sigma_2(i+1) \rangle)$ où $\langle \sigma_2(i), \sigma_2(i+1) \rangle$ est vu comme la valuation qui associe la valeur de $\sigma_2(i)$ à x et celle de $\sigma_2(i+1)$ à $\mathsf{X}x$. Par construction il est évident que $\sigma_2 \models \rho$. De plus d'après le Lemme 20 **(II)**, nous avons pour tout v tel que $sv(v) = \rho(i)$ et pour toute contrainte atomique α de ϕ, $v \models \alpha$ ssi $\sigma_2, i \models \alpha$. Ceci nous permet de prouver que $\sigma_2, i \models \alpha$ ssi $\rho(i) \models_{\mathrm{symb}} \alpha$, par définition de la relation de satisfaction symbolique. Puisque la relation de satisfaction symbolique ne diffère de la relation de $\mathrm{CLTL}_1^1(\mathrm{DL}^+, \mathrm{PROP})$ que pour le cas des contraintes atomiques héritées de $\mathrm{CLTL}^1_1(\mathrm{DL}^+)$, nous pouvons conclure que $\langle\sigma_1, \rho\rangle \models_{\mathrm{symb}} \phi$.

Réciproquement, supposons qu'un modèle symbolique $\langle\sigma_1, \rho\rangle$ satisfasse symboliquement ϕ et qu'il existe un modèle σ_2 tel que $\sigma_2 \models \rho$. Puisque $\sigma_2 \models \rho$, pour tout $i \in \mathbb{N}$ on a $\sigma_2, i \models \rho(i)$ ce qui signifie que $\langle\sigma(i), \sigma(i+1)\rangle \models \rho(i)$ puisque les éléments de $\mathrm{SV}(\phi)$ contiennent des contraintes qui réfèrent à la valeur de l'état courant et de l'état suivant uniquement. Par définition de la relation $\models_{\mathrm{symb}}$, ceci implique que pour tout $i \in \mathbb{N}$ et toute contrainte atomique α de ϕ on a $\rho(i) \models_{\mathrm{symb}} \alpha$ ssi $\sigma, i \models \alpha$. Nous pouvons alors déduire que $\langle\sigma_1, \sigma_2\rangle \models \phi$ puisque les autres cas de la relation de satisfaction sont identiques à la relation de satisfaction symbolique. □

5.2.3 Approche par automate

Nous montrons maintenant comment à partir d'une formule ϕ de $\mathrm{CLTL}^1_1(\mathrm{DL}^+, \mathrm{PROP})$ nous pouvons construire un automate $\mathcal{A}_\phi$ reconnaissant les modèles symboliques qui satisfont les conditions requises par le Lemme 21. Pour définir cet automate, nous avons besoin d'étendre la définition des $\langle 1, \mathbb{Z}\rangle$-automates à compteur avec un alphabet Σ et des ε-transitions. Les transitions de cette classe d'automates sont donc étiquetées par des éléments de $\Sigma \cup \{\varepsilon\}$. Les autres conditions sur les transitions sont conservées. Puisque cette modification est motivée par le besoin de reconnaître un langage, nous définissons le langage accepté par un $\langle 1, \mathbb{Z}\rangle$-automate $\mathcal{A}$ étendu. Les langages $\mathrm{L}_\varepsilon(\mathcal{A})$ et $\mathrm{L}(\mathcal{A})$ sont définis de la manière suivante.

- $\mathrm{L}_\varepsilon(\mathcal{A}) = \{w : \mathbb{N} \to (\Sigma \cup \{\varepsilon\}) \mid$ il existe une exécution acceptante $\pi : \mathbb{N} \to (Q \times \mathbb{Z})$ telle que $\forall i\ \pi(i) \xrightarrow{w(i), \alpha} \pi(i+1)\}$
- $\mathrm{L}(\mathcal{A}) = \{w^{\setminus\varepsilon} \mid w \in \mathrm{L}'(\mathcal{A})$ et $|w^{\setminus\varepsilon}| = \omega\}$, où $w^{\setminus\varepsilon}$ est obtenu à partir de w en effaçant les occurrences de ε.

Nous précisons que la condition $|w^{\setminus\varepsilon}| = \omega$ de $\mathrm{L}(\mathcal{A})$ signifie en fait qu'il existe un nombre infini d'éléments de Σ dans w. L'ensemble $\mathrm{L}(\mathcal{A})$ est le langage reconnu par $\mathcal{A}$ (composé donc de séquences infinies).

D'après le résultat du Lemme 21, nous construisons l'automate $\mathcal{A}_\phi$ comme l'intersection de deux automates $\mathcal{A}_{\mathrm{symb}}$ et $\mathcal{A}_{\mathrm{sat}}$ tels que $\mathrm{L}(\mathcal{A}_{\mathrm{symb}})$ est l'ensemble des modèles symboliques qui satisfont symboliquement ϕ et $\mathrm{L}(\mathcal{A}_{\mathrm{sat}})$ l'ensemble des modèles symboliques $\langle\sigma_1, \rho\rangle$ tels que ρ est satisfaisable. Nous définissons ces deux automates sous la forme de $\langle 1, \mathbb{Z}\rangle$-automates à compteur simples étendus avec l'alphabet $2^{\mathrm{PROP}} \times \mathrm{SV}(\phi)$ et des ε-transitions. Néanmoins l'automate $\mathcal{A}_{\mathrm{symb}}$ est essentiellement un automate Büchi classique car il ne comporte pas d'ε-transitions et les compteurs ne sont pas vraiment nécessaires dans sa construction.

L'automate $\mathcal{A}_{\mathrm{symb}}$ est à nouveau une adaptation de la construction pour LTL de [VW94] que nous modifions au niveau du traitement des formules atomiques. Nous définissons la clôture $cl(\phi)$ de ϕ en considérant à la fois les contraintes atomiques issues de $\mathrm{CLTL}^1_1(\mathrm{DL}^+)$ et variables propositionnelles comme des formules atomiques. Un atome de ϕ est toujours défini comme étant un sous-ensemble maximalement cohérent de $cl(\phi)$. Nous construisons

$\mathcal{A}'_{\text{symb}} = \langle Q, I, F, \delta \rangle$ l'automate de Büchi généralisé sur l'alphabet $2^{\text{PROP}} \times \text{SV}(\phi)$ tel que :

- Q est l'ensemble des atomes de ϕ et $I = \{At \in Q \mid \phi \in At\}$,
- $At \xrightarrow{\langle P, sv \rangle, \top, u} At'$ ssi

  (prop.) pour toute variable propositionnelle p appartenant à ϕ, on a $p \in At$ ssi $p \in P$,
  (contr.) pour toute contrainte atomique α de At on a $sv \models_{\text{symb}} \alpha$,
  (1-pas) pour toute formule $\mathsf{X}\psi \in cl(\phi)$ on a $\mathsf{X}\psi \in At$ ssi $\psi \in At'$,
- Soit $\{\psi_1 \mathsf{U} \phi_1, \ldots, \psi_n \mathsf{U} \phi_n\}$ l'ensemble des formules "until" de $cl(\phi)$. Si cet ensemble est vide, $F = \{Q\}$. Sinon, nous posons $F = \{F_1, \ldots, F_n\}$ tel que pour tout $i \in \{1, \ldots, n\}$ on a $F_i = \{At \in Q : \psi_i \mathsf{U} \phi_i \notin At \text{ ou } \phi_i \in X\}$.

L'automate $\mathcal{A}_{\text{symb}}$ est l'automate de Büchi non généralisé équivalent à $\mathcal{A}'_{\text{symb}}$. Nous rappelons que cette construction peut s'effectuer en espace logarithmique par rapport $|\mathcal{A}'_{\text{symb}}|$.

Nous consacrons la dernière partie de cette section à la construction de $\mathcal{A}_{\text{sat}}$ et nous montrons qu'elle peut se faire en espace polynomial par rapport à $|\phi|$. Nous mettons donc cette partie de la construction de côté pour le moment.

L'automate $\mathcal{A}_\phi$ est obtenu en synchronisant les automates $\mathcal{A}_{\text{symb}}$ et $\mathcal{A}_{\text{sat}}$ de la façon suivante. Nous posons $\mathcal{A}_{\text{symb}} = \langle Q_{sy}, I_{sy}, F_{sy}, \delta_{sy} \rangle$ et $\mathcal{A}_{\text{sat}} = \langle Q_{sa}, I_{sa}, F_{sa}, \delta_{sa} \rangle$. L'automate $\mathcal{A}_\phi = \langle Q, I, F, \delta \rangle$ est défini par :

- $Q = Q_{sy} \times Q_{sa}$,
- $I = I_{sy} \times I_{sa}$,
- $F = F_{sy} \times Q_{sa}$,
- $\langle q_1, q_2 \rangle \xrightarrow{\varepsilon, t, u} \langle q'_1, q'_2 \rangle \in \delta$ ssi $q_1 = q'_1$ et $q_2 \xrightarrow{\varepsilon, t, u} q'_2 \in \delta_{sa}$,
- $\langle q_1, q_2 \rangle \xrightarrow{\langle P, sv \rangle, t, u} \langle q'_1, q'_2 \rangle \in \delta$ ssi $q_1 \xrightarrow{\langle P, sv \rangle, \top, 0} q'_1 \in \delta_{sy}$ et $q_2 \xrightarrow{\langle P, sv \rangle, t, u} q'_2 \in \delta_{sa}$.

Donc $\mathcal{A}_\phi$ peut être construit en espace polynomial par rapport à $|\phi|$ grâce à la façon dont $\mathcal{A}_{\text{sat}}$ est construit (voir la Section 5.2.4). De plus nous montrons dans la Section 5.4 que le problème du vide pour les $\langle 1, \mathbb{Z} \rangle$-automates à compteur simples étendus avec un alphabet est NLOGSPACE-complet. Nous obtenons alors le résultat suivant.

Théorème 27 *Le problème de la satisfaisabilité pour la logique* $\text{CLTL}^1_1(\text{DL}^+, \text{PROP})$ *est* PSPACE*-complet.*

Preuve : La taille de l'automate $\mathcal{A}_\phi$ est exponentielle par rapport à $|\phi|$ mais la construction peut se faire en espace polynomial. Comme le problème du vide est dans NLOGSPACE pour cet automate, nous obtenons une procédure en espace polynomial non-déterministe par composition des deux procédures [BDG88, Lemme 3.35]. D'après le théorème de Savitch cette procédure est donc dans PSPACE.

La preuve de PSPACE-dureté est immédiate par réduction du problème de la satisfaisabilté pour LTL. En effet, puisque le nombre de variables propositionnelles n'est pas restreint dans le fragment $\mathrm{CLTL}_1^1(\mathrm{DL}^+, \mathrm{PROP})$, toute formule de LTL appartient aussi à $\mathrm{CLTL}_1^1(\mathrm{DL}^+, \mathrm{PROP})$. □

Nous obtenons aussi la même borne de complexité pour les problèmes de $\mathrm{CLTL}(\mathrm{DL}^+)$.

Corollaire 10 *Les problèmes de satisfaisabilité et de model-checking pour* $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$ *sont* PSPACE*-complets.*

Preuve : La preuve que ces problèmes sont dans PSPACE est une conséquence directe du Théorème 27 et du Lemme 19.

Nous montrons la PSPACE-dureté par réduction du problème de la satisfaisabilité pour le fragment de LTL avec une variable. Ce problème est PSPACE-complet d'après [DS02, Corollaire 3.2]. Étant donnée une formule ϕ de LTL ayant une unique variable propositionnelle p, nous construisons une formule ϕ' de $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$ en remplaçant simplement les occurrences de p par la contrainte $x = 0$. Il est évident que ϕ est satisfaisable ssi ϕ' est satisfaisable. Enfin, nous rappelons que la satisfaisabilité se réduit au model-checking pour $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$ (voir la fin de la Section 5.1.3), ce qui implique la PSPACE-dureté de ce dernier problème. □

Un corollaire de ces résultats est que le problème du model-checking des automates discrets à une horloge de [DPK03] avec $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$ est dans PSPACE. Ce résultat contraste avec le résultat d'indécidabilité de [DPK03, Section 6]. Les résultats du Corollaire 10 peuvent aussi être étendus en autorisant l'utilisation de variables propositionnelles dans les formules et automates.

Comme dans le chapitre précédent, une particularité de la méthode symbolique que nous utilisons réside dans le fait que la partie logique temporelle est traitée séparément de la partie liée à l'arithmétique de Presburger. Ainsi, nous pouvons adapter la construction afin de traiter n'importe quelle extension de LTL dont les modèles d'une formule donnée peuvent être reconnus par un automate de Büchi. Il nous suffit juste de modifier l'automate $\mathcal{A}_{\mathrm{symb}}$ par rapport à la méthode de construction de cette automate. Ceci inclus les extensions de LTL avec des modalités référant au passé, des opérateurs définis par des automates [Wol83] ou des opérateurs de points fixes [Var88].

Théorème 28 *Les problèmes de satisfaisabilité et model-checking pour* $\mathrm{C}x\mathrm{LTL}_1^1(\mathrm{DL}^+)$ *sont dans* PSPACE*.*

Par conséquent, le model-checking du μ-calcul linéaire étendu avec des contraintes de DL sur les DL-automates à une variable est PSPACE-complet, ce qui raffine un résultat de [BEM97].

5.2.4 Construction de l'automate $\mathcal{A}_{\mathrm{sat}}$

Nous décrivons maintenant la construction de l'automate $\mathcal{A}_{\mathrm{sat}}$ qui reconnaît l'ensemble des modèles symboliques $\langle \sigma_1, \rho \rangle$ tels que ρ est satisfaisable. Nous rappelons que l'on suppose sans perte de généralité que les constantes de C_{loc} vérifient $d_0 = 0$, $d_{\max} \geq 0$ et $d_{\min} \leq 0$, et que celles de C_{step} vérifient $e_0 = 0$, $e_{\max} \geq 0$ et $e_{\min} \leq 0$.

Nous définissons la construction l'automate $\mathcal{A}_{\text{sat}}$ de manière modulaire. L'alphabet de $\mathcal{A}_{\text{sat}}$ est $2^{\text{PROP}} \times \text{SV}(\phi)$ mais comme l'ensemble des variables propositionnelles n'intervient pas dans la satisfaction de ρ, nous allégeons les notations en omettant la partie de l'alphabet correspondant à 2^{PROP}. Dans la suite lorsque nous définissons une non-epsilon transition de la forme $q \xrightarrow{sv,test,maj} q'$, nous avons en fait $q \xrightarrow{\langle sv,X\rangle,test,maj} q'$ pour tout $X \in 2^{\text{PROP}}$.

L'automate $\mathcal{A}_{\text{sat}}$ est divisé en différentes composantes connectées entre elles. Chaque composante est un $\langle 1, \mathbb{Z}\rangle$-automate à compteur simple particulier sur l'alphabet $\text{SV}(\phi)$ de la forme $\langle Q, I, F, \delta\rangle$ tel que

– les ensembles d'états initiaux et finaux I et F sont réduits à des singletons,

– δ est un sous-ensemble de $(Q \setminus F) \times \{\varepsilon\} \times \{\top, =, \neq, >, <\} \times \{-1, 0, 1\} \times (Q \setminus I)$.

Nous appelons entrée de la composante l'unique état de I et sortie l'unique état de F. Les composantes sont reliées par des transitions allant de la sortie de l'une vers l'entrée de la suivante. Chaque composante de $\mathcal{A}_{\text{sat}}$ à pour rôle de vérifier une contrainte de C_x, ou de modifier la valeur du compteur par rapport à une contrainte de Mod_x ou un couple de contraintes de $\text{Mod}_{\mathsf{X}x} \times \text{C}_{\text{step}}$ (vu comme la conjonction de ces deux contraintes). Nous définissons ci-dessous les différentes composantes de $\mathcal{A}_{\text{sat}}$, notées $\mathcal{A}_{\alpha,sv}$ où $\alpha \in \text{C}_x \cup \text{Mod}_x \cup (\text{Mod}_{\mathsf{X}x} \times \text{C}_{\text{step}})$ et $sv \in \text{SV}(\phi)$. Nous notons respectivement $q_{in}^{\alpha,sv}$ et $q_{out}^{\alpha,sv}$ l'entrée et la sortie de $\mathcal{A}_{\alpha,sv}$ ou simplement q_{in} et q_{out} lorsque le contexte est clair.

Pour toute valuation symbolique $sv = \langle \alpha_x, \theta, \alpha'_x, \theta', \alpha_s\rangle \in \text{SV}(\phi)$, nous definissons les composantes suivantes :

– La composante $\mathcal{A}_{\alpha_x,sv}$ teste si la valeur du compteur satisfait la contrainte α_x. Cette composante vérifie la propriété suivante :

$$\text{pour tout } c \in \mathbb{Z} \text{ on a } \langle q_{in}^{\alpha_x,sv}, c\rangle \rightarrow^* \langle q_{out}^{\alpha_x,sv}, c'\rangle \text{ ssi } c = c' \text{ et } [x \leftarrow c] \models \alpha_x.$$

Les Figures 5.2, 5.3 et 5.4 représentent des composantes de ce type vérifiant respectivement les contraintes $x = d_i$, $d_i < x \wedge x < d_{i+1}$ et $d_{\max} < x$ lorsque $d_i \geq 0$. La construction est définie de la même manière dans le cas où $d_i \leq 0$ (on remplace les incrémentations par des décrémentations et les tests $x > 0$ par $x < 0$).

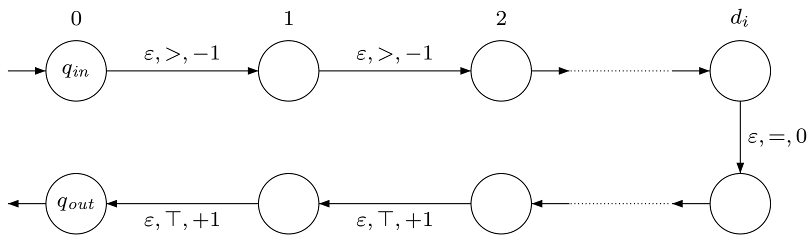


Figure 5.2: Composante $\mathcal{A}_{x=d_i,sv}$

– La composante $\mathcal{A}_{\langle\theta',\alpha_s\rangle,sv}$ met à jour la valeur du compteur par rapport à la contrainte $\theta' \wedge \alpha_s$. Ceci signifie que cette composante est telle que

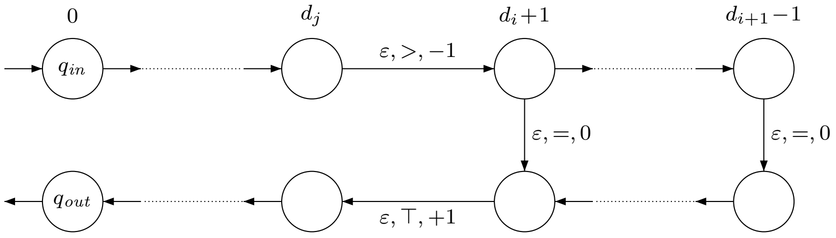


Figure 5.3: Composante $\mathcal{A}_{d_i<x<d_{i+1},sv}$

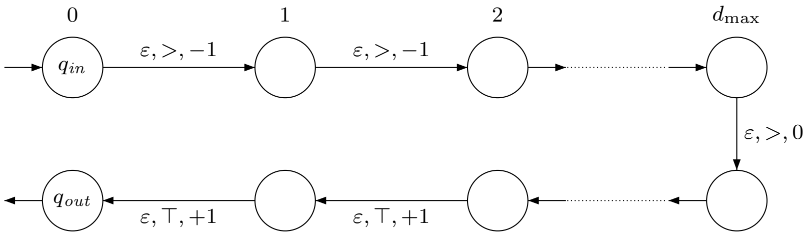


Figure 5.4: Composante $\mathcal{A}_{d_{\max}<x,sv}$

$$\text{pour tout } c \in \mathbb{Z} \text{ on a } [x \leftarrow c] \models \theta \text{ et } \langle q_{in}^{\langle\theta',\alpha_s\rangle,sv}, c\rangle \rightarrow^* \langle q_{out}^{\langle\theta',\alpha_s\rangle,sv}, c'\rangle \text{ ssi } [x \leftarrow c, \mathsf{X}x \leftarrow c'] \models \theta' \wedge \alpha_s.$$

où θ est la contrainte de sv appartenant à Mod_x. Par exemple, la Figure 5.5 représente la composante $\mathcal{A}_{\langle\theta',\alpha_s\rangle,sv}$ telle que θ vaut $x \equiv_2 1$, θ' vaut $\mathsf{X}x \equiv_2 0$ et α_s vaut $x < \mathsf{X}x \wedge \mathsf{X}x < x+7$. Pour construire cette composante, nous devons d'abord calculer à partir de θ, θ' et α_s que $\mathsf{X}x = x + i$ pour $i \in \{1,3,5\}$.

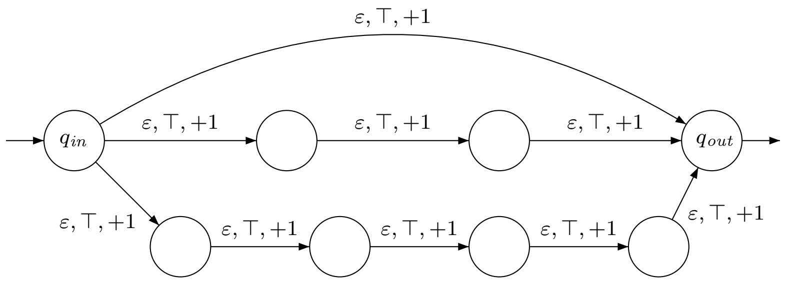


Figure 5.5: Composante $\mathcal{A}_{\langle\theta',\alpha'_x\rangle,sv}$

– La composante $\mathcal{A}_{\theta,sv}$ met à jour le compteur par rapport à la contrainte θ. Cette mise à jour n'est utilisée que pour initialiser la valeur du compteur au début de l'exécution (ceci sera explicité lorsque nous définissons les connections entre les composantes). Cette composante est telle que

$$\text{pour tout } c \in \mathbb{Z} \text{ on a } \langle q_{in}^{\theta,sv}, 0\rangle \rightarrow^* \langle q_{out}^{\theta,sv}, c\rangle \text{ ssi } [x \leftarrow c] \models \theta.$$

La Figure 5.6 représente une composante de la forme $\mathcal{A}_{x\equiv_K c,sv}$.

Le principe de la construction de $\mathcal{A}_{\text{sat}}$ consiste à connecter les différentes composantes de la forme $\mathcal{A}_{\alpha,sv}$ entre elles de manière à ce qu'elles imposent que les différentes valeurs du compteur satisfassent bien les valuations symboliques successives. L'automate

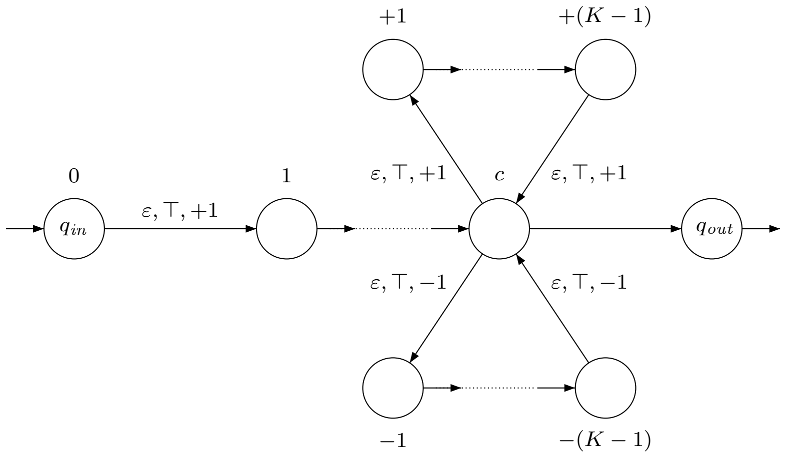


Figure 5.6: Composante $\mathcal{A}_{x \equiv_K c, sv}$

$\mathcal{A}_{\text{sat}} = \langle Q, I, F, \delta \rangle$ est l'union disjointe des différentes composantes $\mathcal{A}_{\alpha,sv}$ définies ci-dessus qui comporte un état initial supplémentaire s_0 et les transitions suivantes :

- Pour tout $sv = \langle \alpha_x, \theta, \alpha'_x, \theta', \alpha_s \rangle \in \mathrm{SV}(\phi)$ on a une transition

  $$s_0 \xrightarrow{\varepsilon,\top,0} q_{in}^{\theta,sv} \in \delta.$$

  Ces transitions correspondent au choix de la première valuation symbolique. Nous définissons donc une transition vers la composante $\mathcal{A}_{\theta,sv}$ afin de s'assurer que la première valeur du compteur vérifie bien θ.
- Pour tout $sv = \langle \alpha_x, \theta, \alpha'_x, \theta', \alpha_s \rangle \in \mathrm{SV}(\phi)$ on a une transition

  $$q_{out}^{\theta,sv} \xrightarrow{\varepsilon,\top,0} q_{in}^{\alpha_x,sv} \in \delta.$$

  Lorsque l'automate est dans l'état de contrôle $q_{out}^{\theta,sv}$ nous savons par définition de la composante $\mathcal{A}_{\theta,sv}$ que la valeur du compteur satisfait θ. Nous voulons maintenant vérifier la prochaine contrainte α_x. Avec ces transitions nous imposons que la seule façon de continuer l'exécution est d'entrer dans la composante $\mathcal{A}_{\alpha_x,sv}$.
- Pour tout $sv = \langle \alpha_x, \theta, \alpha'_x, \theta', \alpha_s \rangle \in \mathrm{SV}(\phi)$ on a une transition

  $$q_{out}^{\alpha_x,sv} \xrightarrow{\varepsilon,\top,0} q_{in}^{\langle \theta', \alpha_s \rangle, sv} \in \delta.$$

  Lorsque l'automate est dans l'état de contrôle $q_{out}^{\alpha_x,sv}$, la valeur du compteur satisfait α_x d'après la définition de $\mathcal{A}_{\alpha_x,sv}$. Nous mettons alors à jour la valeur du compteur par rapport aux contraintes que celle-ci doit vérifier au pas suivant, c'est-à-dire θ' et α_s. C'est pourquoi nous imposons que la seule façon de continuer l'exécution soit d'entrer dans la composante $\mathcal{A}_{\langle \theta', \alpha_s \rangle, sv}$.
- Enfin, pour tout $sv_1 = \langle (\alpha_x)_1, (\theta)_1, (\alpha'_x)_1, (\theta')_1, (\alpha_s)_1 \rangle \in \mathrm{SV}(\phi)$ et $sv_2 = \langle (\alpha_x)_2, (\theta)_2, (\alpha'_x)_2, (\theta')_1, (\alpha_s)_1 \rangle \in \mathrm{SV}(\phi)$ tels que $(\alpha'_x)_1[\mathsf{X}x \leftarrow x] = (\alpha_x)_2$ et $(\theta')_1[\mathsf{X}x \leftarrow x] = (\theta)_2$ on a

  $$q_{out}^{\langle (\theta')_1, (\alpha'_x)_1 \rangle, sv_1} \xrightarrow{sv_1,\top,0} q_{in}^{(\alpha_x)_2, sv_2} \in \delta$$

Si l'automate est dans l'état $q_{out}^{\langle(\theta')_1,(\alpha'_x)_1\rangle,sv_1}$ alors d'après les connections définies au dessus et la définition de $\mathcal{A}_{\langle(\theta')_1,(\alpha'_x)_1\rangle,sv_1}$ toutes les vérifications et mises à jour liées à sv_1 ont été faites avec succès. Nous pouvons donc lire la lettre correspondant à la valuation symbolique sv_1. Nous devons aussi choisir une nouvelle valuation symbolique sv_2 qui concorde avec sv_1 sur les contraintes impliquant la valeur du compteur à l'état suivant. Pour poursuivre l'exécution nous imposons donc d'entrer dans une composante $\mathcal{A}_{(\alpha_x)_2,sv_2}$ telle que sv_2 vérifie cette concordance. Notons que la contrainte $(\theta)_2$ n'a pas besoin d'être vérifiée puisqu'elle l'a été par la composante $\mathcal{A}_{\langle(\theta')_1,(\alpha'_x)_1\rangle,sv_1}$ d'après la correspondance entre $(\theta)_2$ et $(\theta')_1$.

Enfin, l'ensemble des états finaux F est égal à Q. Par construction de $\mathcal{A}_{\text{sat}}$ nous avons la propriété suivante.

Lemme 22 *Pour tout couple de valuations symboliques* $sv_1 = \langle(\alpha_x)_1, (\theta)_1, (\alpha'_x)_1, (\theta')_1, (\alpha_s)_1\rangle$ *et* $sv_2 = \langle(\alpha_x)_2, (\theta)_2, (\alpha'_x)_2, (\theta')_2, (\alpha_s)_2\rangle$ *appartenant à* $\mathrm{SV}(\phi)$ *et pour tout* $c, c' \in \mathbb{Z}$, *les propositions suivantes sont équivalentes :*

(I) $[x \leftarrow c] \models (\theta)_1$ *et* $\langle q_i^{(\alpha_x)_1,sv_1}, c\rangle \xrightarrow{\varepsilon}{}^* \langle q_{in}^{\langle(\theta')_1,(\alpha_s)_1\rangle,sv_1}, c\rangle \xrightarrow{\varepsilon}{}^* \langle q_{out}^{\langle(\theta')_1,(\alpha_s)_1\rangle,sv_1}, c'\rangle \xrightarrow{sv_1} \langle q_{in}^{(\alpha_x)_2,sv_2}, c'\rangle \xrightarrow{\varepsilon}{}^* \langle q_{out}^{(\alpha_x)_2,sv_2}, c'\rangle$,

(II) $[x \leftarrow c, \mathsf{X}x \leftarrow c'] \models sv_1$.

Il suffit juste de considérer les différents modules traversés et les propriétés qu'ils vérifient.

Nous pouvons maintenant prouver le résultat attendu établissant que l'ensemble des modèles symboliques satisfaisables est exactement le langage reconnu par $\mathcal{A}_{\text{sat}}$.

Lemme 23 *Pour tout modèle symbolique* $\langle\sigma_1, \rho\rangle$, $\langle\sigma_1, \rho\rangle$ *est reconnu par* $\mathcal{A}_{\text{sat}}$ *ssi* ρ *est satisfaisable.*

Preuve : Soit $\langle\sigma_1, \rho\rangle$ un modèle symbolique satisfaisable. Nous posons pour tout $i \in \mathbb{N}$, $\rho(i) = \langle(\alpha_x)_i, (\theta)_i, (\alpha'_x)_i, (\theta')_i, (\alpha_s)_i\rangle$. Puisque ρ est satisfaisable, il existe un modèle $\sigma_2 : \mathbb{N} \to \mathbb{Z}$ tel que $\sigma_2 \models \rho$. Par construction des différentes composantes de $\mathcal{A}_{\text{sat}}$ et leurs interconnexions, la séquence σ_2 correspond à une exécution de la forme

$$s_0 \xrightarrow{\varepsilon} \langle q_{in}^{(\theta)_0,\rho(0)}, \sigma_2(0)\rangle \xrightarrow{\varepsilon}{}^* \langle q_{in}^{(\alpha_x)_0,\rho(0)}, \sigma_2(0)\rangle \xrightarrow{\varepsilon}{}^* \langle q_{in}^{\langle(\theta')_0,(\alpha_s)_0\rangle,\rho_0(0)}, \sigma_2(0)\rangle \xrightarrow{\varepsilon}{}^*$$

$$\langle q_{out}^{\langle(\theta')_0,(\alpha_s)_0\rangle,\rho(0)}, \sigma_2(1)\rangle \xrightarrow{\rho(0)} \langle q_{in}^{(\alpha_x)_1,\rho(1)}, \sigma_2(1)\rangle \xrightarrow{\varepsilon}{}^* \langle q_{in}^{\langle(\theta')_1,(\alpha'_x)_1\rangle,\rho(1)}, \sigma_2(1)\rangle \xrightarrow{\varepsilon}{}^*$$

$$\langle q_{out}^{\langle(\theta')_1,(\alpha'_x)_1\rangle,\rho(1)}, \sigma_2(2)\rangle \xrightarrow{\rho(1)} \langle q_{in}^{(\alpha_x)_2,\rho(2)}, \sigma_2(2)\rangle \ldots$$

En effet, une analyse des propriétés vérifiées par chacune des composantes montre que cette exécution est valide. Ceci implique que $\langle\sigma_1, \rho\rangle$ appartient à $\mathrm{L}(\mathcal{A}_{\text{sat}})$.

Réciproquement, supposons que $\langle\sigma, \rho\rangle \in \mathrm{L}(\mathcal{A}_{\text{sat}})$. Par construction de $\mathcal{A}_{\text{sat}}$ il existe une exécution acceptante de la forme

$$s_0 \xrightarrow{\varepsilon} \langle q_{in}^{(\theta)_0,sv_0}, c_0\rangle \xrightarrow{\varepsilon}{}^* \langle q_{in}^{(\alpha_x)_0,sv_0}, c_0\rangle \xrightarrow{\varepsilon}{}^* \langle q_{in}^{\langle(\theta')_0,(\alpha'_x)_0\rangle,sv_0}, c_0\rangle \xrightarrow{\varepsilon}{}^*$$

$$\langle q_{out}^{\langle(\theta')_0,(\alpha'_x)_0\rangle,sv_0}, c_1\rangle \xrightarrow{\rho(0)} \langle q_{in}^{(\alpha_x)_1,sv_1}, c_1\rangle \xrightarrow{\varepsilon}^* \langle q_{in}^{\langle(\theta')_1,(\alpha_s)_1\rangle,sv_1}, c_1\rangle \xrightarrow{\varepsilon}^*$$
$$\langle q_{out}^{\langle(\theta')_1,(\alpha_s)_1\rangle,sv_1}, c_2\rangle \xrightarrow{\rho(1)} \langle q_{in}^{(\alpha_x)_2,sv_1}, c_2\rangle \ldots$$

D'après la définition des différentes composantes de $\mathcal{A}_{\text{sat}}$ et le Lemme 22, cette exécution vérifie $[x \leftarrow c_0] \models (\theta)_0$ et pour tout $i \in \mathbb{N}$, $[x \leftarrow c_i, \mathsf{X}x \leftarrow c_{i+1}] \models (\alpha_x)_i \wedge (\theta)_i \wedge (\alpha'_x)_i \wedge (\theta')_i \wedge (\alpha_s)_i$. Il en résulte que le modèle $\sigma_2 : \mathbb{N} \to \mathbb{Z}$ défini par $\sigma_2(i) = c_i$ pour tout $i \in \mathbb{N}$ satisfait ρ. $\square$

5.3 Model-checking de $\mathrm{CLTL}_1^1(\mathrm{QFP})$

Une question naturelle est de se demander si le résultat du Corollaire 10 est optimal par rapport aux fragments de l'arithmétique de Presburger que nous considérons. Le résultat suivant répond positivement à cette question puisque le problème de la satisfaisabilité est indécidable pour le fragment $\mathrm{CLTL}_1^1(\mathrm{QFP})$. Nous rappelons que l'ensemble des contraintes du fragment sans quantificateur de l'arithmétique de Presburger peut être défini par

$$\alpha ::= \sum_{i\in I} a_i x_i \sim d \mid \sum_{i \in I} a_i x_i \equiv_k c \mid \alpha \wedge \alpha \mid \neg\alpha$$

où $\sim \in \{<, =\}$, $a_i \in \mathbb{Z}$ et I est un ensemble fini d'indices.

Lemme 24 *Les problèmes de la satisfaisabilté et du model-checking pour* $\mathrm{CLTL}_1^1(\mathrm{QFP})$ *sont* Σ_1^1*-complets.*

Preuve La dureté est obtenue par réduction de l'exécution d'une machine de Minsky à deux compteurs. Une configuration $\langle q_i, c_1, c_2\rangle$ de la machine peut se coder par la valeur du compteur $2^i 3^{c_1} 5^{c_2}$. L'incrémentation d'un compteur se fait alors à l'aide d'une multiplication, par exemple l'opération $\mathsf{X}x = 3x$ correspond à l'incrémentation du compteur c_1, et les tests à zero peuvent s'exprimer par une contrainte de périodicité, par exemple $\neg(x \equiv_3 0)$ est vraie ssi le compteur c_1 vaut zéro. $\square$

La solution que nous choisissons dans cette section pour regagner la décidabilité est de restreindre la classe d'automates pour le problème du model-checking aux $\langle 1, \mathbb{Z}\rangle$-automates à compteur. Nous montrons que le problème du model-checking sur les $\langle 1, \mathbb{Z}\rangle$-automates à compteur est décidable pour LTL étendu avec des contraintes de QFP *restreint à une variable mais sans borne sur la longueur temporelle.* Le comportement du compteur est en effet plus contraint dans les $\langle 1, \mathbb{Z}\rangle$-automates à compteur, ce qui explique que nous retrouvions la décidabilité.

Soit $\mathcal{A} = \langle Q_{\mathcal{A}}, I_{\mathcal{A}}, F_{\mathcal{A}}, \delta_{\mathcal{A}}\rangle$ un $\langle 1, \mathbb{Z}\rangle$-automate à compteur dont l'ensemble des mises à jour du compteur est inclus dans $\{\mathsf{X}x = x + u \mid u \in \{u_{\min}, u_{\min}+1, \ldots, u_{\max}\}\}$ tel que $u_{\min} = -u_{\max}$. Étant donné un ensemble fini de formules atomiques X de $\mathrm{CLTL}_1(\mathrm{QFP})$ nous considérons les ressources syntaxiques suivantes :

- l est le plus grand entier tel qu'un terme de la forme $\mathsf{X}^l x$ apparaît dans X.
- K est le plus petit commun multiple des entiers k tels que $\equiv_k$ apparaît dans X.
- CONS est l'ensemble des constantes d telles qu'il existe une contrainte de la forme $\sum a_i \mathsf{X}^i x \sim d$ dans X. Nous notons M et m les éléments maximal et minimal de cet ensemble et supposons sans perte de généralité que $M = -m$.
- COEF est l'ensemble des coefficients entiers a_i tels qu'il existe une contrainte de la forme $\sum a_i \mathsf{X}^i x \sim d$ dans X.

Sans perte de généralité, nous supposons que $a_{\min} = -a_{\max}$ où $a_{\min}$ et $a_{\max}$ sont respectivement les plus petit et plus grand éléments de COEF. Pour des raisons techniques (voir la preuve du Lemme 25), nous posons $\mathrm{CONS}' = \{d_{\min}, \dots, d_{\max}\}$ tel que

$$d_{\max} = -d_{\min} = M + \frac{l(l+1)}{2} a_{\max} u_{\max}.$$

Nous définissons une variante de l'abstraction de la Section 5.2.2 tenant compte des spécificités du nouveau problème. L'ensemble des valuations symboliques $\mathrm{SV}(\mathcal{A}, X)$ par rapport à un ensemble fini de formules atomiques X et $\mathcal{A}$ est défini comme le produit $\mathrm{C}_x \times \mathrm{Mod}_x \times \mathrm{C}^1_{\mathrm{step}} \times \cdots \times \mathrm{C}^l_{\mathrm{step}}$ tel que

- C_x est l'ensemble de contraintes composé de
  - $x < d_{\min}$,
  - $d_{\max} < x$,
  - $x = d$ pour tout $d \in \mathrm{CONS}'$.
- Mod_x est l'ensemble composé des contraintes $x \equiv_K c$ pour tout $c \in \{0, \dots, K-1\}$.
- $\mathrm{C}^i_{\mathrm{step}}$ est l'ensemble composé des contraintes $\mathsf{X}^i x = \mathsf{X}^{i-1} x + u$ pour tout $\{u_{\min}, \dots, u_{\max}\}$.

Nous notons $\mathrm{SV}(\mathcal{A}, \phi)$ l'ensemble des valuations symboliques construites par rapport à l'ensemble des contraintes atomiques d'une formule ϕ de $\mathrm{CLTL}_1(\mathrm{QFP})$ et de l'automate $\mathcal{A}$. Pour tout chemin fini $\pi = \langle q_0, c_0 \rangle \dots \langle q_{|\pi|}, c_{|\pi|} \rangle$ dans l'automate $\mathcal{A}$, nous notons $v_\pi : \{x, \mathsf{X}x, \dots, \mathsf{X}^{|\pi|} x\} \to \mathbb{Z}$ la valuation telle que pour tout $i \in \{0, \dots, |\pi|\}$ on a $v_\pi(\mathsf{X}^i x) = c_i$. Nous montrons maintenant la correction de cette nouvelle abstraction.

Lemme 25 *Soit ϕ une formule de* $\mathrm{CLTL}_1(\mathrm{QFP})$ *et $\mathcal{A}$ un$\langle 1, \mathbb{Z} \rangle$-automate à compteur.*
(I) *Pour tout chemin fini π dans $\mathcal{A}$ tel que $|\pi| = l$, il existe une unique valuation symbolique $sv(v_\pi) \in \mathrm{SV}(\mathcal{A}, \phi)$ telle que $v_\pi \models sv(v)$.*
(II) *Pour tout couple de valuations $v, v' : \{x, \mathsf{X}x, \dots, \mathsf{X}^l x\} \to \mathbb{Z}$ telles que $sv(v) = sv(v')$ et pour toute contrainte atomique α de ϕ on a $v \models \alpha$ ssi $v' \models \alpha$.*

Preuve **(I)** La définition des différents ensembles qui composent l'ensemble des valuations symboliques $\mathrm{SV}(\mathcal{A}, \phi)$ induit une partition de $\mathbb{Z}^{l+1}$. Cette propriété peut être démontrée de la même manière que le Lemme 20 **(I)**.

(II) Nous procédons par induction sur la structure de la contrainte α.

Si α est de la forme $\sum a_i \mathsf{X}^i x \sim d$ alors nous substituons les différents termes $\mathsf{X}^i x$ en utilisant les contraintes $\mathsf{X}^i x = \mathsf{X}^{i-1} x + u_{b_i}$ appartenant à $sv(v)$. Donc la contrainte

$a_0x + a_1\mathsf{X}x + a_2\mathsf{X}^2x \cdots \sim d$ devient

$a_0x + a_1(x + u_{b_1}) + a_2(\mathsf{X}x + u_{b_2}) \cdots \sim d$ puis

$(a_0 + a_1)x + a_2(x + u_{b_1} + u_{b_2}) \cdots \sim d - a_1u_{b_1}$,

$(a_0 + a_1 + a_2)x + \cdots \sim d - ((a_1 + a_2)u_{b_1} + a_2u_{b_2}), \ldots$

... et ainsi de suite jusqu'à se réduire à une contrainte de la forme $ax \sim d'$. Notons que par définition de $d_{\max}$ et $d_{\min}$, la partie droite de l'expression est toujours un élément de CONS$'$. En effet, la partie de droite est toujours de la forme

$$\mathrm{D} = d - \sum_{i=1}^{n} \sum_{j=i}^{n} a_j u_{b_i}$$

pour un $n < l$. Comme pour tout $0 \leq i \leq n$ on a $u_{\min} \leq u_{b_i} \leq u_{\max}$ et $a_{\min} \leq a_i \leq a_{\max}$, la valeur absolue de D est bornée par

$$|d - \sum_{j=i}^{n} a_{\max} u_{b_i}| \leq |d - \frac{n(n+1)}{2} a_{\max} u_{\max}|.$$

Ce qui nous assure que $d_{\min} \leq \mathrm{D} \leq d_{\max}$ puisque $n \leq l$ et $|d| \leq M$.

Considérons la contrainte $ax \sim d'$ obtenue après réduction. Par la propriété ci-dessus on a $d_{\min} \leq d' \leq d_{\max}$. Comme a est un entier et l'intervalle des constantes est symétrique par rapport à 0, d'/a est aussi compris entre $d_{\min}$ et $d_{\max}$. Ainsi, si $v \models sv$ alors la contrainte α_x de sv correspondant à l'ensemble C_x doit être telle que la formule $\alpha_x \Rightarrow ax \sim d'$ est valide, car C_x est construit par rapport à CONS$'$. Puisque $sv(v) = sv(v')$, nous avons $v' \models_{\mathrm{symb}} \alpha_x$ et donc $v' \models_{\mathrm{symb}} \alpha$

Le cas de $\sum a_i \mathsf{X}^i x \equiv_k d$ est identique. En effet, la contrainte se réduit à $ax \equiv_k d'$ après substitutions successives. Nous pouvons alors conclure en utilisant la contrainte de $sv(v)$ correspondant à l'ensemble Mod_x.

Enfin, les cas des connecteurs Booléens est évident par induction. □

Comme dans la Section 5.2.2, nous définissons une relation de satisfaction symbolique. Nous notons $sv \models_{\mathrm{symb}} \alpha$ ssi pour tout chemin π de longueur l dans $\mathcal{A}$ tel que $sv(v_\pi) = sv$ on a $v_\pi \models \alpha$. Notons que la technique de substitution utilisée dans la preuve précédente montre que la satisfaisabilité d'une contrainte atomique par une valuation symbolique se teste en temps polynomial. Nous étendons aussi la relation de satisfaction symbolique aux modèles symboliques (séquences de valuations symboliques) par rapport aux formules de $\mathrm{CLTL}_1(\mathrm{QFP})$.

La relation entre les modèles symboliques de $\mathrm{SV}(\mathcal{A}, \phi)$ et les exécutions de $\mathcal{A}$ est la suivante. Pour tout modèle symbolique $\rho : \mathbb{N} \to \mathrm{SV}(\mathcal{A}, \phi)$, une exécution acceptée par $\mathcal{A}$ de la forme $\langle q_0, c_0\rangle, \langle q_1, c_1\rangle, \langle q_2, c_3\rangle, \ldots$ satisfait ρ ssi le modèle $\sigma = c_0 \cdot c_1 \cdot c_2 \cdots$ est tel que pour tout $i \in \mathbb{N}$ on a $\sigma, i \models \rho(i)$. L'automate $\mathcal{A}$ satisfait ρ ssi il existe une exécution acceptée par $\mathcal{A}$ qui satisfait ρ. Nous notons alors $\mathcal{A} \models \rho$.

Lemme 26 *Soit ϕ une formule de $\mathrm{CLTL}_1(\mathrm{QFP})$ et $\mathcal{A}$ un $\langle 1, \mathbb{Z}\rangle$-automate à compteur. On a $\mathcal{A} \models \phi$ ssi il existe un modèle symbolique $\rho : \mathbb{N} \to \mathrm{SV}(\mathcal{A}, \phi)$ tel que $\rho \models_{\mathrm{symb}} \phi$ et $\mathcal{A} \models \rho$.*

Preuve : Cette preuve est similaire à celle du Lemme 21. Si la formule ϕ est satisfaite par une exécution $\langle q_0, c_0\rangle \to \langle q_1, c_1\rangle \to \langle q_2, c_2\rangle \dots$ alors le modèle symbolique ρ tel que pour tout $i \in \mathbb{N}$ on a $\rho(i) = sv(v_{\pi_i})$ où $\pi_i = \langle q_i, c_i\rangle \to \langle q_{i+1}, c_{i+1}\rangle \to \cdots \to \langle q_{i+l}, c_{i+l}\rangle$ satisfait symboliquement ϕ. Cette affirmation se montre facilement en utilisant le même développement que dans le Lemme 21 et le Lemme 25 ci-dessus.

Réciproquement, si $\rho \models_{\text{symb}} \phi$ alors toute exécution de $\mathcal{A}$ qui satisfait ρ satisfait aussi ϕ. En effet, nous pouvons facilement montrer que pour tout $i \in \mathbb{N}$, le chemin π_i de longueur l commençant à la position i de l'exécution est tel que v_{π_i} satisfait $\rho(i)$. La fin de la preuve est alors identique à la preuve du Lemme 21. $\square$

Nous construisons donc un automate $\mathcal{A}_\phi^{\text{MC}} = \mathcal{A}_{\text{symb}} \cap \mathcal{A}_{\text{sat}}$ tel que $\mathcal{A}_{\text{symb}}$ reconnaît l'ensemble des modèles symboliques qui satisfont symboliquement ϕ et $\mathcal{A}_{\text{sat}}$ reconnaît maintenant l'ensemble des modèles symboliques qui sont satisfaits par une exécution de $\mathcal{A}$. La définition de $\mathcal{A}_{\text{symb}}$ ainsi que la synchronisation entre les deux automates sont similaires à la Section 5.2.3 en considérant le nouvel alphabet $\text{SV}(\mathcal{A}, \phi) \cup \{\varepsilon\}$ et la relation de satisfaction symbolique définie dans cette section.

Lemme 27 *Étant donnée une formule de* $\text{CLTL}_1(\text{QFP})$ *et un* $\langle 1, \mathbb{Z}\rangle$*-automate à compteur* $\mathcal{A}$*, on peut construire un* $\langle 1, \mathbb{Z}\rangle$*-automate à compteur simple* $\mathcal{A}_{\text{sat}}$ *sur l'alphabet* $\text{SV}(\mathcal{A}, \phi) \cup \{\varepsilon\}$ *tel que* $\text{L}(\mathcal{A}_{\text{sat}})$ *est l'ensemble des modèles symboliques* ρ *tels que* $\mathcal{A} \models \rho$*.*

Preuve : Soit $\mathcal{A} = \langle Q_\mathcal{A}, I_\mathcal{A}, F_\mathcal{A}, \delta_\mathcal{A}\rangle$ un $\langle 1, \mathbb{Z}\rangle$-automate à compteur et ϕ une formule de $\text{CLTL}_1(\text{QFP})$. L'automate $\mathcal{A}_{\text{sat}}$ est un $\langle 1, \mathbb{Z}\rangle$-automate à compteur simple sur l'alphabet $\Sigma = SV(\mathcal{A}, \phi)$ avec des ε-transitions. Comme dans la Section 5.2.4, cet automate est constitué de différentes composantes vérifiant les contraintes des valuations symboliques. La principale différence est que nous devons considérer lors de la connexion de ces composantes le comportement de $\mathcal{A}$. Nous adaptons de manière naturelle les notations utilisées dans la Section 5.2.4 pour référer aux différents attributs des composantes de $\mathcal{A}_{\text{sat}}$ (entrée et sortie par exemple). Pour chaque valuation symbolique $sv = \langle \alpha_x, \theta, \alpha_s^1, \dots, \alpha_s^l\rangle$ et pour chaque état $q_a \in Q_\mathcal{A}$ nous définissons les composantes suivantes :

- La composante $\mathcal{A}_{\alpha_x, sv}^{q_a}$ vérifie que la valeur du compteur satisfait la contrainte α_x.

  Pour tout $c \in \mathbb{Z}$ on a $\langle q_a, q_{in}^{\alpha_x, sv}, c\rangle \to^* \langle q_a, q_{out}^{\alpha_x, sv}, c'\rangle$ ssi $c = c'$ et $[x \leftarrow c] \models \alpha_x$.
- La composante $\mathcal{A}_{\alpha_s^1, sv}^{q_a}$ met à jour la valeur du compteur par rapport à α_s^1.

  Pour tout $c \in \mathbb{Z}$ on a $\langle q_a, q_{in}^{\alpha_s^1, sv}, c\rangle \to^* \langle q_a, q_{out}^{\alpha_s^1, sv}, c'\rangle$ ssi $[x \leftarrow c, \mathsf{X}x \leftarrow c'] \models \alpha_s^1$.

Ces différentes composantes peuvent facilement être définies en s'inspirant de la construction de la Section 5.2.4 (voir les Figures 5.2 à 5.6).

Ces composantes sont connectées entre elles et avec un état initial supplémentaire s_0 de la façon suivante :

- Pour toute valuation symbolique $sv = \langle \alpha_x, \theta, \alpha_s^1, \dots, \alpha_s^l\rangle \in \text{SV}(\phi, \mathcal{A})$ telle que
  - α_x vaut $x = 0$,

– θ vaut $x \equiv_K 0$,

et pour tout état initial $q_0 \in I_{\mathcal{A}}$ de l'automate $\mathcal{A}$, on a une transition

$$s_0 \xrightarrow{\varepsilon,\top,0} \langle q_0, q_{in}^{\alpha_x,sv}\rangle.$$

Au début de l'exécution de $\mathcal{A}$, le compteur doit être égal à zéro. Contrairement à la construction définie dans la Section 5.2.4, nous n'avons donc pas besoin de mettre à jour la valeur du compteur par rapport à θ. Nous imposons même que cette contrainte soit $x \equiv_K 0$. Nous choisissons donc une valuation vérifiant ces critères et ajoutons une transition vers la composante correspondante qui vérifie qu'on a bien $x = 0$. Notons que cette vérification est facultative, mais elle permet de définir l'automate de manière plus homogène.

– Pour toute valuation symbolique $sv = \langle \alpha_x, \theta, \alpha_s^1, \ldots, \alpha_s^l\rangle \in \mathrm{SV}(\phi, \mathcal{A})$ et pour tout état $q_a \in Q_{\mathcal{A}}$ on a une transition

$$\langle q_a, q_{out}^{\alpha_x,fr}\rangle \xrightarrow{\varepsilon,\top,0} \langle q_a, q_{in}^{\alpha_s^1,sv}\rangle \in \delta.$$

Une fois que la contrainte α_x est vérifiée, nous devons mettre à jour la valeur du compteur par rapport à la mise à jour définie par α_s^1. Nous imposons donc que la suite de l'exécution passe par la composante $\mathcal{A}_{\alpha_s^1,sv}$.

– Pour tout couple de valuations symboliques $sv_1 = \langle (\alpha_x)_1, (\theta)_1, (\alpha_s^1)_1, \ldots, (\alpha_s^l)_1\rangle \in \mathrm{SV}(\phi, \mathcal{A})$ et $sv_2 = \langle (\alpha_x)_2, (\theta)_2, (\alpha_s^1)_2, \ldots, (\alpha_s^l)_2\rangle \in \mathrm{SV}(\phi, \mathcal{A})$ et tout couple d'états $q_a, q_a' \in Q_{\mathcal{A}}$ vérifiant

1. il existe une transition $q_a \xrightarrow{test,maj} q_a' \in \delta_{\mathcal{A}}$ telle que la formule $\alpha_x \Rightarrow test$ est valide et $maj = (\alpha_s^1)_1$,
2. $(\alpha_s^1)_1 \wedge (\theta)_1 \Rightarrow (\theta)_2$ est valide,
3. pour tout $1 \leq i \leq l-1$, $(\alpha_s^i)_2 = (\alpha_s^{i+1})_1[\mathsf{X}^{i+1}x \leftarrow \mathsf{X}^i x, \mathsf{X}^i x \leftarrow \mathsf{X}^{i-1}x]$,

on a une transition

$$\langle q_a, q_{out}^{\alpha_s^1,sv_1}\rangle \xrightarrow{sv_1,\top,0} \langle q_a', q_{in}^{(\alpha_x)_2,sv_2}\rangle \in \delta.$$

Lorsque l'automate est dans l'état $\langle q_a, q_{out}^{\alpha_s^1,sv_1}\rangle$ toute les opérations nécessaires ont été faites sur le compteur. Nous pouvons donc lire la valuation symbolique sv_1 sous certaines conditions :

– Les contraintes de sv_1 sont en accord avec le comportement de l'automate (1).
– Les contraintes de la valuation symbolique suivante sont en accord avec celles de sv_1 (2 et 3).

Ces conditions peuvent être vérifiées en temps polynomial.

Enfin, l'ensemble des états finaux de $\mathcal{A}_{\mathrm{sat}}$ est $\{\langle q_f, q_{out}^{\alpha_x,sv}\rangle \mid q_f \in F_{\mathcal{A}}\}$.

Cette construction vérifie la propriété suivante :

pour sous-chemin fini $\langle q_i, q_{in}^{(\alpha_x)_i,sv_i}, c_i\rangle \rightarrow \cdots \xrightarrow{sv_i} \langle q_{i+1}, q_{in}^{(\alpha_x)_{i+1},sv_{i+1}}, c_{i+1}\rangle \rightarrow \cdots \xrightarrow{sv_{i+1}} \cdots \xrightarrow{sv_{i+l-1}} \langle q_{i+l}, q_{in}^{(\alpha_x)_{i+l},sv_{i+l}}, c_{i+l}\rangle$ d'une exécution de $\mathcal{A}_{\mathrm{sat}}$, on a $\langle c_i, \ldots, c_{i+l}\rangle \models sv_i$.

Cette propriété nous permet de terminer cette preuve en utilisant les mêmes arguments que dans le Lemme 23. $\square$

L'analyse de cette construction montre que $\mathcal{A}_\phi^{\mathrm{MC}}$ peut être construit en espace polynomial par rapport à $|\phi|$. Les arguments utilisés sont les mêmes que ceux de la Section 5.2.3. Ceci nous permet de conclure à propos de la complexité du problème de model-checking qui nous intéresse.

Théorème 29 *Le problème du model-checking de* $\mathrm{CLTL}_1(\mathrm{QFP})$ *sur les* $\langle 1, \mathbb{Z}\rangle$*-automates à compteur est* PSPACE*-complet.*

La borne supérieure est une conséquence de la complexité de la construction définie dans le Lemme 27 et du Théorème 31 sur le test du vide pour les $\langle 1, \mathbb{Z}\rangle$-automates à compteur simples (voir Section 5.4). La dureté peut être obtenue en utilisant le résultat de [DS02] sur la PSPACE-dureté pour LTL restreint à une proposition. Comme le Théorème 28, ce résultat peut être étendu aux extensions de $\mathrm{CLTL}_1(\mathrm{QFP})$ avec des opérateurs définissables dans MSO.

Théorème 30 *Le problème du model-checking de* $\mathrm{C\text{X}LTL}_1(\mathrm{QFP})$ *sur les* $\langle 1, \mathbb{Z}\rangle$*-automates à compteur est* PSPACE*-complet.*

Ainsi, nous raffinons un peu plus le résultat de [BEM97] sur la vérification de propriétés régulières sur le contenu de la pile d'un automate à pile car le model-checking du μ-calcul linéaire avec des contraintes de QFP sur les $\langle 1, \mathbb{Z}\rangle$-automate à compteur est PSPACE-complet.

5.4 Problème du vide pour les $\langle 1, \mathbb{Z}\rangle$-automates à compteur

Pour conclure ce chapitre, nous montrons que tester si le langage $\mathrm{L}(\mathcal{A})$ accepté par un $\langle 1, \mathbb{Z}\rangle$-automate à compteur simple $\mathcal{A}$ avec un alphabet et des ε-transitions est non-vide est un problème dans NLOGSPACE. Ce problème est équivalent à tester si $\mathcal{A}$ admet une exécution acceptante dans laquelle il n'existe pas de position après laquelle on a uniquement des ε-transitions. Ce résultat est utilisé dans les preuves de complexité des Théorèmes 27 et 29. La preuve que nous développons dans cette section utilise plusieurs réductions pour se ramener au problème de l'existence d'une exécution acceptante dans une classe d'automates plus simple à manipuler, puis la complexité est obtenue en analysant les exécutions acceptantes et en montrant une propriété sur leur longueur minimale. Pour alléger la présentation nous ne précisons pas à chaque fois que les automates que nous manipulons sont des automates à compteur simples. Néanmoins, nous supposons pour le reste de cette section que l'ensemble des mises à jour du compteur est restreint à $\{-1, 0, +1\}$.

5.4.1 Réduction vers les $\langle 1, \mathbb{N}\rangle$-automates à compteur

Nous montrons d'abord que tester si $\mathrm{L}(\mathcal{A}) \neq \emptyset$ pour un $\langle 1, \mathbb{Z}\rangle$-automate à compteur $\mathcal{A}$ avec un alphabet et des ε-transitions peut se réduire au problème de l'existence d'une exécution acceptante pour une sous-classe de $\langle 1, \mathbb{N}\rangle$-automates à compteur. Nous définissons pour cela plusieurs réductions successives. La première étape consiste à se débarrasser de l'alphabet et des ε-transitions. La principale difficulté est d'éviter d'avoir toujours des ε-transitions après une certaine position de l'exécution.

Lemme 28 *Tester si le langage reconnu par un $\langle 1, \mathbb{Z}\rangle$-automate à compteur avec un alphabet et des ε-transitions est non-vide se réduit en espace logarithmique au problème de l'existence d'une exécution acceptante pour les $\langle 1, \mathbb{Z}\rangle$-automates à compteur classiques.*

Preuve Nous montrons qu'étant donné un $\langle 1, \mathbb{Z}\rangle$-automate à compteur $\mathcal{A}$ avec un alphabet et des ε-transitions, nous pouvons construire en espace logarithmique un $\langle 1, \mathbb{Z}\rangle$-automate à compteur $\mathcal{A}'$ classique tel que $\mathrm{L}(\mathcal{A})$ est non vide ssi $\mathcal{A}'$ admet une exécution acceptante. La construction de $\mathcal{A}'$ consiste à utiliser deux copies de $\mathcal{A}$ telle que la copie qui contient les états finaux n'est accessible lors d'une exécution qu'en franchissant une non-ε-transition. Puis lorsqu'un état final est visité, et si la prochaine transition est une ε-transition, on retourne dans la copie sans états finaux, ce qui nous oblige à attendre une nouvelle non-ε-transition avant de visiter à nouveau un état final. La condition de Büchi empêche alors qu'une exécution acceptante de $\mathcal{A}'$ franchisse uniquement des ε-transitions après une certaine position. Si c'était le cas, on resterait bloqué dans la copie sans états acceptants.

Formellement, l'automate $\mathcal{A}' = \langle Q', \delta', I', F'\rangle$ est défini à partir de $\mathcal{A} = \langle Q, \delta, I, F\rangle$ de la façon suivante :

- $Q' = Q \times \{0, 1\}$, $I' = I \times \{0\}$ et $F' = F \times \{1\}$,
- La relation de transition δ' est telle que :
  - pour toute transition $q \xrightarrow{a,t,u} q' \in \delta$ de $\mathcal{A}$ telle que $a \neq \varepsilon$ on a une transition $\langle q, i\rangle \xrightarrow{t,u} \langle q', 1\rangle$ pour tout $i \in \{0, 1\}$,
  - pour toute transition de la forme $q \xrightarrow{\varepsilon,t,u} q' \in \delta$ de $\mathcal{A}$
    - si $q \in F$ on a une transition $\langle q, i\rangle \xrightarrow{t,u} \langle q', 0\rangle \in \delta'$ pour tout $i \in \{0, 1\}$,
    - sinon on a une transition $\langle q, i\rangle \xrightarrow{t,u} \langle q', i\rangle \in \delta'$ pour tout $i \in \{0, 1\}$.

D'après les remarques plus haut, il est évident que cette construction vérifie que $\mathrm{L}(\mathcal{A})$ est non vide ssi $\mathcal{A}'$ admet une exécution acceptante. En effet, il existe une exécution acceptante de la forme

$$\langle q_0, 0\rangle \xrightarrow{\varepsilon,t_0,u_0} \cdots \xrightarrow{a_{i_1-1},t_{i_1-1},u_{i_1-1}} \langle q_{i_1}, c_{i_1}\rangle \xrightarrow{\varepsilon,t_{i_1},u_{i_1}} \langle q_{i_1+1}, c_{i_1+1}\rangle \rightarrow \\ \cdots \rightarrow \langle q_{i_2}, c_{i_2}\rangle \xrightarrow{a_{i_2},t_{i_2},u_{i_2}} \langle q_{i_2+1}, c_{i_2+1}\rangle \rightarrow \cdots$$

dans $\mathcal{A}$ avec $q_{i_2} \in F$ et $a_{i_1} \neq \varepsilon$, ssi il existe une exécution acceptante de la forme

$$\langle\langle q_0, 0\rangle, 0\rangle \xrightarrow{t_0,u_0} \cdots \xrightarrow{t_{i_1-1},u_{i_1-1}} \langle\langle q_{i_1}, c_{i_1}\rangle, 0\rangle \xrightarrow{t_{i_1},u_{i_1}} \langle\langle q_{i_1+1}, 1\rangle, c_{i_1+1}\rangle \rightarrow \\ \cdots \rightarrow \langle\langle q_{i_2}, 1\rangle, c_{i_2}\rangle \xrightarrow{t_{i_2},u_{i_2}} \langle\langle q_{i_2+1}, b\rangle, c_{i_2+1}\rangle \rightarrow \cdots$$

dans $\mathcal{A}'$ où $b = 1$ si $a_{i_2} \neq \varepsilon$ et $b = 0$ sinon. Nous omettons les étiquettes des transitions qui n'influent pas l'état de contrôle d'arrivée. $\square$

Nous réduisons maintenant le problème de l'existence d'une exécution acceptante pour les $\langle 1, \mathbb{Z} \rangle$-automates à compteur au même problème pour les $\langle 1, \mathbb{N} \rangle$-automates à compteur, plus simples à étudier.

Lemme 29 *Le problème de l'existence d'une exécution acceptante pour les $\langle 1, \mathbb{Z} \rangle$-automates à compteur se réduit en espace logarithmique au même problème pour les $\langle 1, \mathbb{N} \rangle$-automates à compteur.*

Preuve Nous montrons qu'étant donné un $\langle 1, \mathbb{Z} \rangle$-automate à compteur simple $\mathcal{A}$ nous pouvons construire en espace logarithmique un $\langle 1, \mathbb{N} \rangle$-automate à compteur simple $\mathcal{A}'$ tel que $\mathcal{A}$ admet une exécution acceptante ssi $\mathcal{A}'$ admet aussi une exécution acceptante. Ici encore, la construction de $\mathcal{A}'$ est basée sur deux copies différentes de l'automate initial $\mathcal{A}$. Une copie est utilisée pour simuler les valeurs positives du compteur et l'autre les valeurs négatives. Le passage d'une copie à l'autre ne se fait que lorsque le compteur vaut zéro et le signe change. Nous pouvons tester dans quelle copie se trouve l'exécution en utilisant les tests $x > 0$ et $x < 0$.

Formellement, l'automate $\mathcal{A}' = \langle Q', \delta', I', F' \rangle$ est défini à partir de $\mathcal{A} = \langle Q, \delta, I, F \rangle$ de la façon suivante :

- $Q' = Q \times \{+, -\}$, $I' = I \times \{+\}$, $F' = F \times \{+, -\}$,
- La relation de transition δ' et définie à partir de δ de la manière suivante :
  - pour toute transition $q \xrightarrow{t,0} q' \in \delta$ de $\mathcal{A}$ telle que $t \in \{=, \neq\}$, on a des transitions
    - $\langle q, - \rangle \xrightarrow{t,0} \langle q', - \rangle \in \delta'$ et
    - $\langle q, + \rangle \xrightarrow{t,0} \langle q', + \rangle \in \delta'$,
  - pour toute transition $q \xrightarrow{>,0} q' \in \delta$ de $\mathcal{A}$, on a une transition $\langle q, + \rangle \xrightarrow{\neq,0} \langle q', + \rangle$,
  - pour toute transition $q \xrightarrow{<,0} q' \in \delta$ de $\mathcal{A}$, on a une transition $\langle q, - \rangle \xrightarrow{\neq,0} \langle q', - \rangle$,
  - pour toute transition $q \xrightarrow{=,-1} q' \in \delta$ de $\mathcal{A}$, on a des transitions
    - $\langle q, + \rangle \xrightarrow{=,+1} \langle q', - \rangle \in \delta'$ et
    - $\langle q, - \rangle \xrightarrow{=,+1} \langle q', - \rangle \in \delta'$,
  - pour toute transition $q \xrightarrow{>,-1} q' \in \delta$ de $\mathcal{A}$, on a une transition $\langle q, + \rangle \xrightarrow{\neq,-1} \langle q', + \rangle \in \delta'$,
  - pour toute transition $q \xrightarrow{<,-1} q' \in \delta$ de $\mathcal{A}$, on a une transition $\langle q, - \rangle \xrightarrow{\neq,+1} \langle q', - \rangle \in \delta'$,
  - pour toute transition $q \xrightarrow{\top,-1} q' \in \delta$ de $\mathcal{A}$, on a des transitions
    - $\langle q, + \rangle \xrightarrow{\neq,-1} \langle q', + \rangle \in \delta'$,
    - $\langle q, + \rangle \xrightarrow{=,+1} \langle q', - \rangle \in \delta'$ et

- $\langle q, - \rangle \xrightarrow{\top,+1} \langle q', - \rangle \in \delta'$,
- pour toute transition $q \xrightarrow{\neq,-1} q' \in \delta$, on a des transitions
  - $\langle q, + \rangle \xrightarrow{\neq,-1} \langle q', + \rangle \in \delta'$ et
  - $\langle q, - \rangle \xrightarrow{\neq,+1} \langle q', - \rangle \in \delta'$,
- les cas restants concernant l'incrémentation $+1$ peuvent être traités de façon symétrique.

Il est facile de montrer que $\mathcal{A}$ admet une exécution acceptante ssi $\mathcal{A}'$ admet une exécution acceptante en analysant cette construction et utilisant les remarques du début de la preuve. Par exemple, l'exécution suivante de $\mathcal{A}$

$$\langle q_0, 0 \rangle \xrightarrow{\top,+1} \langle q_1, 1 \rangle \xrightarrow{>,-1} \langle q_2, 0 \rangle \xrightarrow{=,-1} \langle q_3, -1 \rangle \xrightarrow{\top,-1} \langle q_4, -2 \rangle \xrightarrow{<,+1} \langle q_5, -1 \rangle \rightarrow \cdots$$

correspond à l'exécution de $\mathcal{A}'$

$$\begin{aligned} \langle \langle q_0, + \rangle, 0 \rangle &\xrightarrow{\top,+1} \langle \langle q_1, + \rangle, 1 \rangle \xrightarrow{>,-1} \langle \langle q_2, + \rangle, 0 \rangle \xrightarrow{=,+1} \langle \langle q_3, - \rangle, 1 \rangle \\ &\xrightarrow{\top,+1} \langle \langle q_4, - \rangle, 2 \rangle \xrightarrow{<,-1} \langle \langle q_5, - \rangle, 1 \rangle \rightarrow \cdots \end{aligned}$$

□

Pour terminer, nous simplifions une dernière fois le problème en restreignant la classe de $\langle 1, \mathbb{N} \rangle$-automates à compteur que nous considérons.

Lemme 30 *Le problème de l'existence d'une exécution acceptante pour les $\langle 1, \mathbb{N} \rangle$-automates à compteur se réduit en espace logarithmique au même problème pour des $\langle 1, \mathbb{N} \rangle$-automates sans test de la forme $x \neq 0$.*

Pour prouver ce résultat il suffit d'étendre l'ensemble des états du $\langle 1, \mathbb{N} \rangle$-automates de départ et de remplacer chaque transition de la forme $q \xrightarrow{\neq,u} q'$ par les transitions $q \xrightarrow{\top,-1} q_1 \xrightarrow{\top,+1} q_2 \xrightarrow{\top,u} q'$ où q_1 et q_2 sont de nouveaux états. En effet, cette suite de transitions ne peut être franchie sans avoir de valeur négative si le compteur est égal à zéro.

5.4.2 Exécutions sans test à zéro

Les résultats qui précèdent impliquent que tester si le langage reconnu par un $\langle 1, \mathbb{Z} \rangle$-automate à compteur avec un alphabet et des ε-transitions est non-vide se réduit en espace logarithmique au problème de l'existence d'une exécution acceptante pour un $\langle 1, \mathbb{N} \rangle$-automate à compteur sans test de la forme $x \neq 0$. En effet, la composition de réductions en espace logarithmique est une réduction en espace logarithmique. Nous nous restreignons donc maintenant à la classe de $\langle 1, \mathbb{N} \rangle$-automates à compteur où les seuls tests autorisés sont $x = 0$ et $\top$.

Soit $\mathcal{A} = \langle Q, \delta, I, F \rangle$ un $\langle 1, \mathbb{N} \rangle$-automate vérifiant ces propriétés. Une exécution $\pi = \langle q_0, c_0 \rangle \xrightarrow{t_0,u_0} \langle q_1, c_1 \rangle \xrightarrow{t_1,u_1} \ldots$ est sans test à zéro ssi pour tout $i \in \mathbb{N}$ on a $t_i = \top$. La même définition est valable pour les sous-exécutions finies. Nous traitons d'abord le cas de ce type de sous-exécutions en montrant plusieurs propriétés utiles pour la suite.

Lemme 31 *Soit* $\pi = \langle q_1, c_1 \rangle \xrightarrow{t_1, u_1} \dots \xrightarrow{t_{|\pi|-1}, u_{|\pi|-1}} \langle q_{|\pi|}, c_{|\pi|} \rangle$ *une sous-exécution finie sans test à zéro. Il existe une sous-exécution finie* $\pi' = \langle q'_1, c'_1 \rangle \xrightarrow{t'_1, u'_1} \dots \xrightarrow{t'_{|\pi'|-1}, u'_{|\pi'|-1}} \langle q'_{|\pi'|}, c'_{|\pi'|} \rangle$ *telle que*

1. $q'_1 = q_1$,
2. $q_{|\pi|} = q'_{|\pi'|}$,
3. $c'_1 = c_1$ *et*
4. $|\pi'| \leq |Q|^2 + 2|Q| + 1$.

Preuve : D'abord, remarquons que s'il existe deux indices i et j dans π tels que $q_i = q_j$ et $c_i \geq c_j$ la partie $\langle q_i, c_i \rangle \to \cdots \to \langle q_j, c_j \rangle$ peut être supprimée. Les valeurs qui suivent la configuration $\langle q_j, c_j \rangle$ dans π ont besoin d'être mises à jour en conséquence (en enlevant $c_i - c_j$ aux valeurs successives du compteur) mais comme il est évident que les nouvelles valeurs obtenues pour le compteur sont supérieures, toutes les transitions de la fin de π peuvent être franchies sans avoir de valeurs négatives pour le compteur. Notons que ceci ne peut se faire qu'en l'absence de test $x = 0$, sinon on peut être bloqué par un tel test. Nous supposons donc, sans perte de généralité, que pour tout couple d'indice i, j de π tels que $q_i = q_j$ on a $c_i < c_j$.

S'il n'existe pas d'indices i, j tels que $q_i = q_j$ dans π, alors la taille du chemin est bornée par $|Q|$. Donc nous posons $\pi = \pi'$.

Sinon, il existe un unique couple d'indices i_0, j_0 tel que

- $i_0 < j_0$,
- $q_{i_0} = q_{j_0}$
- les séquences $q_1, \dots, q_{i_0}$ et $q_{i_0}, \dots, q_{j_0-1}$ sont composées d'états de contrôle distincts.

Ces indices correspondent au premier état de π qui se répète. Notons que d'après l'hypothèse ci-dessus sur les états de contrôle qui se répètent dans π nous avons aussi $c_{\text{diff}} = c_{j_0} - c_{i_0} > 0$. En examinant cette définition, on peut facilement montrer que $i_0 \leq |Q|$ de même $j_0 - i_0 \leq |Q|$ car les états avant i_0 et entre i_0 et j_0 doivent être distincts.

Puisque $\langle q_{j_0}, c_{j_0} \rangle \xrightarrow{t_{j_0}, u_{j_0}} \dots \xrightarrow{t_{|\pi|-1}, u_{|\pi|-1}} \langle q_{|\pi|}, c_{|\pi|} \rangle$ est une sous-exécution de $\mathcal{A}$ sans test à zéro, il doit exister un chemin $q'_1 \to q'_2 \to \dots \to q'_m$ tel que $q'_1 = q_{j_0}$ et $q'_m = q_{|\pi|}$ de longueur inférieure à $|Q|$ dans le graphe orienté $G = \langle Q, \{\langle r, s \rangle : \langle r, \top, u, s \rangle \in \delta\} \rangle$ qui est le graphe des états de contrôle accessibles dans $\mathcal{A}$ sans effectuer de test $x = 0$. Nous utilisons alors l'observation suivante sur les $\langle 1, \mathbb{N} \rangle$-automates à compteur :

S'il existe un chemin de longueur c entre un état q et un autre état q' dans G alors on peut atteindre l'état de contrôle q' à partir de la configuration $\langle q, c \rangle$ dans $\mathcal{A}$.

En effet, la valeur du compteur est suffisamment élevée pour pouvoir franchir c transitions quelles que soient les mises à jour (le pire cas étant c décrémentations) et on n'a pas de test $x = 0$ qui pourrait nous bloquer par définition de G.

Nous construisons alors l'exécution finie π' comme la séquence définie par la suite de sous-exécutions suivantes :

- $\langle q_1, c_1\rangle \xrightarrow{t_1,u_1} \ldots \xrightarrow{t_{i_0-1},u_{i_0-1}} \langle q_{i_0}, c_{i_0}\rangle \xrightarrow{t_{i_0},u_{i_0}} \ldots \xrightarrow{t_{j_0-1},u_{j_0-1}} \langle q_{j_0}, c_{j_0}\rangle$,
- $\langle q_{i_0}, c_{i_0} + c_{\text{diff}}\rangle \xrightarrow{t_{i_0},u_{i_0}} \ldots \xrightarrow{t_{j_0-1},u_{j_0-1}} \langle q_{j_0}, c_{j_0} + c_{\text{diff}}\rangle$,
- $\langle q_{i_0}, c_{i_0} + 2c_{\text{diff}}\rangle \xrightarrow{t_{i_0},u_{i_0}} \ldots \xrightarrow{t_{j_0-1},u_{j_0-1}} \langle q_{j_0}, c_{j_0} + 2c_{\text{diff}}\rangle$,

$\vdots$

- $\langle q_{i_0}, c_{i_0} + (|Q|-1)c_{\text{diff}}\rangle, \xrightarrow{t_{i_0},u_{i_0}} \ldots \xrightarrow{t_{j_0-1},u_{j_0-1}} \langle q_{j_0}, c_{j_0} + (|Q|-1)c_{\text{diff}}\rangle$,
- $\langle q_{j_0}, c_{j_0} + |Q|c_{\text{diff}}\rangle \xrightarrow{t'_1,u'_1} \langle q'_2, c'_2\rangle \xrightarrow{t'_2,u'_2} \ldots \xrightarrow{t'_{m-1},u'_{m-1}} \langle q'_m, c'_m\rangle$.

Ce chemin est bien défini car $q_{i_0} = q_{j_0} = q'_1$, $c_{i_0} + c_{\text{diff}} = c_{j_0}$ et la valeur du compteur est assez grande pour parcourir la dernière partie qui correspond au chemin dans G. Cette dernière propriété est assurée par la répétition du sous-chemin entre i_0 et j_0.

La restriction de π' par rapport aux états de contrôles visités est la suivante :

$$\overbrace{q_1 \ldots q_{i_0-1}}^{\text{longueur } \leq |Q|}(\overbrace{q_{i_0} q_{i_0+1} \ldots q_{j_0-1} q_{j_0}}^{\text{longueur } \leq |Q|})^{|Q|} \quad \overbrace{q'_2 \ldots q'_m}^{\text{longueur } \leq |Q|} \quad .$$

ce qui montre bien que la taille de ce chemin est inférieure à $|Q|^2 + 2|Q| + 1$. □

Notons que dans cette construction, en répétant un plus grand nombre de fois la boucle centrale du chemin, nous pouvons obtenir une valeur aussi grande que l'on veut pour la valeur finale du compteur. Ceci nous permet en particulier d'établir qu'à partir de π et d'un entier $d > 0$ on peut construire une exécution π' telle que $q'_1 = q_1$, $q_{|\pi|} = q'_{|\pi'|}$ et $c'_{|\pi|} > c_{|\pi|} + d$ dont la taille est bornée en fonction de $|Q|$ et d.

Corollaire 11 *Soit $\pi = \langle q_1, c_1\rangle \xrightarrow{t_1,u_1} \ldots \xrightarrow{t_{|\pi|-1},u_{|\pi|-1}} \langle q_{|\pi|}, c_{|\pi|}\rangle$ une sous-exécution finie sans test à zéro et $d \in \mathbb{N}$. Il existe une sous-exécution finie $\pi' = \langle q'_1, c'_1\rangle \xrightarrow{t'_1,u'_1} \ldots \xrightarrow{t'_{|\pi'|-1},u'_{|\pi'|-1}} \langle q'_{|\pi'|}, c'_{|\pi'|}\rangle$ telle que*

1. $q'_1 = q_1$,
2. $q_{|\pi|} = q'_{|\pi'|}$,
3. $c'_{|\pi|} > c_{|\pi|} + d$ *et*
4. $|\pi'| \leq |Q|^2 + (2+d)|Q|$.

Nous considérons maintenant les exécutions sans tests à zéro qui comportent un état final répété infiniment souvent. Le but est d'extraire un cycle ayant pour origine cet état final et qui peut se répéter infiniment souvent. En effet, à partir d'un tel cycle on peut construire une exécution vérifiant la condition d'acceptation de Büchi.

Lemme 32 *Soit $\pi = \langle q_1, c_1\rangle \xrightarrow{t_1,u_1} \langle q_2, c_2\rangle \ldots$ une exécution infinie sans test à zéro telle qu'il existe un état final $q_f \in F$ visité infiniment souvent par π. Il existe un indice $i \in \mathbb{N}$ et une sous-exécution finie $\pi' = \langle q'_1, c'_1\rangle \xrightarrow{t'_1,u'_1} \ldots \xrightarrow{t'_{|\pi'|-1},u'_{|\pi'|-1}} \langle q'_{|\pi'|}, c'_{|\pi'|}\rangle$ tels que $|\pi'| > 1$ et $\langle q'_1, c'_1\rangle = \pi(i)$ vérifiant*

1. $q'_1 = q'_{|\pi'|} = q_f$,

2. $c'_{|\pi'|} \geq c'_1$

3. *et* $|\pi'| \leq 2|Q|^2 + 3|Q|$.

Preuve : La preuve de ce résultat a de nombreuses similitudes avec celle du précédent. Puisque l'ordre $\langle \mathbb{N}, < \rangle$ est bien fondé et que l'ensemble des états Q est fini, il existe deux indices i_0, j_0 dans π tels que

- $i_0 < j_0$,
- $q_{i_0} = q_{j_0} = q_f$ et
- $c_{i_0} \leq c_{j_0}$.

La sous-exécution π_0 de π correspondant à la partie entre ces indices vérifie presque toute les propriétés de l'énoncé. Seule la taille de π_0 peut poser problème. Nous procédons alors de la même manière que dans la preuve du Lemme 31. Sans perte de généralité, nous supposons que pour tout couple d'indice $i, j \in \{i_0, \ldots, j_0\}$ de π tels que $q_i = q_j$ on a $c_i < c_j$. En effet si $c_i \geq c_j$, la sous-exécution correspondante peut être supprimée (nous avons alors besoin de mettre à jour la valeur du compteur en conséquence).

S'il n'existe pas d'indices $i, j \in \{i_0, i_0 + 1, \ldots, j_0\}$ tels que $q_i = q_j$ dans π, alors la taille du chemin est bornée par $|Q|$ et nous posons $\pi' = \pi_0$.

Sinon, il existe un unique couple d'indices $i_1, j_1 \in \{i_0, i_0 + 1, \ldots, j_0\}$ tel que

- $i_1 < j_1$,
- $q_{i_1} = q_{j_1}$
- les séquences $q_{i_0}, \ldots, q_{i_1}$ et $q_{i_1}, \ldots, q_{j_1 - 1}$ sont composées d'états de contrôle distincts.

Ces indices correspondent au premier état de π_0 qui se répète. Nous posons $c_{\text{diff}} = c_{j_1} - c_{i_1} > 0$. De plus, cette définition implique que $i_1 - i_0 \leq |Q|$ et $j_1 - i_1 \leq |Q|$.

Comme $\langle q_{j_1}, c_{j_1} \rangle \xrightarrow{t_{j_1}, u_{j_1}} \ldots \xrightarrow{t_{j_0 - 1}, u_{j_0 - 1}} \langle q_{j_0}, c_{j_0} \rangle$ est une sous-exécution sans test à zéro de $\mathcal{A}$ il doit exister un chemin $q'_1 \to q'_2 \to \ldots \to q'_m$ de longueur inférieure à $|Q|$ dans le graphe orienté $G = \langle Q, \{\langle r, s \rangle : \langle r, \top, u, s \rangle \in \delta\} \rangle$. L'observation faite à propos de G dans la preuve du Lemme 31 nous permet de construire la sous-exécution π' comme la succession des séquences :

- $\langle q_{i_0}, c_{i_0} \rangle \xrightarrow{t_{i_0}, u_{i_0}} \ldots \xrightarrow{t_{i_1 - 1}, u_{i_1 - 1}} \langle q_{i_1}, c_{i_1} \rangle \xrightarrow{t_{i_1}, u_{i_1}} \ldots \xrightarrow{t_{j_1 - 1}, u_{j_1 - 1}} \langle q_{j_1}, c_{j_1} \rangle$,
- $\langle q_{i_1}, c_{i_1} + c_{\text{diff}} \rangle \xrightarrow{t_{i_1}, u_{i_1}} \ldots \xrightarrow{t_{j_1 - 1}, u_{j_1 - 1}} \langle q_{j_1}, c_{j_1} + c_{\text{diff}} \rangle$,
- $\langle q_{i_0}, c_{i_0} + 2c_{\text{diff}} \rangle \xrightarrow{t_{i_1}, u_{i_1}} \ldots \xrightarrow{t_{j_1 - 1}, u_{j_1 - 1}} \langle q_{j_1}, c_{j_1} + 2c_{\text{diff}} \rangle$,

$\vdots$

- $\langle q_{i_0}, c_{i_0} + 2|Q|c_{\text{diff}} \rangle \xrightarrow{t_{i_1}, u_{i_1}} \ldots \xrightarrow{t_{j_1 - 1}, u_{j_1 - 1}} \langle q_{j_1}, c_{j_1} + 2|Q|c_{\text{diff}} \rangle$,
- $\langle q_{j_1}, c_{i_0} + (2|Q| + 1)c_{\text{diff}} \rangle \xrightarrow{t'_1, u'_1} \langle q'_2, c'_2 \rangle \xrightarrow{t'_2, u'_2} \ldots \xrightarrow{t'_{m-1}, u'_{m-1}} \langle q'_m, c'_m \rangle$.

Ce chemin est bien défini car $q_{i_1} = q_{j_1} = q'_1$, $c_{i_1} + c_{\text{diff}} = c_{j_1}$ et la valeur du compteur est assez grande pour parcourir la dernière partie qui correspond au chemin dans G.

La restriction de π' aux état de contrôle est la suivante :

$$\overbrace{q_1 \cdots q_{i_0-1}}^{\text{longueur } \leq |Q|} (\overbrace{q_{i_0} q_{i_0+1} \cdots q_{j_0-1} q_{j_0}}^{\text{longueur } \leq |Q|})^{2|Q|+1} \overbrace{q'_2 \cdots q'_m}^{\text{longueur } \leq |Q|} .$$

Il est évident que la longueur de ce chemin est bornée par $2|Q|^2 + 3|Q|$. Pour montrer que $c'_m > c_{i_0}$ nous étudions les mises à jour du compteur. Dans le pire des cas, la première partie et la dernière décrémentent $|Q|$ fois le compteur chacune. Or la partie du milieu, par construction, incrémente le compteur de $2|Q|c_{\text{diff}}$. Nous pouvons donc conclure que $c'_m > c_{i_0}$. □

5.4.3 Exécutions avec tests à zéro

Nous étudions maintenant les exécutions qui comportent des tests à zéro (de la forme $x = 0$). Nous rappelons d'abord un résultat qui nous sera utile ensuite.

Lemme 33 *[LLT05] Soit $\pi = \langle q_1, c_1 \rangle \xrightarrow{t_1,u_1} \dots \xrightarrow{t_{|\pi|-1},u_{|\pi|-1}} \langle q_{|\pi|}, c_{|\pi|} \rangle$ une exécution finie de $\mathcal{A}$ telle que $\langle q_2, c_2 \rangle \xrightarrow{t_2,u_2} \dots \xrightarrow{t_{|\pi|-1},u_{|\pi|-1}} \langle q_{|\pi|}, c_{|\pi|} \rangle$ est sans test à zéro et $c_{|\pi|} = c_1 = 0$. Il existe une exécution finie $\pi' = \langle q'_1, c'_1 \rangle \xrightarrow{t'_1,u'_1} \dots \xrightarrow{t'_{|\pi'|-1},u'_{|\pi'|-1}} \langle q'_{|\pi'|}, c'_{|\pi'|} \rangle$ telle que*

1. $q'_1 = q_1$,
2. $q'_{|\pi'|} = q_{|\pi|}$,
3. $c'_{|\pi'|} = c'_1 = 0$ *et*
4. *pour tout* $i \in \{1, \dots, |\pi'|\}$ *on a* $c'_i \leq |Q|^3 + |Q|^2$.

Preuve : Nous supposons sans perte de généralité qu'entre la première et la dernière position de π où la valeur du compteur est supérieure à $|Q|^2$, la valeur du compteur reste toujours supérieure à $|Q|^2$. Si le compteur repasse plusieurs fois en dessous de $|Q|^2$, le chemin π peut être découpé en plusieurs sous-chemins vérifiant cette propriété qui sont chacun traités de la même manière que ci dessous.

Supposons qu'il existe une position $i_{\max}$ de π où la configuration $\pi(i_{\max}) = \langle q_i, c_{i_{\max}} \rangle$ est telle que $c_{i_{\max}} > |Q|^3 + |Q|^2$. Nous allons supprimer des morceaux de chemin afin de diminuer la valeur du compteur. Nous utilisons l'observation suivante :

Dans tout chemin fini $\langle q_0, c_0 \rangle \to \cdots \langle q_{|\pi|}, c_s \rangle$ tel que $c_s = c_0 + |Q|$, il existe un sous-chemin de la forme $\langle q_i, c \rangle \to \cdots \to \langle q_j, c + d \rangle$ tel que $q_i = q_j$ et $0 < d \leq |Q|$.

Cette affirmation est facile à prouver considérant qu'une transition incrémente le compteur d'au plus 1 et que par conséquent $s \geq |Q|$.

Considérons le sous-chemin π^+ entre la première position où le compteur vaut $|Q|^2$ et la dernière position avant $i_{\max}$ où le compteur vaut $|Q|^3 + |Q|^2$. Nous pouvons découper ce chemin en $|Q|^2$ parties $\pi^+_1, \dots, \pi^+_{|Q|^2}$ telles que pour tout $1 \leq i \leq |Q|^2$ le chemin π^+_i débute dans une configuration $\langle q^+_i, c^+_i \rangle$ et se termine dans une configuration de la forme

$\langle q^+_{i+1}, c^+_i + |Q|\rangle$. Ces chemins vérifient les conditions de notre observation précédente. Il existe donc dans chaque π^+_i un sous-chemin de la forme $\langle q, c\rangle \rightarrow^* \langle q, c + d^+_i\rangle$. Comme pour tout $1 \leq i \leq |Q|^2$ on a $0 < d^+_i \leq |Q|$, il existe une valeur d^+ qui apparaît au moins $|Q|$ fois dans $\{d_1, \dots, d_{|Q|^2}\}$. On peut découper de la même façon le sous-chemin entre la dernière position après $i_{\max}$ où le compteur vaut $|Q|^3 + |Q|^2$ et la première position après $i_{\max}$ où le compteur vaut $|Q|^2$ et trouver une valeur d^- qui apparaît au moins $|Q|$ fois dans les sous-chemins de la forme $\langle q, c\rangle \rightarrow^* \langle q, c - d^-_i\rangle$.

Nous pouvons supprimer alors d^- chemins de la forme $\langle q, c\rangle \rightarrow^* \langle q, c + d^+\rangle$ dans π^+ et d^+ chemins de la forme $\langle q, c\rangle \rightarrow^* \langle q, c - d^-\rangle$ dans π^- en mettant les valeurs du compteur à jour en conséquence. Le fait que nous considérons des parties du chemin où la valeur du compteur est supérieure à $|Q|^2$ assure que la valeur du compteur reste positive après ces suppressions. En effet, cette construction décrémente le compteur d'une position donnée d'au plus $d^+ \times d^-$.

Nous pouvons réitérer cette opération tant qu'il existe une position où la valeur du compteur dépasse la borne. $\square$

La preuve du résultat suivant est une conséquence directe du Lemme [LLT05, Lemme 42].

Lemme 34 *Soit $\pi = \langle q_1, c_1\rangle \xrightarrow{t_1, u_1} \dots \xrightarrow{t_{|\pi|-1}, u_{|\pi|-1}} \langle q_{|\pi|}, c_{|\pi|}\rangle$ une exécution finie de $\mathcal{A}$ telle que $c_{|\pi|} = c_1 = 0$. Il existe une exécution finie $\pi' = \langle q'_1, c'_1\rangle \xrightarrow{t'_1, u'_1} \dots \xrightarrow{t'_{|\pi'|-1}, u'_{|\pi'|-1}} \langle q'_{|\pi'|}, c'_{|\pi'|}\rangle$ telle que*

1. $q'_1 = q_1$,
2. $q'_{|\pi'|} = q_{|\pi|}$,
3. $c'_{|\pi'|} = c'_1 = 0$ *et*
4. $|\pi'| \leq |Q|^5 + |Q|^4$.

Preuve : Soit π un chemin vérifiant les propriétés de l'énoncé. Nous posons $1 = i_1 < i_2 < \dots < i_m = n$ l'ensemble des indices du chemin tels que $c_{i_1} = c_{i_2} = \dots = c_{i_m} = 0$ et aucune autre configuration n'a pour valeur du compteur zéro. Notons que les sous-exécutions cycliques de la forme $\langle q, c\rangle \rightarrow \dots \rightarrow \langle q, c\rangle$ peuvent être supprimées de l'exécution. Sans perte de généralité, nous supposons donc que $m \leq |Q|$. Pour tout $1 \leq j \leq m$, les sous-exécutions de π entre la position i_j et la position i_{j+1} vérifient les conditions du Lemme 33. Donc pour tout j il existe sous-exécution π'_j commençant dans la configuration $\langle q_{i_j}, 0\rangle$ et terminant dans la configuration $\langle q'_{i_{j+1}}, 0\rangle$ telle que la valeur du compteur est bornée par $|Q|^3 + |Q|^2$. Comme les cycles peuvent être éliminés, nous pouvons supposer que la longueur de π'_j est bornée par $|Q| \times (|Q|^3 + |Q|^2)$. Le chemin π' obtenu en concaténant les chemins $\pi_1, \dots, \pi_{m-1}$ vérifie toutes les conditions requises. En particulier, sa taille est bornée par $(m-1) \times |Q| \times (|Q|^3 + |Q|^2) \leq |Q|^5 + |Q|^4$. $\square$

Nous considérons enfin les exécutions acceptantes qui comportent une infinité de tests à zéro. Nous adaptons d'abord le Lemme 33.

Lemme 35 *Soient* $q_f \in F$ *un état final et* $\pi = \langle q_1, c_1\rangle \xrightarrow{t_1,u_1} \ldots \xrightarrow{t_{|\pi|-1},u_{|\pi|-1}} \langle q_{|\pi|}, c_{|\pi|}\rangle$ *une exécution finie de* $\mathcal{A}$ *telle que* $\langle q_2, c_2\rangle \xrightarrow{t_2,u_2} \ldots \xrightarrow{t_{|\pi|-1},u_{|\pi|-1}} \langle q_{|\pi|}, c_{|\pi|}\rangle$ *est sans test à zéro,* $c_{|\pi|} = c_1 = 0$ *et* $q_f \in \{q_1, \ldots, q_{|\pi|}\}$*. Il existe une exécution finie* $\pi' = \langle q'_1, c'_1\rangle \xrightarrow{t'_1,u'_1} \ldots \xrightarrow{t'_{|\pi'|-1},u'_{|\pi'|-1}} \langle q'_{|\pi'|}, c'_{|\pi'|}\rangle$ *telle que*

1. $q'_1 = q_1$,
2. $q'_{|\pi'|} = q_{|\pi|}$,
3. $c'_{|\pi'|} = c'_1 = 0$, $q_f \in \{q'_1, \ldots, q'_n\}$ *et*
4. *pour tout* $i \in \{1, \ldots, |\pi'|\}$ *on a* $c'_i \leq 2|Q|^3 + |Q|^2$.

Preuve : Nous développons uniquement les arguments qui permettent d'adapter la preuve de [LLT05, Lemma 42] (voir le Lemme 33).

La différence avec l'énoncé du Lemme 33 est la présence de q_f dans l'exécution. Nous ne pouvons pas supprimer des sous-chemins comme bon nous semble au risque de supprimer toutes les occurrences de q_f. C'est pourquoi nous avons besoin d'un intervalle supplémentaire entre les valeurs $|Q|^3 + |Q|^2$ et $2|Q|^3 + |Q|^2$. Ainsi nous avons la possibilité de procéder aux suppressions de sous-chemins soit dans la partie du chemin où la valeur du compteur est comprise entre $|Q|^3 + |Q|^2$ et $2|Q|^3 + |Q|^2$, soit dans la partie du chemin où la valeur du compteur est comprise entre $|Q|^2$ et $|Q|^3 + |Q|^2$. Ceci nous permet de ne pas supprimer une occurrence de q_f que nous voulons garder en choisissant de supprimer dans une autre partie. □

Nous établissons enfin un résultat qui nous permet de traiter la répétition d'un état final dans les exécutions acceptantes qui comportent une infinité de tests à zéro.

Lemme 36 *Soit* $q_f \in F$ *un état final et* $\pi = \langle q_1, c_1\rangle \xrightarrow{t_1,u_1} \ldots \xrightarrow{t_{|\pi|-1},u_{|\pi|-1}} \langle q_{|\pi|}, c_{|\pi|}\rangle$ *une exécution finie de* $\mathcal{A}$ *telle que* $c_{|\pi|} = c_1 = 0$ *et* $q_f \in \{q_1, \ldots, q_n\}$*. Il existe une exécution finie* $\pi' = \langle q'_1, c'_1\rangle \xrightarrow{t'_1,u'_1} \ldots \xrightarrow{t'_{|\pi'|-1},u'_{|\pi'|-1}} \langle q'_{|\pi'|}, c'_{|\pi'|}\rangle$ *telle que*

1. $q'_1 = q_1$,
2. $q'_{|\pi'|} = q_{|\pi|}$,
3. $c'_{|\pi'|} = c'_1 = 0$, $q_f \in \{q'_1, \ldots, q'_n\}$ *et*
4. $|\pi'| \leq 2|Q|^5 + 4|Q|^4 + |Q|^3$.

Preuve : Nous procédons de la même façon que dans le Lemme 34. Nous notons $1 = i_1 < i_2 < \ldots < i_m = n$ les indices du chemin tels que $c_{i_1} = c_{i_2} = \cdots = c_{i_m} = 0$. Aucune autre configuration n'a pour valeur du compteur zéro. Comme les sous-exécutions cycliques de la forme $\langle q, c\rangle \to \ldots \to \langle q, c\rangle$ peuvent être supprimées de l'exécution, nous supposons donc que $m \leq |Q|$. Il existe un entier $j \in \{1, \ldots, m\}$ tel qu'une configuration de la forme $\langle q_f, c\rangle$ appartient à la sous-exécution entre la position i_j et i_{j+1}. Cette sous-exécution vérifie les conditions du Lemme 35. Il existe donc une sous-exécution π'_1 commençant dans la configuration $\langle q_{i_j}, 0\rangle$ et terminant dans la configuration $\langle q'_{i_{j+1}}, 0\rangle$ telle que la valeur du compteur est bornée par $2|Q|^3+|Q|^2$. Comme les cycles peuvent être éliminés, nous pouvons supposer que la longueur de π'_1 est bornée par $|Q| \times (2|Q|^3 + |Q|^2)$.

Les sous-exécutions entre la position 0 et i_j et entre la position i_{j+1} et $|\pi|$ vérifie les conditions du Lemme 34. Pour ces sous-exécutions il existe donc des sous-exécutions π'_0 et π'_2 chacun de longueur bornée par $|Q|^5 + |Q|^4$ allant respectivement de $\langle q_0, 0\rangle$ à $\langle q_{i_j}, 0\rangle$ et de $\langle q_{i_j}, 0\rangle$ à $\langle q_{|\pi|}, 0\rangle$.

Le chemin $\pi = \pi'_0 \cdot \pi'_1 \cdot \pi'_2$ obtenu par concaténation de ces différentes exécutions finies vérifie les conditions requises. □

5.4.4 Borne sur la taille des exécutions acceptantes

Nous disposons maintenant de tous les éléments pour établir une borne sur la taille minimale d'un témoin fini d'une exécution acceptante dans un $\langle 1, \mathbb{N}\rangle$-automate à compteur sans test $x \neq 0$.

Lemme 37 *Soit $P_1(x) = x^5 + x^4 + x^3 + 9/2x^2 + 5x$ et $P_2(x) = 3x^5 + 5x^4 + x^3$ deux polynômes. Pour tout $\langle 1, \mathbb{N}\rangle$-automate à compteur $\mathcal{A}$ sans test de la forme $x \neq 0$, il existe une exécution acceptante pour $\mathcal{A}$ ssi l'une des propositions suivantes est vraie :*

($\star$) *Il existe une exécution finie $\pi = \langle q_1, c_1\rangle \xrightarrow{t_1,u_1} \dots \xrightarrow{t_{|\pi|-1},u_{|\pi|-1}} \langle q_{|\pi|}, c_{|\pi|}\rangle$ et un indice $i_0 < |\pi|$ tels que*

1. $|\pi| \leq P_1(|Q|)$
2. $q_1 \in I$ *et* $c_1 = 0$,
3. $q_{i_0} = q_{|\pi|} \in F$ *sont des états finaux,*
4. $c_{i_0} \leq c_{|\pi|}$,
5. *la sous-exécution $\pi' = \langle q_{i_0}, c_{i_0}\rangle \xrightarrow{t_{i_0},u_{i_0}} \dots \xrightarrow{t_{|\pi|-1},u_{|\pi|-1}} \langle q_{|\pi|}, c_{|\pi|}\rangle$ est sans test à zéro.*

($\star\star$) *Il existe une exécution finie $\pi = \langle q_1, c_1\rangle \xrightarrow{t_1,u_1} \dots \xrightarrow{t_{|\pi|-1},u_{|\pi|-1}} \langle q_{|\pi|}, c_{|\pi|}\rangle$ et un indice $i_0 < |\pi|$ tels que*

1. $|\pi| \leq P_2(|Q|)$
2. $q_1 \in I$ *et* $c_1 = 0$,
3. $\{q_{i_0}, \dots, q_{|\pi|}\} \cap F \neq \emptyset$
4. $c_{i_0} = c_{|\pi|} = 0$
5. *la sous-exécution $\pi' = \langle q_{i_0}, c_{i_0}\rangle \xrightarrow{t_{i_0},u_{i_0}} \dots \xrightarrow{t_{|\pi|-1},u_{|\pi|-1}} \langle q_{|\pi|}, c_{|\pi|}\rangle$ contient des tests à zéro $x = 0$.*

Preuve Si ($\star$) ou ($\star\star$) est vrai, alors une exécution acceptante peut facilement être définie en répétant une infinité de fois le suffixe π' (dans les deux cas).

Réciproquement, s'il existe exécution acceptante π et une position $i_0 \in \mathbb{N}$ dans cette exécution après laquelle plus aucun test à zéro n'est fait, alors nous pouvons montrer que ($\star$) est vérifié. Pour tout $i \in \mathbb{N}$ nous notons $\pi(i) = \langle q_i, c_i \rangle$. Sans perte de généralité, nous supposons que $i_0 > 0$ et $c_{i_0-1} = 0$. En effet si π est sans test à zéro nous pouvons poser $i_0 = 1$ puisque la valeur initiale du compteur est zéro, sinon i_0 est la position suivant le dernier test à zéro. D'après le Lemme 34 il existe une exécution finie π_1 entre $\langle q_0, 0 \rangle$ et $\langle q_{i_0-1}, 0 \rangle$ de longueur inférieure à $|Q|^5 + |Q|^4$. Notons que cette étape est triviale dans le cas où l'exécution est sans test à zéro.

Comme la sous-exécution commençant après la position i_0 est sans test à zéro, nous pouvons appliquer le Lemme 32. Ceci nous permet de construire une exécution finie π_3 débutant dans une configuration $\langle q_{i_1}, c_1 \rangle$ et finissant dans une configuration $\langle q_2, c_2 \rangle$ telle que i_1 correspond à une position de π telle que $i_1 \geq i_0$, $q_{i_1} = q_2 \in F$ et $c_1 \leq c_2$. La longueur de π_3 est bornée par $2|Q|^2 + 3|Q|$ et d'après la construction définie dans la preuve du Lemme 32 cette sous-exécution est sans test à zéro. Puisque une transition décrémente le compteur d'au plus une unité et $c_1 \leq c_2$ les transitions successives de π_3 peuvent toutes être franchies pour toute valeur initiale $c_1 > |Q|^2 + 3/2|Q|$.

Il reste à relier la fin du chemin π_1 au début de π_3. En utilisant le Corollaire 11, on peut montrer qu'il existe une exécution π_2 entre $\langle q_{i_0-1}, 0 \rangle$ et $\langle q_{i_1}, c_1 \rangle$ telle que $c_1 > |Q|^2 + 3/2|Q|$. La longueur de cette exécution est bornée par $(2 + |Q|^2 + 3/2|Q|) \times |Q| + |Q|^2$. En concaténant π_1, π_2 et la variante π_3 débutant dans la configuration $\langle q_{i_1}, c_1 \rangle$ d'arrivée de π_2, nous obtenons une exécution finie qui satisfait ($\star$).

Supposons maintenant que pour toute exécution acceptante π il existe une infinité de tests à zéro. Nous montrons alors que ($\star\star$) est vraie. S'il existe une infinité de tests à zéro dans π, il existe en particulier deux indices i_0, i_1 de π vérifiant $q_{i_0} = q_{i_1}$, $c_{i_0} = c_{i_1} = 0$ et tels qu'il existe un état final entre ces deux positions. D'après le Lemme 34 il existe une exécution finie π_1 entre $\langle q_0, 0 \rangle$ et $\langle q_{i_0}, 0 \rangle$ de longueur inférieure à $|Q|^5 + 2|Q|^4 + |Q|^2$.

De plus en utilisant le Lemme 36, on sait qu'il existe une exécution finie π_2 entre $\langle q_{i_0}, 0 \rangle$ et $\langle q_{i_1}, 0 \rangle = \langle q_{i_0}, 0 \rangle$ de longueur inférieure à $2|Q|^5 + 4|Q|^4 + |Q|^3$ telle qu'une position j de cette exécution vérifie $q_j \in F$. Ce chemin comporte des tests à zéro, sinon l'exécution

$$\langle q_0, 0 \rangle \to \ldots \to \langle q_{i_0}, 0 \rangle (\to \ldots \to \langle q_j, c_j \rangle \to \ldots \to \langle q_{i_0}, 0 \rangle)^\omega.$$

est acceptante et contredit l'hypothèse qu'il existe une infinité de tests à zéro pour toute exécution acceptante. L'exécution finie obtenue en concaténant π_1 et π_2 vérifie les conditions de ($\star\star$). $\square$

Nous pouvons maintenant montrer le résultat principal de cette section.

Théorème 31 *Tester si le langage* $\mathrm{L}(\mathcal{A})$ *reconnu par un* $\langle 1, \mathbb{Z} \rangle$*-automate à compteur simple* $\mathcal{A}$ *avec un alphabet et des* ε*-transitions est non-vide est un problème* NLOGSPACE*-complet.*

Preuve : Nous rappelons d'abord que comme la composition de réduction en espace logarithmique est une réduction en espace logarithmique, ce problème se réduit en espace logarithmique au problème de l'existence d'une exécution acceptante pour un $\langle 1, \mathbb{N}\rangle$-automate à compteur sans test $x \neq 0$. Ceci est démontré dans la sous-section 5.4.1. De plus le Lemme 37 permet de montrer que ce dernier problème est dans NLOGSPACE en distinguant les exécution ayant un nombre fini et infini de tests à zéro. En effet, nous pouvons vérifier qu'une exécution finie de taille polynomiale vérifie les conditions de l'énoncé du Lemme 37 en espace logarithmique.

La composition d'une fonction en espace logarithmique avec une fonction non-déterministe en espace logarithmique nous donne une fonction non-déterministe en espace logarithmique. Donc le problème est dans NLOGSPACE. Comme souvent, la NLOGSPACE-dureté est obtenue facilement par réduction du problème d'accessibilité dans les graphes. $\square$

Pour conclure cette section, notons que le même problème pour la classe générale des $\langle 1, \mathbb{Z}\rangle$-automates à compteur, où les mises à jour sont de la forme $\mathsf{X}x = x+u$ pour $u \in \mathbb{Z}$, est NP-dur. En effet, le problème NP-complet SUBSETSUM se réduit facilement à ce problème. De plus, ce problème est dans PSPACE puisqu'il peut être réduit en espace polynomial au cas des $\langle 1, \mathbb{Z}\rangle$-automates à compteur simples. Nous ignorons cependant la complexité exacte.

Synthèse I

Le tableau suivant résume les différents résultats de cette partie relatifs aux extensions de LTL avec des contraintes induites par l'arithmétique de Presburger (voir [Gas05]).

	MC ($\mathcal{D}$-auto.)	SAT	MC ($\langle n, \mathbb{Z}\rangle$-auto.)
CLTL(IPC*)	PSPACE-c., Th. 23	PSPACE-c., Cor. 5	Σ^1_1-c. [Min67, AH94]
CLTL$^1_3$(DL)	Σ^1_1-c. [CC00]	Σ^1_1-c. [CC00]	Σ^1_1-c. [CC00]
CLTL$^\omega_2$(DL)	Σ^1_1-c. [Min67, AH94]	Σ^1_1-c. [DD07]	Σ^1_1-c. [Min67, AH94]
CLTL$^2_1$(DL)	Σ^1_1-c., Cor. 8	Σ^1_1-c., Th. 26	PSPACE-c, Th. 29 + [DS02]
CLTL$^1_2$(DL)	Σ^1_1-c. [Min67, AH94]	Σ^1_1-c., Th. 26	Σ^1_1-c. [Min67, AH94]
CLTL$^1_1$(DL$^+$)	PSPACE-c., Cor. 10	PSPACE-c., Cor. 10	PSPACE-c, Th. 29 + [DS02]
CLTL$^1_1$(QFP)	Σ^1_1-c., Lem. 24	Σ^1_1-c., Lem. 24	PSPACE-c, Th. 29 + [DS02]
CLTL$^\omega_1$(QFP)	Σ^1_1-c., Lem. 24	Σ^1_1-c., Lem. 24	PSPACE-c., Th. 29 + [DS02]

Nous avons établi une classification détaillée des fragments de LTL étendu avec l'arithmétique de Presburger selon plusieurs critères de restrictions :

- le fragment de l'arithmétique de Presburger considéré (qualitatif vs quantitatif),
- le nombre de variables des formules,
- la longueur temporelle des formules,
- dans une moindre mesure, le modèle opérationnel considéré dans le problème du model-checking.

La première remarque que nous pouvons faire est que l'introduction de contraintes quantitatives provoque plus facilement l'indécidabilité de l'extension de LTL correspondante. Ce n'est pas le cas pour les logique arborescentes où la logique CCTL(Z^c) restreinte au fragment qualitatif est déjà indécidable (voir [Čer94]). Pour les fragments quantitatifs, nous avons établi des limites de décidabilité fines par rapport aux restrictions du nombre de variables et de la longueur temporelle.

Les résultats de décidabilité que nous avons montrés reposent sur des représentations symboliques des modèles et une adaptation de la construction d'automate de LTL [VW94]. A chaque fois, nous avons besoin de définir une relation de satisfaction symbolique puis de caractériser l'ensemble des modèles symboliques qui correspondent à un modèle concret. Dans chaque cas, l'abstraction que nous définissons permet de représenter de manière fini un ensemble infini de valuations. Les approches que nous définissons diffèrent néanmoins et introduisent à chaque fois une particularité nouvelle par rapport à la construction classique d'automate pour LTL. Par exemple, les automates que nous utilisons pour montrer la décidabilité de $\mathrm{CLTL}_1^1(\mathrm{DL}^+)$ sont des automates à un compteur et la construction pour $\mathrm{CLTL}(\mathrm{IPC}^\star)$ utilise une approximation de l'ensemble des modèles symboliques qui correspondent à un modèle concret.

Nous discuterons plus tard des problèmes liés aux extensions arborescentes des logiques introduites dans cette partie. Pour conclure cette partie nous notons que le domaine concret $\langle \mathbb{N}, <, = \rangle$ est équivalent au domaine sur les chaînes de caractères $\langle \{0\}^*, \preceq, = \rangle$ où l'alphabet est un singleton. L'étude des extensions de LTL sur des domaines concrets où les variables sont interprétées par des mots sur un alphabet quelconque permettrait peut être de généraliser plusieurs des résultats de cette partie, en particulier concernant les fragments qualitatifs. Enfin, d'autres restrictions sur le comportement des compteurs pourraient aussi permettre de définir des fragments décidables.

Troisième partie

Opérateur de stockage

Chapitre 6

LTL avec contraintes de répétition

Sommaire

6.1 Mécanismes de stockage dans les logiques

Nous considérons dans ce chapitre une extension des logiques temporelles sur des domaines concrets dont la définition est légèrement différente de celle utilisée jusqu'à présent puisque nous introduisons une forme restreinte de quantification du premier ordre. Ceci nous permet de spécifier des propriétés plus riches et en particulier de définir des mécanismes de stockage d'une valeur. Dans la modélisation des systèmes informatiques, il est en effet utile de pouvoir stocker des informations telle que la valeur d'une horloge ou d'un registre, l'adresse d'un noeud, l'origine du modèle, etc... Les instructions du type `let x=...in...` dans certains langages de programmation ou plus simplement la quantification de la logique classique $\exists x\ \phi(x)$ sont aussi des exemples de mémorisation d'une information.

De nombreux formalismes logiques introduisent ainsi des opérateurs permettant de stocker une information. Par exemple, dans les logiques hybrides des opérateurs permettant de stocker un état particulier ont été considérés [BS95, Gor96] : la formule $\downarrow_x \phi(x)$ est vérifiée si $\phi(x)$ est vraie dans une structure de Kripke où la variable propositionnelle x n'est vraie que dans l'état courant. C'est aussi le cas de la λ-abstraction des prédicats [Fit02] utilisée pour interpréter les constantes dans la logique modale du premier ordre.

Dans le cadre des logiques temporelles qui nous intéresse plus particulièrement, plusieurs travaux utilisent de tels mécanismes. Par exemple dans la logique temporisée TPTL de [AH94] la formule $x.\phi(x)$ est vraie si la formule ϕ dans laquelle on remplace toutes les occurrences de x par la valeur courante de l'horloge est vérifiée. L'opérateur Now permet de changer le début du modèle pour le fixer à l'état courant, ce qui à pour conséquence d'oublier le passé (voir par exemple [LMS02]). Ainsi la formule Now $\mathsf{X}^{-1}\top$ n'est pas satisfaisable puisque lorsque l'opérateur Now est utilisé l'origine du modèle est modifiée. Par ailleurs, dans [KV06] est définie une logique temporelle arborescente avec mémoire permettant de référer au noeud courant dans un chemin à partir de la racine du modèle à l'aide d'une proposition particulière. La logique $\mathrm{CLTL}^{\Diamond}(\langle\mathbb{N},=\rangle)$ que nous introduisons est une extension de $\mathrm{CLTL}(\langle\mathbb{N},=\rangle)$ qui utilise aussi un tel mécanisme de stockage et permet d'exprimer des contraintes de répétition d'une valeur dans le modèle. Par exemple $\mathrm{CLTL}^{\Diamond}(\langle\mathbb{N},=\rangle)$ permet d'exprimer qu'il existe une position dans le futur où la valeur de la variable y est égale à la valeur courante de la variable x grâce à la contrainte atomique $x = \Diamond y$. Contrairement aux travaux cités ci-dessus, le mécanisme de stockage dans la logique $\mathrm{CLTL}^{\Diamond}(\langle\mathbb{N},=\rangle)$ est implicite. Néanmoins l'évaluation d'une contrainte de répétition implique de mémoriser la valeur à répéter.

Enfin, nous avons omis dans la liste précédente, ceci afin d'en parler plus en détail maintenant, plusieurs travaux utilisant *l'opérateur freeze* pour étendre LTL sur des domaines concrets (voir [DLN07, DL06, Laz06]). Cet opérateur définit d'une manière explicite et générale le mécanisme de stockage. L'utilisation d'un opérateur freeze, par exemple dans la formule $\downarrow_{r=x} \phi$, permet de stocker la valeur de x dans le registre r. La formule ϕ est alors évaluée en remplaçant les occurrences de r par la valeur courante de x. L'extension de LTL sur des domaines concrets avec l'opérateur freeze est assez expressive pour coder certains formalismes cités ci-dessus et en particulier $\mathrm{CLTL}^{\Diamond}(\langle\mathbb{N},=\rangle)$. Cependant, cette expressivité est telle que ce formalisme permet facilement de coder le problème de l'arrêt des machines de Minsky. Il est donc intéressant de considérer des fragments de cette logique comme $\mathrm{CLTL}^{\Diamond}(\langle\mathbb{N},=\rangle)$ qui permettent de retrouver la décidabilité.

Avant de définir formellement $\mathrm{CLTL}^{\Diamond}(\langle\mathbb{N},=\rangle)$, nous nous attardons un peu sur la définition de cette logique et les résultats qui peuvent être déduits pour certains de ces fragments à partir de résultats connus. Les résultats que nous démontrons ensuite complètent les résultats de décidabilité connus pour divers fragments de la logique temporelle linéaire avec l'opérateur freeze.

Ce chapitre développe les résultats présentés dans [DDG07].

6.1.1 LTL avec l'opérateur freeze

Afin de mieux situer les résultats que nous énonçons par la suite, nous présentons quelques résultats connus sur les extensions de LTL utilisant l'opérateur freeze. Étant donné un ensemble d'opérateurs temporels $\mathcal{O}$ inclus dans $\{\mathsf{U}, \mathsf{S}, \mathsf{X}^i, \mathsf{X}^i\mathsf{F}, \mathsf{X}^{-i}, \mathsf{X}^{-i}\mathsf{F}^{-1} \mid i \in \mathbb{N}\}$, la logique $\mathrm{LTL}^{\downarrow}(\mathcal{O})$ est définie par la grammaire suivante :

$$\phi ::= x = y \mid x = r \mid \neg\phi \mid \phi \wedge \phi \mid \mathrm{O}(\phi_1, \ldots, \phi) \mid \downarrow_{r=\mathsf{X}^i x} \phi.$$

où x, y, r sont des variables, O est un opérateur temporel de $\mathcal{O}$ et l'opérateur $\downarrow$ est appelé *opérateur freeze*. L'ensemble des variables est interprété dans $\mathbb{N}$ et est partitionné en deux sous-ensembles

- un ensemble V_f de *variables flexibles* qui sont des variables standards par rapport aux logiques introduites jusqu'à présent,
- un ensemble V_r de *variables rigides* qui sont utilisées comme des registres permettant de stocker des valeurs et dont la valeur peut être modifiée uniquement par l'utilisation d'un opérateur freeze.

Dans la définition ci-dessus, nous avons $x, y \in V_f$ et $r \in V_r$. Nous notons $\mathrm{LTL}^{\downarrow}_{n,m}(\mathcal{O})$ le fragment de $\mathrm{LTL}^{\downarrow}(\mathcal{O})$ restreint aux formules avec n variables et m registres. La logique $\mathrm{LTL}^{\downarrow}(\{\mathsf{X}, \mathsf{U}\})$ étend $\mathrm{CLTL}(\langle\mathbb{N}, =\rangle)$ puisqu'il est démontrer dans [DLN07] qu'elle est strictement plus expressive. Par exemple une propriété telle que *"la variable x ne prend jamais deux fois la même valeur"* (x est un nonce) ne peut être exprimée dans $\mathrm{CLTL}(\langle\mathbb{N}, =\rangle)$ et correspond à la formule $\mathsf{G} \downarrow_{r=x} \mathsf{G}(r \neq x)$ de $\mathrm{LTL}^{\downarrow}(\{\mathsf{X}, \mathsf{U}\})$.

Nous considérons pour la logique $\mathrm{LTL}^{\downarrow}(\mathcal{O})$ à la fois les modèles finis qui sont des séquences finies de valuations de la forme $\{0, 1, 2, \ldots, n\} \to (V_f \to \mathbb{N})$ et les modèles infinis de la forme $\mathbb{N} \to (V_f \to \mathbb{N})$ qui associent à chaque position une valeur aux variables flexibles. Nous notons $|\sigma|$ la taille de σ qui est égale à la cardinalité de l'ensemble de définition de σ si σ est un modèle fini ou ω si σ est infini. La relation de satisfaction est paramétrée par un contexte κ qui associe une valeur aux variables rigides. Nous notons $\kappa[r \leftarrow i]$ le contexte affectant la valeur $i \in \mathbb{N}$ à la variable r et $\kappa(r')$ à toute autre variable r' différente de r. La relation de satisfaction $\models_\kappa$ dépendant du contexte κ est définie de la façon suivante :

- $\sigma, i \models_\kappa x = y$ ssi $\sigma(i)(x) = \sigma(i)(y)$,
- $\sigma, i \models_\kappa x = r$ ssi $\sigma(i)(x) = \kappa(r)$,
- $\sigma, i \models_\kappa \neg\phi$ ssi $\sigma, i \not\models_\kappa \phi$,
- $\sigma, i \models_\kappa \phi \wedge \phi'$ ssi $\sigma, i \models_\kappa \phi$ et $\sigma, i \models_\kappa \phi'$,
- $\sigma, i \models_\kappa \downarrow_{r=\mathsf{X}^j x} \phi$ ssi $\sigma, i \models_{\kappa[r \leftarrow \sigma(i+j)(x)]} \phi$,
- Les opérateurs temporels que nous utilisons dans $\mathrm{LTL}^{\downarrow}(\mathcal{O})$ par la suite sont standard et les formules de la forme $\mathrm{O}(\phi_1, \ldots, \phi)$ sont interprétées par rapport à la sémantique de l'opérateur correspondant à O en tenant compte de la taille du modèle. Par exemple, nous avons

– $\sigma, i \models_\kappa \mathsf{X}^j\phi$ ssi $i + j < |\sigma|$ et $\sigma, i + j \models_\kappa \phi$,
– $\sigma, i \models_\kappa \mathsf{X}^j\mathsf{F}\phi$ ssi il existe $i + j \leq j' < |\sigma|$ tel que $\sigma, j' \models_\kappa \phi$.

Le problème de la satisfaisabilité dans le cas fini (respectivement infini) consiste à déterminer s'il existe un modèle fini (respectivement infini) satisfaisant une formule de $\mathrm{LTL}^{\downarrow}(\mathcal{O})$ donnée. Il est démontré dans [DLN07] que le problème de la satisfaisabilité fini et infini est indécidable pour $\mathrm{LTL}^{\downarrow}(\{\mathsf{X}, \mathsf{U}\})$ (ce résultat peut aussi être obtenu en utilisant [LP05]). Le statut du problème de la satisfaisabilité pour différents fragments de la forme $\mathrm{LTL}^{\downarrow}_{1,m}(\mathcal{O})$ est établi par les résultats de [DL06].

– Le problème de la satisfaisabilité dans le cas fini est décidable pour $\mathrm{LTL}^{\downarrow}_{1,1}(\{\mathsf{X}, \mathsf{U}\})$. Néanmoins, la complexité pour $\mathrm{LTL}^{\downarrow}_{1,1}(\{\mathsf{X}, \mathsf{F}\})$ est déjà non-primitive récursive.
– Le problème de la satisfaisabilité dans le cas fini est indécidable pour le fragment $\mathrm{LTL}^{\downarrow}_{1,1}(\{\mathsf{X}, \mathsf{F}, \mathsf{X}^{-1}, \mathsf{F}^{-1}\})$.
– Le problème de la satisfaisabilité dans le cas infini est indécidable pour $\mathrm{LTL}^{\downarrow}_{1,1}(\{\mathsf{X}, \mathsf{F}\})$.
– Le problème de la satisfaisabilité dans les cas fini et infini est indécidable pour le fragment $\mathrm{LTL}^{\downarrow}_{1,2}(\{\mathsf{X}, \mathsf{F}\})$.

Pour résumer, la satisfaisabilité dans le cas fini est décidable pour le fragment de la logique $\mathrm{LTL}^{\downarrow}(\{\mathsf{X}, \mathsf{U}\})$ avec un registre et une variable flexible, mais ce problème devient indécidable lorsque l'on considère le cas infini, ou lorsque l'on ajoute des opérateurs temporels passés, ou encore lorsque l'on utilise plus d'un registre. Les preuves de certains de ces résultats utilisent des réductions vers des problèmes pour des classes d'automates à compteurs avec erreur d'incrémentation. Nous montrons par la suite que le problème de la satisfaisablité pour LTL avec des contraintes de répétition se traduit aussi vers un problème pour un classe particulière d'automates à compteurs.

Il est aussi établi dans [DL06] que tout fragment simple de $\mathrm{LTL}^{\downarrow}_{1,1}(\{\mathsf{X}, \mathsf{U}\})$ est équivalent à une instance la logique du premier ordre sur les mots de données avec deux variables de la forme $FO_2(\sim, +1, +2, \ldots, +k, <)$ telle que

– les variables représentent des positions dans le mot,
– $\sim$ est une relation d'équivalence entre les positions
– $+i$ ($i \in \mathbb{N}$) représente la relation $x = y + i$.

Un fragment simple de $\mathrm{LTL}^{\downarrow}_{1,1}(\{\mathsf{X}, \mathsf{U}\})$ correspond à la restriction d'un fragment de la forme $\mathrm{LTL}^{\downarrow}_{1,1}(\mathcal{O})$ tel que $\mathcal{O} = \{\mathsf{X}, \mathsf{X}^2, \ldots, \mathsf{X}^m, \mathsf{X}^{m+1}\mathsf{F}, \mathsf{X}^{-1}, \mathsf{X}^{-2}, \ldots, \mathsf{X}^{-m}, \mathsf{X}^{-(m+1)}\mathsf{F}^{-1}\}$ et dans lequel l'utilisation des opérateurs de $\mathcal{O}$ n'est permise que si elle est directement précédée d'un opérateur freeze. D'après cette équivalence, le problème de la satisfaisabilité dans le cas fini des fragments simples est décidables puisque ce problème est décidable pour la logique du premier ordre $FO_2(\sim, +1, +2, \ldots, +k, <)$ [BMS+06]. Ce résultat nous sera utile par la suite pour déduire la satisfaisabilité dans le cas fini de certains fragments de notre logique (voir la preuve du Théorème 32).

6.2 LTL avec des contraintes de répétitions

Nous utilisons une définition étendue des logiques temporelles sur des domaines concrets dans ce chapitre. Soit $\mathrm{VAR} = \{x_1, x_2, \ldots\}$ un ensemble infini dénombrable de variables. Les formules de la logique avec contraintes de répétition $\mathrm{CLTL}^{\Diamond}(\langle D, =\rangle)$ sont définies par la grammaire suivante

$$\phi ::= \mathsf{X}^i x = \mathsf{X}^j y \mid x = \Diamond y \mid \phi \wedge \phi \mid \neg\phi \mid \mathsf{X}\phi \mid \phi \mathsf{U} \phi \mid \mathsf{X}^{-1}\phi \mid \phi \mathsf{S} \phi$$

où $x, y \in \mathrm{VAR}$ et $i, j \in \mathbb{N}$. La principale différence avec la définition utilisée jusqu'à présent est l'introduction de contraintes de la forme $x = \Diamond y$ exprimant une contrainte de répétition de la valeur prise par la variable x dans un état futur de la variable y. Ce type de contrainte, introduit initialement dans le cadre de raisonnement spatio-temporel [WZ00], peut être vu comme la disjonction infinie $\bigvee_{i\in\mathbb{N}}(x = \mathsf{X}^i y)$. Nous utilisons aussi les opérateurs temporels référant au passé X^{-1} et S ainsi que les abréviations usuelles F^{-1} et G^{-1}, afin de pouvoir mieux situer les résultats qui suivent par rapport aux résultats cités plus tôt concernant $\mathrm{LTL}^{\downarrow}(\mathcal{O})$. Étant donné un entier $k \geq 0$, nous notons $\mathrm{CLTL}^{\Diamond}_k(D, =)$ le fragment de $\mathrm{CLTL}^{\Diamond}(D, =)$ restreint aux formules ayant au plus k variables.

Un modèle de $\mathrm{CLTL}^{\Diamond}(\langle D, =\rangle)$ est une séquence finie ou infinie de valuations de la forme $\mathrm{VAR} \to D$. Nous fixons $D = \mathbb{N}$ pour la suite mais les résultats peuvent être transposés à n'importe quel autre domaine d'interprétation fini ou infini puisque les seules contraintes autorisées sont des tests d'égalité. La relation de satisfaction $\sigma \models \phi$ de $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, =\rangle)$ est définie de la façon suivante.

- $\sigma, i \models \mathsf{X}^j x = \mathsf{X}^{j'} y$ ssi $i + j \leq |\sigma| - 1$, $i + j' \leq |\sigma| - 1$ et $\sigma(i + j)(x) = \sigma(i + j')(y)$,
- $\sigma, i \models x = \Diamond y$ ssi il existe $i < j \leq |\sigma| - 1$ tel que $\sigma(i)(x) = \sigma(i + j)(y)$,
- $\sigma, i \models \phi \wedge \phi'$ ssi $\sigma, i \models \phi$ et $\sigma, i \models \phi'$,
- $\sigma, i \models \neg\phi$ ssi $\sigma, i \not\models \phi$,
- $\sigma, i \models \mathsf{X}\phi$ ssi $i + 1 \leq |\sigma| - 1$ et $\sigma, i + 1 \models \phi$,
- $\sigma, i \models \mathsf{X}^{-1}\phi$ ssi $i > 0$ et $\sigma, i - 1 \models \phi$,
- $\sigma, i \models \phi \mathsf{U} \phi'$ ssi il existe $i \leq j \leq |\sigma| - 1$ tel que $\sigma, j \models \phi'$ et pour tout $i \leq l < j$, $\sigma, l \models \phi$.
- $\sigma, i \models \phi \mathsf{S} \phi'$ ssi il existe $0 \leq j \leq i$ tel que $\sigma, j \models \phi'$ et pour tout $j \leq l < i$, $\sigma, l \models \phi$.

Nous notons $\sigma \models \phi$ ssi $\sigma, 0 \models \phi$. Toujours dans le but de raffiner les résultats relatifs à $\mathrm{LTL}^{\downarrow}(\mathcal{O})$, nous considérons les problèmes de satisfaisabilité pour $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, =\rangle)$ dans les cas fini et infini. Étant donnée une formule ϕ de $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, =\rangle)$ nous voulons déterminer l'existence d'un modèle respectivement fini ou infini satisfaisant la formule ϕ.

Pour la logique propositionnelle LTL, la satisfaisabilité dans le cas fini peut se réduire en espace logarithmique au cas infini en introduisant une variable propositionnelle p supplémentaire et en ajoutant la contrainte que la formule $p\mathsf{U}\mathsf{G}\neg p$ soit vérifiée dans le modèle infini. Ainsi, la formule ϕ doit être satisfaite avant la position où on a $\neg p$, ce qui implique de relativiser les formules. Par exemple, nous transformons $\phi \mathsf{U} \phi'$ en $(p \Rightarrow \phi)\mathsf{U}(p \wedge \phi')$ (voir le Lemme 1). Cependant, la même réduction ne fonctionne pas pour $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ à cause des contraintes de la forme $x = \Diamond y$. En effet, nous n'avons pas d'information sur la position où y prend la même valeur que x à l'état courant et il est donc impossible d'imposer que la propriété p soit vérifiée à cette position sans étendre la logique. Nous distinguons donc ces deux problèmes par la suite.

Afin de ne pas causer d'ambiguïté, nous n'introduisons pas dans notre définition la contrainte $\mathsf{X}^i x = \Diamond y$. Cette expression peut être interprétée de différentes façons selon si l'on considère que la valeur future de y représentée par $\Diamond y$ doit être recherchée à partir de la position courante ou après la $i^{\text{ème}}$ position suivante (c'est-à-dire après la valeur représentée par $\mathsf{X}^i x$). Les deux cas peuvent s'exprimer dans la logique. Ainsi le premier cas se traduit $\bigvee_{0 \leq j < i} (\mathsf{X}^j y = \mathsf{X}^i x) \vee \mathsf{X}^i (x = \Diamond y)$ et le second simplement $\mathsf{X}^i(x = \Diamond y)$. Notons aussi la différence entre la formule $\neg(x = \Diamond y)$, qui signifie qu'aucune valeur prise par y dans le futur n'est égale à la valeur courante de x, et la contrainte $x \neq \Diamond y$ signifiant qu'il existe une valeur future de y différente de la valeur courante de x. Cette dernière contrainte n'est pas présente formellement dans la définition de $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ mais peut être exprimée grâce à l'équivalence suivante.

$$x \neq \Diamond y \;\Leftrightarrow\; (\neg(x = \mathsf{X}y) \wedge \mathsf{X}\top) \vee \big((x = \mathsf{X}y) \wedge \mathsf{X}((y = \mathsf{X}y)\mathsf{U}((y \neq \mathsf{X}y) \wedge \mathsf{X}\top))\big)$$

Si le modèle est infini, les occurrences de $\mathsf{X}\top$ peuvent être omises. De même, les contraintes de la forme $x = \mathsf{X}^{-i} y$ peuvent être exprimées dans la logique en utilisant l'équivalence

$$x = \mathsf{X}^{-i} y \Leftrightarrow \mathsf{X}^{-i}\top \wedge \mathsf{X}^{-i}(y = \mathsf{X}^i x).$$

La logique $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ peut aussi exprimer qu'une variable est un nonce avec la formule $\mathsf{G}\neg(x = \Diamond x)$. Enfin, il est possible d'exprimer dans $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ des propriétés plus complexes sur des données évoluant au cours du temps. Par exemple, considérons un ensemble de données $\{d_1, \ldots, d_n\}$ qui change au cours du temps et deux variables send et receive. La propriété suivante

$$\bigwedge_{i=1,\ldots,n} \mathsf{G}(d_i \neq \Diamond d_i) \wedge \bigwedge_{i=1,\ldots,n} \mathsf{G}\big((\mathrm{send} = d_i) \Rightarrow (d_i = \Diamond \mathrm{receive})\big)$$

exprime que si send prend la valeur d'une donnée alors il existe une valeur dans le futur où receive prend cette même valeur. La première partie de la formule exprime que les données ne sont jamais constantes. D'autres propriétés sur les systèmes à jetons qui évoluent au cours du temps citées dans [LP05, Sect.3] peuvent s'exprimer dans $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$, ce qui témoigne du pouvoir d'expressivité de cette logique.

Nous pouvons définir un problème de model-checking pour la logique $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ sur des automates à contraintes dont les transitions sont étiquetées par des combinaisons booléennes de contraintes atomiques de $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$. Un $\langle \mathbb{N}, = \rangle^{\Diamond}$-automate avec k variables $\mathcal{A} = \langle Q, I, F, \delta \rangle$ est un $\langle \mathbb{N}, = \rangle$-automate étendu tel que

– Q est un ensemble fini d'états de contrôle,

- $I \subseteq Q$ est un ensemble d'états initiaux,
- $F \subseteq Q$ est un ensemble d'états finaux,
- $\delta \subseteq Q \times 1SC_k^{\Diamond}(\langle \mathbb{N}, = \rangle) \times Q$ est une relation de transition finie,

  où $1SC_k^{\Diamond}(\langle \mathbb{N}, = \rangle)$ est l'ensemble des formules obtenues par combinaisons Booléennes de contraintes transitionnelles de $\mathrm{CLTL}(\langle \mathbb{N}, = \rangle)$ et de contraintes de répétition de la forme $x = \Diamond y$ définies avec les variables $\{x_1, \ldots, x_k\}$.

Une configuration de $\mathcal{A}$ est une paire $\langle q, v \rangle \in Q \times \mathbb{N}^k$ composée d'un état de contrôle et d'un k-tuple d'entiers naturels vu comme une valuation pour les variables $\{x_1, \ldots, x_k\}$. La sémantique d'un $\langle \mathbb{N}, = \rangle^{\Diamond}$-automate est un peu différente de celle des $\langle \mathbb{N}, = \rangle$-automates à cause des contraintes de répétition. Pour évaluer une contrainte de répétition, nous avons en effet besoin de connaître la totalité de l'exécution. Il n'est donc pas possible de définir une transition sur un seul pas. Une exécution linéaire finie de $\mathcal{A}$ est une séquence finie de la forme $\pi = \langle q_0, \sigma(0) \rangle \xrightarrow{\alpha_0} \langle q_1, \sigma(1) \rangle \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{|\pi|-1}} \langle q_{|\pi|}, \sigma(|\pi|) \rangle$ telle que σ est un modèle fini de $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ qui associe à chaque position une valuation de la forme $\{x_1, \ldots, x_k\} \rightarrow \mathbb{N}$ et pour tout $0 \leq i \leq |\pi| - 1$, on a $\langle q_i, \alpha_i, q_{i+1} \rangle \in \delta$ et $\sigma, i \models \alpha_i$. Dans ce cas nous disons toujours que σ réalise π. Une exécution linéaire infinie est définie de la même manière par rapport à un modèle infini. Une exécution finie est acceptante ssi elle termine dans un état final. Une exécution infinie est acceptante ssi elle visite infiniment souvent un état final.

Le problème du model-checking dans le cas fini (respectivement infini) pour la logique $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ prend comme entrées une formule ϕ de $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ et un $\langle \mathbb{N}, = \rangle^{\Diamond}$-automate et consiste à déterminer s'il existe un modèle fini (respectivement infini) σ qui réalise une exécution acceptante de $\mathcal{A}$ et satisfait ϕ. Comme le comportement d'un $\langle \mathbb{N}, = \rangle^{\Diamond}$-automate peut facilement se traduire en une formule de $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ de la même manière que pour les extensions de LTL sur domaine concret, le problème du model-checking est équivalent au problème de la satisfaisabilité de $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ aussi bien dans le cas fini que infini. C'est pourquoi nous considérons uniquement le problème de la satisfaisabilité dans la suite.

Il est clair que la logique $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ est incluse dans le fragment de la logique $\mathrm{LTL}^{\downarrow}(\mathsf{X}, \mathsf{U}, \mathsf{X}^{-1}, \mathsf{S})$ restreint à un registre $\mathrm{LTL}^{\downarrow}_{\omega,1}(\mathsf{X}, \mathsf{U}, \mathsf{X}^{-1}, \mathsf{S})$ puisque la contrainte atomique $\mathsf{X}^i x = \mathsf{X}^j y$ telle que $i \leq j$ est équivalente à $\mathsf{X}^i \downarrow_{r=x} \mathsf{X}^{j-i}(r = y)$ et $x = \Diamond y$ à $\downarrow_{r=x} \mathsf{XF}(r = y)$. De la même manière que pour $\mathrm{LTL}^{\downarrow}(\mathcal{O})$, la logique $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ est strictement plus expressive que son fragment sans les contraintes de la forme $x = \Diamond y$ puisqu'une propriété comme le nonce ne peut pas être exprimée en l'absence de tout opérateur permettant de stocker une valeur courante. Par contre, la logique $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ n'est ni englobée par le fragment "safety" de $\mathrm{LTL}^{\downarrow}_{1,1}(\mathsf{X}, \mathsf{U})$ considéré dans [Laz06] ni non plus par le fragment plat de $\mathrm{LTL}^{\downarrow}_{\omega,1}(\mathsf{X}, \mathsf{U}, \mathsf{X}^{-1}, \mathsf{S})$. Le fragment safety restreint l'utilisation de l'opérateur temporel U dans un nombre impair de négations et le fragment plat est obtenu en restreignant l'utilisation de l'opérateur freeze dans les sous-formules de la forme $\phi = \phi_1 \mathsf{U} \phi_2$ en fonction du nombre de négation : si ϕ est dans la portée d'un nombre pair de négations alors ϕ_1 ne contient pas d'occurrence de l'opérateur freeze, sinon ϕ_2 ne contient pas d'occurrence de l'opérateur freeze. Contrairement à ces logiques, $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ contient des opérateurs temporels référant au passé et n'a aucune restriction relatives à l'utilisation des opérateurs temporels ou de la négation dans les formules. Cependant des réductions à certains fragments de $\mathrm{LTL}^{\downarrow}_{\omega,1}(\mathsf{X}, \mathsf{U}, \mathsf{X}^{-1}, \mathsf{S})$ nous permettent d'établir déjà la décidabilité du problème de la satisfaisabilité pour des fragments de $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ dans le cas fini.

Théorème 32
(I) *Le problème de la satisfaisabilité pour* $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ *restreint aux opérateurs temporels futurs est décidable dans le cas fini.*
(II) *Le problème de la satisfaisabilité pour* $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ *restreint aux opérateurs temporels de l'ensemble* $\{\mathsf{X}, \mathsf{X}^{-1}, \mathsf{F}, \mathsf{F}^{-1}\}$ *est décidable dans le cas fini.*

Preuve :
(I) Toute formule de la logique $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ utilisant uniquement les opérateurs temporels X et U peut être transformée en une formule de $\mathrm{LTL}^{\downarrow}_{\omega,1}(\mathsf{X}, \mathsf{U})$ en exprimant les formules atomiques à l'aide de l'opérateur freeze (voir plus haut). De plus, d'après [DLN07, Proposition 4], toute formule ϕ de $\mathrm{LTL}^{\downarrow}_{\omega,1}(\mathsf{X}, \mathsf{U})$ peut par la suite être transformée de manière uniforme en une formule ϕ' ayant une seule variable flexible et le même nombre de registres telle que ϕ est satisfaisable ssi ϕ' est satisfaisable. La transformation définie nous permet donc de réduire le problème de la satisfaisabilité dans le cas fini de $\mathrm{LTL}^{\downarrow}_{\omega,1}(\mathsf{X}, \mathsf{U})$ au problème de la satisfaisabilité dans le cas fini de $\mathrm{LTL}^{\downarrow}_{1,1}(\mathsf{X}, \mathsf{U})$ qui est décidable (voir [DL06, Corollaire 13] et Section 6.1.1).

(II) Soit ϕ une formule de $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ utilisant uniquement les opérateurs temporels de l'ensemble $\{\mathsf{X}, \mathsf{X}^{-1}, \mathsf{F}, \mathsf{F}^{-1}\}$, $V = \{x_1, \ldots, x_k\}$ l'ensemble des variables de ϕ et $l = |\phi|_{\mathsf{X}}$. Pour alléger un peu la suite de la preuve, nous supposons que toutes les contraintes atomiques sont de la forme $x = \mathsf{X}^i y$ ou $x = \Diamond y$. Nous pouvons nous ramener à ce cas en transformant les contraintes de la forme $\mathsf{X}^i x = \mathsf{X}^j y$ telles que $i \leq j$ en $\mathsf{X}^i(x = \mathsf{X}^{j-i} y)$. La formule ϕ se transforme en formule de $\mathrm{LTL}^{\downarrow}_{\omega,1}(\{\mathsf{X}, \mathsf{X}^{-1}, \mathsf{F}, \mathsf{F}^{-1}\})$ en utilisant les équivalences suivantes

(X) $x = \mathsf{X}^i y \Leftrightarrow \downarrow_{r=x} \mathsf{X}^i(r = y)$,
($\Diamond$) $x = \Diamond y \Leftrightarrow \downarrow_{r=x} \mathsf{XF}(r = y)$.

Nous posons $N = 3k(l + 1)$ et $\mathcal{O}_N$ l'ensemble des opérateurs temporels suivants :

$$\mathcal{O}_N = \{\mathsf{X}, \mathsf{X}^2, \ldots \mathsf{X}^N, \mathsf{X}^{N+1}\mathsf{F}, \mathsf{X}^{-1}, \mathsf{X}^{-2}, \ldots \mathsf{X}^{-N}, \mathsf{X}^{-(N+1)}\mathsf{F}^{-1}\}.$$

En utilisant les mêmes techniques de preuve que celles du Lemme 3, nous construisons une formule ϕ' appartenant au fragment simple de $\mathrm{LTL}^{\downarrow}_{1,1}(\mathcal{O}_N)$, c'est-à-dire le fragment où chaque opérateur temporel $\mathrm{O} \in \mathcal{O}_N$ est dans la portée directe d'un opérateur freeze $\downarrow_{r=x} \mathrm{O}\phi$, et aucune autre utilisation de l'opérateur freeze n'est permise. Nous rappelons que le problème de la satisfaisabilité dans le cas fini est décidable pour le fragment simple de $\mathrm{LTL}^{\downarrow}_{1,1}(\mathcal{O}_N)$ [BMS+06, DL06].

Nous codons un état d'un modèle ayant k variables en $3k$ états d'un modèle avec une seule variable. Nous reprenons l'idée du Lemme 3 qui est d'utiliser un état sur trois dans le modèle à une variable pour coder les valeurs du modèle original et les valeurs intermédiaires pour marquer le début d'une séquence de $3k$ états correspondant à un état du modèle original. Pour reprendre l'exemple du Lemme 3, si $k = 2$ alors le modèle suivant

$$\begin{pmatrix} y_1^0 \\ y_2^0 \end{pmatrix} \begin{pmatrix} y_1^1 \\ y_2^1 \end{pmatrix} \cdots$$

est codé par le modèle à une variable

$$\boxed{y_1^0 = \circ} \neq \circ \neq y_2^0 \neq \circ \neq \circ \neq \boxed{y_1^1 = \circ} \neq \circ \neq y_2^1 \neq \circ \neq \circ \ldots$$

où $\circ$ dénote des valeurs arbitraires vérifiant les contraintes spécifiées sur la figure avec ses voisins. Notons que de telles valeurs existent toujours car le domaine d'interprétation $\mathbb{N}$ comporte 3 valeurs distinctes. Le début d'une séquence codant un état du modèle original est marqué par la répétition de la même valeur. Donc dans le modèle à une variable, la contrainte $x = \mathsf{X}x$ est vérifiée lorsque l'on se trouve à une position codant le début d'un état du modèle à k variables. On accède à la valeur correspondant à $\mathsf{X}^i x_j$ dans le modèle à k variables en référant au terme $\mathsf{X}^{3ik+3j}x$ dans le modèle à une variable.

Nous imposons que la contrainte $x = \mathsf{X}x$ soit vérifiée exactement tous les $3k$ états grâce à la formule suivante

$$\phi_{3k} \stackrel{\text{def}}{\equiv} (x = \mathsf{X}x) \wedge \bigwedge_{0<i<3k} \mathsf{X}^i(x \neq \mathsf{X}x) \wedge$$

$$\mathsf{G}\big(((x = \mathsf{X}x) \wedge \mathsf{X}^{3k+1}\top) \Leftrightarrow \mathsf{X}^{3k}(x = \mathsf{X}x)\big) \wedge \mathsf{G}((x = \mathsf{X}x) \Rightarrow \bigwedge_{0<i<3k} \mathsf{X}^i\top)$$

qui est équivalente en utilisant l'équivalence (X) à

$$\downarrow_{r=x} \mathsf{X}(r = x) \wedge \bigwedge_{0<i<3k} \mathsf{X}^i \downarrow_{r=x} \mathsf{X}(r \neq x) \wedge \neg\mathsf{F}\big((\downarrow_{r=x} \mathsf{X}(r = x) \wedge \mathsf{X}^{3k+1}\top$$

$$\wedge \mathsf{X}^{3k} \downarrow_{r=x} \mathsf{X}(r \neq x)) \vee ((\downarrow_{r=x} \mathsf{X}(r \neq x) \vee \neg\mathsf{X}^{3k+1}\top) \wedge \mathsf{X}^{3k} \downarrow_{r=x} \mathsf{X}(r = x))\big)\big) \wedge$$

$$\neg\mathsf{F}(\downarrow_{r=x} \mathsf{X}(r = x) \wedge (\bigvee_{0<i<3k} \neg\mathsf{X}^i\top)).$$

Cette formule peut s'exprimer dans le fragment simple de $\mathrm{LTL}^{\downarrow}_{1,1}(\mathcal{O}_N)$ en utilisant les équivalences suivantes

- ($\star$) $\mathsf{F}\phi \Leftrightarrow \phi \vee \mathsf{X}\phi \vee \cdots \vee \mathsf{X}^N\phi \vee \mathsf{X}^{N+1}\mathsf{F}\phi$ (idem pour l'opérateur F^{-1}) ,
- ($\star\star$) $\mathrm{O}\phi \Leftrightarrow \downarrow_{r=x} \mathrm{O}\phi$ si dans ϕ toutes les occurrences de $r = x$ sont dans la portée d'un opérateur freeze.

Nous définissons maintenant une fonction f transformant la formule ϕ en formule ϕ' avec une seule variable et imposant que l'évaluation de ϕ' respecte les conditions de la réduction d'un modèle avec k variables définie ci-dessus.

- (X) $f(x_m = \mathsf{X}^i x_n) = \downarrow_{r=\mathsf{X}^{3m}x} \mathsf{X}^{3ik}(r = \mathsf{X}^{3n}x)$
car $x_m = \mathsf{X}^i x_n \Leftrightarrow \downarrow_{r=x_m} \mathsf{X}^i(r = x_n)$,

– ($\Diamond$) $f(x_m = \Diamond x_n) = \downarrow_{r=\mathsf{X}^{3m}x} \mathsf{X}^{3k}\mathsf{F}(\mathsf{X}^{-3n}(x = \mathsf{X}x) \wedge (r = x))$
puisque $x_m = \Diamond x_n \Leftrightarrow \downarrow_{r=x_m} \mathsf{XF}(r = x_n)$;

pour des raisons liées à l'utilisation de l'opérateur freeze, cette contrainte est traduite de la façon suivante : nous cherchons une valeur dans le futur qui est égale à la valeur de l'état codant la valeur courante de x_m puis nous revenons en arrière afin de vérifier que cette valeur code bien un état de x_n (si c'est le cas on doit avoir $\mathsf{X}^{-3n}(x = \mathsf{X}x)$),

– f est un homomorphisme par rapport aux opérateurs Booléens,

– $f(\mathsf{X}\phi) = \mathsf{X}^{3k}f(\phi)$,

– $f(\mathsf{F}\phi) = \mathsf{F}(\downarrow_{r=x} \mathsf{X}(r = x) \wedge f(\phi))$;
il est nécessaire de s'assurer que l'on s'arrête sur un état codant le début d'un configuration du modèle original avant de vérifier $f(\phi)$, ce qui explique l'ajout de la contrainte $\downarrow_{r=x} \mathsf{X}(r = x)$,

– $f(\mathsf{X}^{-1}\phi) = \mathsf{X}^{-3k}f(\phi)$,

– $f(\mathsf{F}^{-1}\phi) = \mathsf{F}^{-1}(\downarrow_{r=x} \mathsf{X}(r = x) \wedge f(\phi))$;
la même remarque que pour le cas de $\mathsf{F}\phi$ et valable.

Nous montrons maintenant que toute formule obtenue en appliquant la fonction f peut s'exprimer dans le fragment simple de $\mathrm{LTL}^{\downarrow}_{1,1}(\mathcal{O}_N)$. Nous développons les cas de base concernant (X) et ($\Diamond$) :

(X) $\downarrow_{r=\mathsf{X}^{3m}x} \mathsf{X}^{3ik}(r = \mathsf{X}^{3n}x)$
$\equiv \mathsf{X}^{3m} \downarrow_{r=x} \mathsf{X}^{3(ik-m)}(r = \mathsf{X}^{3n}x)$
$\equiv \downarrow_{r=x} \mathsf{X}^{3m} \downarrow_{r=x} \mathsf{X}^{3ik-m+n}(r = x)$.

($\Diamond$) $\downarrow_{r=\mathsf{X}^{3m}x} \mathsf{X}^{3k}\mathsf{F}(\mathsf{X}^{-3n}(x = \mathsf{X}x) \wedge (r = x))$
$\equiv \mathsf{X}^{3m} \downarrow_{r=x} \mathsf{X}^{3(k-m)}\mathsf{F}(\mathsf{X}^{-3n} \downarrow_{r=x} \mathsf{X}(r = x) \wedge (r = x))$.
$\equiv \downarrow_{r=x} \mathsf{X}^{3m} \downarrow_{r=x} \mathsf{X}^{3(k-m)}\mathsf{F}(\downarrow_{r=x} \mathsf{X}^{-3n} \downarrow_{r=x} \mathsf{X}(r = x) \wedge (r = x))$.

Nous voyons bien que les formules obtenues ont une seule variable et utilisent un seul registre. De plus, tous les opérateurs temporels sont dans la portée directe d'un quantificateur. Le cas des opérateurs de $\{\mathsf{X}, \mathsf{X}^{-1}, \mathsf{F}, \mathsf{F}^{-1}\}$ peut être traité en utilisant les équivalences ($\star$) et ($\star\star$) et celui des opérateurs booléens par induction.

Nous montrons enfin que la formule ϕ est satisfaisable ssi $\phi_{3k} \wedge f(\phi)$ est satisfaisable. Considérons une séquence d'entiers σ_1 et un modèle $\sigma_k : \mathbb{N} \to (V \to \mathbb{N})$ défini par $\sigma_k(i)(x_j) = \sigma_1(3ik + 3j)$ pour tout $i \in \mathbb{N}$ et $x_j \in V$. Notons que, pour tout modèle σ_k, on peut construire un modèle σ_1 tel que pour tout $i \in \mathbb{N}$ et $x_j \in V$ on a $\sigma_k(i)(x_j) = \sigma_1(3ik+3j)$ et $\sigma_1 \models \phi_{3k}$ car $\mathbb{N}$ contient assez de valeurs distinctes pour pouvoir définir assigner aux états intermédiaires des valeurs satisfaisant la contrainte exprimée par ϕ_{3k}. Sans perte de généralité nous supposons donc que $\sigma_1 \models \phi_{3k}$. Nous pouvons montrer par induction sur la structure de ϕ que pour tout $i \in \mathbb{N}$ on a $\sigma_1, 3ik \models \phi_{3k} \wedge f(\phi)$ ssi $\sigma_k, i \models \phi$.

– Si ϕ est de la forme $(x_m = \mathsf{X}^j x_n)$ alors la formule $\phi_{3k} \wedge f(\phi)$ est équivalente à $\phi_{3k} \wedge \downarrow_{r=\mathsf{X}^{3m}x} \mathsf{X}^{3jk}(r = \mathsf{X}^{3n}x)$. Comme pour tout $i \in \mathbb{N}$ on a $\sigma_1(3ik + 3m) =$

$\sigma_1(3(i+j)k+3n)$ ssi $\sigma_k(i)(x_m) = \sigma_k(i+j)(x_n)$, la propriété est vérifiée.

- Si ϕ est de la forme $(x_m = \Diamond x_n)$ alors la formule $\phi_{3k} \wedge f(\phi)$ est équivalente à $\phi_{3k} \wedge \downarrow_{r=\mathsf{X}^{3m}x} \mathsf{X}^{3k}\mathsf{F}(\mathsf{X}^{-3n}(x = \mathsf{X}x) \wedge (r = x))$. Supposons que $\sigma_1, 3ik \models \phi_{3k} \wedge f(\phi)$. D'après ϕ_{3k}, pour tout $i \in \mathbb{N}$ une position $j > 3ik$ vérifie $\sigma_1(j) = \sigma(j+1)$ ssi $j = 3j'k$ pour $j' \in \mathbb{N}$. De plus si $\sigma_1, 3ik \models \phi_{3k} \wedge f(\phi)$ alors $\sigma_1(3ik+3m) = \sigma_1(3(i+j')k+3n)$ ce qui signifie que $\sigma_k(i)(x_m) = \sigma_k(i+j')(x_n)$ et donc que $\sigma_k \models \phi$.
  L'implication inverse est directe par construction puisque l'on suppose que $\sigma_1 \models \phi_{3k}$.
- Les cas restants peuvent facilement être traités par induction. □

Dans le cas infini, il semble difficile d'utiliser le même genre de réduction puisque le problème de la satisfaisabilité de $\mathrm{LTL}^{\downarrow}_{1,1}(\mathsf{X}, \mathsf{F})$ est déjà indécidable. Les résultats qui suivent généralisent le Théorème 32 en utilisant une méthode de traduction vers des systèmes à compteurs. Nous montrons que le problème de la satisfaisabilité de $\mathrm{CLTL}^{\Diamond}(\langle\mathbb{N}, =\rangle)$ sans restriction sur les opérateurs temporels est décidable à la fois pour les cas fini et infini.

6.3 Approche symbolique pour $\mathrm{CLTL}^{\Diamond}(\langle\mathbb{N}, =\rangle)$

Nous introduisons dans cette section une représentation symbolique des modèles de la logique $\mathrm{CLTL}^{\Diamond}(\langle\mathbb{N}, =\rangle)$. Nous expliquons ensuite comment cette abstraction est utilisée pour définir une procédure de décision pour le problème de la satisfaisabilité en réduisant ce problème à un problème pour les réseaux de Petri. Nous passons d'abord par une étape intermédiaire de traduction vers le problème du vide pour une classe particulière d'automates à compteurs.

6.3.1 Représentation symbolique des modèles

Soit X un ensemble fini de formules atomiques de $\mathrm{CLTL}^{\Diamond}(\langle\mathbb{N}, =\rangle)$. Nous posons $V = \{x_1, \ldots x_k\}$ l'ensemble des variables utilisées dans X et l le plus grand entier tel qu'un terme de la forme $\mathsf{X}^l x$ apparaît dans X. Nous définissons Ω_k^l comme étant l'ensemble composé des contraintes de la forme $\mathsf{X}^i x = \mathsf{X}^j y$ et $\mathsf{X}^i(x = \Diamond y)$ telles que $x, y \in \{x_1, \ldots x_k\}$ et $i, j \in \{0, \ldots, l\}$ ainsi que leurs négations.

L'abstraction des modèles que nous définissons est légèrement différente si l'on considère le cas fini ou bien le cas infini. Dans le cas infini, une valuation symbolique par rapport à X est un ensemble $sv \subseteq \Omega_k^l$ maximalement cohérent dans le sens où il satisfait les conditions suivantes :

(F1) Pour toute contrainte $\alpha \in \Omega_k^l$, soit α appartient à sv ou bien $\neg\alpha \in sv$, mais pas les deux (on identifie α et $\neg\neg\alpha$).

(F2) Pour tout $i \in \{0, \ldots, l\}$ et $x \in V$, on a $\mathsf{X}^i x = \mathsf{X}^i x \in sv$.

(F3) Pour tout $i, j \in \{0, \ldots, l\}$ et $x, y \in V$, on a $\mathsf{X}^i x = \mathsf{X}^j y \in sv$ ssi $\mathsf{X}^j y = \mathsf{X}^i x \in sv$.

(F4) Pour tout $i, j, j' \in \{0, \ldots, l\}$ et $x, y, z \in \{x_1, \ldots, x_k\}$, si $\{\mathsf{X}^i x = \mathsf{X}^j y, \mathsf{X}^j y = \mathsf{X}^{j'} z\} \subseteq sv$ alors $\mathsf{X}^i x = \mathsf{X}^{j'} z \in sv$.

(F5) Pour tout $i, j \in \{0, \ldots, l\}$ et $x, y \in V$ tels que $\mathsf{X}^i x = \mathsf{X}^j y \in sv$, on a :

- si $i = j$ alors pour tout $z \in \{x_1, \ldots, x_k\}$ on a $\mathsf{X}^i(x = \Diamond z) \in sv$ ssi $\mathsf{X}^j(y = \Diamond z) \in sv$;
- si $i < j$ alors $\mathsf{X}^i(x = \Diamond y) \in sv$, et pour tout $z \in V$, on a $\mathsf{X}^i(x = \Diamond z) \in sv$ ssi soit $\mathsf{X}^j(y = \Diamond z) \in sv$ ou il existe $i < j' \leq j$ tel que $\mathsf{X}^i x = \mathsf{X}^{j'} z \in sv$.

Notons que les conditions (F2) à (F4) expriment seulement que l'égalité est une relation d'équivalence. Les conditions de (F5) expriment la cohérence des contraintes de répétition. Nous notons $\mathrm{SV}(X)$ l'ensemble des valuations symboliques construites par rapport à X. Par extension, $\mathrm{SV}(\phi)$ est l'ensemble des valuations symboliques construit par rapport a l'ensemble des contraintes atomiques d'une formule ϕ de $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$. Un modèle infini σ satisfait une valuation symbolique sv à la position i ssi pour toute contrainte α de sv on a $\sigma, i \models \alpha$. Dans ce cas nous notons $\sigma, i \models sv$.

Une valuation symbolique telle qu'elle est définie ci-dessus exprime des contraintes sur l états consécutifs des variables. Dans le cas fini, il est nécessaire de considérer que le modèle peut se terminer avant la $l^{\text{ème}}$ position. Ceci permet entre autre de traiter le cas particulier où il existe un modèle dont la taille est inférieure à la longueur temporelle de la formule. Dans ce but nous définissons un ensemble de valuations symboliques $\mathrm{SV}'(X)$ différent pour le cas fini dont les éléments sont des paires composées d'un ensemble de contraintes maximalement cohérent et d'une information sur la taille du modèle. L'indication sur la taille du modèle est un élément de $\{0, \ldots, l\} \uplus \{\mathrm{nd}\}$ où nd est un élément particulier signifiant que le modèle ne termine pas avant la fin de la valuation symbolique. L'ensemble des conditions requises a besoin d'être adapté en fonction de cette nouvelle indication puisque si le modèle termine à la position définie par $i \in \{0, \ldots, l\}$ nous n'avons besoin de vérifier les conditions (F1) à (F5) qu'entre les positions 0 et i. Nous avons donc $\langle sv, i \rangle \in \mathrm{SV}'(X)$ ssi $i = \mathrm{nd}$ et sv vérifie les conditions (F1) à (F5) du cas infini ou bien $i \in \{0, \ldots, l\}$ et $sv \subseteq \Omega_k^i$ tel que les conditions suivantes sont respectées.

(F1') Pour toute contrainte $\alpha \in \Omega_k^i$, soit α appartient à sv ou bien $\neg\alpha \in sv$, mais pas les deux.

(F2') Pour tout $j \in \{0, \ldots, i\}$ et $x \in V$, on a $\mathsf{X}^j x = \mathsf{X}^j x \in sv$.

(F3') Pour tout $j, j' \in \{0, \ldots, i\}$ et $x, y \in V$, on a $\mathsf{X}^j x = \mathsf{X}^{j'} y \in sv$ ssi $\mathsf{X}^{j'} y = \mathsf{X}^j x \in sv$.

(F4') Pour tout $j, j', j'' \in \{0, \ldots, i\}$ et $x, y, z \in \{x_1, \ldots, x_k\}$, si $\{\mathsf{X}^j x = \mathsf{X}^{j'} y, \mathsf{X}^{j'} y = \mathsf{X}^{j''} z\} \subseteq sv$ alors $\mathsf{X}^j x = \mathsf{X}^{j''} z \in sv$.

(F5') Pour tout $j, j' \in \{0, \ldots, i\}$ et $x, y \in V$ tels que $\mathsf{X}^j x = \mathsf{X}^{j'} y \in sv$, on a :

- si $j = j'$ alors pour tout $z \in \{x_1, \ldots, x_k\}$ on a $\mathsf{X}^j(x = \Diamond z) \in sv$ ssi $\mathsf{X}^{j'}(y = \Diamond z) \in sv$;
- si $j < j'$ alors $\mathsf{X}^j(x = \Diamond y) \in sv$, et pour tout $z \in V$, on a $\mathsf{X}^j(x = \Diamond z) \in sv$ ssi soit $\mathsf{X}^{j'}(y = \Diamond z) \in sv$ ou il existe $j < j'' \leq j'$ tel que $\mathsf{X}^j x = \mathsf{X}^{j''} z \in sv$.

(F6') Pour toute contrainte $\mathsf{X}^j(x = \Diamond y) \in sv$ telle que $j \in \{0, \ldots, i\}$, il existe $j' \in \{j+1, \ldots, i\}$ tel que $\mathsf{X}^j x = \mathsf{X}^{j'} y \in sv$.

La dernière condition exprime que toutes les contraintes de répétitions doivent être satisfaites avant la fin du modèle. Nous notons $\mathrm{SV}'(\phi)$ l'ensemble des valuations symboliques construit par rapport à l'ensemble des contraintes atomiques d'une formule ϕ de

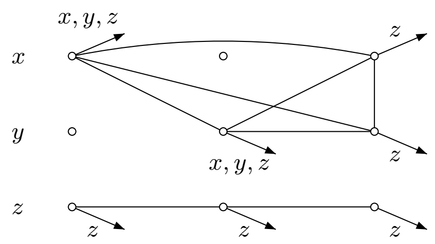


Figure 6.1: Exemple de représentation graphique d'une valuation symbolique

CLTL$^{\Diamond}(\langle \mathbb{N}, = \rangle)$. Un modèle σ satisfait une valuation symbolique $\langle sv, i \rangle \in \mathrm{SV}'(X)$ à la position j ssi

- soit $i = \mathrm{nd}$ et $|\sigma| - (j+1) \geq l+1$
  ou $i = |\sigma| - (j+1)$
- et pour toute contrainte α de sv on a $\sigma, j \models \alpha$.

Nous notons alors $\sigma, j \models \langle sv, i \rangle$.

Lemme 38 *Soit X un ensemble de contraintes atomiques de* CLTL$^{\Diamond}(\langle \mathbb{N}, = \rangle)$ *et σ un modèle construit sur le même ensemble de variables. Pour tout $0 \leq i \leq |\sigma| - 1$, il existe une unique valuation symbolique $sv \in \mathrm{SV}(X)$ (respectivement $\langle sv, j \rangle \in \mathrm{SV}'(X)$) telle que $\sigma, i \models sv$ (respectivement $\sigma, i \models \langle sv, j \rangle$).*

Preuve : Par définition, l'ensemble $\{\{\sigma \mid \sigma \models sv\} \mid sv \in \mathrm{SV}(X)\}$ est une partition de l'ensemble des modèles de la forme $\mathbb{N} \to (V \to \mathbb{N})$ où V est l'ensemble des variables de X. Donc si σ est une séquence infinie, le $i^{\text{ème}}$ préfixe de σ a forcément une unique abstraction.

Si σ est un modèle fini alors deux cas se présentent. Si $|\sigma| - (i+1) \geq l+1$ alors la preuve est similaire au cas précédent car l'ensemble de contraintes vérifie les mêmes conditions que dans le cas infini. Sinon, la partition à considérer est $\{\{\sigma \mid \sigma \models \langle sv, |\sigma| - (i+1) \rangle\} \mid \langle sv, |\sigma| - (i+1) \rangle \in \mathrm{SV}'(X)\}$. $\square$

Une valuation symbolique sv de $\mathrm{SV}(X)$ ou $\mathrm{SV}'(X)$ peut être représentée par un graphe non orienté G_{sv} dont l'ensemble des sommets est composé des paires $\langle x, i \rangle$ telles que $x \in \{x_1, \ldots, x_k\}$ et $i \in \{0, \ldots, l\}$. Nous identifions le terme $\mathsf{X}^i x$ avec le sommet $\langle x, i \rangle$. Il existe une arête entre deux sommets $\langle x, i \rangle$ et $\langle y, j \rangle$ de G_{sv} ssi la contrainte $\mathsf{X}^i x = \mathsf{X}^j y$ appartient à sv. Chaque sommet du graphe $\langle x, i \rangle$ est annoté par un arc sans destination (ouvert) étiqueté par l'ensemble des *obligations futures* lié à ce sommet qui est défini par

$$\Diamond_{sv}(\langle x, i \rangle) \stackrel{\text{def}}{=} \{y \mid \mathsf{X}^i(x = \Diamond y) \in sv\}.$$

La Figure 6.1 représente une valuation symbolique dans le cas infini sur les variables $\{x, y, z\}$ avec une longueur temporelle égale à 2. Par souci de lisibilité, certaines arêtes qui peuvent être obtenues par transitivité ne sont pas représentées.

Le niveau d'un sommet $\langle x, i \rangle$ est sa composante i. Nous définissons la *classe d'équivalence de x au niveau i dans sv* comme étant l'ensemble

$$[\langle x, i \rangle]_{sv} \stackrel{\text{def}}{=} \{y \mid \mathsf{X}^i x = \mathsf{X}^i y \in sv\}.$$

Dans le cas fini, nous posons $\Diamond_{\langle sv,j\rangle}(\langle x,i\rangle) = [\langle x,i\rangle]_{\langle sv,j\rangle} = \emptyset$ si $i \neq \mathrm{nd}$ et $i > j$. La définition est similaire au cas infini lorsque $i = \mathrm{nd}$ ou $i \leq j$.

Une paire de valuations symboliques $\langle sv, sv'\rangle \in \mathrm{SV}(\phi)^2$ est *cohérente sur un pas* ssi elle vérifie les propriétés suivantes :

- pour tout $i, j \in \{1, \ldots, l\}$, on a $\mathsf{X}^i x = \mathsf{X}^j y \in sv$ ssi $\mathsf{X}^{i-1} x = \mathsf{X}^{j-1} y \in sv'$,
- pour tout $i \in \{1, \ldots, l\}$, on a $\mathsf{X}^i(x = \Diamond y) \in sv$ ssi $\mathsf{X}^{i-1}(x = \Diamond y) \in sv'$.

Une paire de valuations symboliques $\langle\langle sv, i\rangle, \langle sv', i'\rangle\rangle \in \mathrm{SV}'(\phi)^2$ dans le cas fini est cohérente sur un pas ssi

- $\langle sv, sv'\rangle$ vérifie les conditions du cas infini,
- $i \neq 0$,
- si $i \in \{1, \ldots, l\}$ alors $i' = i - 1$ sinon $i' \in \{l + 1, \mathrm{nd}\}$.

Un *modèle symbolique* par rapport à une formule ϕ est une séquence ρ de valuations symboliques telle que pour tout $0 \leq i < |\rho| - 1$, la paire $\langle \rho(i), \rho(i+1)\rangle$ est cohérente sur un pas. Nous disons que la séquence ρ est *cohérente pas à pas*. Un modèle symbolique fini doit terminer par une valuation symbolique de la forme $\langle sv, 0\rangle$. Cette condition, associée à la cohérence pas à pas, implique qu'à toute position i d'un modèle symbolique fini ρ on a

- si $|\rho| - (i+1) \geq l + 1$ alors $\rho(i)$ est de la forme $\langle sv, \mathrm{nd}\rangle$,
- sinon $\rho(i)$ est de la forme $\langle sv, |\rho| - (i+1)\rangle$.

Un modèle σ satisfait un modèle symbolique ρ ssi $|\sigma| = |\rho|$ et pour tout $0 \leq i \leq |\rho| - 1$ on a $\sigma, i \models \rho(i)$.

Nous définissons une relation de satisfaction symbolique notée $\rho, i \models_{\mathrm{symb}} \phi$ entre les modèles symboliques et les formules de $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, =\rangle)$. Cette relation est définie de la même manière que la relation de satisfaction de $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, =\rangle)$ excepté au niveau atomique. Étant donnés une contrainte atomique α, un modèle symbolique ρ et une position $i \in \mathbb{N}$, nous notons

$$\rho, i \models_{\mathrm{symb}} \alpha \text{ ssi } \alpha \in \rho(i).$$

Par abus de notation, nous notons $\alpha \in \langle sv, i\rangle$ dans le cas fini si la contrainte α appartient à l'ensemble sv. Dans le cas fini, si $0 \leq |\rho| - (i+1) \leq l$ alors $\rho(i)$ ne peut satisfaire de contrainte référant à des états hors du modèle car la valuation symbolique est alors construite par rapport à $\Omega_k^{|\rho|-(i+1)}$.

Soit ϕ une formule de $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, =\rangle)$ construite avec un ensemble fini de variables V telle que $|\phi|_{\mathsf{X}} = l$. Un modèle symbolique ρ par rapport à ϕ peut être représenté par un graphe G_ρ de la même façon qu'une valuation symbolique de $\mathrm{SV}(\phi)$. Les sommets de G_ρ sont des paires de la forme $\langle x, i\rangle \in V \times \mathbb{N}$ et on a une arête entre $\langle x, i\rangle$ et $\langle y, j\rangle$ ssi $0 \leq j - i \leq l$ et il existe une arête entre $\langle x, 0\rangle$ et $\langle y, j - i\rangle$ dans la représentation graphique définie pour la valuation symbolique $G_{\rho(i)}$. Les annotations concernant les obligations futures sont ajoutées de la même manière. Notons que cette construction est bien définie car la séquence est cohérente pas à pas. Nous étendons les notations relatives aux obligations futures et aux

$$\begin{array}{lcccccc} x: & 4 & 1 & 4 & 9 & 4 & 3\cdots \\ y: & 7 & 4 & 8 & 4 & 7 & 4\cdots \end{array} \quad \models$$

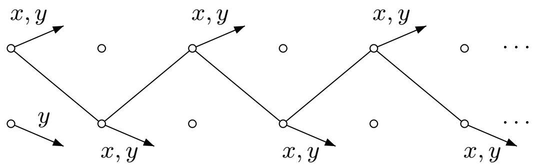


Figure 6.2: Exemple de représentation graphique d'un modèle symbolique

classes d'équivalence d'une valuation symbolique pour qu'elles s'appliquent aux modèles symboliques de la façon suivante :

$$\Diamond_\rho(\langle x, i\rangle) = \Diamond_{\rho(i)}(\langle x, 0\rangle) \qquad \text{et} \qquad [\langle x, i\rangle]_\rho = [\langle x, 0\rangle]_{\rho(i)}.$$

Dans Figure 6.2 nous donnons la représentation graphique d'un modèle symbolique avec une longueur temporelle égale à 1 ainsi qu'un modèle qui satisfait ce modèle symbolique. Un chemin dans G_ρ est une séquence (finie ou infinie) de sommets $n_0, n_1 \ldots$ telle que pour tout i les sommets n_i et n_{i+1} sont connectés par une arête. Un chemin est en avant si pour tout i le niveau de n_i et inférieur à celui de n_{i+1}. Par la suite, nous identifions parfois ρ avec sa représentation graphique G_ρ.

6.3.2 Automate Symbolique

L'approche symbolique que nous utilisons repose sur le résultat suivant.

Lemme 39 *Une formule ϕ de* CLTL$^\Diamond(\langle \mathbb{N}, =\rangle)$ *est satisfaisable ssi il existe un modèle symbolique ρ tel que $\rho \models_{\text{symb}} \phi$ et un modèle σ tel que $\sigma \models \rho$.*

Preuve : Nous développons d'abord le cas infini. Supposons qu'une formule ϕ de la logique CLTL$^\Diamond(\langle \mathbb{N}, =\rangle)$ soit satisfaite par un modèle infini σ et que $\mathrm{SV}(\phi) \subseteq \mathcal{P}(\Omega^l_k)$. À partir de σ, nous pouvons construire un modèle symbolique ρ tel que pour tout $i \in \mathbb{N}$ on a $\sigma, i \models \rho(i)$. D'après le Lemme 38, il existe pour tout $i \in \mathbb{N}$ une unique valuation symbolique $sv_i \in \mathrm{SV}(\phi)$ telle que $\sigma, i \models sv_i$. Nous posons $\rho(i) = sv_i$ pour tout $i \in \mathbb{N}$. Il reste à montrer que la séquence ρ est cohérente pas à pas. Supposons qu'il existe un indice $i \in \mathbb{N}$ tel que la paire $\langle \rho(i), \rho(i+1)\rangle$ n'est pas cohérente sur un pas. D'après la condition **(F1)** de la définition des valuations symboliques, il existe donc une contrainte $\alpha \in \Omega^l_k$ telle que $\mathsf{X}\alpha \in \rho(i)$ et $\neg\alpha \in \rho(i+1)$ où $\mathsf{X}\alpha$ est la contrainte de Ω^l_k obtenue à partir de α en remplaçant les occurrences de X^i par X^{i+1} pour tout $i \in \{0, \ldots, l-1\}$. Nous obtenons donc une contradiction car il est impossible d'avoir à la fois $\sigma, i \models \mathsf{X}\alpha$ et $\sigma, i+1 \models \neg\alpha$.

Par définition de la relation de satisfaction symbolique et la propriété **(F1)**, si $\sigma, i \models \alpha$ alors $\rho(i) \models_{\text{symb}} \alpha$. Par induction sur la structure de ϕ, nous pouvons alors prouver que $\rho \models_{\text{symb}} \phi$ car la relation de satisfaction symbolique est différente de la relation de satisfaction de CLTL$^\Diamond(\langle \mathbb{N}, =\rangle)$ seulement au niveau atomique.

Réciproquement, supposons qu'il existe un modèle infini σ et un modèle symbolique ρ tels que $\rho \models_{\text{symb}} \phi$ et $\sigma \models \rho$. Par définition, pour tout $i \in \mathbb{N}$ nous avons $\sigma, i \models \rho(i)$. Pour toute contrainte atomique α de ϕ telle que $\rho \models_{\text{symb}} \alpha$ on a $\sigma, i \models \alpha$ car $\alpha \in \rho(i)$ d'après la relation de satisfaction symbolique. Nous pouvons alors procéder par induction sur la structure de ϕ. D'après ce qui précède pour les contraintes atomiques et le fait que la relation de satisfaction symbolique corresponde à la relation de satisfaction de $\mathrm{CLTL}^{\Diamond}(\langle\mathbb{N},=\rangle)$ pour les autres cas, il est facile de conclure.

Le cas fini est semblable puisque d'après le Lemme 38 pour tout $i \in \{0,\ldots,|\sigma|\}$ il existe aussi une unique valuation symbolique $\langle sv_i, j_i\rangle \in \mathrm{SV}'(\phi)$ telle que $\sigma, i \models sv_i$. Nous posons $\rho(i) = \langle sv_i, j_i\rangle$ pour tout $i \in \{0,\ldots,|\sigma|\}$. Cette séquence est cohérente pas à pas puisque comme dans le cas infini les ensembles de contraintes successifs sont construits par rapport à des parties de σ qui se superposent partiellement et que l'indication de taille correspond aux nombres de positions restantes dans σ après la position courante. Nous pouvons alors procéder par induction sur la structure de ϕ. Le cas des contraintes atomiques peut être traité en utilisant la définition de la relation de satisfaction symbolique et la propriété (F1') (ou (F1) selon les cas). Nous faisons juste remarquer que par définition de la relation de satisfaction au niveau atomique, $\sigma, i \models \alpha$ et $|\sigma| - (i+1) \leq l$ implique que $\alpha \in \Omega_k^{|\sigma|-(i+1)}$ sinon α fait référence à des états en dehors du modèle. Les autres cas sont directs puisque la relation de satisfaction symbolique correspond à celle de $\mathrm{CLTL}^{\Diamond}(\langle\mathbb{N},=\rangle)$ dans ces cas. La réciproque peut aussi être traitée de manière analogue au cas infini. $\square$

Considérons une formule ϕ de $\mathrm{CLTL}^{\Diamond}(\langle\mathbb{N},=\rangle)$ construite avec un ensemble fini de variables $V = \{x_1,\ldots,x_k\}$ et telle que $|\phi|_{\mathsf{X}} = l$. Pour déterminer si ϕ est satisfaisable, nous étendons la construction d'automate de Büchi de [VW94] par rapport à notre abstraction. Nous définissons un automate $\mathcal{A}_\phi$ qui est l'intersection de trois automates $\mathcal{A}_{\mathrm{pap}}$, $\mathcal{A}_{\mathrm{symb}}$ et $\mathcal{A}_{\mathrm{sat}}$ tels que

- $\mathcal{A}_{\mathrm{pap}}$ reconnaît l'ensemble des modèles symboliques valides (cohérents pas à pas + conditions supplémentaires pour le cas fini),
- $\mathcal{A}_{\mathrm{symb}}$ reconnaît l'ensemble des modèles symboliques qui satisfont symboliquement ϕ et
- $\mathcal{A}_{\mathrm{sat}}$ reconnaît l'ensemble des modèles symboliques qui sont satisfaisables.

L'automate $\mathcal{A}_{\mathrm{pap}}$ vérifie que la séquence en entrée correspond bien à un modèle symbolique valide. Dans le cas infini, il faut juste vérifier la cohérence pas à pas. Pour le cas fini, nous devons aussi nous assurer que la dernière valuation symbolique est de la forme $\langle sv, 0\rangle$ et marque donc bien la fin du modèle. Nous donnons ci-dessous une définition générique de $\mathcal{A}_{\mathrm{pap}} = \langle Q, I, F, \delta\rangle$ pour les cas fini et infini.

- Q est l'ensemble des valuations symboliques par rapport à ϕ et $I = Q$,
- l'alphabet est aussi égal à l'ensemble des valuations symboliques par rapport à ϕ,
- la relation de transition est définie par $sv_1 \xrightarrow{sv} sv_2$ (respectivement $\langle sv_1, i_1\rangle \xrightarrow{\langle sv,i\rangle} \langle sv_2, i_2\rangle$) ssi
  - $sv_1 = sv$ (respectivement $\langle sv_1, i_1\rangle = \langle sv, i\rangle$) et
  - la paire $\langle sv_1, sv_2\rangle$ (respectivement $\langle\langle sv_1, i_1\rangle, \langle sv_2, i_2\rangle\rangle$) est cohérente sur un pas,
- l'ensemble des états finaux est tel que

– dans le cas infini $F = Q$,
– dans le cas fini $F = \{\langle sv, 0\rangle \mid \langle sv, 0\rangle \in \mathrm{SV}'(\phi)\}$.

Une exécution infinie est acceptée ssi elle visite infiniment souvent un état final (condition de Büchi). Dans le cas fini, une exécution est acceptante ssi elle termine dans un état final.

L'automate $\mathcal{A}_{\mathrm{symb}}$ est l'adaptation de l'automate défini pour la version propositionnelle de LTL dans [VW94]. Nous modifions cette construction en tenant compte des formules atomiques et de la relation de satisfaction symbolique de $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, =\rangle)$. Nous adaptons la clôture $cl(\phi)$ définie dans le cas propositionnel (avec les opérateurs passés) en considérant les formules atomiques comme des propositions. L'ensemble des atomes de ϕ que nous notons $\mathrm{Atom}(\phi)$ est composé des ensembles maximalement cohérents d'éléments de $cl(\phi)$. Pour le cas infini, l'automate $\mathcal{A}_{\mathrm{symb}}$ est l'automate de Büchi généralisé tel que

– l'ensemble des états est $Q = \mathrm{Atom}(\phi)$,
– l'ensemble des états initiaux I est le sous-ensemble de Q composé des atomes At tels que
  – $\phi \in At$,
  – pour toute formule $\mathsf{X}^{-1}\phi' \in cl(\phi)$ on a $\neg\mathsf{X}^{-1}\phi' \in At$,
– l'alphabet de $\mathcal{A}_{\mathrm{symb}}$ est $\mathrm{SV}(\phi)$,
– $At \xrightarrow{sv} At'$ ssi
  pour toute formule atomique α de At on a $\alpha \in sv$,
  pour tout $\mathsf{X}\psi \in cl(\phi)$, on a $\mathsf{X}\psi \in At$ ssi $\psi \in At'$ et
  pour tout $\mathsf{X}^{-1}\psi \in cl(\phi)$, on a $\psi \in At$ ssi $\mathsf{X}^{-1}\psi \in At'$.
– soit $\{\psi_1 \mathsf{U} \phi_1, \ldots, \psi_r \mathsf{U} \phi_r\}$ l'ensemble des formules “until” dans $cl(\phi)$. Si cet ensemble est vide alors $F = Q$. Sinon nous posons $F = \{F_1, \ldots, F_r\}$ tel que pour tout $i \in \{1, \ldots, r\}$ on a $F_i = \{At \in Q : \psi_i \mathsf{U} \phi_i \notin At \text{ ou } \phi_i \in At\}$.

Dans le cas fini, l'automate $\mathcal{A}_{\mathrm{symb}}$ est un automate fini sur l'alphabet $\mathrm{SV}'(\phi)$ qui accepte les entrées sur lesquelles l'exécution termine dans un état marquant la fin du modèle. Nous devons aussi tenir compte de la fin du modèle pour évaluer les sous-formules qui réfèrent à un état futur. Formellement, ceci se traduit par

– $Q = \mathrm{Atom}(\phi) \times \mathrm{SV}'(\phi)$,
– $I \subseteq Q$ tel que pour tout $\langle At, \langle sv, i\rangle\rangle \in I$ on a
  – $\phi \in At$,
  – pour toute formule $\mathsf{X}^{-1}\phi' \in cl(\phi)$ on a $\neg\mathsf{X}^{-1}\phi' \in At$,
– $\langle At, \langle sv, i\rangle\rangle \xrightarrow{\langle sv, i\rangle} \langle At', \langle sv', i'\rangle\rangle$ ssi
  si $i = \mathrm{nd}$ alors pour toute formule atomique α dans At on a $\alpha \in sv$,
  sinon pour toute formule atomique $\alpha \in \Omega^i_k \cap At$ on a $\alpha \in sv$,
  si $i = 0$ alors il n'existe pas de formule de la forme $\mathsf{X}\phi$ dans At.
  sinon pour tout $\mathsf{X}\psi \in cl(\phi)$, on a $\mathsf{X}\psi \in At$ ssi $\psi \in At'$.

pour tout $\mathsf{X}^{-1}\psi \in cl(\phi)$, on a $\psi \in At$ ssi $\mathsf{X}^{-1}\psi \in At'$.

– F est l'ensemble des états de la forme $\langle At, \langle sv, 0\rangle\rangle$.

La construction de $\mathcal{A}_{\mathrm{sat}}$ est développée plus tard. En admettant pour le moment que cet automate peut être construit, nous pouvons montrer le résultat suivant.

Théorème 33 *Une formule ϕ de* $\mathrm{CLTL}^{\Diamond}(\langle\mathbb{N}, =\rangle)$ *est satisfaisable ssi le langage accepté par $\mathcal{A}_\phi$ est non vide.*

Ce résultat est une conséquence directe du Lemme 39 et de la construction de $\mathcal{A}_\phi$ comme l'intersection de $\mathcal{A}_{\mathrm{pap}}$, $\mathcal{A}_{\mathrm{symb}}$ et $\mathcal{A}_{\mathrm{sat}}$.

6.4 Caractérisation des modèles symboliques réalisables

Pour déterminer si un modèle symbolique ρ est satisfaisable, nous associons à ρ un ensemble de compteurs qui mémorisent le nombre de contraintes de la forme $x = \Diamond y$ à satisfaire dans le futur. Par exemple, si la conjonction $x = \Diamond y_1 \wedge \cdots \wedge x = \Diamond y_n$ doit être satisfaite à la position courante et qu'aucune de ces contraintes n'est satisfaite par la valuation symbolique courante, nous incrémentons un compteur indexé par l'ensemble $\{y_1, \ldots, y_n\}$ afin de stocker cette obligation de répéter la valeur de x. Pour qu'un modèle fini soit satisfaisable il est nécessaire que toutes les obligations soit satisfaites avant la fin du modèle. Pour les modèles infinis, certaines obligations peuvent ne plus apparaître après une certaine position du modèle et les autres doivent être satisfaites infiniment souvent. Nous définissons la condition exacte dans cette section.

6.4.1 Séquence de comptage

Nous considérons une formule ϕ construite avec un ensemble de variables $V = \{x_1, \ldots, x_k\}$ et telle que $|\phi|_{\mathsf{X}} = l$. Pour chaque ensemble $X \in \mathcal{P}^+(V)$ non vide, nous introduisons un compteur représentant le nombre de valeurs distinctes qui doivent se répéter dans un état futur de chacune des variables de X. Nous identifions l'ensemble de ces compteurs avec $\mathcal{P}^+(V)$. Une valuation des compteurs est une fonction de la forme $v : \mathcal{P}^+(V) \to \mathbb{N}$ associant un entier naturel à chaque compteur. Par exemple, $v(\{x, y\})$ est la valeur du compteur $\{x, y\}$ c'est-à-dire le nombre de valeurs distinctes à répéter à la fois dans un état futur de x et un état futur de y.

Étant donnés une valuation symbolique $sv \in \mathrm{SV}(\phi)$ et un compteur $X \in \mathcal{P}^+(V)$, un *point d'incrémentation* pour X dans sv est une classe d'équivalence de la forme $[\langle x, 0\rangle]_{sv}$ telle que

– $\Diamond_{sv}(\langle x, 0\rangle) = X$ et
– $\langle x, 0\rangle$ n'est connecté à aucun autre noeud de sv par une arête vers l'avant, c'est-à-dire qu'il n'existe aucune arête entre $\langle x, 0\rangle$ et un autre noeud de la forme $\langle y, j\rangle$ tel que $j \in \{1, \ldots l\}$.

Une telle classe d'équivalence correspond à une valeur qui ne se répète plus dans un état futur de la valuation symbolique courante, elle ne se *propage* pas à un niveau supérieur. Nous parlons de propagation lorsqu'il existe une arête vers l'avant entre $\langle x, 0\rangle$ et $\langle y, j\rangle$ puisque un sous-ensemble des obligations associées à $\langle x, 0\rangle$ se retrouve dans $\langle y, j\rangle$ (condition (F5) des valuations symboliques). Si une valeur ne se propage pas, nous avons besoin de mémoriser les variables pour lesquelles cette valeur doit se répéter en incrémentant le compteur correspondant. Un *point de décrémentation* pour X dans sv est défini de manière symétrique. Il s'agit d'une classe d'équivalence de la forme $[\langle x, l\rangle]_{sv}$ telle que

- $\Diamond_{sv}(\langle x, l\rangle) \cup [\langle x, l\rangle]_{sv} = X$ et
- $\langle x, l\rangle$ n'est connecté à aucun autre noeud de sv par une arête vers l'arrière, c'est-à-dire qu'il n'existe aucune arête entre $\langle x, l\rangle$ et un autre noeud de la forme $\langle y, j\rangle$ tel que $j \in \{0, \dots l-1\}$.

Intuitivement, comme une telle classe d'équivalence n'est connectée à aucun état passé, nous pouvons fixer arbitrairement une valeur pour les variables de cette classe, ce qui permet de satisfaire au moins partiellement une obligation laissée insatisfaite jusqu'à présent. Notons que comme on a $X = \Diamond_{sv}(\langle x, l\rangle) \cup [\langle x, l\rangle]_{sv}$, on est assuré de ne pas oublier des obligations. En effet, les obligations de X qui ne sont pas satisfaites par la classe d'équivalence $[\langle x, l\rangle]_{sv}$ doivent être satisfaite plus tard car la valeur affectée à la classe d'équivalence doit se répéter dans au moins un état futur de chacune des variables de $\Diamond_{sv}(\langle x, l\rangle)$. Notons que la définition n'autorise pas de point d'incrémentation ou de décrémentation après la fin du modèle dans le cas fini car après la dernière position du modèle symbolique la classe d'équivalence de chaque sommet est vide. Nous notons u^+_{sv} la valuation des compteurs qui associe à chaque compteur X le nombre de points d'incrémentation pour X dans sv. De même, u^-_{sv} est la valuation des compteurs qui associe à chaque compteur X le nombre de points de décrémentation pour X dans sv.

Considérons un modèle symbolique ρ de la forme $\mathbb{N} \to \mathrm{SV}(\phi)$ (ou $\{0, \dots, |\rho|-1\} \to \mathrm{SV}'(\phi)$). Pour tout ensemble $X \in \mathcal{P}^+(\{x_1, \dots, x_k\})$, un point d'incrémentation pour X dans ρ est une classe d'équivalence de la forme $[\langle x, i\rangle]_\rho$ telle que $[\langle x, 0\rangle]_{\rho(i)}$ est un point d'incrémentation pour X dans la valuation symbolique $\rho(i)$. Un point de décrémentation pour X dans ρ est une classe d'équivalence de la forme $[\langle x, i\rangle]_\rho$ telle que $[\langle x, l\rangle]_{\rho(i-l)}$ est un point de décrémentation pour X dans la valuation symbolique $\rho(i-l)$.

Une séquence canonique de valuations symboliques γ_ρ par rapport au modèle symbolique ρ, ou *séquence de comptage*, compte les différentes obligations laissées insatisfaites à chaque position. Soit $\dot{+}$ la variante de l'opération d'addition sur les entiers définie par $n \dot{+} m = \max(0, n+m)$. La séquence de comptage γ_ρ est définie par induction de la façon suivante. Pour tout $X \in \mathcal{P}^+(V)$ on a

- $\gamma_\rho(0)(X) = 0$ et
- pour tout $0 \leq i \leq |\rho|-1$,

$$\gamma_\rho(i+1)(X) = \gamma_\rho(i)(X) \dot{+} (u^+_{\rho(i)}(X) - u^-_{\rho(i+1)}(X)).$$

6.4.2 Condition sur les modèles symboliques satisfaisables

Nous caractérisons les modèles symboliques satisfaisables en utilisant les séquences de comptage.

Lemme 40
(I) *Un modèle symbolique fini ρ est satisfaisable ssi la valeur finale de tous les compteurs de la séquence de comptage γ_ρ vaut* 0.

(II) *Un modèle symbolique infini ρ est satisfaisable ssi les conditions suivantes sont vérifiées.*

(C1) Il n'existe pas de chemin infini en avant dans G_ρ et de compteur X tels que chaque noeud du chemin a pour future obligation X et il existe une variable $y \in X$ telle qu'aucun noeud de la forme $\langle y, i\rangle$ n'est connecté à ce chemin par une arête en avant.

(C2) La séquence γ_ρ est telle que chaque compteur X satisfait l'une des conditions suivantes :
- *(a) il existe une position après laquelle la valeur du compteur X vaut toujours* 0 *et il n'existe plus de point d'incrémentation pour X,*
- *(b) il existe une infinité de points de décrémentation pour X qui sont connectés par un chemin en avant à un noeud $\langle x, i\rangle$ tel que $\Diamond_\rho(\langle x, i\rangle) \subset X$ dans G_ρ,*
- *(c) pour chaque $x \in X$ il existe une infinité de points de décrémentation pour X qui sont connectés par un chemin en avant à un noeud de la forme $\langle x, i\rangle$ dans G_ρ.*

Preuve : Soit ρ un modèle symbolique satisfaisant les conditions de l'énoncé et γ_ρ la séquence de comptage correspondante. Nous définissons ci-dessous un algorithme pour construire un modèle σ satisfaisant ρ.

Pour chaque compteur $X \in \mathcal{P}^+(\{x_1, \ldots, x_k\})$, nous introduisons une file FIFO $\texttt{Repeat}_X$ dont les éléments sont des paires de $\mathbb{N} \times X$ contenant une valeur à répéter et la prochaine variable de X qui doit prendre cette valeur. Le deuxième élément est utilisé dans le cas où le compteur est décrémenté infiniment souvent afin de s'assurer que toutes les variables de X prennent la valeur à répéter dans le futur. Pour les mêmes raisons, nous fixons arbitrairement un ordre sur les variables tel que $x_1 < x_2 < \ldots < x_k$ qui est utilisé comme ordre de répétition pour les valeurs. Nous introduisons aussi pour chaque variable $x \in \{x_1, \ldots, x_k\}$ un ensemble $\texttt{Forbid}_x$ qui contient toutes les valeurs que la variable x ne peut plus prendre. Chaque file $\texttt{Repeat}_X$ est initialisée par la file vide et chaque ensemble $\texttt{Forbid}_x$ par l'ensemble vide.

La construction de σ est équivalente à la construction d'un étiquetage pour la représentation graphique de ρ qui respecte les contraintes d'égalité et les obligations. Nous définissons un étiquetage qui associe à chaque sommet une valeur dans $\mathbb{N} \times (\{x_1, \ldots, x_k\} \cup \{\emptyset\})$, le second argument servant à garder une trace de certaines obligations laissées insatisfaites. Nous procédons par induction sur le niveau des sommets.

(I0) Pour la représentation de la première valuation symbolique, il existe une valuation qui respecte les relations d'égalité puisqu'une valuation symbolique est cohérente. Nous affectons les valeurs de cette valuation au premier élément de l'étiquette de chaque sommets. Le second élément de chacune des étiquettes de ces sommets est $\emptyset$.

(Ind) Supposons maintenant que le graphe est étiqueté jusqu'à la position $i+l$, c'est-à-dire jusqu'à la représentation de la valuation symbolique $\rho(i)$. L'étiquetage au niveau suivant est défini de la façon suivante.

Nous traitons d'abord les points d'incrémentation qui correspondent à une obligation qui n'est pas satisfaite et ne se propage pas dans la valuation symbolique courante. Il faut donc mémoriser cette obligation.

(Ind1) Supposons que l'étiquette de $\langle x, i\rangle$ est $\langle a, b\rangle$ et que $[\langle x, 0\rangle]_{\rho(i)}$ est un point d'incrémentation. Nous ajoutons à la fin de la queue $\texttt{Repeat}_{\Diamond_{\rho(i)}(\langle x,0\rangle)}$ la paire $\langle a', b'\rangle$ telle que

- $a = a'$,
- si $b \neq \emptyset$ alors $b' = b$
  sinon b' est la variable la plus petite dans $\Diamond_{\rho(i)}(\langle x, 0\rangle)$ (par rapport à l'ordre fixé).

Puis nous ajoutons la valeur a à chaque ensemble $\texttt{Forbid}_y$ tel que $y \notin \Diamond_{\rho(i)}(\langle x, 0\rangle)$. En effet, nous ne pouvons plus à partir de ce moment affecter la valeur a à une variable y qui n'est pas dans $\Diamond_{\rho(i)}(\langle x, 0\rangle)$ puisque cette absence correspond à une contrainte de la forme $\neg(x = \Diamond y)$ (d'après la condition **(F1)** des valuations symboliques).

(Ind2) Une fois ces opérations effectuées, nous définissons la valeur des étiquettes associées aux sommets du prochain niveau, c'est-à-dire les sommets de la forme $\langle x, i+l+1\rangle$. Plusieurs cas se présentent :

(Ind2.1) Supposons que le sommet $\langle x, i+l+1\rangle$ soit connecté par une relation d'égalité à un autre sommet $\langle y, j\rangle$ tel que $i < j < i + l + 1$ et soit $\langle a, b\rangle$ l'étiquette de $\langle y, j\rangle$. Le sommet $\langle x, i + l + 1\rangle$ est étiqueté par $\langle a', b'\rangle$ tel que

- $a = a'$,
- si $\Diamond(\langle y, j\rangle) = \Diamond(\langle x, i + l + 1\rangle)$ alors $b = b'$
  sinon $b = \emptyset$.

(Ind2.2a) Si $\langle x, i + l + 1\rangle$ n'est connecté à aucun autre sommet de niveau inférieur alors $[\langle x, l\rangle]_{\rho(i+1)}$ est un point de décrémentation pour un compteur X.

Si la file $\texttt{Repeat}_X$ est vide alors nous affectons l'étiquette $\langle a, \emptyset\rangle$ à tous les sommets de $[\langle x, l\rangle]_{\rho(i+1)}$ telle que a est une valeur qui n'est ni interdite, ni déjà utilisée afin d'éviter des égalités non souhaitées par la suite. Formellement, ceci nous donne

$$a \notin \bigcup_{y\in[\langle x,l\rangle]_{\rho(i+1)}} \texttt{Forbid}_y \cup \bigcup_{X\in\mathcal{P}^+(\{x_1,\ldots,x_k\})} \texttt{Repeat}_X$$

et a vérifie les contraintes imposées par les arcs.

(Ind2.2b) Si la file $\texttt{Repeat}_X$ n'est pas vide, alors nous avons encore plusieurs cas. Si nous considérons une séquence finie, ou que le compteur X satisfait la condition (C2a), ou encore si l'on peut atteindre par un chemin en avant un sommet de la forme $\langle y, j\rangle$ à partir de $\langle x, i + l + 1\rangle$ tel que $\Diamond(\langle y, j\rangle)$ est strictement inclus dans X alors nous enlevons le premier élément $\langle a, b\rangle$ de la file $\texttt{Repeat}_X$ et nous affectons à tous les sommets de $[\langle x, l\rangle]_{\rho(i+1)}$ l'étiquette $\langle a, \emptyset\rangle$.

(Ind2.2c) Dans le cas restant, nous enlevons le premier élément $\langle a, b\rangle$ de la file $\texttt{Repeat}_X$ tel que $b \in [\langle x, l\rangle]_{\rho(i+1)}$ et nous affectons à tous les sommets de $[\langle x, l\rangle]_{\rho(i+1)}$ l'étiquette $\langle a', b'\rangle$ telle que

- $a = a'$

– b' est la plus petite variable supérieure à b qui n'est pas accessible par un chemin en avant à partir du sommet $\langle x, i + l + 1\rangle$, si une telle variable n'existe pas alors $b = \emptyset$.

S'il n'existe pas d'élément de $\texttt{Repeat}_X$ satisfaisant les bonnes conditions alors nous procédons comme dans le cas (Ind 2.2.a) où la file est vide.

Nous démontrons maintenant que cet étiquetage satisfait bien toutes les contraintes induites par le graphe. Les contraintes d'égalité représentées par des arcs sont locales et sont bien respectées d'après les règles **(I0)**, **(Ind2.1)** et **(Ind2.2.a)**. En effet, pour toute position dans le modèle symbolique les valeurs affectées aux différentes classes d'équivalences de la valuation symbolique courante sont distinctes. Cette propriété est évidente pour la position 0. Si l'on suppose par induction que pour toute position $j < i$ chaque classe d'équivalence de la valuation symbolique courante a une valeur distincte alors une même valeur ne peut être ajoutée deux fois dans une file (voir **(Ind1)**). Ceci assure que l'ensemble des valeurs présentent dans les files et la valuation symbolique courante ne contient pas de doublon. À la position $i + 1$, les valeurs des classes d'équivalences sont affectées de l'une des façons suivantes :

– par propagation au niveau suivant,

– par suppression d'un élément d'une file (cas des points de décrémentation),

– par introduction d'une nouvelle variable **(Ind2.2.a)**.

Ainsi à la position $i+1$ toutes les valeurs affectées aux classes d'équivalences sont distinctes puisque les valeurs affectées dans les deux premiers cas sont distinctes par hypothèse, de même que dans le dernier cas par définition de l'opération **(Ind2.2.a)**.

La mise à jour des ensembles $\texttt{Forbid}_x$ (voir **(Ind1)**) et le choix des valeurs arbitraires (voir **(Ind2.2.a)**) impliquent que les contraintes d'interdiction d'une valeur pour chaque variable sont respectées. En particulier, si l'ensemble des obligations est vide alors celui-ci est satisfait. De plus cette remarque nous permet de procéder par induction sur le nombre de variables dans lesquelles la valeur doit se répéter que nous appelons taille de l'obligation. Le cas de base où $X = \emptyset$ est traité ci-dessus.

Considérons maintenant un ensemble d'obligations X et un sommet du graphe $\langle x, i\rangle$ tel que $\Diamond(\langle x, i\rangle) = X$ et nous supposons que pour tout sommet dont l'ensemble des obligations est strictement inclus dans X, ces obligations sont satisfaites **(H1)**.

(I) Considérons d'abord le cas fini. Si on n'atteint pas de point d'incrémentation par un chemin en avant à partir de $\langle x, i\rangle$ alors on atteint un sommet dont l'ensemble des obligations est strictement inclus dans $\Diamond(\langle x, i\rangle)$. En effet, les conditions sur les valuations symboliques dans le cas fini assure que les obligations sont satisfaites avant la fin du modèle, ce qui implique que lorsque l'on suit un chemin continu jusqu'à la dernière position du modèle l'ensemble des obligations associées à ce chemin diminue avant la fin. Dans ce cas, l'hypothèse **(H1)** permet de conclure.

Si on atteint un point d'incrémentation sans atteindre de sommet dont l'ensemble des obligations est strictement inclus dans $\Diamond(\langle x, i\rangle)$ alors la valeur *val* associée à $\langle x, i\rangle$ est insérée dans $\texttt{Repeat}_\texttt{X}$. Comme à la fin de la séquence de comptage tous les compteurs valent 0, on a autant de point de décrémentation pour X que de point d'incrémentation. De plus, d'après la règle **(Ind2.2b)**, pour chaque point de décrémentation on retire un élément de $\texttt{Repeat}_\texttt{X}$ ce qui permet de prouver que toutes les files sont vidées avant la fin de l'algorithme. En particulier, pour la valeur *val* qui nous intéresse deux cas se présentent.

- Soit le point de décrémentation utilisé pour retirer *val* de la file est tel qu'on peut atteindre un sommet dont l'ensemble des obligations est strictement inclus dans $\Diamond(\langle x, i\rangle)$. Dans ce cas, nous pouvons conclure.
- Sinon, on atteint un nouveau point d'incrémentation et *val* est réinsérée dans la file. En effet, nous rappelons qu'il est impossible d'atteindre la fin de la séquence sans qu'une obligation non-vide propagée le long du chemin ne diminue. Or, ceci ne peut se produire qu'un nombre fini de fois et donc il existe dans le futur un point de décrémentation qui permet de retirer *val* de la file et vérifie les conditions du point précédent.

Donc l'obligation X est bien satisfaite. Ceci prouve que si ρ est finie et que la valeur finale de tous les compteurs de la séquence de comptage γ_ρ vaut 0 alors ρ est satisfaisable.

Réciproquement, si ρ est satisfaisable alors la valeur de tous les compteurs à la fin de la séquence de comptage est bien zéro. Sinon, le compteur qui reste positif signale une obligation laissée insatisfaite d'après la définition d'un point d'incrémentation.

(II) Nous traitons maintenant le cas infini. Nous supposons que **(H1)** est vraie et nous montrons que l'obligation X de $\langle x, i\rangle$ est satisfaite par l'étiquetage. Plusieurs cas se présentent.

- *Supposons que le compteur associé à X satisfait la contraintes C2(a).*
  Avant tout, nous remarquons une propriété liée à la construction. Pour chaque compteur X vérifiant C2(a), il existe un entier que nous notons i_X tel que pour tout $j \geq i_X$, $\gamma_\rho(X)(j) = 0$. De plus, comme chaque point de décrémentation pour un compteur vérifiant (C2a) provoque la suppression d'un élément de la file correspondante lorsqu'elle n'est pas vide **(Ind2.2b)**. En comparant cette règle avec la définition d'une séquence de comptage, on s'aperçoit que pour tout $i \in \mathbb{N}$, le nombre d'éléments dans $\texttt{Repeat}_X$ après i itérations de l'étiquetage est égal à $\alpha_\rho(X)(i)$. En particulier, la file $\texttt{Repeat}_X$ est vide après i_X .
  Si $i \geq i_X$ alors le compteur n'est plus incrémenté par la suite. Il existe donc un chemin vers l'avant à partir de $\langle x, i\rangle$ tel que

  1. soit ce chemin passe par un sommet dont l'ensemble des obligations est strictement inclus dans X,
  2. soit ce chemin est infini.

  Un tel chemin existe sinon nous avons un point d'incrémentation pour X. Dans le premier cas, l'hypothèse d'induction nous permet de conclure. Pour le deuxième cas, la condition (C1) assure que toutes les contraintes de répétition imposées par X sont satisfaites.
  Nous procédons par induction pour les positions restantes. Nous venons de montrer que la propriété est vérifiée pour $i_0 \geq i_X$. Nous montrons que si la propriété est vérifiée

pour tout sommet $\langle x, i\rangle$ tel que $\Diamond(\langle x, i\rangle)_\rho = X$ et $(i_X - j) \leq i \leq i_X$ **(H2)** alors elle l'est aussi pour tout sommet $\langle y, (i_X - j - 1)\rangle$ tel que $\Diamond(\langle y, (i_X - j - 1)\rangle)_\rho = X$.
S'il existe une arête entre $\langle y, (i_X - j - 1)\rangle$ et un sommet $\langle y', j'\rangle$ tel que $j' > i_X - j - 1$ alors

- soit l'ensemble d'obligations associé à $\langle y', j'\rangle$ est strictement inclus dans X et nous pouvons utiliser l'hypothèse d'induction **(H1)** pour conclure,
- soit l'ensemble d'obligations associé à $\langle y', j'\rangle$ est égal à X et nous pouvons utiliser l'hypothèse d'induction **(H2)** pour conclure.

Sinon, $[\langle y, (i_X - j - 1)\rangle]_\rho$ est un point d'incrémentation et il doit exister un point de décrémentation $[\langle y', j'\rangle]_\rho$ tel que $(i_X - j - 1) < j' \leq i_X$ car la file doit être vide après i_X. Nous pouvons alors conclure de la même manière en utilisant l'hypothèse d'induction **(H1)** ou **(H2)** en fonction des obligations associées à $\langle y', j'\rangle$.

- *Si le compteur associé à X ne vérifie pas la condition (C2a) alors nous distinguons encore plusieurs cas.*

  S'il existe un chemin infini vers l'avant à partir de $\langle x, i\rangle$ alors

  - soit ce chemin passe par un sommet dont l'ensemble des obligations est strictement inclus dans X et nous pouvons conclure en utilisant **(H1)**,
  - soit ce chemin est infini et la condition (C1) nous assure que l'obligation est satisfaite.

  Sinon, nous pouvons atteindre à partir d'un chemin fini en avant un point d'incrémentation de la forme $[\langle x', i'\rangle]$ tel que $i \leq i'$. Si l'ensemble des obligation associé à $\langle x', i'\rangle$ est strictement inclus dans X nous pouvons toujours conclure en utilisant **(H1)**. Autrement, d'après la règle **(Ind1)** la valeur affectée à $\langle x, i\rangle$ est insérée dans $\texttt{Repeat}_X$ au cours de l'algorithme. Nous montrons maintenant que toute valeur insérée dans la file $\texttt{Repeat}_X$ satisfait les obligations correspondant à X. Notons que dans ce cas une obligation n'est pas satisfaite ssi un élément reste infiniment dans une file.
  Le cas pour lequel il existe une infinité de points de décrémentation pour X à partir desquels on peut atteindre un sommet dont l'ensemble des obligations est inclus strictement dans X est le plus facile (C2b). En effet, la règle **(Ind2.2b)** assure que tous les éléments peuvent être sortis de la file et nous pouvons utiliser l'hypothèse d'induction **(H1)** pour assurer que les obligations seront satisfaites.
  Sinon, à partir d'un certain point les éléments de $\texttt{Repeat}_X$ sont sortis uniquement à l'aide de la règle **(Ind2.2c)**. Notons, qu'aucun élément ne peut rester bloqué dans la file grâce à la condition (C3b) et le système de file FIFO. Alors, le système de rotation des variables de la deuxième composante des étiquettes imposé par **(Ind2.2c)** assure que toutes les obligations de X sont satisfaites.

Réciproquement, supposons que ρ est satisfaisable et ne satisfait pas (C1) ou (C2). *Si la condition (C1)* n'est pas vérifiée alors il existe un chemin infini en avant dans G_ρ dont tous les sommets ont pour obligation X et une variable $x \in X$ qui n'est jamais connectée par une arête avant à ce chemin. Alors l'obligation de répéter la valeur associée à ce chemin dans un état futur de x ne peut être satisfaite car la valeur de x dans tous les états futurs est différente des sommets du chemin (sinon on aurait une arête). Ceci contredit le fait que ρ soit satisfaisable.

Si ρ vérifie (C1) et pas (C2) alors il existe un compteur X qui ne vérifie aucune des conditions (C2a), (C2b) et (C2c). Puisque X ne vérifie pas C2(a), il existe une infinité de points d'incrémentation pour X, c'est-à-dire que le compteur est infiniment souvent positif. S'il n'existe qu'un nombre fini de points de décrémentation pour X alors il est évident que certaines obligations ne sont pas satisfaites ce qui nous conduit à une contradiction.

Nous supposons alors qu'il existe un nombre infini de points d'incrémentation et de décrémentation pour X. Puisque X ne satisfait pas (C2b), il n'existe qu'un nombre fini de points de décrémentation reliés par un chemin en avant à une sommet dont les obligations sont strictement incluses dans X. Après le dernier points de décrémentation vérifiant cette propriété, les autres points de décrémentation pour X ont pour obligation X et tous les sommets accessibles par des chemins en avant ont le même ensemble d'obligations. Or, comme X ne vérifie pas (C2c) il existe une variable $y \in X$ qui n'est reliée par un chemin en avant à ces points de décrémentation qu'un nombre fini de fois. Comme il existe une infinité de points d'incrémentation et de décrémentation qui propagent l'obligation X, il existe un position après laquelle l'obligation de répétition pour la variable y ne peut plus être satisfaite. Et nous obtenons à nouveau une contradiction. $\square$

Nous allons maintenant montrer que cette condition peut être reconnue par un automate à compteur particulier dont le test du vide est décidable.

6.5 Procédure de décision

6.5.1 Automates à compteurs avec test à zéro définitif

Nous introduisons maintenant une classe d'automates à compteurs avec une disjonction de condition de Büchi généralisée qui a pour particularité que dans toute exécution au plus un test à zéro est effectué. Cette classe d'automate nous permet de vérifier la condition de satisfaisabilité d'un modèle symbolique infini du Lemme 40.

Un *automate à compteurs avec test à zéro définitif* $\mathcal{A}$ est une structure $\langle Q, I, \mathcal{F}, \Sigma, C, \rightarrow \rangle$ telle que :

- Q est un ensemble fini d'états,
- Σ est un alphabet fini,
- C est un ensemble fini de compteurs,
- $I \subseteq Q$ est l'ensemble des états initiaux,
- l'ensemble des états finaux est donné sous la forme $\mathcal{F} = \{F_0, F_1, \dots, F_n\}$ où $n \geq 0$ et pour chaque $i = \{1, \dots, n\}$ l'ensemble F_i est un sous-ensemble de $\mathcal{P}(Q)$,
- la relation de transition $\rightarrow$ est un sous-ensemble fini de $Q \times \mathcal{P}(C) \times \mathbb{Z}^C \times \Sigma \times Q$.

Les transitions sont notées $q \xrightarrow{C_{taz},up,a} q'$ où la composante $C_{taz} \subseteq C$ est interprétée comme un test à zéro sur tous les compteurs appartenant à l'ensemble C_{taz}. Une configuration $\langle q, v \rangle$ est un élément de $Q \times \mathbb{N}^C$ composé d'un état et d'une valuation des compteurs. Nous définissons la sémantique sur un pas de la façon suivante : $\langle q, v \rangle \to \langle q', v' \rangle$ ssi il existe une transition $q \xrightarrow{Y,up,a} q'$ dans l'automate $\mathcal{A}$ telle que pour tout compteur $X \in C$ on a $v(X) = 0$ si $X \in Y$ et $v'(X) = v(X) + up(X)$. Une exécution est une séquence de configurations régie par la relation de transition de $\mathcal{A}$. Une exécution infinie est acceptée ssi il existe un ensemble $F_i \in \mathcal{F}$ tel que chaque ensemble d'états de F_i est visité infiniment souvent. L'ensemble des séquences de Σ^ω qui étiquettent les transitions successives des exécutions acceptées forme le langage reconnu par $\mathcal{A}$.

La graphe des configurations d'un automate à compteurs avec test à zéro définitif $\mathcal{A}$ doit vérifier certaines conditions supplémentaires. Il existe une partition $\{Q_0, \ldots, Q_n\}$ de l'ensemble des états Q et des ensembles correspondants de compteurs $C_0, C_1, \ldots, C_n$ tels que

- $C_0 = \emptyset$,
- $I \subseteq Q_0$,
- pour tout $i \in \{1, \ldots, n\}$ une transition d'un état de Q_0 vers un état de Q_i ne peut être franchie que si les compteurs de C_i sont égaux zéro,
- toute transition d'un état de $Q_i \neq Q_0$ va vers un autre état de Q_i,
- toute transition vers un état de Q_i ne modifie pas la valeur des compteurs de C_i.

Une conséquence des ces propriétés est que lorsqu'une exécution entre dans une configuration dont l'état de contrôle appartient à $Q_i \neq Q_0$, les compteurs de C_i sont à zéro et le reste par la suite. Enfin, pour tout $i \in \{0, \ldots, n\}$, on a $F_i \subseteq \mathcal{P}(Q_i)$. Formellement, ces conditions se résument par :

1. $Q = Q_0 \uplus \cdots \uplus Q_n$ et $I \subseteq Q_0$,
2. $\mathcal{F} = \{F_0, F_1, \ldots, F_n\}$ où chaque $F_i \subseteq \mathcal{P}(Q_i)$,
3. il existe $n+1$ ensembles de compteurs $C_0, \ldots, C_n \subseteq C$ tels que $C_0 = \emptyset$ et la relation de transition $\to \subseteq Q \times \mathcal{P}(C) \times \mathbb{Z}^C \times \Sigma \times Q$ vérifie les conditions ci-dessous :
   pour tout $i, i' \in \{0, \ldots, n\}$, $q \in Q_i$ et $q' \in Q_{i'}$, les transitions de q vers q' sont de la forme $q \xrightarrow{Y,up,a} q'$ où
   (a) si $i \neq i'$ alors $i = 0$ et $Y = C_{i'}$,
   (b) si $i = i'$ alors $Y = \emptyset$,
   (c) pour tout $X \in C_{i'}$, $up(X) = 0$.

Étant donnée une formule ϕ construite sur un ensemble de variables V, l'automate à compteurs que nous construisons à la fin de cette section est tel que $C = \mathcal{P}^+(V)$ et $n = 2^{|V|} - 1$, ce qui correspond aux compteurs des différentes obligations.

Le résultat ci-dessous établit la décidabilité du problème du vide pour les automates à compteurs avec test à zéro définitif grâce un résultat de [Jan90] sur les réseaux de Petri.

Lemme 41 *Le problème du vide pour les automates à compteurs avec test à zéro définitif est décidable.*

Preuve : Nous définissons une réduction de ce problème au problème $\mathbb{P}_{\text{temp}}$ qui consiste à vérifier des propriétés d'équité sur les réseaux de Petri. Ce problème est décidable d'après [Jan90].

Nous rappelons d'abord le fragment de ce problème qui nous intéresse. Un réseau de Petri $N = \langle S, T, W, M_0 \rangle$ est une structure telle que

- S est un ensemble fini de places,
- T est un ensemble fini de transitions,
- $W : (S \times T) \cup (T \times S) \to \mathbb{N}$ est une fonction de poids,
- M_0 est un marquage initial des places.

Nous supposons que le lecteur est familier avec la sémantique de ce modèle (pour plus d'informations, voir par exemple [Pet81]). En quelques mots, $W(s,t)$ définit le nombre de jetons nécessaires pour activer la transition t. Lorsqu'il existe une transition t telle que pour tout $s \in S$ le nombre de jetons dans la place s est supérieur à $W(s,t)$ cette transition peut être franchie. Alors, pour toute place $s \in S$ un nombre $W(s,t)$ de jetons est consommé dans la place s puis $W(t,s)$ jetons sont produits cette même place.

Nous considérons le fragment du langage $L(Q', \mathsf{GF})$ de [Jan90] dont les formules sont définies par la grammaire suivante :

$$\phi ::= s = i \;\mid\; \phi \vee \phi \;\mid\; \phi \wedge \phi \;\mid\; \mathsf{GF}\phi,$$

où $s \in S$ et $i \in \mathbb{N}$. La formule atomique $s = i$ exprime que le nombre de jetons dans la place s est i. Les opérateurs G et F ont leur signification habituelle et donc la formule $\mathsf{GF}\phi$ signifie que ϕ est vérifiée infiniment souvent. Dans sa définition générale, le problème $\mathbb{P}_{\text{temp}}$ prend comme entrée une formule ϕ de $L(Q', \mathsf{GF})$ et un marquage initial M_0 pour un réseau de Petri N et vérifie qu'il existe une exécution infinie qui satisfait ϕ.

Considérons un automate à compteurs avec test à zéro définitif $\mathcal{A} = \langle Q, I, \mathcal{F}, \Sigma, C, \to \rangle$ tel que $\{Q_0, \dots, Q_n\}$ est la partition de Q qui correspond aux différents tests à zéros. Nous construisons à partir de cet automate un réseau de Petri $N_{\mathcal{A}}$ qui simule le comportement de $\mathcal{A}$ sans les tests à zéro. Cette traduction d'un automate à compteurs sans test à zéro vers un réseau de Petri est standard [Reu90].

1. Pour chaque état de contrôle q de $\mathcal{A}$ nous introduisons une place s_q dans $N_{\mathcal{A}}$ et pour chaque compteur c nous introduisons une place s_c.
2. Pour chaque transition de la forme $q \xrightarrow{Y,up,a} q'$ appartenant à l'automate $\mathcal{A}$, nous définissons une transition dans $N_{\mathcal{A}}$ qui
   - consomme un jeton de la place s_q,
   - produit un jeton dans la place $s_{q'}$ et,
   - pour tout $c \in C$, produit ou consomme $up(c)$ jetons dans la place s_c selon si $up(c)$ est respectivement positif ou négatif.

Les tests à zéro de l'automate de départ sont pris en compte dans la formule de $L(Q', \mathsf{GF})$ définie plus bas. Un marquage initial pour $N_{\mathcal{A}}$ contient un unique jeton dans une place de la forme s_{q_0} telle que $q_0 \in I$ est un état initial de $\mathcal{A}$. Il n'y a aucun autre jeton dans les autres places, en particulier celles de la forme s_c, ce qui signifie que la valeur initiale des compteurs est 0. Notons qu'à partir de ce marquage initial, tous les marquages obtenus sont tels qu'une unique place de la forme s_q telle que $q \in Q$ a un jeton (voir la relation de transition). Donc à tout moment de l'exécution, un seul état de contrôle de l'automate à compteur de départ est actif.

Nous affirmons que tester si l'automate $\mathcal{A}$ admet une exécution acceptante est équivalent à vérifier si la formule suivante est vraie dans $N_{\mathcal{A}}$.

$$\phi \equiv \bigvee_{0 \leq i \leq n} \big(\mathsf{GF}(\bigwedge_{c \in C_i} s_c = 0) \wedge (\bigwedge_{Y \in F_i} \mathsf{GF}(\bigvee_{q \in Y} s_q = 1)\big)$$

Par construction, nous avons une correspondance entre la relation de transition de $\mathcal{A}$ et celle de $N_{\mathcal{A}}$. Donc s'il existe un $i \in \{0, \ldots, n\}$ tel que pour tout $Y \in F_i$ on a infiniment souvent $(\bigvee_{q \in Y} s_q = 1)$ alors la condition d'acceptation est vérifiée dans $\mathcal{A}$.

Pour montrer qu'une exécution de $N_{\mathcal{A}}$ qui vérifie ϕ correspond à une exécution acceptante de $\mathcal{A}$, il nous reste à montrer que lorsque l'on franchit une transition qui correspond à un passage vers un état de Q_i alors les compteurs de C_i valent bien 0. Si la formule ϕ est vérifiée alors il existe un $i \in \{0, \ldots, n\}$ tel qu'on reste bloquer dans la composante correspondant aux états de Q_i à cause de la condition d'acceptation. En effet, comme $N_{\mathcal{A}}$ est obtenu par traduction d'un automate à compteurs avec tests à zéro définitif, si on obtient un marquage tel que $s_q = 1$ pour un état $q \in Q_i$ tel que $Q_i \neq Q_0$ alors pour tout marquage obtenu par la suite il existe $q' \in Q_i$ tel que $s_{q'} = 1$ puisque toutes les transitions sortant d'un état de Q_i vont vers un autre état de Q_i. De plus, pour tout marquage obtenu tel que $s_q = 1$ pour un état $q \in Q_i$ le nombre de jetons dans les places s_c telles que $c \in C_i$ ne change pas par rapport au marquage précédent car la transition correspondante dans $\mathcal{A}$ ne modifie pas la valeur des compteurs de c. Donc si on obtient un marquage tel que $s_q = 1$ pour un état $q \in Q_i$ tel que $Q_i \neq Q_0$ alors le nombre de jetons les places s_c telles que $c \in C_i$ n'est plus jamais modifié. Ainsi, tout marquage obtenu où il existe un état $q \in Q_i$ tel que $s_q = 1$ et $Q_i \neq Q_0$ doit vérifier $s_c = 0$ pour tout $c \in C_i$, sinon $\mathsf{GF}(\bigwedge_{c \in C_i} s_c = 0)$ ne peut pas être satisfait car le nombre de jetons dans les places de la forme s_c telle que $c \in C_i$ ne change plus par la suite. Si $Q_i = Q_0$, alors on n'a pas besoin de test à zéro. Le test à zéro est donc bien simulé pour toute exécution de $N_{\mathcal{A}}$ qui satisfait ϕ et nous pouvons conclure que si $N_{\mathcal{A}}$ satisfait ϕ alors $\mathcal{A}$ a une exécution acceptante.

La réciproque est facile à montrer car il est évident qu'une exécution acceptante pour $\mathcal{A}$ est telle qu'il existe $i \in \{0, \ldots, n\}$ tel que pour tout $c \in C_i$ la valeur de c vaut zéro infiniment souvent (c est toujours égal à zéro après le test) et chaque ensemble de F_i est visité infiniment souvent (condition d'acceptation). L'exécution dans $N_{\mathcal{A}}$ correspondant à une telle exécution acceptante de $\mathcal{A}$ vérifie ϕ. $\square$

6.5.2 Automate pour les séquences de comptage satisfaisables

Nous pouvons maintenant construire l'automate $\mathcal{A}_{\text{sat}}$ évoqué dans la Section 6.3.2 qui reconnaît l'ensemble des modèles symboliques satisfaisables construits par rapport à ϕ. Nous commençons par décrire la construction correspondant au cas infini. L'automate à compteurs simple $\mathcal{A}_{\text{sat}}$ est défini comme l'intersection des automates $\mathcal{A}^1$ et $\mathcal{A}^2$ qui vérifient respectivement les conditions (C1) et (C2) du Lemme 40 (nous rappelons ces conditions plus bas). Comme la cohérence du modèle symbolique est vérifiée par l'automate $\mathcal{A}_{\text{pap}}$ dans la construction de $\mathcal{A}_{\phi}$, nous pouvons supposer que les séquences de valuations symboliques que nous considérons sont des modèles symboliques valides dans la construction de ces automates.

Le langage reconnu par $\mathcal{A}^1$ est l'ensemble des modèles symboliques $\rho : \mathbb{N} \to \mathrm{SV}(\phi)$ dans lesquels il n'existe pas de chemin infini en avant dans G_ρ et de compteur X dans la séquence de comptage associée à ρ tels que chaque noeud du chemin a pour future obligation X et il existe une variable $y \in X$ telle qu'aucun noeud de la forme $\langle y, i\rangle$ n'est connecté à ce chemin par une arête en avant. Nous définissons en fait l'automate de Büchi $\mathcal{A}^1$ à partir d'un automate $\tilde{\mathcal{A}}^1$ qui reconnaît le complément du langage accepté par $\mathcal{A}^1$. L'automate $\tilde{\mathcal{A}}^1$ accepte donc les séquences de valuations symboliques pour lesquelles il existe un chemin infini en avant dans la représentation graphique tel que

- tous les sommets du chemin ont la même obligation X,
- il existe une variable $x \in X$ telle qu'aucun sommet de la forme $\langle x, i\rangle$ n'est relié par une arête en avant au chemin.

Nous posons $Q = \{q_0\} \uplus (V \times \{0, \ldots l\})$ comme étant l'ensemble des états de $\tilde{\mathcal{A}}^1$, $I = \{q_0\}$ et $F = Q \setminus \{q_0\}$. La relation de transition de $\tilde{\mathcal{A}}^1$ est définie par

- $q_0 \xrightarrow{sv} q_0$ pour tout $sv \in \mathrm{SV}(\phi)$
  Cette règle permet de passer les différentes valuations symboliques de la séquence en attendant la position à laquelle commence le chemin.
- $q_0 \xrightarrow{sv} (\langle x, i\rangle, y)$ pour tout $sv \in \mathrm{SV}(\phi)$ tel que $y \in \Diamond(\langle x, i\rangle)_{sv}$
  Cette règle correspond au choix non-deterministe du début du chemin et de la variable y qui n'est jamais reliée à ce chemin.
- $(\langle x, i\rangle, y) \xrightarrow{sv} (\langle x, i-1\rangle, y)$ pour tout $i > 0$ et $sv \in \mathrm{SV}(\phi)$
  Cette règle sert juste à attendre que le sommet courant du chemin soit à la limite de la valuation symbolique courante avant de passer au prochain sommet.
- $(\langle x, 0\rangle, y) \xrightarrow{sv} (\langle x', i'\rangle, y)$ pour tout $sv \in \mathrm{SV}(\phi)$ vérifiant les conditions suivantes :
  1. $x = \mathsf{X}^{i'+1}x'$ est une contrainte de sv,
  2. $\Diamond(\langle x, 0\rangle)_{sv} = \Diamond(\langle x', i'+1\rangle)_{sv}$,
  3. pour tout $j \in \{1, \ldots, l\}$ la contrainte $x = \mathsf{X}^j y$ n'appartient pas à sv.

  La première condition exprime que $\langle x', i'\rangle$ est le prochain sommet du chemin et la deuxième que l'ensemble des obligations sur le chemin reste le même. Enfin, la dernière condition assure qu'aucun sommet correspondant à la variable y n'est connecté au chemin par une arête en avant.

L'automate $\mathcal{A}^1$ est l'automate de Büchi obtenu en complémentant l'automate ci-dessus.

Nous présentons maintenant l'automate $\mathcal{A}^2 = \langle Q, I, F, \Sigma, C, \to\rangle$ tel que $\Sigma = \mathrm{SV}(\phi)$, $C = \mathcal{P}^+(\{x_1, \ldots, x_k\})$ et $Q = \mathrm{SV}(\phi) \uplus \bigcup_{Z \subseteq C} Q_{\mathcal{A}_Z}$ où $Q_{\mathcal{A}_Z}$ est l'ensemble des états de l'automate $\mathcal{A}_Z$ défini plus bas et vérifiant les différentes conditions de (C2) énoncées dans le Lemme 40 après que l'on ait effectué un test à zéro sur les compteurs appartenant à l'ensemble Z. L'ensemble des états initiaux est défini par $I = \mathrm{SV}(\phi)$ et la relation de transition de $\mathcal{A}^2$ vérifie les règles suivantes.

– $sv \xrightarrow{\emptyset,up,sv} sv'$
pour tout $sv, sv' \in \mathrm{SV}(\phi)$ et up vérifiant $u^+_{sv}(X) - u^-_{sv'}(X) \leq up(X) \leq u^+_{sv}(X)$ pour tout $X \in C$.
– $sv \xrightarrow{Z,up,sv} q_{0,\mathcal{A}_Z}$
pour tout $sv \in \mathrm{SV}(\phi)$, où $q_{0,\mathcal{A}_Z}$ est l'état initial de l'automate $\mathcal{A}_Z$ et up vérifiant $u^+_{sv}(X) - u^-_{sv'}(X) \leq up(X) \leq u^+_{sv}(X)$ pour tout $X \in C \setminus Z$ et $up(X) = u^+_{sv}(X) = 0$ pour tout $X \in Z$. Notons que la condition impose qu'il n'existe aucun point d'incrémentation pour un compteur de Z dans sv.

L'automate de Büchi $\mathcal{A}_Z$ est défini par :

$$\mathcal{A}_Z = \mathcal{A}^{2a}_Z \cap \bigcap_{X \in C \setminus Z} (\mathcal{A}^{2b}_X \cup \mathcal{A}^{2c}_X) \qquad \text{où} \qquad \mathcal{A}^{2c}_X = \bigcup_{x \in X} \mathcal{A}_{X,x}.$$

tel que les différentes composantes de cet automate sont définies de la façon suivante.

– L'automate de Büchi $\mathcal{A}^{2a}_Z$ vérifie la condition (C2.a) pour les compteurs de Z et donc reconnaît l'ensemble des modèles symboliques dans lesquels il n'existe aucun point d'incrémentation pour un compteur appartenant à Z. Ceci assure que dans $\mathcal{A}_Z$ les compteurs de Z ne sont plus jamais modifiés et restent à zéro (puisque les compteurs de cet ensemble ont subi un test à zéro).
Cet automate ne comporte qu'un seul état $Q = \{q_0\}$ tel que $I = F = \{q_0\}$ et une exécution peut se poursuivre uniquement si on lit une valuation symbolique qui ne comporte aucun point d'incrémentation pour un compteur de Z. Formellement ceci nous donne la relation de transition suivante.

$q_0 \xrightarrow{sv} q_0$ pour toute valuation symbolique $sv \in \mathrm{SV}(\phi)$ telle que pour tout compteur X appartenant à Z, il n'existe pas de point d'incrémentation pour X dans sv.

– L'automate $\mathcal{A}^{2b}_X$ vérifie la condition (C2.b) pour un compteur $X \notin Z$. Cet automate accepte donc les modèles symboliques pour lesquels il existe une infinité de points d'incrémentation pour X à partir desquels on peut atteindre par un chemin vers l'avant un noeud de la forme $\langle x, i \rangle$ dont l'ensemble des obligations futures est strictement inclus dans X. Pour ce faire, $\mathcal{A}^{2b}_X$ procède de la façon suivante. L'automate choisit de façon non déterministe un sommet appartenant à point de décrémentation pour X. Il vérifie ensuite l'existence d'un chemin entre ce sommet et un sommet dont les obligations sont strictement incluse dans X. Lorsqu'un tel sommet est atteint, on entre dans l'état final. Un mot est accepté si cet état final est atteint infiniment souvent. Formellement, ceci nous donne :

– $Q = \{q_0, q_f\} \uplus (V \times \{0, \ldots, l\})$ est l'ensemble des états de $\mathcal{A}^{2b}_X$,
– $I = \{q_0\}$ et $F = \{q_f\}$,

- la relation de transition est définie par
  - $q_0 \xrightarrow{sv} q_0$ pour tout $sv \in \mathrm{SV}(\phi)$,
  - $q_0 \xrightarrow{sv} \langle x, l\rangle$ pour toute valuation symbolique $sv \in \mathrm{SV}(\phi)$ telle que $[\langle x, l\rangle]_{sv}$ est un point de décrémentation pour X,
  - $\langle x, i\rangle \xrightarrow{sv} \langle x, i-1\rangle$ pour tout $i > 0$ et $sv \in \mathrm{SV}(\phi)$,
  - $\langle x, 0\rangle \xrightarrow{sv} \langle y, i-1\rangle$ si $\Diamond(\langle x, 0\rangle)_{sv} = X$, $i > 0$ et $x = \mathsf{X}^i y$ est une contrainte de sv.
  - $\langle x, 0\rangle \xrightarrow{sv} q_f$ si $\Diamond(\langle x, 0\rangle)_{sv}$ est strictement inclus dans X.
  - $q_f \xrightarrow{sv} q_0$ pour tout $sv \in \mathrm{SV}(\phi)$.

- L'automate $\mathcal{A}_X^{2c}$ vérifie la condition (C2.c) pour un compteur $X \notin Z$. Sa construction est similaire au cas précédent sauf que l'on ne passe dans l'état final qu'à chaque fois que pour toute variable x de X on a deviné un point de décrémentation à partir duquel on peut atteindre un état de x.

Il est facile de vérifier que l'automate obtenu est un automate à compteurs avec test à zéro définitif. En effet la relation de transition de $\mathcal{A}^2$ assure que pour tout $Z \subseteq C$ les compteurs de Z valent bien zéro avant d'entrer dans $\mathcal{A}_Z$ puis la composante $\mathcal{A}_Z^{2a}$ de $\mathcal{A}_Z$ assure que les compteurs de Z ne sont plus modifiés. De plus toutes les transitions de $\mathcal{A}_Z$ restent dans $\mathcal{A}_Z$.

Dans le cas fini, la construction de $\mathcal{A}_{\mathrm{sat}}$ est plus simple. Cette construction reprend des éléments de la construction de $\mathcal{A}^2$. L'automate $\mathcal{A}_{\mathrm{sat}}$ pour le cas fini est un automate à compteur sans test à zéro $\langle Q, I, F, \Sigma, C, \rightarrow\rangle$ tel que

- $Q = \mathrm{SV}'(\phi)$ est un ensemble fini d'états, $I = Q$ et $F = Q$
- $\Sigma = \mathrm{SV}'(\phi)$,
- $C = \mathcal{P}^+(\{x_1, \ldots, x_k\})$ est un ensemble de compteurs,
- La relation de transition est définie par

$$\langle sv, i\rangle \xrightarrow{\emptyset, up, \langle sv, i\rangle} \langle sv', i'\rangle$$

  pour tout $\langle sv, i\rangle, \langle sv', i'\rangle \in \mathrm{SV}(\phi)$ et up vérifiant $u^+_{sv}(X) - u^-_{sv'}(X) \leq up(X) \leq u^+_{sv}(X)$ pour tout $X \in C$.

La condition d'acceptation pour cet automate est un peu particulière. Une exécution finie est acceptée ssi elle termine dans un état final avec tous les compteurs égaux à zéro. Notons que le problème du vide pour un tel automate se réduit à l'accessibilité d'un marquage dans un réseau de Petri puisqu'un automate à compteur sans test à zéro se traduit facilement en réseau de Petri (voir par exemple la preuve du Lemme 41).

Lemme 42 *Un modèle symbolique ρ est satisfaisable ssi ρ appartient au langage reconnu par $\mathcal{A}_{\mathrm{sat}}$.*

Preuve : Considérons le cas infini. Par construction, toutes les conditions du Lemme 40 sont testées par $\mathcal{A}_{\mathrm{sat}}$. Cependant, la séquence de comptage peut ne pas être canonique.

En effet, par définition on peut franchir dans une exécution acceptante π une transition $\rho(i) \xrightarrow{\emptyset, up, sv} \rho(i)$ dans $\mathcal{A}^2$ telle qu'il existe X où $\gamma_\rho(i)(X) + up(X) > \gamma_\rho(i)(X) \dot{+} (u^+_{\rho(i)}(X) - u^-_{\rho(i+1)}(X))$.

Cependant, la relation de transition de $\mathcal{A}^2$ autorise toutes les mises à jours entre $u^+_{\rho(i)}(X) - u^-_{\rho(i+1)}(X)$ et $u^+_{\rho(i)}(X)$ entre deux états $\rho(i)$ et $\rho(i+1)$ appartenant à $\mathrm{SV}(\phi)$. Ceci permet de trouver une exécution acceptante dans $\mathcal{A}^2$ suivant les mêmes états de contrôle que π telle que pour toute position i on a $\rho(i) \xrightarrow{\emptyset, up, sv} \rho(i+1)$ ssi pour tout $X \in C$ on a $\gamma_\rho(i)(X) + up(X) = \gamma_\rho(i)(X) \dot{+} (u^+_{\rho(i)}(X) - u^-_{\rho(i+1)}(X))$. Il suffit de prendre la transition qui décrémente le plus possible les compteurs à chaque étape (sans être négatif). Par construction, la valeur de chaque compteur est inférieure à celle de l'exécution originale. La position après laquelle les compteurs qui satisfont (C2a) valent toujours 0 peut donc être antérieure mais cette condition est toujours vérifiée. Les autres conditions ne sont pas influencées par la nouvelle valeur des compteurs. Cette exécution témoigne donc que ρ est satisfaisable.

Réciproquement, un modèle symbolique satisfaisable vérifie les conditions du Lemme 40. Donc il existe une exécution acceptante dans $\mathcal{A}^1$ puisque la séquence vérifie (C1). Le cas qui pose problème est la condition (C2.a) qui nous oblige à considérer la séquence de comptage. Or par définition pour tout $i \in \mathbb{N}$,

$$\gamma_\rho(i+1)(X) = \gamma_\rho(i)(X) \dot{+} (u^+_{\rho(i)}(X) - u^-_{\rho(i+1)}(X)).$$

ce qui signifie que

$$\gamma_\rho(i)(X) + (u^+_{\rho(i)}(X) - u^-_{\rho(i+1)}(X)) \leq \gamma_\rho(i+1)(X) \leq \gamma_\rho(i)(X) + u^+_{\rho(i)}(X).$$

Par définition de $\mathcal{A}^2$, il existe donc toujours une transition qui permet de mettre à jour les compteurs de la même manière que dans la séquence de comptage γ. Donc s'il existe un ensemble de compteurs Z qui restent à zéro après une certaine position i dans γ, alors il existe aussi une exécution de $\mathcal{A}^2$ où les compteurs de Z restent aussi à zéro après la position i. Par définition de $\mathcal{A}^2$ on peut ensuite entrer dans la composante $\mathcal{A}_Z$ qui accepte la séquence puisqu'elle vérifie la condition (C2) (par hypothèse les compteurs de Z ne sont plus modifiés après la position i).

De la même manière que dans le cas infini, nous pouvons construire dans le cas fini une séquence de comptage canonique pour ρ à partir d'une exécution acceptante dans $\mathcal{A}_{\mathrm{sat}}$. La condition imposant que la valeur finale de tous les compteur vaut 0 est préservée puisque la valeur des compteurs diminue. Réciproquement, toute séquence de comptage peut être simulée par l'automate $\mathcal{A}_{\mathrm{sat}}$ dans le cas fini. L'idée est la même que dans le cas infini puisque la définition de la relation de transition est similaire. D'après le Lemme 40 si ρ est satisfaisable alors il existe une séquence de comptage dont la valeur finale de tous les compteurs vaut 0. L'exécution qui correspond à cette séquence de comptage dans l'automate $\mathcal{A}_{\mathrm{sat}}$ est acceptante. □

Nous pouvons maintenant prouver le résultat principal de ce chapitre.

Théorème 34 *Le problème de la satisfaisabilité pour la logique* $\mathrm{CLTL}^{\diamond}(\langle \mathbb{N}, = \rangle)$ *est décidable dans les cas fini et infini.*

Preuve : Soit ϕ une formule de $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$. Nous construisons l'automate à compteurs simple $\mathcal{A}_\phi$ qui accepte l'intersection des langages reconnus par les automates $\mathcal{A}_{\mathrm{symb}}$, $\mathcal{A}_{\mathrm{sat}}$ et $\mathcal{A}_{\mathrm{pap}}$ de la manière suivante.

L'automate $\mathcal{A}_{\mathrm{symb}} \cap \mathcal{A}_{\mathrm{pap}}$ est un automate de Büchi standard construit en faisant le produit synchronisé des deux automates. Nous synchronisons ensuite cet automate avec $\mathcal{A}_{\mathrm{sat}}$ de la façon suivante : pour tout $q, q' \in \mathcal{A}_{\mathrm{symb}} \cap \mathcal{A}_{\mathrm{pap}}$ et $q_{\mathrm{sat}}, q'_{\mathrm{sat}} \in \mathcal{A}_{\mathrm{sat}}$, on a $\langle q, q_{\mathrm{sat}} \rangle \xrightarrow{X,up,sv} \langle q', q'_{\mathrm{sat}} \rangle$ ssi $q \xrightarrow{sv} q'$ est une transition de $\mathcal{A}_{\mathrm{symb}} \cap \mathcal{A}_{\mathrm{pap}}$ et $q \xrightarrow{X,up,sv} q'$ une transition de $\mathcal{A}_{\mathrm{pap}}$. Puisque $\mathcal{A}_{\mathrm{symb}} \cap \mathcal{A}_{\mathrm{pap}}$ n'a pas de compteur, l'automate obtenu reste un automate à compteurs avec test à zéro définitif une fois les conditions d'acceptation combinées. D'après le Lemme 41 tester que le langage qu'il reconnaît est non-vide est un problème décidable. On peut donc en déduire que le problème de satisfaisabilité est décidable en utilisant le Théorème 33.

La construction est la même dans le cas fini mais nous obtenons un automate à compteur fini. Étant donné la condition d'acceptation de $\mathcal{A}_{\mathrm{sat}}$ dans le cas fini, le problème de satisfaisabilité se réduit à un problème d'accessibilité dans les réseaux de Petri [Kos82] en utilisant la même méthode que dans le Lemme 41. La construction du réseau de Petri est identique et l'on cherche à savoir si l'on peut atteindre un marquage où toutes les places représentant un compteur sont vides et il existe une place représentant un état final ayant un jeton. Il est facile de montrer que l'accessibilité d'une telle configuration est équivalente à l'existence d'une exécution finie pour $\mathcal{A}_{\mathrm{sat}}$. $\square$

Les Théorèmes 33 et 34 peuvent être étendus à toute extension de LTL du moment que les opérateurs temporels sont définissables dans MSO grâce à [Büc62].

Corollaire 12 *Le problème de la satisfaisabilité pour la logique* $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ *étendue avec des opérateurs définissables dans MSO est décidable dans les cas fini et infini.*

6.5.3 Fragment dans PSPACE

Nous étudions maintenant le fragment $\mathrm{CLTL}_1^{\Diamond}(\langle \mathbb{N}, = \rangle)$ correspondant à la restriction de la logique $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ avec une seule variable. Notons que les modèles de ce fragment sont des séquences d'entiers et le seul compteur dont on a besoin est de la forme $\{x\}$ où x est la variable utilisée dans la formule. Pour toute séquence de comptage γ, nous identifions la valeur $\gamma(i)(\{x\})$ de ce compteur à une position i avec $\gamma(i)$. Étant donné un modèle symbolique ρ sur l'alphabet $\mathrm{SV}(\phi)$, la séquence de comptage γ_ρ est telle que pour tout $0 \leq i < |\rho| - 1$, on a $\gamma_\rho(i+1) = \gamma_\rho(i) + u_i^+ - u_{i+1}^-$ où $u_i^+, u_{i+1}^- \in \{0, 1\}$. D'après le Lemme 40, les conditions pour qu'un modèle symbolique construit sur une seule variable soit satisfaisable se réduisent à l'existence d'une séquence de comptage telle que l'unique compteur reste toujours égal à zéro après une certaine position ou bien est décrémenté infiniment souvent.

Lemme 43 *Un modèle symbolique* ρ *de la forme* $\{0, \ldots, |\rho| - 1\} \to \mathrm{SV}(\phi)$ *ou* $\mathbb{N} \to \mathrm{SV}(\phi)$ *tel que* ϕ *est une formule de* $\mathrm{CLTL}_1^{\Diamond}(\langle \mathbb{N}, = \rangle)$ *est satisfaisable ssi*

(I) *la dernière valeur de* γ_ρ *vaut* 0 *dans le cas fini,*

(II) *pour le cas infini*

- *soit il existe une position i_0 telle que pour tout $i > i_0$ on a $\gamma_\rho(i) = 0$,*
- *soit il existe un infinité de point de décrémentation pour le compteur.*

La vérification de cette condition est simplifiée par le fait que le compteur de toute séquence de comptage par rapport à un tel modèle symbolique est borné.

Lemme 44 *Pour tout modèle symbolique ρ construit par rapport à une formule du fragment* $\mathrm{CLTL}_1^\Diamond(\langle\mathbb{N},=\rangle)$*, la séquence de comptage γ_ρ vérifie $\gamma(i) \leq l$ pour tout $0 \leq i < |\rho| - 1$.*

Preuve : Soit ϕ une formule de $\mathrm{CLTL}_1^\Diamond(\langle\mathbb{N},=\rangle)$ et ρ un modèle symbolique infini de ϕ. Nous montrons que pour tout $i \in \mathbb{N}$, le nombre de classes d'équivalence distinctes dans $\rho(i)$, noté $\sharp(\rho(i))$, est borné par $l + 1 - \gamma_\rho(i)$. Nous procédons par induction sur i.

Pour $i = 0$, la valeur initiale de γ_ρ vaut 0. Il nous faut donc montrer que le nombre de classes d'équivalence dans $\rho(0)$ est inférieur ou égal à $l + 1$. Notons que cette propriété est toujours vraie pour n'importe quelle valuation symbolique car le pire cas qui puissent arriver est que tous les termes soient différents deux à deux, ce qui donne $l+1$ classes d'équivalence (une par terme).

Nous supposons maintenant que l'on a $\sharp(\rho(i)) \leq l + 1 - \gamma_\rho(i)$ et nous montrons que la propriété est vérifiée pour $i + 1$. Nous distinguons plusieurs cas

- Si $\gamma_\rho(i + 1) = \gamma_\rho(i) + 1$, alors nous avons obligatoirement $u_i^+ = 1$ et $u_i^- = 0$. Ceci implique que $[\langle x, 0\rangle]_{\rho(i)}$ est un point d'incrémentation et $[\langle x, l\rangle]_{\rho(i+1)}$ n'est pas un point de décrémentation. Par définition des points d'incrémentation et de décrémentation, $\langle x, i\rangle$ n'est égal à aucun sommet $\langle x, j\rangle$ pour $i < j \leq i + l$ mais par contre il existe un arête entre $\langle x, i + l + 1\rangle$ et un sommet $\langle x, j\rangle$ tel que $i < j \leq i + l$. Par conséquent le nombre de classes d'équivalence dans $\rho(i + 1)$ diminue de 1 par rapport à $\rho(i)$, ce qui signifie que $\sharp(\rho(i)) = \sharp(\rho(i+1)) - 1$. L'inégalité $\sharp(\rho(i+1)) \leq l+1-\gamma_\rho(i+1)$ est donc bien valable.
- Si $\gamma_\rho(i + 1) = \gamma_\rho(i) - 1$ alors nous avons obligatoirement $u_i^+ = 0$ et $u_i^- = 1$. Ceci implique que $[\langle x, 0\rangle]_{\rho(i)}$ n'est pas un point d'incrémentation et $[\langle x, l\rangle]_{\rho(i+1)}$ est un point de décrémentation. En suivant le même raisonnement que le cas précédent, nous obtenons que le nombre de classes d'équivalence augmente de 1 car ce cas est symétrique. L'inégalité $\sharp(\rho(i + 1)) \leq l + 1 - \gamma_\rho(i + 1)$ est donc aussi vérifiée.
- Si $\gamma_\rho(i + 1) = \gamma_\rho(i)$ et $u_i^+ = u_i^- = 1$ alors $[\langle x, 0\rangle]_{\rho(i)}$ est un point d'incrémentation et $[\langle x, l\rangle]_{\rho(i+1)}$ est un point de décrémentation. Ce cas peut se traiter de manière similaire aux précédents.
- Le cas restant nécessite un peu plus d'attention. En effet, si $\gamma_\rho(i+1) = \gamma_\rho(i)$ et $u_i^+ = 0$ alors deux cas peuvent se présenter :
  - soit $[\langle x, l\rangle]_{\rho(i+1)}$ n'est pas un point de décrémentation,
  - soit $[\langle x, l\rangle]_{\rho(i+1)}$ est un point de décrémentation mais on ne peut pas décrémenter car $\gamma_\rho(i) = 0$.

  Bien entendu, $[\langle x, 0\rangle]_{\rho(i)}$ lui n'est pas un point d'incrémentation.

Dans le premier cas, nous pouvons utiliser le même développement que pour les cas précédents pour montrer que $\sharp(\rho(i)) = \sharp(\rho(i+1))$ et ainsi déduire que l'inégalité est vérifié.
Dans le deuxième cas, nous avons $\gamma_\rho(i+1) = \gamma_\rho(i) = 0$. Nous pouvons alors utiliser la remarque faite au début de cette preuve pour montrer que $\sharp(\rho(i+1))$ est bien borné par $l+1$.

Puisque le nombre de classes d'équivalence dans chaque valuation est positif, pour tout $i \in \mathbb{N}$ nous avons donc $l + 1 - \gamma_\rho(i) > 0$ ce qui est équivalent à $l + 1 > \gamma_\rho(i)$.

La preuve dans le cas fini est essentiellement la même mais nécessite de considérer les valuations symboliques qui marquent la fin du modèle symbolique, ce qui donne un peu plus de cas différents. Ces valuations symboliques ne posent pas de problèmes car les termes hors du modèle n'introduisent pas de nouvelle classe d'équivalence. □

Ceci permet de construire un automate $\mathcal{A}^1_{\text{sat}}$ sans compteur dans le cas du fragment à une variable $\text{CLTL}^{\Diamond}_1(\langle\mathbb{N},=\rangle)$.

Lemme 45 *L'ensemble des modèles symboliques satisfaisables pour une formule du fragment* $\text{CLTL}^{\Diamond}_1(\langle\mathbb{N},=\rangle)$ *est reconnu par un automate de Büchi dans le cas infini, et par un automate fini dans le cas fini.*

Preuve : D'après le Lemme 43, dans le cas infini la séquence ρ est satisfaisable ssi il existe une position après laquelle la valeur du compteur vaut toujours 0 ou bien il existe une infinité de points de décrémentation pour le compteur. Comme le compteur est toujours compris entre 0 et l d'après le Lemme 44 nous pouvons construire l'automate de Büchi $\mathcal{A}^1_{\text{sat}} = \langle Q, I, F, \delta\rangle$ en codant la valeur du compteur dans les états de l'automate.

- L'ensemble des états Q est égal à $\text{SV}(\phi) \times \{0, \ldots, l\} \times \{\texttt{dec}, \neg\texttt{dec}, \texttt{zero}\}$. La deuxième composante code donc la valeur du compteur et la troisième est utilisée pour spécifier la condition d'acceptation.
- L'ensemble des états initiaux est $I = \{\langle sv, 0, \neg\texttt{dec}\rangle \mid sv \in \text{SV}(\phi)\}$.
- L'ensemble des états finaux est $F = \{\langle sv, 0, \texttt{zero}\rangle \mid sv \in \text{SV}(\phi)\} \cup \{\langle sv, i, \texttt{dec}\rangle \mid sv \in \text{SV}(\phi)$ et $i \in \{0, \ldots, l\}\}$.
- **(T1)** Pour tout $sv, sv' \in \text{SV}(\phi)$, $i \in \{0, \ldots, l\}$ et $u \in \{\texttt{dec}, \neg\texttt{dec}\}$ on a une transition $\langle sv, i, u\rangle \xrightarrow{sv} \langle sv', i', u'\rangle$ ssi la paire $\langle sv, sv'\rangle$ est cohérente sur un pas et
  - si $u^+_{sv} = 1$ et $u^-_{sv'} = 0$, alors $i' = i + 1$ et $u' = \neg\texttt{dec}$,
  - si $u^+_{sv} = 1$ et $u^-_{sv'} = 1$, alors $i' = i$ et $u' = \texttt{dec}$,
  - si $u^+_{sv} = 0$ et $u^-_{sv'} = 1$, alors
    soit $i' = i - 1 = 0$ et $u' \in \{\texttt{dec}, \texttt{zero}\}$
    ou $i' = i - 1 > 0$ et $u' = \texttt{dec}$,
    ou $i' = i = 0$ et $u' \in \{\neg\texttt{dec}, \texttt{zero}\}$,
  - si $u^+_{sv} = u^-_{sv'} = 0$, alors
    soit $i' = i = 0$ et $u' \in \{\neg\texttt{dec}, \texttt{zero}\}$,

ou $i' = i > 0$ et $u' = \neg\texttt{dec}$.

– **(T2)** Pour tout $sv \in \mathrm{SV}(\phi)$, on a une transition $\langle sv, 0, \texttt{zero}\rangle \xrightarrow{sv} \langle sv', 0, \texttt{zero}\rangle$ ssi la paire $\langle sv, sv'\rangle$ est cohérente sur un pas et $u^+_{sv} = u^-_{sv'}$.

Par construction un modèle symbolique satisfaisable est accepté par l'automate. Nous analysons les différents cas décrits dans le Lemme 43. S'il existe une position après laquelle le compteur reste égal à 0 alors d'après la relation de transition, il existe une exécution sur cette entrée où l'on entre dans un état de la forme $\langle sv, 0, \texttt{zero}\rangle$. D'après les règles de **(T1)** on a en effet toujours la possibilité d'entrer dans un tel état lorsque la deuxième composante, qui correspond à la valeur du compteur, vaut 0. Comme le compteur reste ensuite à zéro, la règle de transition **(T2)** peut toujours être utilisée. Si le compteur est décrémenté infiniment souvent mais qu'il est infiniment souvent positif alors il existe un exécution où l'on passe infiniment souvent par un états de la forme $\langle sv, i, \texttt{dec}\rangle$. En effet, les règles de **(T1)** permettent que l'on passe par un tel état à chaque fois que le compteur est décrémenté.

Réciproquement, par construction une séquence acceptée est cohérente pas à pas et correspond donc à un modèle symbolique de ϕ. De plus une séquence est acceptée si

– soit l'exécution reste infiniment dans la sous-composante formée par les états de la forme $\langle sv, 0, \texttt{zero}\rangle$ (voir la règle **(T2)**), ce qui signifie que le compteur reste toujours à 0 après une certaine position,

– soit on visite infiniment souvent un état de la forme $\langle sv, i, \texttt{dec}\rangle$, ce qui signifie que le compteur est décrémenté infiniment souvent (voir la règle **(T1)**).

Dans les deux cas la séquence est satisfaisable d'après le Lemme 43.

Pour le cas fini, nous voulons que l'exécution se termine avec le compteur égal à zéro. La construction se simplifie de la manière suivante :

– $Q = \mathrm{SV}'(\phi) \times \{0, \ldots, l\}$.

– $I = \{\langle\langle sv, i\rangle, 0\rangle \mid \langle sv, i\rangle \in \mathrm{SV}'(\phi)\}$.

– $F = \{\langle\langle sv, 0\rangle, 0\rangle \mid \langle sv, 0\rangle \in \mathrm{SV}'(\phi)\}$.

– Pour tout $\langle sv, i\rangle, \langle sv', i'\rangle \in \mathrm{SV}'(\phi)$ et $j \in \{0, \ldots, l{+}1\}$ on a une transition $\langle\langle sv, i\rangle, j\rangle \xrightarrow{\langle sv, i\rangle} \langle\langle sv', i'\rangle, j\rangle$ ssi la paire $\langle\langle sv, i\rangle, \langle sv', i'\rangle\rangle$ est cohérente sur un pas et $j' = \max(0, j + (u^+_{sv} - u^-_{sv'}))$. Notons que d'après le Lemme 44 $i' \in \{0, \ldots, l\}$.

La preuve de correction de cet automate est directe. Une exécution acceptante de l'automate $\mathcal{A}^1_{\mathrm{sat}}$ simule une séquence de comptage et la condition d'acceptation impose que le compteur vaut 0. □

La taille de l'automate $\mathcal{A}^1_{\mathrm{sat}}$ est exponentielle par rapport à $|\phi|$ mais cet automate peut être construit en espace polynomial. De plus, tester si le langage reconnu par le produit de cet automate avec l'automate $\mathcal{A}_{\mathrm{pap}} \cap \mathcal{A}_{\mathrm{symb}}$ de la Section 6.3.2 peut se faire en espace logarithmique non-deterministe (par rapport à la taille de l'automate obtenu) puisqu'il s'agit d'un automate de Büchi (ou d'un automate fini dans le cas fini). Ces éléments nous permettent d'établir le résultat de complexité suivant.

Théorème 35 *Le problème de la satisfaisablité pour* $\mathrm{CLTL}_1^{\Diamond}(\langle \mathbb{N}, = \rangle)$ *est* PSPACE*-complet dans les cas fini et infini.*

Preuve : La construction de $\mathcal{A}_\phi$ de la section précédente peut être simplifiée en utilisant $\mathcal{A}^1_{\mathrm{sat}}$ au lieu de $\mathcal{A}_{\mathrm{sat}}$. La propriété établissant que ϕ est satisfaisable ssi le langage reconnu par $\mathcal{A}_\phi$ est non-vide est préservée car $\mathcal{A}^1_{\mathrm{sat}}$ reconnaît l'ensemble des modèles symboliques satisfaisables et les autres composantes ne changent pas. D'après les remarques au dessus, la construction de l'automate de Büchi $\mathcal{A}_\phi$ peut se faire en espace polynomial. En composant, cette construction avec le test du vide pour un automate de Büchi nous obtenons une procédure dans PSPACE.

La borne supérieure est obtenue en utilisant la PSPACE-dureté du problème de la satisfaisabilité pour la logique $\mathrm{CLTL}_1(\langle \mathbb{N}, = \rangle)$ qui est incluse dans $\mathrm{CLTL}_1^{\Diamond}(\langle \mathbb{N}, = \rangle)$. En effet, le problème de la satisfaisabilité pour $\mathrm{CLTL}(\langle \mathbb{N}, = \rangle)$ est PSPACE-dur car on peut y réduire le problème de la satisfaisabilité pour LTL en codant les variables propositionnelles par des contraintes de la forme $x = y$. D'après le Corollaire 1 le fragment $\mathrm{CLTL}_1(\langle \mathbb{N}, = \rangle)$ est aussi PSPACE-dur.

La preuve pour le cas fini est similaire. □

Une conséquence directe de ce résultat est que le problème de la satisfaisablité dans les cas fini et infini de $\mathrm{LTL}^{\downarrow}_{1,1}(\{\mathsf{X}, \mathsf{X}^{-1}, \mathsf{U}, \mathsf{S}\})$ est dans PSPACE lorsque l'on restreint l'utilisation de l'opérateur freeze aux sous-formules de la forme $\downarrow_{r=x} \mathsf{XF}(r = x)$ et $\downarrow_{r=x} \mathsf{X}^i(r = x)$ (où r est l'unique registre).

6.5.4 Répétition de valeurs dans le passé

Nous montrons maintenant que nous pouvons étendre le langage logique pour exprimer des contraintes de répétition dans le passé tout en préservant la décidabilité. Nous considérons l'extension de $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, = \rangle)$ avec des contraintes de la forme $x = \Diamond^{-1} y$ que nous notons $\mathrm{CLTL}^{\Diamond, \Diamond^{-1}}(\langle \mathbb{N}, = \rangle)$. La relation de satisfaction est étendue de la façon suivante.

$$\sigma, i \models x = \Diamond^{-1} y \text{ ssi il existe } 0 \leq j < i \text{ tel que } x = \sigma(j)(y).$$

Notons que de la même manière que pour $x \neq \Diamond y$, la contrainte $x \neq \Diamond^{-1} y$ peut être exprimée dans le langage logique même si elle est omise dans la définition

$$x \neq \Diamond^{-1} y \;\Leftrightarrow\; (\neg(x = \mathsf{X}^{-1} y) \wedge \mathsf{X}^{-1}\top) \vee \big((x = \mathsf{X}^{-1} y) \wedge \mathsf{X}^{-1}((y = \mathsf{X}^{-1} y) \mathsf{S}((y \neq \mathsf{X}^{-1} y) \wedge \mathsf{X}^{-1}\top))\big).$$

Pour traiter le problème de la satisfaisabilité de $\mathrm{CLTL}^{\Diamond, \Diamond^{-1}}(\langle \mathbb{N}, = \rangle)$, nous devons étendre la représentation symbolique. Nous définissons Ω'^l_k l'extension de Ω^l_k avec les contraintes de la forme $\mathsf{X}^i(x = \Diamond^{-1} y)$ et leur négation pour $i \in \{0, \ldots, l\}$. Une valuation symbolique est toujours un ensemble maximalement cohérent de contraintes de Ω'^l_k. En plus des conditions

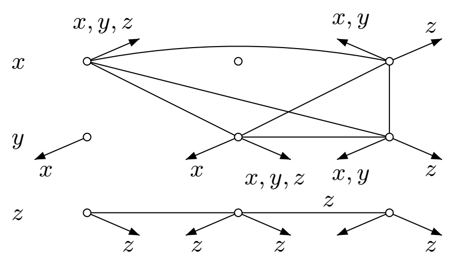


Figure 6.3: Représentation graphique d'une valuation symbolique avec obligations passées

(F1) à (F5) définie dans la Section 6.3.1 pour la représentation symbolique des modèles de $\mathrm{CLTL}^{\Diamond}(\langle\mathbb{N},=\rangle)$, une valuation symbolique de $\mathrm{CLTL}^{\Diamond,\Diamond^{-1}}(\langle\mathbb{N},=\rangle)$ doit vérifier les conditions supplémentaires suivantes exprimant la cohérence des répétitions passées :

(FP) pour tout $i,j \in \{0,\ldots,l\}$ et $x,y \in \{x_1,\ldots x_k\}$, si $\mathsf{X}^i x = \mathsf{X}^j y$ appartient à sv alors
- si $i = j$, alors pour tout $z \in \{x_1,\ldots,x_k\}$ on a $\mathsf{X}^i(x = \Diamond^{-1}z) \in sv$ ssi $\mathsf{X}^j(y = \Diamond^{-1}z) \in sv$,
- si $i > j$ alors $\mathsf{X}^i(x = \Diamond^{-1}y) \in sv$, et pour tout $z \in \{x_1,\ldots,x_k\}$, on a $\mathsf{X}^i(x = \Diamond^{-1}z) \in sv$ ssi soit $\mathsf{X}^j(y = \Diamond^{-1}z) \in sv$ ou bien il existe $i > j' \geq j$ tel que $\mathsf{X}^i x = \mathsf{X}^{j'} z$ appartient à sv.

Un modèle symbolique est toujours une séquence de valuations symboliques cohérente pas à pas, considérant notre nouvelle définition de valuations symboliques. Ceci signifie qu'une paire de valuations symboliques $\langle sv, sv'\rangle$ est cohérente sur un pas ssi

- pour tout $i,j \in \{1,\ldots,l\}$, on a $\mathsf{X}^i x = \mathsf{X}^j y \in sv$ ssi $\mathsf{X}^{i-1} x = \mathsf{X}^{j-1} y \in sv'$,
- pour tout $i \in \{1,\ldots,l\}$, on a $\mathsf{X}^i(x = \Diamond y) \in sv$ ssi $\mathsf{X}^{i-1}(x = \Diamond y) \in sv'$.
- pour tout $i \in \{1,\ldots,l\}$, on a $\mathsf{X}^i(x = \Diamond^{-1} y) \in sv$ ssi $\mathsf{X}^{i-1}(x = \Diamond^{-1} y) \in sv'$.

Toutes ces extensions s'appliquent naturellement au cas fini en considérant la définition de valuation symbolique pour le cas fini.

Nous étendons la représentation graphique des modèles à cette nouvelle abstraction. La Figure 6.3 est un exemple de cette nouvelle représentation graphique. La différence avec la Figure 6.1 est la présence d'arcs ouverts vers le passé représentant les répétitions dans le passé ou *obligations passées*. Nous posons

$$\Diamond^{-1}_{sv}(\mathsf{X}^i x) = \{y \mid \mathsf{X}^i(x = \Diamond^{-1} y) \in sv\}.$$

Comme nous devons maintenant aussi tenir compte des obligations passées, les compteurs sont des paires de la forme $\langle X_p, X_f\rangle$ appartenant à $\mathcal{P}^+(\{x_1,\ldots,x_k\})\times\mathcal{P}^+(\{x_1,\ldots,x_k\})$ telle que X_p représente les obligations passées et X_f les obligations futures. Les valuations pour les compteurs sont adaptées en conséquence. Pour préciser davantage, la valeur associée à $\langle X_p, X_f\rangle$ représente le nombre de valeurs distinctes qui sont apparues dans des états passés de chacune des variables de X_p et doivent se répéter dans un ou plusieurs états futurs de chacune des variables de X_f.

Nous expliquons maintenant comment la séquence de comptage est étendue. Un point d'incrémentation pour un compteur $\langle X_p, X_f \rangle$ dans une valuation symbolique sv est une classe d'équivalence de la forme $[\langle x, 0\rangle]_{sv}$ telle que

- $\Diamond_{sv}(\langle x, 0\rangle) = X_f$,
- $\langle x, 0\rangle$ n'est connecté à aucun autre noeud de sv par une arête vers l'avant
- et $[\langle x, 0\rangle]_{sv} \cup \Diamond_{fr}^{-1}(x, 0) = X_p$.

Un point de décrémentation pour un compteur $\langle X_p, X_f \rangle$ dans une valuation symbolique sv est une classe d'équivalence de la forme $[\langle x, l\rangle]_{sv}$ vérifiant

- $\Diamond_{sv}(\langle x, l\rangle) \cup [\langle x, l\rangle]_{sv} = X_f$,
- $\langle x, l\rangle$ n'est connecté à aucun autre noeud de sv par une arête vers l'arrière,
- et $\Diamond_{sv}^{-1}\langle x, l\rangle = X_p$.

Ces nouvelles définitions permettent d'enregistrer les répétitions des valeurs dans le passé. Les valeurs u_{sv}^{+} et u_{sv}^{-} dénotent toujours respectivement le nombre de points d'incrémentation et de décrémentation de chaque compteur dans sv. Étant donné un modèle symbolique ρ, la séquence de comptage γ_ρ est maintenant définie inductivement par

- $\gamma(0)(\langle X_p, X_f\rangle) = 0$ et
- pour tout $i \in \{0, \dots |\rho| - 1\}$, on a
$$\gamma(i+1)(\langle X_p, X_f\rangle) = \gamma(i)(\langle X_p, X_f\rangle) + (u_{\rho(i)}^{+}(\langle X_p, X_f\rangle) - u_{\rho(i+1)}^{-}(\langle X_p, X_f\rangle)).$$

Notons que la différence avec le cas de $\mathrm{CLTL}^{\Diamond}(\langle \mathbb{N}, =\rangle)$ est le caractère obligatoire de la décrémentation. En effet si un point de décrémentation porte pour information une répétition passée, alors on doit absolument trouver une valeur qui correspond parmi les valeurs qui doivent se répéter. Dans cette définition, le compteur peut donc devenir négatif mais de telles séquences de comptage seront écartées par les conditions introduites par la suite. Intuitivement, si le compteur devient négatif ceci signifie qu'une contrainte de répétition passée ne peut pas être satisfaite. Bien que nous ayons besoin de plus de compteurs, l'introduction de contraintes de répétitions passées n'introduit pas de réelles complications. Ceci peut être comparé à l'ajout d'opérateurs référant au passé dans LTL. En fait, le passé étant fini les informations peuvent être accumulées facilement.

Lemme 46 *Un modèle symbolique ρ pour une formule de $\mathrm{CLTL}^{\Diamond, \Diamond^{-1}}(\langle \mathbb{N}, =\rangle)$ est satisfaisabe ssi la séquence de comptage γ_ρ est telle que pour tout $i \in \{0, \dots |\rho| - 1\}$ on a $\gamma(i)(\langle X_p, X_f\rangle) \geq 0$ et les conditions suivantes sont satisfaites :*

(I) *Si ρ est un modèle symbolique fini alors la valeur finale de tous les compteurs de γ_ρ vaut 0.*

(II) *Si ρ est un modèle symbolique infini alors*

(C1) Il n'existe pas de chemin infini en avant dans G_ρ et de compteur $\langle X_p, X_f\rangle$ tels que chaque noeud du chemin a pour future obligation $\langle X_p, X_f\rangle$ et il existe une variable $y \in X_f$ telle qu'aucun noeud de la forme $\langle y, i\rangle$ n'est connecté à ce chemin par une arête en avant.

(C2) *La séquence γ_ρ est telle que chaque compteur $\langle X_p, X_f \rangle$ satisfait l'une des conditions suivantes :*
  (a) *il existe une position après laquelle la valeur du compteur $\langle X_p, X_f \rangle$ vaut toujours 0 et il n'existe plus de point d'incrémentation pour $\langle X_p, X_f \rangle$,*
  (b) *il existe une infinité de points de décrémentation pour $\langle X_p, X_f \rangle$ qui sont connectés par un chemin en avant à un noeud $\langle x, i \rangle$ tel que $\Diamond_\rho(\langle x, i \rangle) \subset X_f$ dans G_ρ,*
  (c) *pour chaque $x \in X_f$ il existe une infinité de points de décrémentation pour $\langle X_p, X_f \rangle$ qui sont connectés par un chemin en avant à un noeud de la forme $\langle x, i \rangle$ dans G_ρ.*

Preuve : La preuve de ce résultat est très semblable à celle du Lemme 40. Nous insistons surtout sur les changements à opérer dans l'algorithme.

Soit ρ un modèle symbolique satisfaisant les conditions de l'énoncé et γ_ρ la séquence de comptage correspondante. Pour chaque compteur d'obligation $\langle X_p, X_f \rangle \in \mathcal{P}^+(\{x_1, \ldots, x_k\}) \times \mathcal{P}^+(\{x_1, \ldots, x_k\})$ nous introduisons une file FIFO $\texttt{Repeat}_{\langle X_p, X_f \rangle}$ contenant des éléments de l'ensemble $\mathbb{N} \times X_f$ correspondant à la valeur à répéter et la variable de X_f dont nous attendons qu'elle prenne cette valeur. Le second argument est toujours utilisé pour les compteurs qui sont décrémentés infiniment souvent. Nous fixons dans ce but un ordre sur les variables tel que $x_1 < x_2 < \ldots < x_k$. Nous utilisons aussi des ensembles $\texttt{Forbid}_x$ pour chaque $x \in V$ afin de mémoriser les valeurs interdites pour la variable x. Ces éléments sont initialisés respectivement avec la file vide et l'ensemble vide. Nous modifions notre procédure d'étiquetage de la représentation graphique de ρ. Le principal changement est qu'un point de décrémentation ne peut plus être ignoré dans le sens où nous devons obligatoirement lui affecter une valeur en attente dans la file correspondante.

(I0) Pour la première valuation symbolique, il existe une valuation des sommets qui satisfait les relations d'égalités car l'ensemble de contraintes est maximalement cohérent. Nous affectons à chaque sommet une étiquette composée de la valeur qui lui est assignée par cette valuation et de l'ensemble vide.

(Ind) Supposons maintenant que la séquence est étiquetée jusqu'à la valuation symbolique $\rho(i)$. Nous définissons l'étiquetage de $\rho(i+1)$. Comme la séquence est cohérente pas à pas, nous devons juste définir les étiquettes des sommets de niveau $i+l+1$.

(Ind1) Nous nous occupons d'abord des points d'incrémentation de la forme $[\langle x, i \rangle]$. Soit $\langle a, b \rangle$ l'étiquette affectée à $\langle x, i \rangle$. Nous insérons à la fin de la queue $\texttt{Repeat}_{\langle X_p, X_f \rangle}$ telle que $X_p = \Diamond^{-1}_{\rho(i)}(\langle x, 0 \rangle) \cup [\langle x, 0 \rangle]_{\rho(i)}$ et $X_f = \Diamond_{\rho(i)}(\langle x, 0 \rangle)$ la paire $\langle a', b' \rangle$ vérifiant

– $a = a'$,
– si $b \neq \emptyset$ alors $b' = b$
  sinon b' est la variable la plus petite dans $\Diamond_{\rho(i)}(x, 0)$ (par rapport à l'ordre fixé).

Ensuite, nous ajoutons la valeur a à tous les ensembles de la forme $\texttt{Forbid}_y$ tels que $y \notin \Diamond_{\rho(i)}(x, 0)$.

(Ind2) Nous définissons ensuite la valeur des étiquettes associées aux sommets de la forme $\langle x, i+l+1 \rangle$ de la façon suivante :

(Ind2.1) Si le sommet $\langle x, i+l+1\rangle$ est connecté par une relation d'égalité à un autre sommet $\langle y, j\rangle$ tel que $i < j < i+l+1$ alors soit $\langle a, b\rangle$ l'étiquette de $\langle y, j\rangle$. Nous affectons à $\langle x, i+l+1\rangle$ la paire $\langle a', b'\rangle$ telle que

– $a = a'$,

– si $\Diamond(\langle y, j\rangle) = \Diamond(\langle x, i+l+1\rangle)$ alors $b = b'$
sinon $b = \emptyset$.

(Ind2.2) Si $\langle x, i+l+1\rangle$ n'est connecté à aucun autre sommet de niveau inférieur alors plusieurs cas se présentent.

(Ind2.2a) Si $\Diamond^{-1}_{\rho(i+1)}(\langle x, l\rangle) = \emptyset$ alors nous affectons l'étiquette $\langle a, \emptyset\rangle$ à tous les sommets de $[\langle x, l\rangle]_{\rho(i+1)}$ telle que a vérifie les contraintes imposées par les arcs et

$$a \notin \bigcup_{y\in[\langle x,l\rangle]_{\rho(i+1)}} \texttt{Forbid}_y \cup \bigcup_{\langle X_p,X_f\rangle\in\mathcal{P}^+(\{x_1,\ldots,x_k\})^2} \texttt{Repeat}_{\langle X_p,X_f\rangle}.$$

Sinon $[(x, l)]_{\rho(i+1)}$ est un point de décrémentation pour un compteur $\langle X_p, X_f\rangle$. Nous supposons alors que $\texttt{Repeat}_{\langle X_p,X_f\rangle}$ n'est pas vide. Nous montrons plus tard que si c'était le cas, le compteur $\langle X_p, X_f\rangle$ serait négatif. Il reste alors les cas suivants.

(Ind2.2b) Si la séquence est finie, ou si X est un compteur qui satisfait la condition (C2a), ou bien si l'on peut atteindre par un chemin en avant un sommet de la forme $\langle y, j\rangle$ à partir de $\langle x, i+l+1\rangle$ tel que $\Diamond(\langle y, j\rangle)$ est strictement inclus dans X alors nous enlevons le premier élément $\langle a, b\rangle$ de la file $\texttt{Repeat}_{\langle X_p,X_f\rangle}$ et nous affectons aux sommets de la classe d'équivalence $[\langle x, l\rangle]_{\rho(i+1)}$ l'étiquette $\langle a, \emptyset\rangle$.

(Ind2.2c) Sinon, nous enlevons le premier élément $\langle a, b\rangle$ de la file $\texttt{Repeat}_{\langle X_p,X_f\rangle}$ tel que $b \in [\langle x, l\rangle]_{\rho(i+1)}$ et nous affectons à tous les sommets de $[\langle x, l\rangle]_{\rho(i+1)}$ l'étiquette $\langle a', b'\rangle$ telle que

– $a = a'$

– b' est la plus petite variable supérieure à b qui n'est pas accessible par un chemin en avant à partir du sommet $\langle x, i+l+1\rangle$
si une telle variable n'existe pas alors $b = \emptyset$.

S'il n'existe pas d'élément de $\texttt{Repeat}_{\langle X_p,X_f\rangle}$ satisfaisant les bonnes conditions alors nous procédons comme dans le cas **(Ind2.2a)**.

La principale différence avec l'algorithme du Lemme 40 est que dans tous les cas où l'on traite un point de décrémentation, un élément de la file correspondante est enlevé. Comme pour chaque point d'incrémentation un élément est inséré et que dans la relation de comptage incrémentations et décrémentations sont obligatoires, nous pouvons en conclure que pour tout $0 \leq i \leq |\rho|$ et pour tout compteur $\langle X_p, X_f\rangle$, la taille de la file $\texttt{Repeat}_{\langle X_p,X_f\rangle}$ à l'étape i est égale à $\gamma(\langle X_p, X_f\rangle)(i)$. C'est pourquoi nous pouvons supposer dans les point **(Ind2.2.b)** et **(Ind2.2.c)** que la file et non-vide. En utilisant le même développement que dans la preuve du Lemme 40, nous pouvons montrer que cet étiquetage satisfait toutes les contraintes du modèle symbolique à la fois dans les cas finis et infinis.

Réciproquement, nous pouvons aussi utiliser la même méthode pour le Lemme 40 afin de montrer que tout modèle symbolique satisfaisable vérifie les conditions de l'énoncé. La seule différence est qu'il nous faut montrer que la valeur des compteurs n'est jamais négative. Si un compteur $\langle X_p, X_f \rangle$ devient négatif, c'est qu'à ce moment il existe plus de point de décrémentation que d'incrémentation pour ce compteur dans la séquence, d'après la définition de la relation de comptage. Ce qui signifie que toutes les obligations correspondant à $\langle X_p, X_f \rangle$ ont été satisfaites et qu'il n'existe pas de valeur qui est apparue dans les variables de X_p et doit se répéter dans le futur dans des états des variables de X_f. La séquence n'est donc pas satisfaisable, ce qui établit une contradiction. $\square$

En conséquence, nous pouvons facilement mettre à jour la construction de $\mathcal{A}_\phi$ par rapport aux nouveaux compteurs et à la nouvelle définition des séquences de comptage. L'automate obtenu est toujours un automate à compteurs avec test à zéro définitif dans le cas infini et un automate à compteur sans test à zéro dans le cas fini, ce qui nous permet d'étendre la preuve de décidabilité du Théorème 34.

Théorème 36 *Le problème de la satisfaisabilité pour* $\mathrm{CLTL}^{\Diamond,\Diamond^{-1}}(\langle \mathbb{N}, = \rangle)$ *dans les cas fini et infini est décidable.*

Comme corollaire, nous obtenons toujours la décidabilité du problème de la satisfaisabilité pour les extensions de $\mathrm{CLTL}^{\Diamond,\Diamond^{-1}}(\langle \mathbb{N}, = \rangle)$ avec des opérateurs temporels définissables dans MSO.

Corollaire 13 *Le problème de la satisfaisabilité pour la logique* $\mathrm{CLTL}^{\Diamond,\Diamond^{-1}}(\langle \mathbb{N}, = \rangle)$ *étendue avec des opérateurs définissables dans MSO est décidable dans les cas fini et infini.*

Notons enfin que le Lemme 44 s'adapte aussi facilement à l'extension $\mathrm{CLTL}_1^{\Diamond,\Diamond^{-1}}(\langle \mathbb{N}, = \rangle)$ ce qui permet d'établir la complexité pour le fragment à une variable.

Théorème 37 *Le problème de la satisfaisabilité pour* $\mathrm{CLTL}_1^{\Diamond,\Diamond^{-1}}(\langle \mathbb{N}, = \rangle)$ *dans les cas fini et infini est* PSPACE-*complet.*

Synthèse II

Nous avons démontré dans cette partie des résultats concernant une extension de la logique $\mathrm{CLTL}(\langle\mathbb{N},=\rangle)$ avec une forme restreinte de quantification du premier ordre permettant d'exprimer des contraintes de répétition d'une valeur. Pour cette logique nommée $\mathrm{CLTL}^{\Diamond}(\langle\mathbb{N},=\rangle)$, nous avons montré un résultat de décidabilité en réduisant le problème de la satisfaisabilité d'une formule vers un problème sur les réseaux de Petri. Cette réduction est plutôt surprenante puisqu'à priori la logique ne contient que des contraintes d'égalité et aucun mécanisme de comptage contrairement à CLTL(DL) (voir Chapitre 5). Ce résultat établit une nouvelle relation entre les extensions de logiques temporelles sur des domaines concrets manipulant des entiers et les machines à compteurs.

Comparé aux résultats concernant l'extension de LTL avec l'opérateur freeze, la logique que nous considérons ne contient aucune restriction syntaxique sur le nombre de variables, la longueur temporelle et les opérateurs logiques. Le choix qui a été fait ici est de réduire l'utilisation de l'opérateur freeze tout en gardant l'expressivité du langage logique, afin d'obtenir une preuve homogène de la décidabilité de la satisfaisabilité dans les cas fini et infini. Cependant, la complexité de ce problème reste à définir puisque la complexité pour les problèmes concernant les réseaux de Petri que nous utilisons n'est pas établie. Dans le fragment à une variable de la logique, la borne de PSPACE-complétude peut néanmoins être prouvée car le compteur que nous utilisons dans la réduction est borné. La complexité générale de ce problème est le principal problème laissé ouvert dans cette partie.

Les extensions possibles pour la logique $\mathrm{CLTL}^{\Diamond}(\langle\mathbb{N},=\rangle)$ concernent deux directions non orthogonales. D'une part, il est envisageable d'étendre la logique afin de considérer des contraintes plus riches. Dans les contraintes de répétition de $\mathrm{CLTL}^{\Diamond}(\langle\mathbb{N},=\rangle)$ nous n'avons aucune information sur la position à laquelle la valeur se répète. Cette contrainte peut être affaiblie en autorisant la vérification d'une autre contrainte à la position où la répétition est satisfaite. Par exemple $x = \Diamond y^{(z=\mathsf{X}z)}$ signifie que dans le futur il existe une position où la valeur de y est égale à la valeur courante de x et qu'à cette position la valeur de z est égale à la prochaine valeur de z. Nous pouvons aussi imposer un ordre sur les répétitions avec de telles contraintes. On peut envisager par exemple d'introduire des contraintes telles que

$$x = \Diamond y^{(y=\Diamond z^{(z=\Diamond x)})}$$

qui exprimerait que la valeur courante de x se répète en y puis en z et à nouveau en x. Notons qu'une telle extension permettrai de réduire la satisfaisabilité dans le cas fini au cas infini. Nous conjecturons que certaines méthodes définies dans cette partie peuvent être réutilisée pour montrer la décidabilité du problème de la satisfaisabilité au moins pour certains fragments de cette logique.

Enfin, il est aussi possible d'enrichir les contraintes en considérant des domaines plus riches autorisant d'autres relations que l'égalité, par exemple $\langle\mathbb{N},<,=\rangle$. Pour information, la logique sur les mots de données $\mathrm{FO}(\preceq,=,+1)$ avec une ordre sur les données est indécidables [BMS$^+$06] mais le fait qu'on n'ait aucune information sur la position où la contrainte $x < \Diamond y$ est satisfaite fait que la preuve d'indécidabilité ne peut être adaptée de manière triviale.

Quatrième partie

CTL* sur des domaines concrets

Chapitre 7

CCTL*($\mathcal{D}$) avec abstraction vérifiable localement

Sommaire

Nous avons jusqu'à présent considéré uniquement des logiques temporelles linéaires sur des domaines concrets. Nous nous intéressons maintenant aux extensions de logiques temporelles arborescentes avec des domaines concrets. Le but est de généraliser dans la mesure du possible une partie des résultats obtenus dans le cas linéaire. Nous commençons par introduire dans ce chapitre une extension des formalismes utilisés pour l'approche à base d'automates et nous définissons un résultat général de décidabilité pour une sous-famille de logiques temporelles arborescentes sur des domaines concrets.

7.1 Logiques arborescentes sur domaines concrets

Comparé aux nombreuses extensions de LTL sur des domaines concrets, il existe a notre connaissance moins de travaux en relation avec les logiques temporelles arborescentes étendues avec des domaines concrets dans la littérature. La plupart de ces travaux concernent surtout des problèmes de model-checking de propriétés exprimées par des contraintes de Presburger sur des systèmes à compteurs. Ce problème en général est indécidable puisque la simple accessibilité d'un état de contrôle dans une machine de Minsky à deux compteurs est indécidable [Min67]. Pour les logiques temporelles arborescentes étendues avec des domaines concrets, on sait aussi d'après un résultat de [Čer94] que problème model-checking de CCTL(Z) sur les Z-automates est indécidable. Rappelons que le domaine concret Z est égal à $\langle \mathbb{Z}, <, = \rangle$. Donc ce résultat implique que la restriction à des ensembles de contraintes qualitatives sur les entiers n'est pas suffisante pour retrouver la décidabilité dans le cas arborescent (contrairement au cas linéaire).

Une solution possible pour définir des fragments décidables de ce problème consiste à restreindre le modèle considéré. Nous rappelons que de nombreuses restrictions des systèmes à compteurs ont été introduites dans la littérature telle que les machines à compteurs dont les valeurs ont un nombre borné d'alternations entre croissance et décroissance [Iba78], les systèmes à compteur plats [CJ98] ou les systèmes à compteurs plats avec des fonctions de mise à jour affines formant un monoïde fini [Boi98, FL02, BFLP03] pour citer quelques exemples. Concernant le model-checking de propriétés exprimées dans des logiques temporelles arborescentes avec des contraintes sur les compteurs, il est défini dans [DFGvD06] une classe d'automates à compteurs pour laquelle le problème du model-checking de CTL* étendu avec des contraintes de Presburger est décidable. De même, il est montré dans [BDR03] que le model-checking des automates temporisés discrets avec une version temporisée et paramétrée de CTL est décidable. Nous avons aussi parlé précédemment des travaux de [FWW97, EKS03] qui établissent des résultats de décidabilité et complexité pour le problème du model-checking des automates à piles avec un extension de CTL* comportant des contraintes de régularité sur le contenu de pile. Rappelons que les automates à pile généralisent la classe des automates à un compteur. Ces travaux sont aussi un exemple pour lequel l'ensemble d'interprétation des variables n'est pas les entiers.

Des fragments de ces problèmes peuvent être exprimés en termes de problèmes de model-checking des logiques temporelles arborescentes sur des domaines concrets, mais ils introduisent des restrictions du problème général soit sur le modèle, soit sur les ressources syntaxiques des formules. Nous voulons considérer dans un premier temps un cadre aussi général que possible pour étendre la méthode à base d'automate que nous utilisons dans le cas linéaire. De plus, nous désirons définir une approche permettant de traiter à la fois les problèmes de satisfaisabilité et de model-checking. Dans ce chapitre, nous définissons une nouvelle représentation symbolique pour les modèles d'une formule appartenant à une logique arborescente de la forme CCTL$^\star(\mathcal{D})$. Cette abstraction utilise et généralise les différentes représentations que nous avons introduites dans le cadre linéaire jusqu'à présent. Nous utilisons ensuite cette représentation symbolique des modèles pour résoudre les problèmes de la satisfaisabilité et du model-checking pour une sous-famille des logiques de la forme CCTL$^\star(\mathcal{D})$ dans lesquelles le domaine concret vérifie une propriété générale qui permet de prouver l'existence d'un modèle correspondant à une abstraction donnée en effectuant uniquement des tests locaux sur le modèle symbolique, ce qui facilite la procédure de décision. Bien que cette propriété puisse paraître à première vue restrictive, de nombreux travaux introduisent des extensions de LTL sur des domaines concrets la vérifiant (voir par exemple [BC02, DD07, Dem04]). D'autres logiques telle que l'extension de LTL avec la logique de séparation de [BDL07] utilisent le même genre d'abstraction et entrent donc dans cette classe de langages logiques. Nos résultats généralisent donc les différents résultats de décidabilité démontrés dans ces travaux et raffinent certaines bornes de complexité établies dans le cadre des logiques temporelles linéaires, bien souvent des bornes de PSPACE-complétude. Une comparaison plus détaillée avec ces travaux sera faite au cours de ce chapitre.

Nous privilégions toujours une approche à base d'automates mais nous manipulons maintenant des structures arborescentes. Nous adaptons différentes constructions d'automates d'arbres alternants introduites dans le cadre des logiques temporelles arborescentes propositionnelles. Contrairement à [KVW00], nous ne définissons pas de nouvelle classe d'automates, mais nos constructions nécessitent plusieurs adaptations non triviales et combinent différents résultats liés aux automates d'arbres alternants. En effet, la construction des automates doit tenir compte de la spécificité de la logique $\mathrm{CCTL}^\star(\mathcal{D})$ qui autorise entre autre la comparaison de variables à différents états du modèles. Cette spécificité en particulier est prise en compte à la fois dans la représentation symbolique des modèles et la construction des relations de transition dans les automates. Les résultats qui suivent généralisent fidèlement les résultats connus sur les fragment linéaires $\mathrm{CLTL}(\mathcal{D})$ des logiques étudiées puisque nous utilisons le même type d'approche et que la borne de complexité optimale peut être retrouvée pour le cas linéaire en utilisant cette approche.

Notons enfin sur le plan technique que le traitement du problème du model-checking n'est pas une conséquence directe du résultat sur le problème de la satisfaisabilité et nécessite que nous définissions une construction intermédiaire pour abstraire les exécutions d'un $\mathcal{D}$-automate. En effet, si un domaine $\mathcal{D}$ possède un ensemble d'interprétation ayant un nombre infini d'éléments, le graphe des configurations d'un $\mathcal{D}$-automate peut avoir un degré infini et nous manipulons des automates d'arbres à branchement fini. Nous rappelons que cet argument prouve que le problème du model-checking pour les logiques arborescentes sur des domaines concrets ne se réduit pas toujours au problème de la satisfaisabilité puisque le langage logique ne nous permet pas d'exprimer qu'un état possède un nombre infini de successeurs.

7.2 Représentation symbolique des modèles arborescents

Nous présentons ici une représentation symbolique des modèles pour une formule appartenant à une logique arborescente de la forme $\mathrm{CCTL}^\star(\mathcal{D})$. Cette représentation généralise les représentations symboliques qui ont été définies dans le cadre des logiques temporelles linéaires.

7.2.1 Rappels sur l'abstraction dans le cas linéaire

Nous rappelons d'abord quelques propriétés des abstractions des modèles d'une formule de $\mathrm{CLTL}(\mathcal{D})$ que nous utilisons par la suite pour l'extension de l'abstraction au cas arborescent. Dans le cas linéaire, une valuation symbolique par rapport à un ensemble X de contraintes atomiques de $\mathrm{CLTL}(\mathcal{D})$, pour un domaine concret quelconque $\mathcal{D} = \langle D, \mathcal{R} \rangle$, peut être définie comme un ensemble cohérent de contraintes de $\mathcal{D}$ par rapport à l'ensemble Term des termes utilisés dans X. Pour toute valuation symbolique sv, nous notons $v \models sv$ ssi v satisfait toutes les contraintes de sv. L'ensemble $\mathrm{SV}(X)$ des valuations symboliques par rapport à X définit des classes d'équivalences pour les valuations de la forme $\mathrm{Term} \to D$. Pour qu'une abstraction sous la forme de valuations symboliques satisfasse cette condition, l'ensemble $\mathrm{SV}(X)$ doit donc vérifier les propriétés suivantes :

(SV1) Pour toute valuation $v : \mathrm{Term} \to D$ il existe une unique valuation symbolique dans $\mathrm{SV}(X)$ notée $sv(v)$ telle que $v \models sv(v)$.

(SV2) Pour toute paire de valuations $v, v' : \mathrm{Term} \to D$ telle que $sv(v) = sv(v') \in \mathrm{SV}(X)$ et toute contrainte atomique de X, on a $v \models \alpha$ ssi $v' \models \alpha$.

Ces conditions imposent que $\{\{v \models sv\} \mid v : \mathrm{Term} \rightarrow D \text{ et } sv \in \mathrm{SV}(X)\}$ soit une partition finie de l'ensemble des valuations de la forme Term $\rightarrow D$. Notons que toutes les abstractions introduites dans les chapitres précédents vérifient cette définition générale.

Nous pouvons toujours supposer sans perte de généralité que le nombre de relations du domaine concret que l'on considère est fini car le nombre de relations apparaissant dans X est fini. De même, l'ensemble des termes à considérer est aussi fini. Alors, la façon la plus facile de définir une abstraction vérifiant les propriétés ci-dessus consiste à prendre les ensembles maximalement cohérents de contraintes sur les termes de X par rapport aux relations utilisées dans X. Ainsi on est sûr que les conditions **(SV1)** et **(SV2)** sont vérifiées car l'ensemble maximalement cohérents de contraintes satisfaites par une valuation donnée v est unique et toute contrainte de X satisfaite par v doit appartenir à $sv(v)$ sinon l'ensemble n'est pas maximal par rapport aux relations et aux termes de X.

Cependant, cette représentation n'est pas toujours la plus astucieuse en terme de complexité (voir les Chapitres 4 et 5). Par exemple, lorsque l'on considère un ensemble de contraintes de périodicité utilisant les relations de congruence $\equiv_2$ et $\equiv_3$, il est plus judicieux et concis de considérer un ensemble de contraintes maximalement cohérent par rapport à la relation $\equiv_6$. En effet comme 2 et 3 divisent 6 il est possible de déterminer la satisfaction de contraintes utilisant les relations $\equiv_2$ et $\equiv_3$ en connaissant la valeurs des différentes variables modulo 6.

7.2.2 Modèles symboliques arborescents

Soit ϕ une formule de $\mathrm{CCTL}^\star(\mathcal{D})$ sous forme positive. Nous notons $V \subseteq \mathrm{VAR}$ l'ensemble des variables utilisées dans ϕ, $|\phi|_{\mathsf{X}} = l$ et $E_\sharp(\phi) + 1 = d$. Pour évaluer la satisfaction d'une contrainte atomique de ϕ, il est nécessaire d'anticiper les configurations d'un modèle le long de tous les chemins finis de taille l. De plus, dans l'optique de résoudre le problème de la satisfaisabilité, nous pouvons restreindre la recherche de modèles pour ϕ à la classe des d-arbres d'après le Corollaire 3. C'est pourquoi l'élément de base de notre représentation symbolique des modèles est un arbre fini donnant les informations sur toutes les l prochaines configurations possibles dans un d-arbre.

Supposons que l'on peut construire une abstraction des valuations satisfaisant les conditions **(SV1)** et **(SV2)** par rapport à tout ensemble de contraintes atomiques $\mathrm{CLTL}(\mathcal{D})$. Nous considérons l'ensemble de valuations symboliques noté $\mathrm{SV}(\phi)$ construit par rapport à l'ensemble des contraintes atomiques de ϕ. Nous étendons cette abstraction au cas arborescent de la façon suivante. Une *fenêtre symbolique fr* est une fonction de la forme $\{1,\ldots,d\}^l \rightarrow \mathrm{SV}(\phi)$ qui associe à chaque chemin fini représenté par un mot de $\{1,\ldots,d\}^l$ une valuation symbolique. Pour être cohérente par rapport à la définition d'un arbre, cette fonction doit entre autre coïncider sur les préfixes communs aux chemins, c'est-à-dire :

Pour tout $w_1, w_2 \in \{1,\ldots,d\}^l$ tel que w est le plus grand préfixe commun à w_1 et w_2, on a

$$\alpha \in fr(w_1) \text{ ssi } \alpha \in fr(w_2)$$

pour toute contrainte atomique α de la forme $\mathrm{R}(\mathsf{X}^{i_1}x_{j_1},\ldots,\mathsf{X}^{i_k}x_{j_k})$ où $0 \leq i_1,\ldots,i_k \leq |w|$.

Nous notons $\texttt{Frame}(\phi)$ l'ensemble des fenêtres symboliques construites par rapport à ϕ. Notons que dans le cas général, le fait qu'à chaque chemin de longueur l soit affecté un ensemble de contraintes cohérent ne suffit pas pour affirmer que l'ensemble total de contraintes qui constitue une fenêtre symbolique est satisfaisable. Cependant, cette condition est suffisante dans le cas des domaines que nous considérons par la suite.

Étant donné $a \in \{1, \ldots, d\}$, une paire de fenêtres symboliques $\langle fr, fr' \rangle$ de $\texttt{Frame}(\phi)$ est *a-cohérente sur un pas* ssi pour tout $w \in \{1, \ldots, d\}^{l-1}$ on a

$$\begin{aligned} &\mathrm{R}(\mathsf{X}^{i_1} x_{j_1}, \ldots, \mathsf{X}^{i_k} x_{j_k}) \in fr(a \cdot w) \text{ et } 0 < i_m \leq l \text{ pour tout } 1 \leq m \leq k \\ &\text{ssi } \mathrm{R}(\mathsf{X}^{i_1 - 1} x_{j_1}, \ldots, \mathsf{X}^{i_k - 1} x_{j_k}) \in fr(w \cdot b) \text{ pour tout } b \in \{1, \ldots, d\}. \end{aligned}$$

Un d-arbre $T_{\mathrm{symb}} = \langle N, \{n_0\}, \rightarrow, \Gamma_{\mathrm{symb}} \rangle$ tel que $\Gamma_{\mathrm{symb}} : N \rightarrow \texttt{Frame}(\phi)$ associe une fenêtre symbolique à chaque sommet, est *cohérent pas à pas* ssi pour tout sommet $n \in N$ et $a \in \{1, \ldots, d\}$ le a-successeur n' de n est tel que la paire $\langle \Gamma_{\mathrm{symb}}(n), \Gamma_{\mathrm{symb}}(n') \rangle$ est a-cohérente sur un pas. Nous appelons un tel arbre T_{symb} un *modèle symbolique arborescent* par rapport à la formule ϕ.

Considérant cette abstraction, nous pouvons définir une relation de satisfaction symbolique et une sémantique proche de celle de CTL*. Nous rappelons que dans le cas linéaire, étant données une valuation symbolique $sv \in \mathrm{SV}(\phi)$ et une contrainte atomique α de ϕ nous notons $sv \models_{\mathrm{symb}} \alpha$ ssi pour toute valuation $v \in \mathrm{SV}(\phi)$ telle que $v \models sv$ on a $v \models \alpha$. Notons que cette définition n'a de sens que si l'ensemble des valuations symboliques vérifie les conditions **(SV1)** et **(SV2)**. L'extension de cette relation au cas arborescent est directe en considérant la relation de satisfaction de CCTL$^\star(\mathcal{D})$. La seule différence est au niveau des contraintes atomiques de $\mathcal{D}$. Soit $T_{\mathrm{symb}} = \langle N, \{n_0\}, \rightarrow, \Gamma_{\mathrm{symb}} \rangle$ un modèle symbolique arborescent. Pour tout chemin $\pi = \pi(0) \xrightarrow{a_0} \pi(1) \xrightarrow{a_1} \cdots$ dans un graphe, nous notons w_π la séquence $a_0 \cdot a_1 \cdots$.

Pour tout chemin π dans T_{symb}, on a $\langle T_{\mathrm{symb}}, \pi \rangle \models_{\mathrm{symb}} \alpha$ ssi $\Gamma_{\mathrm{symb}}(\pi(0))(w_\pi^l) \models_{\mathrm{symb}} \alpha$ où w_π^l est le préfixe de longueur l de w_π.

Notons que les contraintes atomiques peuvent être interprétées directement par rapport à un sommet du modèle symbolique arborescent car celui-ci contient toute l'information nécessaire. Cependant, ceci nous oblige à définir une relation de satisfaction plus complexe paramétrée par des chemins. En effet, la satisfaction d'une contrainte atomique dans la portée d'un quantificateur existentiel est conditionnée par un chemin particulier qu'il nous faut mémoriser pour déterminer la satisfaction à partir du sommet courant. Contrairement au cas linéaire, les contraintes atomiques ne peuvent donc pas être considérées exactement comme des variables propositionnelles dans l'approche symbolique.

Nous définissons aussi une relation de satisfaction entre les modèles de CCTL$^\star(\mathcal{D})$ et les modèles symboliques arborescents. Étant donné un graphe $G = \langle N, \{n_0\}, \rightarrow, \Gamma \rangle$ de degré d dont les successeurs de chaque sommet sont ordonnés et un chemin fini π dans ce graphe, nous définissons la valuation v_π telle que :

$$v_\pi(\mathsf{X}^i x) = \Gamma(\pi(i))(x) \text{ pour tout } x \in \mathrm{VAR} \text{ et } i \in \{0, \ldots, |\pi|\}.$$

Pour toute fenêtre symbolique $fr \in \texttt{Frame}(\phi)$, nous notons $\langle G, n \rangle \models fr$ ssi pour tout $w \in \{0, \ldots, d\}^l$ le chemin fini π décrit par w et ayant pour origine n existe dans G et la valuation v_π satisfait $fr(w)$. Le graphe G satisfait le modèle symbolique arborescent

$T_{\text{symb}} = \langle N', \{n'_0\}, \rightarrow', \Gamma_{\text{symb}} \rangle$ ssi pour tout sommet $n \in N$ atteint dans G en suivant un chemin décrit par un mot w depuis le sommet initial de G, le sommet $n' \in N'$ atteint dans T_{symb} en suivant le chemin décrit par le même mot w depuis la racine vérifie $\langle G, n \rangle \models \Gamma_{\text{symb}}(n')$.

Cette relation de satisfaction implique plusieurs relations entre les arcs et les sommets des graphes G et T_{symb}. D'abord, d'après la relation de satisfaction d'une fenêtre symbolique, G doit être un d-graphe. Sinon, certains chemins définis dans une fenêtre symbolique ne sont pas définis à partir de certains noeuds de G. De plus, tous sommets $n \in N$ et $n' \in N'$ atteints respectivement dans G et T_{symb} en se déplaçant le long de chemins décrits par le même mot à partir du sommet initial de chaque graphe sont bisimilaires puisque la relation de satisfaction $\langle G, n \rangle \models \Gamma_{\text{symb}}(n')$ est toujours vérifiée pour un tel couple de sommets. Notons que si G est un graphe quelconque, alors plusieurs sommets de T_{symb} peuvent être bisimilaires au même sommet de G. Néanmoins, lorsque G est un d-arbre alors il existe une bijection $f : N \rightarrow N'$ entre les deux ensembles de sommets telle que l'image du sommet initial de G est la racine de T_{symb} et pour tout $n \in G$ on a

- $\langle G, n \rangle \models \Gamma_{\text{symb}}(f(n))$,
- n' est le a-successeur de n dans G ssi $f(n')$ est le a-successeur de $f(n)$ dans T_{symb}.

Le résultat suivant peut être montré en considérant les définitions de cette section. Il généralise les approches que nous avons définies dans le cas linéaire.

Lemme 47 *Une formule ϕ de* CCTL$^\star(\mathcal{D})$ *est satisfaisable ssi il existe un modèle symbolique arborescent T_{symb} de ϕ et un d-arbre T tel que $T_{\text{symb}} \models_{\text{symb}} \phi$ et $T \models T_{\text{symb}}$.*

Preuve : Soit ϕ une formule de CCTL$^\star(\mathcal{D})$ et supposons que ϕ soit satisfaisable. D'après le Corollaire 3, il existe un d-arbre $T = \langle N, \{n_0\}, \rightarrow, \Gamma \rangle$ tel que $\langle T, n_0 \rangle \models \phi$. Nous définissons un modèle symbolique $T_{\text{symb}} = \langle N', \{n'_0\}, \rightarrow', \Gamma_{\text{symb}} \rangle$ à partir de T qui satisfait les conditions de l'énoncé. Nous notons $N = \{n_0, n_1, \ldots\}$ l'ensemble des sommets de T et $N' = \{n'_0, n'_1, \ldots\}$ l'ensemble des sommets de T_{symb}. La construction de T_{symb} est définie par induction sur la distance par rapport à la racine. Pour tout sommet n et toute séquence $w \in \{1, \ldots, d\}^l$, la paire $\langle n, w \rangle$ est vue comme le chemin fini décrit par w ayant pour origine le sommet n.

- La racine de T_{symb} est le sommet n'_0 et Γ_{symb} vérifie $\Gamma_{\text{symb}}(n'_0)(w) = fr(v_{\langle n_0, w \rangle})$ pour tout $w \in \{1, \ldots, d\}^l$.
- Pour tout sommet n'_i de T_{symb} l'ensemble des successeurs de n'_i est défini par la règle suivante : pour tout a-successeur n_j de n_i dans T nous ajoutons un a-successeur n'_j à n'_i et la fonction Γ_{symb} est mise à jour de façon à ce que $\Gamma_{\text{symb}}(n'_j)(w) = fr(v_{\langle n_j, w \rangle})$ pour tout $w \in \{1, \ldots, d\}^l$.

Cette construction respecte la cohérence pas à pas car la fenêtre symbolique affectée à chaque sommet est définie par rapport à un sous-arbre ayant une partie commune avec le sous-arbre utilisé pour construire la fenêtre symbolique affectée à son père. De plus, on définit explicitement une bijection f entre N et N' telle que pour tout $n_i \in N$ on a $f(n_i) = n'_i$ et d'après la définition de Γ_{symb} cette bijection vérifie $\langle T, n \rangle \models \Gamma_{\text{symb}}(f(n))(w)$

pour tout $n \in N$. De plus la fonction f préserve les chemins puisque $f(n_0) = n'_0$, et pour tout $n, n' \in N$ on a $n \xrightarrow{a} n'$ dans T ssi $f(n) \xrightarrow{a}' f(n')$ dans T_{symb}. Ceci nous permet de déduire qu'on a bien $T \models T_{\text{symb}}$ car la relation de satisfaction est toujours respectée lorsque l'on suit le même chemin dans les deux arbres.

Nous montrons maintenant que $T_{\text{symb}} \models_{\text{symb}} \phi$. Nous procédons par induction sur la structure de ϕ. Considérons une contrainte atomique α de ϕ. D'après la propriété **(SV2)** des valuations symboliques, pour tout sommet n_i de T, $w \in \{1, \ldots, d\}^l$ et toute valuation v telle que $sv(v) = sv(v_{\langle n_i, w\rangle})$ on a $v \models \alpha$ ssi $v_{\langle n_i, w\rangle} \models \alpha$. Ceci implique que si $v_{\langle n_i, w\rangle} \models \alpha$ alors $\Gamma_{\text{symb}}(n'_i)(w) \models_{\text{symb}} \alpha$ par construction de Γ_{symb}. Donc pour tout chemin $\pi = n_{i_0} \cdot n_{i_1} \cdots$ dans T, si $\langle T, \pi\rangle \models \alpha$ alors $\langle T_{\text{symb}}, \pi'\rangle \models_{\text{symb}} \alpha$ où $\pi' = n'_{i_0} \cdot n'_{i_1} \cdots$ puisque $n_i \xrightarrow{a} n_j$ implique $n'_i \xrightarrow{a} n'_j$ car une propriété de f est de préserver les transitions. Les autres cas sont triviaux en utilisant les propriétés de la bijection entre N et N' puisque la relation de satisfaction de CCTL$^\star(\mathcal{D})$ est identique à la relation symbolique pour ces cas.

Réciproquement, supposons que $T_{\text{symb}} \models_{\text{symb}} \phi$ et $T \models T_{\text{symb}}$. Par conséquent, les sommets de T_{symb} et T reliés par une bijection f telle que pour tout n de T on a $\langle T, n\rangle \models \Gamma_{\text{symb}}(f(n))$. Considérons un chemin π et w le préfixe de longueur $|\phi|_{\mathsf{X}}$ de w_π. Si on a $\langle T_{\text{symb}}, \pi\rangle \models_{\text{symb}} \alpha$ pour une contrainte atomique α de ϕ alors par définition on a $\Gamma_{\text{symb}}(\pi(0))(w) \models_{\text{symb}} \alpha$. Or, comme $T \models T_{\text{symb}}$ on a en particulier $\langle T, n\rangle \models \Gamma_{\text{symb}}(\pi(0))$ où n est l'image de $\pi(0)$ par l'inverse de la bijection entre les sommets de T et T_{symb}. Par conséquent, nous avons $v_{\langle n, w\rangle} \models \Gamma_{\text{symb}}(\pi(0))(w)$ ce qui implique d'après la définition de la relation de satisfaction symbolique pour les valuations symboliques $v_{\langle n, w\rangle} \models \alpha$. Donc, nous avons $\langle T, \langle n, w\rangle\rangle \models \alpha$. Nous pouvons alors facilement conclure pour les autres cas en utilisant les propriétés de la bijection entre les sommets de T et ceux de T_{symb} et la correspondance entre la relation de satisfaction de CCTL$^\star(\mathcal{D})$ et la relation de satisfaction symbolique. □

7.3 Un résultat de décidabilité pour la satisfaisabilité

Nous définissons maintenant la construction d'un automate qui reconnaît les modèles d'une formule de CCTL$^\star(\mathcal{D})$ lorsque l'on peut calculer une abstraction des modèles vérifiant une propriété particulière.

7.3.1 Abstraction vérifiable localement

Nous disons qu'une abstraction symbolique des valuations vérifiant les propriétés **(SV1)** et **(SV2)** est *vérifiable localement* si pour toute valuation symbolique sv, toute valuation $v : \text{Term}' \to D$ qui satisfait le sous-ensemble de contraintes de sv impliquant uniquement les termes de Term' peut être étendue en une valuation satisfaisant la totalité des contraintes de sv. Un telle abstraction vérifie la propriété suivante.

Lemme 48 *Pour toute formule ϕ de* CCTL$^\star(\mathcal{D})$*, tout modèle symbolique arborescent qui est construit à l'aide d'une abstraction vérifiable localement par rapport à ϕ est satisfaisable.*

Preuve : Soit $T_{\text{symb}} = \langle N', \{n'_0\}, \rightarrow', \Gamma_{\text{symb}}\rangle$ un modèle symbolique arborescent tel que $N' = \{n'_0, n'_1, \ldots\}$. Nous considérons l'arbre $T = \langle N, \{n_0\}, \rightarrow, \Gamma\rangle$ tel que $N = \{n_0, n_1, \ldots\}$ et pour tout $i \in \mathbb{N}$ on a $n_i \rightarrow n_j$ ssi $n'_i \rightarrow' n'_j$. Nous montrons par induction sur la distance par rapport à la racine comment construire une valuation des sommets de T de manière à ce que $T \models T_{\text{symb}}$. Comme toute fenêtre symbolique est satisfaisable, $\Gamma_{\text{symb}}(n'_0)$ en particulier est satisfaisable. Il existe donc un étiquetage qui associe une valuation à tous les sommets de T dont la distance par rapport à la racine est inférieure ou égale à $|\phi|_{\mathsf{X}}$ satisfaisant les contraintes de cette fenêtre symbolique.

Supposons qu'un étiquetage partiel des sommets de T satisfait toutes les fenêtres symboliques entre la racine et un sommet n. Comme un modèle symbolique arborescent est cohérent pas à pas, si l'affectation partielle des sommets satisfait $\Gamma_{\text{symb}}(n)$ alors elle satisfait aussi un sous-ensemble des contraintes de $\Gamma_{\text{symb}}(n')$ pour tout successeur n' de n. Comme l'abstraction est vérifiable localement, nous pouvons compléter cette solution pour tous les successeurs de n. $\square$

Ce résultat implique que l'on peut vérifier qu'un arbre T_{symb} est un modèle symbolique satisfaisable en effectuant uniquement des tests locaux. La seule condition à tester est en effet la cohérence pas à pas du modèle symbolique. Cette condition porte sur un nombre fini de sommets contrairement à la condition requise par exemple sur les modèles symboliques de CLTL(IPC*) qui implique l'ensemble des positions du modèle (voir le Lemme 15).

Il existe une famille de logiques temporelles sur des domaines concrets particuliers pour lesquelles toute abstraction vérifiant **(SV1)** et **(SV2)** est vérifiable localement. Un domaine concret satisfait la *propriété de complétion* ssi pour tout ensemble de contraintes satisfaisable de ce domaine, toute valuation qui satisfait le sous-ensemble des contraintes mettant en cause les variables du domaine de définition de v peut être étendue en valuation satisfaisant l'ensemble total. Formellement, $\mathcal{D} = \langle D, R\rangle$ satisfait la propriété de complétion ssi pour tout ensemble X de contraintes de $\mathcal{D}$ et toute valuation $v' : V' \rightarrow D$ telle que :

- $V' \subseteq \text{VAR}$ et
- pour toute contrainte de la forme $\text{R}(x_{j_1}, \ldots, x_{j_k}) \in X$ telle que $x_{j_1}, \ldots, x_{j_k} \in V'$ on a $v' \models \text{R}(x_{j_1}, \ldots, x_{j_k})$,

il existe une valuation $v : \text{VAR} \rightarrow D$ telle que pour tout $x \in V'$ on a $v(x) = v'(x)$ et $v \models X$. Pour de tels domaines, il est évident qu'une abstraction correcte des valuations est vérifiable localement puisqu'une valuation symbolique est par définition un ensemble cohérent de contraintes.

Dans [DD07] il est établi que tout domaine concret de la forme $\langle D, <, =\rangle$ dense et ouvert vérifie la propriété de complétion. De nombreux domaines concrets satisfont donc cette propriété qui est parfois introduite sous d'autres noms : cohérence globale [BC02], ω-admissibilité [LM05]... En intelligence artificielle par exemple, le domaine RCC8 (Region Connection Calculus [CBGG97]) exprimant des relations topologiques sur le plan des réels $\mathbb{R}^2$ ou les relations de Allen sur les axes rationnels utilisées pour représenter des connaissances sur des intervalles temporels sont des exemples parmi de nombreux autres dans ce domaine d'applications (une liste plus exhaustive est donnée dans [BC02, Sect.2]). Il est démontré dans [LM05] que l'on peut combiner des domaines concrets vérifiant la propriété de complétion avec des concepts généraux d'inclusion permettant d'exprimer des connaissances acquises pour définir des extensions de logiques de description décidables, alors que ces extensions sont indécidables pour d'autres domaines concrets. Le résultat que nous montrons dans la suite est le pendant de ce résultat pour la logique CTL*. Les exemples qui nous intéressent plus particulièrement concernent les domaines utilisés pour étendre la logique temporelle linéaire LTL qui satisfont la propriété de complétion, par exemple $\langle \mathbb{Q}, <, = \rangle$, $\langle \mathbb{R}, <, = \rangle$ [DD07] ou le système de raisonnement spatial qualitatif de [BC02].

Beaucoup de domaines manipulant des entiers ne sont pas vérifiables localement. Ceci est une conséquence du domaine d'interprétation discret qui complique l'extension des valuations vérifiant une partie d'un ensemble de contraintes. Pour prendre un exemple simple, si on a $x < y < z$ et que l'on fixe la valeur de x et z dans $\mathbb{Z}$ alors il n'existe pas toujours de valeur pour y dans $\mathbb{Z}$. Cependant, il existe une abstraction vérifiable localement pour les logiques temporelles étendues avec le domaine IPC^{++} [Dem04] capturé par IPC*. Nous rappelons que ce domaine est la restriction de IPC* aux contraintes de la forme

$$\begin{aligned} \theta ::= \ & x \equiv_k [c_1, c_2] \mid x \equiv_k y + [c_1, c_2] \mid x = y \mid x < d \mid x = d \mid \\ & \theta \wedge \theta \mid \neg\theta \mid \exists x \ \theta \end{aligned}$$

La propriété de complétion n'est pas vérifiée pour des ensembles de contraintes quelconques. Par exemple, l'ensemble $X = \{x \equiv_2 y + 1, x \equiv_2 z + 1, z \equiv_3 y\}$ est satisfait par la valuation $v = \{x \leftarrow 1, y \leftarrow 0, z \leftarrow 0\}$ mais la valuation $v' = \{y \leftarrow 3, z \leftarrow 6\}$ ne peut être complétée bien qu'elle satisfasse le sous-ensemble $\{z \equiv_3 y\}$. Cependant ce genre de contre-exemple utilise le fait que l'ensemble n'est pas maximalement cohérent et n'est plus valable lorsqu'on considère un ensemble de valuations symboliques dont les éléments sont maximalement cohérents. Dans notre exemple, la contrainte $y \equiv_2 z$ devrait alors appartenir à X puisqu'elle peut être déduite de $x \equiv_2 y + 1$ et $x \equiv_2 z + 1$ ce qui empêche la valuation v' de satisfaire le sous-ensemble de X impliquant seulement y et z.

Enfin, il existe des extensions de logiques temporelles avec des langages de contraintes pour lesquelles on peut définir une abstraction vérifiable localement, bien que la logique n'utilise pas de domaine concret. C'est le cas des logiques considérés dans [BDL07] qui étendent LTL avec des fragments de la logique de séparation. Les abstractions utilisées dans ces travaux sont similaires à notre définition générale et sont vérifiables localement (voir [BDL07, Lemme 4]).

Les problèmes de la satisfaisabilité et du model-checking pour toutes les extensions de LTL citées ci-dessus sont PSPACE-complets. Pour les domaines concrets $\mathcal{D}$ satisfaisant la propriété de complétion, un théorème général de [DD07] établit que les problèmes de satisfaisabilité et model-checking pour les logiques de la forme CLTL($\mathcal{D}$) sont PSPACE-complets. Les résultats qui suivent nous permettent de généraliser tous ces résultats aux extensions arborescentes. Nous établissons la borne optimale de 2EXPTIME-complétude des extensions de CTL* sur des domaines concrets pour lesquelles il existe une abstraction vérifiable localement qui est calculable.

7.3.2 Construction de l'automate pour CCTL*($\mathcal{D}$)

Considérons une logique de la forme CCTL*($\mathcal{D}$) telle qu'il existe un algorithme permettant de calculer une abstraction vérifiable localement pour toute formule ϕ de CCTL*($\mathcal{D}$).

Les résultats des Lemmes 48 et 47 suggèrent l'approche suivante : pour toute formule ϕ de CCTL$^\star(\mathcal{D})$ nous construisons un automate vérifiant la satisfaction symbolique de ϕ et la cohérence pas à pas du modèle symbolique. Cet automate que nous notons $\mathcal{A}_\phi$ est donc défini comme l'intersection de deux automates d'arbres alternants $\mathcal{A}_{\text{pap}}$ et $\mathcal{A}_{\text{symb}}$ tels que le langage reconnu par $\mathcal{A}_{\text{pap}}$ est l'ensemble des modèles symboliques cohérents pas à pas et le langage reconnu par $\mathcal{A}_{\text{symb}}$ est l'ensemble des modèles symboliques qui satisfont symboliquement ϕ.

Étant donnée une formule ϕ, on peut facilement définir un automate d'arbres alternant $\mathcal{A}_{\text{pap}}$ avec une condition d'acceptation de parité qui reconnaît l'ensemble des arbres étiquetés par des éléments de $\texttt{Frame}(\phi)$ qui sont cohérents pas à pas. L'ensemble des états de cet automate ainsi que son alphabet sont égaux à l'ensemble $\texttt{Frame}(\phi)$ et la relation de transition est définie par la seule règle

$$\delta(fr, fr) = \bigwedge_{a\in\{1,\ldots,d\}} \bigvee_{\substack{fr' \,\in\, \texttt{Frame}(\phi) \\ \text{t.q. } \langle fr, fr'\rangle \text{ est } a\text{-cohérent}}} \langle a, fr'\rangle$$

pour tout $fr \in \texttt{Frame}(\phi)$. Pour tout couple de fenêtres symboliques $fr, fr' \in \texttt{Frame}(\phi)$ tel que $fr \neq fr'$, on a donc $\delta(fr, fr') = \perp$. Tout les états sont acceptants ce qui implique que tout chemin infini est acceptant. Cette condition peut être spécifiée par une condition de parité en affectant à tous les états de Q un rang égal à 0. Au contraire, notons que la relation de transition n'autorise pas l'existence de chemin fini dans le modèle symbolique.

Nous nous concentrons maintenant sur la construction de $\mathcal{A}_{\text{symb}}$.

Lemme 49 *Étant donnée une formule ϕ de* CCTL$^\star(\mathcal{D})$*, nous pouvons construire un automate $\mathcal{A}_{\text{symb}}$ tel que pour tout modèle symbolique arborescent T_{symb} par rapport à ϕ, on a $T_{\text{symb}} \models_{\text{symb}} \phi$ ssi T_{symb} est accepté par $\mathcal{A}_{\text{symb}}$.*

Preuve : L'automate $\mathcal{A}_{\text{symb}}$ est un automate d'arbres alternant avec condition de parité sur l'alphabet $\texttt{Frame}(\phi)$. Nous décrivons sa construction par induction sur la structure de ϕ. Cette construction est une adaptation de la construction classique pour CTL* (voir par exemple [KVW00]).

Une sous-formule ϕ' de ϕ est dite maximale ssi ϕ' est une sous-formule d'état stricte de ϕ et il n'existe pas de sous-formule d'état stricte ϕ'' de ϕ telle que ϕ' est une sous-formule stricte de ϕ''. Nous notons $\text{Max}(\phi) = \{\phi_1, \ldots, \phi_s\}$ l'ensemble des sous-formules maximales de ϕ. Pour chaque sous-formule maximale ϕ_i, nous supposons par induction que l'automate $\mathcal{A}^i_{\text{symb}} = \langle Q^i, \{q^i_0\}, \texttt{Frame}(\phi), \delta^i, \text{Rank}^i\rangle$ qui reconnaît les modèles symboliques qui satisfont ϕ_i est défini, ainsi que son complément $\tilde{\mathcal{A}}^i_{\text{symb}} = \langle \tilde{Q}^i, \{\tilde{q}^i_0\}, \texttt{Frame}(\phi), \tilde{\delta}^i, \tilde{\text{Rank}}^i\rangle$. Rappelons que le complément d'un automate d'arbres alternant avec condition de parité se calcule en temps linéaire (voir par exemple la preuve de [Kir02]). Nous supposons sans perte de généralité que tous les états de ces automates sont disjoints.

Si ϕ vaut $\top$ ou $\perp$ alors la construction de $\mathcal{A}_{\text{symb}}$ est triviale.

Si ϕ est une conjonction de la forme $\phi_1 \wedge \phi_2$ alors l'automate $\mathcal{A}_{\text{symb}} = \langle Q^1 \cup Q^2 \cup \{q_0\}, \{q_0\}$

$\mathtt{Frame}(\phi), \delta, \mathrm{Rank}\rangle$ est défini en ajoutant un état initial et en envoyant des copies vers les automates $\mathcal{A}^1_{\mathrm{symb}}$ et $\mathcal{A}^2_{\mathrm{symb}}$. Formellement, ceci nous donne :

- pour tout $q \in Q_1$, on a $\mathrm{Rank}(q) = \mathrm{Rank}^1(q)$ et $\delta(q, fr) = \delta^1(q, fr)$ pour tout $fr \in \mathtt{Frame}(\phi)$,
- pour tout $q \in Q_2$, on a $\mathrm{Rank}(q) = \mathrm{Rank}^2(q)$ et $\delta(q, fr) = \delta^2(q, fr)$ pour tout $fr \in \mathtt{Frame}(\phi)$,
- $\mathrm{Rank}(q_0) = 0$ et pour tout $fr \in \mathtt{Frame}(\phi)$ on ajoute la règle de transition

$$\delta(q_0, fr) = \delta(q_0^1, fr) \wedge \delta(q_0^2, fr).$$

Lorsque ϕ est une disjonction de la forme $\phi_1 \vee \phi_2$, la construction est la même en remplaçant la conjonction par une disjonction dans la règle $\delta(q_0, fr)$.

Si ϕ est de la forme $E\psi$, alors nous commençons par construire l'automate d'arbres alternant $\mathcal{A}_\psi$ qui vérifie si la formule linéaire ψ est satisfaite par une branche de l'arbre lorsque l'on considère les sous-formules maximales comme des propositions. Cet automate n'est pas une adaptation directe de l'automate de $\mathrm{CLTL}(\mathcal{D})$ puisqu'il est nécessaire de connaître à l'avance les l prochaines positions des sommets du chemin que l'on considère dans l'arbre pour évaluer une contrainte atomique et ceci impose donc de respecter certains déplacements dans l'arbre par la suite. Nous devons toujours connaître l coups à l'avance et lorsqu'une transition est franchie, nous suivons en fait le chemin déjà deviné et nous devinons le $l+1^{\text{ème}}$ prochain déplacement. Nous mémorisons les l prochains déplacements à effectuer dans le codage des états de contrôle.

Plus formellement, nous avons d'abord besoin de définir une variante de la clôture habituelle d'une formule de LTL (utilisée par la construction de [VW94]) car nous manipulons des formules sous forme positive utilisant l'opérateur $\tilde{\mathsf{U}}$. La clôture de ψ, notée $cl(\psi)$ est le plus petit ensemble contenant ψ et vérifiant :

- pour toute formule ϕ' appartenant à $cl(\psi)$, la formule $\neg\phi'$ appartient à $cl(\psi)$ (on identifie $\neg\neg\phi'$ et ϕ'),
- pour toute formule de la forme $\phi' \wedge \phi''$ ou $\phi' \vee \phi''$ appartenant à $cl(\psi)$ les formules ϕ' et ϕ'' appartiennent à $cl(\psi)$,
- pour toute formule de la forme $\mathsf{X}\phi'$ appartenant à $cl(\psi)$, la formule ϕ' appartient à $cl(\psi)$,
- pour toute formule de la forme $\phi'\mathsf{U}\phi''$ appartenant à $cl(\psi)$ les formules ϕ', ϕ'' et $\mathsf{X}(\phi'\mathsf{U}\phi'')$ appartiennent à $cl(\psi)$,
- pour toute formule de la forme $\phi'\tilde{\mathsf{U}}\phi''$ appartenant à $cl(\psi)$ les formules ϕ', ϕ'' et $\mathsf{X}(\phi'\tilde{\mathsf{U}}\phi'')$ appartiennent à $cl(\psi)$.

Notons que d'après cette définition, nous considérons à la fois les sous-formules maximales et les contraintes atomiques comme des propositions. L'alphabet de $\mathcal{A}_\psi$ est donc $\mathtt{Frame}(\phi) \times \mathcal{P}(\mathrm{Max}(\phi))$. Un atome est un ensemble maximalement cohérent d'éléments de $cl(\psi)$. Nous notons $\mathrm{Atom}(\psi)$ l'ensemble des atomes de ψ tel que $At \in \mathrm{Atom}(\psi)$ ssi

– pour toute formule ϕ' de $cl(\psi)$, soit $\phi' \in At$ ou bien $\neg\phi' \in At$ mais pas les deux,

– la formule $\phi' \wedge \phi''$ appartient à At ssi les formules ϕ' et ϕ'' appartiennent à At,

– la formule $\phi' \vee \phi''$ appartient à At ssi ϕ' ou ϕ'' appartient à At,

– la formule $\phi' \mathsf{U} \phi''$ appartient à At ssi la formule ϕ'' appartient à At ou les formules ϕ' et $\mathsf{X}(\phi' \mathsf{U} \phi'')$ appartiennent à At,

– la formule $\phi' \tilde{\mathsf{U}} \phi''$ appartient à At ssi $\phi'' \in At$ et la formule ϕ' ou $\mathsf{X}(\phi' \tilde{\mathsf{U}} \phi'')$ appartient à At.

L'automate d'arbres alternant $\mathcal{A}_\psi = \langle Q, \{q_0\}, \mathtt{Frame}(\phi) \times \mathcal{P}(\mathrm{Max}(\phi)), \delta_\psi, F \rangle$ est tel que

– $Q = (\mathrm{Atom}(\psi) \times \{1, \ldots, d\}^l) \uplus \{q_0\}$,

– $\delta_\psi(q_0, \langle fr, X \rangle) = \bigvee_{\phi \in At} \bigvee_{w \in \{1,\ldots,d\}^l} \delta_\psi(\langle At, w \rangle, \langle fr, X \rangle)$,

– $\delta_\psi(\langle At, a \cdot w \rangle, \langle fr, X \rangle) = \bigvee_{b \in \{0,\ldots,d\}} (a, \langle At', w \cdot b \rangle)$ ssi

  – $fr(a \cdot w) \models_{\mathrm{symb}} \alpha$ pour tout $\alpha \in At$,

  – $\phi_i \in X$ pour tout $\phi_i \in At$,

  – pour tout $\mathsf{X}\phi \in cl(\psi)$ on a $\mathsf{X}\phi \in At$ ssi $\phi \in At'$.

– Soit $\{\phi'_1 \mathsf{U} \phi''_1, \ldots, \phi'_r \mathsf{U} \phi''_r\}$ l'ensemble des formules "until" de $cl(\psi)$. Si cet ensemble est vide nous posons $F = Q$, sinon $F = \{F_1, \ldots, F_r\}$ tel que pour tout $i \in \{1, \ldots, r\}$, $F_i = \{\langle At, w \rangle \in Q : \phi'_i \mathsf{U} \phi''_i \notin At \text{ ou } \phi''_i \in At\}$.

D'après les Théorèmes 14 et 15, nous pouvons facilement transformer cet automate en un automate avec condition de parité $\mathcal{A}'_\psi = \langle Q', \{q'_0\}, \mathtt{Frame}(\phi) \times \mathcal{P}(\mathrm{Max}(\phi)), \delta'_\psi, \mathrm{Rank}' \rangle$. Pour obtenir finalement l'automate $\mathcal{A}_{\mathrm{symb}}$ qui reconnaît les modèles symboliques qui satisfont $E\psi$ il nous reste à compléter la construction afin de vérifier que les sous-formules maximales supposées vraies à chaque position de l'exécution de $\mathcal{A}'_\psi$ sont satisfaites. Pour cela, nous envoyons des copies vers les automates vérifiant ces sous-formules en modifiant la relation de transition de la façon suivante :

$$\delta(q, fr) = \bigvee_{X \subseteq \mathrm{Max}(\phi)} \big(\delta'_\psi(q, \langle fr, X \rangle) \wedge \bigwedge_{\phi_i \in X} \delta^i(q^i_0, fr) \wedge \bigwedge_{\phi_i \notin X} \tilde{\delta}^i(\tilde{q}^i_0, fr)\big).$$

Pour le cas où ϕ est de la forme $A\psi$ nous construisons l'automate pour $E\neg\psi$ puis nous le complémentons (en temps linéaire).

Nous montrons pour terminer la correction de cette construction. Nous développons le cas où ϕ est de la forme $E\psi$ qui est le seul cas non-trivial. Si un modèle symbolique T_{symb} satisfait symboliquement la formule ϕ alors il existe un chemin infini π à partir de la racine de T_{symb} tel que $\langle T_{\mathrm{symb}}, \pi \rangle \models_{\mathrm{symb}} \psi$. Par construction, on peut vérifier qu'une exécution de $\mathcal{A}_{\mathrm{symb}}$ qui procède le long de w_π est acceptante.

Réciproquement, si un modèle symbolique T_{symb} est accepté $\mathcal{A}_{\mathrm{symb}}$, il existe une exécution acceptante T_r sur l'entrée T_{symb} et un chemin π de T_r tel que si $\mathcal{A}_\psi$ procède le long de w_π alors T_{symb} est accepté. D'après la construction de $\mathcal{A}_\psi$ on a donc $\langle T_{\mathrm{symb}}, \pi' \rangle \models_{\mathrm{symb}} \psi$ où π' est le chemin décrit par w_π à partir de la racine de T_{symb}. Par conséquent on a bien $T_{\mathrm{symb}} \models_{\mathrm{symb}} \phi$. □

L'automate $\mathcal{A}_\phi$ est l'automate d'arbres alternant avec condition d'acceptation de parité obtenu en faisant l'intersection classique des automates $\mathcal{A}_{\text{pap}}$ et $\mathcal{A}_{\text{symb}}$. Notons que lorsque ϕ est une formule de CLTL($\mathcal{D}$) alors $\mathcal{A}_{\text{pap}}$ et $\mathcal{A}_{\text{symb}}$ sont des automates de mots (puisque $d = 1$) et on peut vérifier que l'automate $\mathcal{A}_{\text{symb}}$ est similaire à l'automate de LTL puisque l'ensemble des sous-formules maximales est vide. De plus, la condition d'acceptation peut être spécifiée par une condition de Büchi. En effet, la condition de parité est nécessaire pour complémenter les automates dans la construction de la preuve du Lemme 49. Cette condition n'est donc pas nécessaire dans le cas de CLTL($\mathcal{D}$). Pour le fragment linéaire, nous retrouvons donc la construction d'un automate de Büchi sur des mots. Il nous reste à prouver que cette construction vérifie bien la propriété désirée.

Lemme 50 *Pour toute formule ϕ de* CCTL*($\mathcal{D}$)*, si on peut construire une abstraction vérifiable localement alors ϕ est satisfaisable ssi le langage accepté par $\mathcal{A}_\phi$ est non vide.*

Preuve : Supposons qu'il existe un arbre T_{symb} qui est accepté par $\mathcal{A}_\phi$. Cette arbre est donc accepté à la fois par $\mathcal{A}_{\text{pap}}$ et $\mathcal{A}_{\text{symb}}$ d'après la construction de $\mathcal{A}_\phi$. Comme l'arbre T_{symb} est reconnu par $\mathcal{A}_{\text{pap}}$, il est cohérent pas à pas ce qui signifie que T_{symb} est un modèle symbolique satisfaisable car l'abstraction considérée est vérifiable localement (Lemme 48). De plus, si T_{symb} est accepté par $\mathcal{A}_{\text{symb}}$ alors ce modèle symbolique satisfait symboliquement la formule ϕ d'après le Lemme 49. En utilisant le Lemme 47 nous pouvons en déduire que ϕ est satisfaisable.

Réciproquement, si ϕ est satisfaisable alors il existe d'après le Lemme 47 un modèle symbolique arborescent T_{symb} et un d-arbre T tel que $T_{\text{symb}} \models_{\text{symb}} \phi$ et $T \models T_{\text{symb}}$. Comme T_{symb} est un modèle symbolique, il est cohérent pas à pas et donc $\mathcal{A}_{\text{pap}}$ accepte cet arbre. De plus, d'après le Lemme 49 cet arbre est aussi accepté par $\mathcal{A}_{\text{symb}}$ puisque $T_{\text{symb}} \models_{\text{symb}} \phi$. Le modèle symbolique T_{symb} appartient donc au langage accepté par $\mathcal{A}_\phi$. $\square$

Comme le test du vide pour un automate d'arbres alternant avec condition d'acceptation de parité est décidable, nous pouvons donc déduire la décidabilité du problème de la satisfaisabilité pour les extensions de CTL* sur des domaines concrets lorsqu'il existe une abstraction vérifiable localement.

Théorème 38 *Le problème de la satisfaisabilité pour* CCTL*($\mathcal{D}$) *est décidable si on peut calculer une abstraction vérifiable localement pour les formules de cette logique.*

Nous étudions maintenant la complexité de cette construction. Nous rappelons d'abord que toute formule de CCTL*($\mathcal{D}$) peut être mise sous forme positive en temps linéaire d'après le Lemme 4. Lorsque la relation de satisfaction symbolique peut se tester en temps exponentiel par rapport à $|\phi|$, l'automate $\mathcal{A}_{\text{symb}}$ peut être construit en temps exponentiel par rapport à $|\phi|$. Il est alors clair qu'à partir d'une formule sous forme positive ϕ, l'automate $\mathcal{A}_\phi$ peut être construit en temps exponentiel par rapport à $|\phi|$ puisque chacune des composantes $\mathcal{A}_{\text{pap}}$ et $\mathcal{A}_{\text{symb}}$ peut être construite en temps exponentiel. Notons que la taille de ces différents automates est aussi exponentielle par rapport à $|\phi|$. Pour construire l'intersection de ces deux automates, on a juste besoin d'ajouter un état initial à l'ensemble des états et une règle de transition qui envoie des copies vers les états initiaux des automates $\mathcal{A}_{\text{pap}}$ et $\mathcal{A}_{\text{symb}}$. Le problème du vide pour les automates d'arbres alternants avec condition de parité étant EXPTIME-complet, nous obtenons une procédure totale dans 2EXPTIME.

Théorème 39 *Le problème de la satisfaisabilité pour* CCTL$^\star(\mathcal{D})$ *est* 2EXPTIME-*complet lorsque l'on peut calculer une abstraction vérifiable localement pour laquelle la relation de satisfaction symbolique se teste en temps exponentiel.*

La 2EXPTIME-dureté de ce problème est obtenue directement puisque la famille de logiques considérée englobe CTL* (lorsque $\mathcal{D}$ n'est pas trivial). En effet, les formules atomiques de CTL* n'expriment pas de contraintes comparant plusieurs variables propositionnelles ce qui fait que le domaine correspondant à CTL* vérifie la propriété de complétion. Notons que les exemples de domaines concrets cités au début de cette section vérifient tous les critères du Théorème 39. Pour le cas linéaire nous retrouvons la borne de PSPACE-complétude lorsque la relation de satisfaction symbolique se teste en espace polynomial puisque nous construisons un automate de Büchi en espace polynomial par rapport à ϕ pour lequel le vide peut être testé en NLOGSPACE.

7.4 Problème du model-checking

Nous traitons maintenant le problème du model-checking des extensions de la forme CCTL$^\star(\mathcal{D})$ pour lesquelles il existe une abstraction localement vérifiable. Le principal problème que nous rencontrons est lié au fait qu'un $\mathcal{D}$-automate peut générer un graphe de configurations ayant un degré de branchement infini. Nous ne pouvons pas manipuler un tel graphe avec les automates que nous avons utilisés dans la section précédente. Nous définissons donc un automate qui reconnaît des exécutions symboliques de l'automate ayant un branchement fini puis combinons cet automate avec la construction définie pour résoudre le problème de la satisfaisabilié.

Soit ϕ une formule de CCTL$^\star(\mathcal{D})$ sous forme positive et $\mathcal{A} = \langle Q, I, F, \delta\rangle$ un $\mathcal{D}$-automate. Nous définissons pour le reste de cette section V l'ensemble des variables utilisées dans ϕ et $\mathcal{A}$ et $l = |\phi|_{\mathsf{X}}$. Sans perte de généralité, nous supposons que $l > 0$. Nous devons considérer maintenant un ensemble de valuations symboliques qui tient compte du comportement de l'automate $\mathcal{A}$. Nous utilisons donc l'ensemble des valuations symboliques construites par rapport à l'ensemble des contraintes atomiques de ϕ et des contraintes utilisées dans la relation de transition de $\mathcal{A}$, que nous notons $\mathrm{SV}(\phi, \mathcal{A})$. Nous supposons que cet ensemble vérifie les conditions **(SV1)** et **(SV2)** que nous avons définies précédemment et que cette abstraction et vérifiable localement. Notons que la condition **(SV2)** est donc valable à la fois pour les contraintes atomiques de ϕ et les contraintes sur les transitions de $\mathcal{A}$. Nous réutilisons aussi les définitions relatives à l'abstraction définie par $\mathrm{SV}(\phi, \mathcal{A})$:

- pour toute valuation v, nous notons $sv(v)$ l'unique valuation symbolique de $\mathrm{SV}(\phi, \mathcal{A})$ telle que $v \models sv(v)$,
- pour toute contrainte α et valuation symbolique $sv \in \mathrm{SV}(\phi, \mathcal{A})$, nous notons $sv \models_{\text{symb}} \alpha$ ssi pour toute valuation v telle que $sv(v) = sv$ on a $v \models \alpha$.

Nous introduisons la notation $\mathrm{SV}_0(\phi, \mathcal{A})$ pour l'ensemble composé de la restriction des éléments de $\mathrm{SV}(\phi, \mathcal{A})$ aux contraintes utilisant exclusivement des variables et non des termes. Donc pour tout ensemble X appartenant à $\mathrm{SV}_0(\phi, \mathcal{A})$ il existe au moins une valuation symbolique $sv \in \mathrm{SV}(\phi, \mathcal{A})$ telle que pour toute relation α de la forme $\mathrm{R}(x_{j_1}, \ldots, x_{j_k}) \in sv$ avec $x_{j_1}, \ldots, x_{j_k} \in V$ on a $\alpha \in X$ ssi $\alpha \in sv$. Une conséquence de cette définition est que $\mathrm{SV}_0(\phi, \mathcal{A})$ vérifie **(SV1)** pour les valuations de la forme $V \to D$ et **(SV2)** pour les contraintes de X impliquant seulement des variables.

Nous utilisons les éléments de $\mathrm{SV}_0(\phi, \mathcal{A})$ pour définir une abstraction finie de l'ensemble infini des états de $\mathcal{A}$. Nous construisons à partir du $\mathcal{D}$-automate $\mathcal{A}$ un automate de Büchi classique $\mathcal{A}_{\mathrm{abs}} = \langle Q', I', F', \delta' \rangle$ sur l'alphabet $\mathrm{SV}(\phi, \mathcal{A})$ qui représente une abstraction de $\mathcal{A}$. L'ensemble des états Q' est égal à $Q \times \mathrm{SV}_0(\phi, \mathcal{A})$ et la relation de transition est définie par $\langle q, X \rangle \xrightarrow{sv} \langle q', X' \rangle$ ssi

- $sv \models X$, ce qui signifie que pour tout $\mathrm{R}(x_{j_1}, \ldots, x_{j_k}) \in X$ on a $\mathrm{R}(x_{j_1}, \ldots, x_{j_k}) \in sv$.
- $sv \models X'[x \leftarrow \mathsf{X}x \mid x \in \mathrm{VAR}]$ où $X'[x \leftarrow \mathsf{X}x \mid x \in \mathrm{VAR}]$ est obtenu à partir de X' en substituant chaque occurrence de x par $\mathsf{X}x$ pour tout $x \in \mathrm{VAR}$. Ceci signifie que pour tout $\mathrm{R}(x_{j_1}, \ldots, x_{j_k}) \in X'$ on a $\mathrm{R}(\mathsf{X}x_{j_1}, \ldots, \mathsf{X}x_{j_k}) \in sv$.
- il existe un chemin fini $q_0 \xrightarrow{\alpha_0} q_1 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{l-1}} q_l$ dans $\mathcal{A}$ tel que
  - $q_0 = q$ et $q_1 = q'$,
  - pour tout $i \in \{0, \ldots, l-1\}$ on a $sv \models_{\mathrm{symb}} \mathsf{X}^i \alpha_i$ où $\mathsf{X}^i \alpha_i$ est la contrainte atomique obtenue en substituant chaque terme $\mathsf{X}^j x$ par $\mathsf{X}^{i+j} x$ dans α_i.

L'ensemble des états initiaux I' est égal $\{\langle q_0, X_0 \rangle \in Q \times \mathrm{SV}_0(\phi, \mathcal{A}) \mid q_0 \in I \text{ et } \vec{0} \models X\}$ où $\vec{0}$ est une valuation initiale, et l'ensemble des états finaux est $F' = F \times \mathrm{SV}_0(\phi, \mathcal{A})$. Cette construction se fait en temps exponentiel par rapport à $|\mathcal{A}|$ lorsque la relation de satisfaction symbolique peut se tester en temps exponentiel. Notons que cette construction implique que pour tout $q \in Q$ et $v : V \to D$ les états $\langle q, v \rangle$ de $\mathcal{A}$ et $\langle q, X \rangle$ de $\mathcal{A}_{\mathrm{abs}}$ tel que $v \models X$ sont bisimilaires. Ce qui nous intéressent plus particulièrement est d'établir une correspondance plus forte entre les exécutions linéaires dans ces deux automates.

Lemme 51 *Il existe une exécution infinie de la forme* $\langle q_0, v_0 \rangle \xrightarrow{\alpha_0} \langle q_1, v_1 \rangle \xrightarrow{\alpha_1} \ldots$ *dans* $\mathcal{A}$ *ssi il existe une exécution infinie de la forme* $\langle q_0, X_0 \rangle \xrightarrow{sv_0} \langle q_1, X_1 \rangle \xrightarrow{sv_1} \ldots$ *dans* $\mathcal{A}_{\mathrm{abs}}$ *telle que les séquences* $\sigma = v_0 \cdot v_1 \cdots$ *et* $\rho = sv_0 \cdots sv_1 \cdots$ *vérifient*

- ρ *est cohérente pas à pas,*
- $\sigma \models \rho$, *c'est-à-dire que pour tout* $i \in \mathbb{N}$ $\sigma, i \models \rho(i)$. *En particulier pour tout* $i \in \mathbb{N}$ *on a* $v_i \models X_i$.

Preuve : Soit $\langle q_0, v_0 \rangle \xrightarrow{\alpha_0} \langle q_1, v_1 \rangle \xrightarrow{\alpha_1} \ldots$ une exécution infinie de $\mathcal{A}$. Par construction de l'abstraction, il existe pour chaque sous-chemin $\langle q_i, v_i \rangle \xrightarrow{\alpha_i} \cdots \xrightarrow{\alpha_{i+l-1}} \langle q_{i+l}, v_{i+l} \rangle$ de longueur l une transition de $\mathcal{A}_{\mathrm{abs}}$ de la forme $\langle q_i, X_i \rangle \xrightarrow{sv(v_i^l)} \langle q_{i+1}, X_{i+1} \rangle$ telle que $v_i \models X_i$, $v_{i+1} \models X_{i+1}$ et v_i^l est la valuation définie par $v_i^l(\mathsf{X}^j x) = v_{i+j}(x)$ pour tout $j \in \{0, \ldots, l\}$. En effet, d'après les conditions **(SV1)** et **(SV2)** et la définition de la relation de satisfaction symbolique on a bien

– $v_i \models X_i$ implique que $sv(v_i^l) \models_{\text{symb}} X_i$,
– $v_{i+1} \models X_{i+1}$ implique que $sv(v_i^l) \models X'[x \leftarrow \mathsf{X}x \mid x \in \text{VAR}]$,
– pour tout $j \in \{0, \dots, l-1\}$, $v_i, v_{i+1} \models \alpha_i$ implique que $sv(v_i^l) \models_{\text{symb}} \mathsf{X}^i\alpha_i$.

Notons que d'après **(SV1)** les ensembles X_i et X_{i+1} sont uniques pour v_i et v_{i+1} donnés. Nous posons $sv_i = sv(v_i^l)$ pour tout $i \in \mathbb{N}$. Comme les valuations symboliques sv_i et sv_{i+1} sont définies par rapport à des chemins qui se superposent partiellement, il est facile de montrer que chaque paire $\langle sv_i, sv_{i+1} \rangle$ est cohérente sur un pas. Donc l'exécution $\langle q_0, X_0 \rangle \xrightarrow{sv_0} \langle q_1, X_1 \rangle \xrightarrow{sv_1} \langle q_2, X_2 \rangle \xrightarrow{sv_2} \cdots$ est telle que la séquence $\rho = sv_0 \cdot sv_1 \cdots$ est cohérente pas à pas. Enfin, comme à chaque position $i \in \mathbb{N}$ on a $sv_i = sv(v_i^l)$, il est évident que la séquence $\sigma = v_0 \cdot v_1 \cdots$ satisfait ρ.

Réciproquement, supposons qu'il existe une exécution infinie $\langle q_0, X_0 \rangle \xrightarrow{sv_0} \langle q_1, X_1 \rangle \xrightarrow{sv_1} \dots$ dans $\mathcal{A}_{\text{abs}}$ telle que la séquence $\rho = sv_0 \cdot sv_1 \cdots$ est cohérente pas à pas. Nous montrons comment construire une exécution linéaire de $\mathcal{A}$ vérifiant les bonnes propriétés par induction sur la position du chemin. Par définition de la relation de transition de $\mathcal{A}_{\text{abs}}$, il existe un chemin $q_0 \xrightarrow{\alpha_0} q_1 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{l-1}} q_l$ de longueur l dans $\mathcal{A}$ tel que pour tout $i \in \{0, \dots, l-1\}$ on a $sv_0 \models_{\text{symb}} \mathsf{X}^i\alpha_i$. De plus, comme sv_0 est satisfaisable il existe une valuation v telle que $v \models sv_0$. Par définition de la relation de satisfaction symbolique, la sous-exécution finie $\langle q_0, v_0 \rangle \xrightarrow{\alpha_0} \langle q_1, v_1 \rangle \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{l-1}} \langle q_l, v_l \rangle$ telle que pour tout $j \in \{0, \dots, l\}$ et $x \in V$ on a $v_j(x) = v(\mathsf{X}^j x)$ est valide, c'est-à-dire que les valuations v_j et v_{j+1} satisfont α_j pour tout $j \in \{0, \dots, l-1\}$.

Supposons maintenant que nous pouvons construire une exécution $\langle q_0, v_0 \rangle \xrightarrow{\alpha_0} \langle q_1, v_1 \rangle \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{i-2+l}} \langle q_{i-1+l}, v_{i-1+l} \rangle$ de $\mathcal{A}$ satisfaisant toutes les valuations symboliques de ρ jusqu'à la position $i-1$. Comme la séquence ρ est cohérente pas à pas, si $sv_{i-1} \models_{\text{symb}} \mathsf{X}^j\alpha_{i-1+j}$ pour tout $j \in \{0, \dots, l-1\}$ alors $sv_i \models_{\text{symb}} \mathsf{X}^j\alpha_{i+j}$ pour tout $j \in \{0, \dots, l-2\}$. De plus, la valuation v_i^{l-1} définie par $v_i^{l-1}(\mathsf{X}^j x) = v_{i+j}(x)$ pour tout $j \in \{0, \dots, l-1\}$ satisfait toute les contraintes de sv_i impliquant uniquement des termes de la forme $\mathsf{X}^i x$ tel que $i < l$.

Dans $\mathcal{A}$, il existe une transition $q_{i+l-1} \xrightarrow{\alpha_{i+l-1}} q_{i+l}$ telle que $sv_i \models_{\text{symb}} \mathsf{X}^l\alpha_{i+l-1}$, sinon comme la séquence ρ est cohérente pas à pas il n'existe pas de transition $q_{i+l-1} \xrightarrow{\alpha_{i+l-1}} q_{i+l}$ telle que $sv_{i+l-1} \models_{\text{symb}} \alpha_{i+l-1}$ et par conséquent la transition $\langle q_{i+l-1}, X_{i+l-1} \rangle \xrightarrow{sv_{i+l-1}} \langle q_{i+l}, X_{i+l} \rangle$ n'existe pas. Comme l'abstraction que nous utilisons est vérifiable localement, il existe une valuation v_{i+l} qui étend v_i^{l-1} en v_i^l telle que $v_i^l(\mathsf{X}^l x) = v_{i+l}(x)$ et $v_i^l \models sv_i$. Par définition de la relation de satisfaction symbolique, on a $v_i^l \models \alpha_{i+l-1}$ et donc l'exécution peut être étendue en utilisant la transition $\langle q_{i+l-1}, v_{i+l-1} \rangle \xrightarrow{\alpha_{i+l-1}} \langle q_{i+l}, v_{i+l} \rangle$. □

Nous posons d comme étant le maximum de $E_\sharp(\phi)+1$ et du degré de $\mathcal{A}_{\text{abs}}$, c'est-à-dire le nombre maximal de transitions issues d'un même état. Nous notons $\texttt{Frame}(\phi, \mathcal{A})$ l'ensemble des fenêtres symboliques de la forme $\{1, \dots, d\}^l \rightarrow \text{SV}(\phi, \mathcal{A})$. Une exécution symbolique de $\mathcal{A}_{\text{abs}}$ est un modèle symbolique arborescent $T_{\text{symb}} = \langle N, \{n_0\}, \rightarrow, \Gamma_{\text{symb}} \rangle$ vérifiant les propriétés suivantes.

– $N \subseteq Q \times \text{SV}_0(\phi, \mathcal{A}) \times \{1, \dots, d\}^*$ où la dernière composante marque le chemin suivi depuis la racine et sert à différencier les multiples copies d'un même état de $\mathcal{A}_{\text{abs}}$,

- $\Gamma_{\text{symb}} : N \rightarrow \texttt{Frame}(\phi, \mathcal{A})$,
- n_0 est de la forme $\langle q_0, X_0, \varepsilon \rangle$ où $q_0 \in I$ et $\Gamma_{\text{symb}}(n_0) = fr_0$ tel que pour tout $w \in \{1, \ldots, d\}^l$ on a $X_0 \subseteq fr_0(w)$,
- Soit $n = \langle q, X, w_n \rangle$ un noeud de N tel que $\Gamma_{\text{symb}}(n) = fr$ et $\{n_1, \ldots, n_d\}$ l'ensemble de ces successeurs. Nous notons $n_j = \langle q_j, X_j, w_n \cdot j \rangle$ pour tout $j \in \{1, \ldots, d\}$. Ces sommets vérifient les propriétés suivantes :
  - pour tout $j \in \{1, \ldots, d\}$ il existe une transition $\langle q, X \rangle \xrightarrow{sv_j} \langle q_j, X_j \rangle$ dans $\mathcal{A}_{\text{abs}}$ et il existe un mot $w \in \{1, \ldots, d\}^{l-1}$ tel que $fr(j \cdot w) = sv_j$,
  - pour toute transition $\langle q, X \rangle \xrightarrow{sv} \langle q', X' \rangle$ dans $\mathcal{A}_{\text{abs}}$ il existe $j \in \{1, \ldots, d\}$ tel que $\langle q', X' \rangle = \langle q_j, X_j \rangle$ et il existe $w \in \{1, \ldots, d\}^{l-1}$ tel que $fr(j \cdot w) = sv_j$.
  - pour tout $j \in \{1, \ldots, d\}$ la paire $\langle fr, \Gamma_{\text{symb}}(n_j) \rangle$ est j-cohérente sur un pas.

Par construction, $\mathcal{A}_{\text{abs}}$ et T_{symb} sont liés par la propriété suivante. Nous réutilisons la définition de satisfaction symbolique $\models_{\text{symb}}$ entre les modèles symboliques arborescents et les sous-formules de ϕ.

Lemme 52 *Pour tout chemin infini $\langle q_0, X_0, w_0 \rangle \rightarrow \langle q_1, X_1, w_1 \rangle \rightarrow \cdots$ dans T_{symb}, il existe une exécution linéaire de la forme $\langle q_0, X_0 \rangle \xrightarrow{sv_0} \langle q_1, X_1 \rangle \xrightarrow{sv_1} \ldots$ dans $\mathcal{A}_{\text{abs}}$ telle que la séquence $sv_0 \cdot sv_1 \cdots$ est cohérente pas à pas.*

Preuve : Considérons un chemin $\pi = \langle q_0, X_0, w_0 \rangle \xrightarrow{a_0} \langle q_1, X_1, w_1 \rangle \xrightarrow{a_1} \cdots$ dans T_{symb}. Par définition de $T_{\text{symb}} = \langle N, \{n_0\}, \rightarrow, \Gamma_{\text{symb}} \rangle$ il existe pour tout $i \in \mathbb{N}$ une transition $\langle q_i, X_i \rangle \xrightarrow{sv_i} \langle q_{i+1}, X_{i+1} \rangle$ dans $\mathcal{A}_{\text{abs}}$ ce qui implique l'existence d'un chemin $q_i \xrightarrow{\alpha_i} q_{i+1} \xrightarrow{\alpha^i_{i+1}} q^i_{i+2} \xrightarrow{\alpha^i_{i+2}} \cdots \xrightarrow{\alpha^i_{i+l-1}} q^i_{i+l}$ dans $\mathcal{A}$ tel que

- $sv_i \models X_i$,
- $sv_i \models X_{i+1}[x \leftarrow \mathsf{X}x \mid x \in \text{VAR}]$,
- $sv_i \models_{\text{symb}} \alpha_i$,
- et il existe un mot $w''_i \in \{1, \ldots, d\}^{l-1}$ tel $fr_i(a_i \cdot w''_i) = sv_i$ où $fr_i = \Gamma_{\text{symb}}(\langle q_i, X_i, i \rangle)$.

Nous posons $w'_i = w_\pi(i) \cdots w_\pi(i+l)$ pour tout $i \in \mathbb{N}$. Par définition des fenêtres symboliques, la valuation symbolique $fr_i(w'_i)$ coïncide avec sv_i sur les termes de la forme x et $\mathsf{X}x$ car $sv_i = fr_i(a_i \cdot w''_i)$. Ceci implique que $fr_i(w'_i) \models X_i$, $fr_i(w'_i) \models X_{i+1}[x \leftarrow \mathsf{X}x \mid x \in \text{VAR}]$ et $fr_i(w'_i) \models_{\text{symb}} \alpha_i$. En réitérant les mêmes arguments, nous pouvons prouver que $fr_{i+j}(w'_{i+j}) \models_{\text{symb}} \alpha_{i+j}$ pour tout $j \in \{1, \ldots, l-1\}$ et comme par définition des modèles symboliques la séquence $fr_i(w'_i) \cdot fr_{i+1}(w'_{i+1}) \cdots fr_{i+l-1}(w'_{i+l-1})$ est cohérente pas à pas cela implique que $fr_i(w'_i) \models \mathsf{X}^j \alpha_{i+j}$ pour tout $j \in \{1, \ldots, l-1\}$. Toutes les conditions sont donc réunies pour montrer l'existence d'une transition de la forme $\langle q_i, X_i \rangle \xrightarrow{fr_i(w_i)} \langle q_{i+1}, X_{i+1} \rangle$ dans $\mathcal{A}_{\text{abs}}$. Par induction, nous pouvons donc construire une exécution $\langle q_0, X_0 \rangle \xrightarrow{fr_0(w'_0)} \langle q_1, X_1 \rangle \xrightarrow{fr_1(w'_1)} \cdots$ telle que $fr_0(w'_0) \cdot fr_1(w'_1) \cdots$ est cohérente pas à pas. □

Les Lemmes 51 et 52 nous permettent d'établir la relation suivante entre $\mathcal{A}$ et les exécutions symboliques de $\mathcal{A}_{\mathrm{abs}}$.

Lemme 53 *Pour toute formule ϕ de* CCTL$^\star(\mathcal{D})$ *et tout $\mathcal{D}$-automate $\mathcal{A}$, on a $\mathcal{A} \models \phi$ ssi il existe une exécution symbolique T_{symb} de $\mathcal{A}_{\mathrm{abs}}$ telle que $T_{\mathrm{symb}} \models_{\mathrm{symb}} \phi$.*

Preuve : Nous procédons par induction sur la structure de ϕ. Le seul cas non trivial est $E\psi$. Nous supposons par induction que la propriété est vraie pour les sous-formules maximales de ϕ. Nous rappelons qu'une sous-formule maximale ϕ' de ϕ est une sous-formule d'état stricte de ϕ telle que pour toute sous-formule d'état stricte ϕ'' de ϕ, ϕ' n'est pas une sous-formule stricte de ϕ''. Dans ce cas, si $\mathcal{A} \models \phi$ alors il existe une exécution $\pi = \langle q_0, v_0\rangle \xrightarrow{\alpha_0} \langle q_1, v_1\rangle \xrightarrow{\alpha_1} \cdots$ de $\mathcal{A}$ qui satisfait ψ. D'après le Lemme 51, il existe donc une exécution $\pi_{\mathrm{abs}} = \langle q_0, X_0\rangle \xrightarrow{sv_0} \langle q_1, X_1\rangle \xrightarrow{sv_1} \ldots$ dans $\mathcal{A}_{\mathrm{abs}}$ telle que la séquence $\rho = sv_0 \cdot sv_1 \cdots$ est cohérente pas à pas et $\sigma = v_0 \cdot v_1 \cdots$ vérifie $\sigma \models \rho$. En conséquence de la définition des valuations symboliques, pour toute contrainte atomique de ϕ nous avons $\sigma, i \models \alpha$ ssi $sv_i \models \alpha$.

D'après la construction définie, il existe une exécution symbolique T_{symb} dont l'un des chemins $\pi_{\mathrm{symb}} = \langle q_0, X_0, w_0\rangle \xrightarrow{a_0} \langle q_1, X_1, w_1\rangle \xrightarrow{a_1} \cdots$ correspond à π_{abs}, c'est-à-dire que la fenêtre symbolique fr_i associée à chaque noeud de la forme $\langle q_i, X_i, w_i\rangle$ est telle que $fr(a_i \cdots a_{i+l}) = sv_i$. Donc pour toute contrainte atomique de ϕ on a $\sigma, i \models \alpha$ ssi $\langle T_{\mathrm{symb}}, \pi^i_{\mathrm{symb}}\rangle \models_{\mathrm{symb}} \alpha$. De plus, l'ensemble des sous-formules d'état satisfaites à une position i de π dans $\mathcal{A}$ sont satisfaites symboliquement à la même position de π_{symb} dans T_{symb} par hypothèse d'induction. Puisque les mêmes formules atomiques sont satisfaites à chaque position des deux chemins et que la relation de satisfaction correspond à la relation symbolique en dehors du cas des contraintes atomiques, nous pouvons conclure que $\langle T_{\mathrm{symb}}, \pi_{\mathrm{symb}}\rangle \models_{\mathrm{symb}} \psi$.

Réciproquement, si T_{symb} satisfait $E\psi$ alors il existe un chemin $\pi_{\mathrm{symb}} = \langle q_0, X_0, w_0\rangle \xrightarrow{a_0} \langle q_1, X_1, w_1\rangle \xrightarrow{a_1} \cdots$ tel que $\langle T_{\mathrm{symb}}, \pi_{\mathrm{symb}}\rangle \models_{\mathrm{symb}} \psi$. D'après la construction utilisée dans la preuve du Lemme 52, il existe une exécution $\pi_{\mathrm{abs}} = \langle q_0, X_0\rangle \xrightarrow{sv_0} \langle q_1, X_1\rangle \xrightarrow{sv_1} \ldots$ de $\mathcal{A}_{\mathrm{abs}}$ telle que pour tout $i \in \mathbb{N}$, on a $fr_i(a_i \cdots a_{i+l}) = sv_i$ où fr_i est la fenêtre symbolique associée au noeud $\langle q_i, X_i, w_i\rangle$. De plus, la séquence $\rho = sv_0 \cdot sv_1 \cdots$ est cohérente pas à pas. D'après le Lemme 51 il existe donc une exécution $\pi = \langle q_0, v_0\rangle \xrightarrow{\alpha_0} \langle q_1, v_1\rangle \xrightarrow{\alpha_1} \cdots$ de $\mathcal{A}$ telle que $\sigma = v_0 \cdot v_1 \cdots$ vérifie $\sigma \models \rho$. Nous pouvons alors déduire en utilisant les propriétés des valuations symboliques et l'hypothèse d'induction que $\pi \models \psi$ et par conséquent $\mathcal{A} \models E\psi$. $\square$

Nous étendons donc la construction de la section précédente en intersectant la variante de l'automate $\mathcal{A}_\phi$ sur l'alphabet $\texttt{Frame}(\phi, \mathcal{A})$ avec un automate $\mathcal{A}'_{\mathrm{abs}}$ qui reconnaît l'ensemble des exécutions symboliques de l'automate $\mathcal{A}_{\mathrm{abs}}$. Nous notons l'automate obtenu $\mathcal{A}^{\mathrm{MC}}_\phi$.

L'ensemble des états de $\mathcal{A}'_{\mathrm{abs}}$ est identique à celui de $\mathcal{A}_{\mathrm{abs}}$, de même que l'ensemble des états initiaux et finaux. L'alphabet de $\mathcal{A}'_{\mathrm{abs}}$ est $\texttt{Frame}(\phi, \mathcal{A})$ et la relation de transition qui reprend les règles de construction d'une exécution symbolique est définie par

$$\delta(\langle q, X\rangle, fr) = \bigwedge_{\langle q,X\rangle \xrightarrow{sv} \langle q',X'\rangle} \quad \bigvee_{i:\exists w, fr(iw)=fr} (i, \langle q', X'\rangle)$$

$$\wedge \bigwedge_{i\in\{0,\ldots,d\}} \bigvee_{\langle q,X\rangle \xrightarrow{sv} \langle q',X'\rangle : \exists w, fr(iw)=fr} (i, \langle q', X'\rangle)$$

pour tout état $\langle q, X\rangle \in Q \times \mathrm{SV}_0(\phi, \mathcal{A})$ et fenêtre symbolique $fr \in \mathtt{Frame}(\phi, \mathcal{A})$. La première partie de cette règle exprime que chaque transition de $\mathcal{A}_{\mathrm{abs}}$ correspond à un successeur dans le modèle symbolique et la seconde partie que chaque successeur correspond à une transition.

La preuve du Lemme 50 peut facilement être adaptée afin de montrer que $\mathcal{A} \models \phi$ ssi le langage accepté par l'automate $\mathcal{A}_\phi^{\mathrm{MC}}$ n'est pas vide. Ceci nous permet de prouver la décidabilité du model-checking.

Théorème 40 *Le problème de model-checking pour* $\mathrm{CCTL}^\star(\mathcal{D})$ *est décidable lorsqu'il existe une abstraction vérifiable localement pour les modèles de cette logique qui est calculable.*

La complexité de cette procédure n'augmente pas par rapport au problème de la satisfaisabilité. En effet, lorsque la relation de satisfaction symbolique peut se tester en temps exponentiel, l'automate $\mathcal{A}_{\mathrm{abs}}$ peut être construit en temps exponentiel par rapport à $\mathcal{A}$ de même que $\mathcal{A}'_{\mathrm{abs}}$ peut être construit en temps polynomial par rapport à $\mathcal{A}_{\mathrm{abs}}$.

Corollaire 14 *Le problème de model-checking pour* $\mathrm{CCTL}^\star(\mathcal{D})$ *est dans* 2EXPTIME *lorsque l'on peut calculer une abstraction vérifiable localement pour laquelle la relation de satisfaction symbolique se teste en temps exponentiel.*

L'écart de complexité avec le model-checking de $\mathrm{CTL}^\star$ s'explique par la construction de $\mathcal{A}_{\mathrm{abs}}$ qui fait qu'il est difficile de raffiner la borne de complexité en utilisant cette méthode.

Chapitre 8

CTL* avec contraintes qualitatives sur les entiers

Sommaire

Nous nous intéressons maintenant à une extension de CTL* sur un domaine dont les variables sont interprétées par des entiers qui n'appartient pas à la sous-famille de domaines concrets définie dans le chapitre précédent. Nous considérons plus précisément le domaine IPC* contenant des contraintes de comparaisons telles que $x < y$ ou $x < d$ et des contraintes de périodicité de la forme $x \equiv_K y + c$ pour $x, y \in \mathrm{VAR}$, $k, c \in \mathbb{N}$ et $d \in \mathbb{Z}$ (voir la définition exacte dans la Section 3.3.2). Nous définissons une méthode permettant de construire l'ensemble des états d'un IPC*-automate qui vérifient une formule du fragment existentiel de CCTL*(IPC*) et nous en déduisons un résultat de décidabilité pour le problème de model-checking correspondant.

Les résultats présentés dans ce chapitre correspondent à l'article [BG06].

8.1 CTL* avec contraintes de Presburger

La méthode développée dans le chapitre précédent ne permet pas de traiter le cas de nombreux domaines concrets manipulant des entiers. En effet, le Lemme 15 illustre que les modèles symboliques de CLTL(IPC*) ne sont pas vérifiables localement car la condition à vérifier porte sur l'ensemble de la séquence de valuations symboliques. En fait, c'est déjà le cas pour les modèles symboliques de CLTL($\langle \mathbb{N}, <, = \rangle$) d'après les résultats de [DD07]. La condition de caractérisation des modèles symboliques satisfaisables pour cette logique utilise en effet une condition proche de celle du Lemme 15. Nous proposons ici une autre méthode permettant de résoudre le problème du model-checking pour le fragment existentiel de CTL* étendu avec un ensemble de contraintes sur les entiers.

Notre objectif est de généraliser une partie des résultats concernant les extensions de LTL avec des contraintes sur les entiers. Étant données les restrictions nécessaires pour les fragments quantitatifs de Presburger, nous nous concentrons sur le fragment qualitatif IPC*. Malheureusement cette restriction n'est pas suffisante non plus car le problème du model-checking de CCTL*(IPC*) est indécidable. D'après un résultat de [Čer94] on peut déjà prouver l'indécidabilité du problème du model-checking pour la restriction de la logique CCTL*(Z) aux formules de longueur temporelle égale à 0. Par contre, ce même papier établit la décidabilité du model-checking pour le fragment existentiel de CCTL*(Z) restreint au formules dont la longueur temporelle est 0 sur une classe d'automates strictement incluse dans celle des Z-automates. Rappelons que nous avons déjà établi dans le Lemme 8 la correspondance entre la classe des IRA utilisée dans [Čer94] et les automates à contraintes que nous considérons. Nous généralisons ici ce résultat concernant le model-checking de ce fragment de CCTL*(Z) :

- en étendant le langage de contraintes,
- en étendant le modèle opérationnel,
- et levant la restriction sur la longueur temporelle des formules.

Contrairement aux résultats des chapitres précédents, le fragment que nous considérons dans ce chapitre restreint le langage logique et pas uniquement le langage de contraintes. Nous n'avons donc qu'un fragment de CCTL*(IPC*) mais nous proposons en contrepartie de traiter un problème plus large que le problème du model-checking. En effet, nous définissons une méthode permettant de construire une représentation de l'ensemble $[\![\phi]\!]_{\mathcal{A}}$ des états de $\mathcal{A}$ qui satisfont ϕ. Cette méthode étend les travaux de [Čer94] et certaines preuves originales complètent certains détails techniques omis dans ces travaux.

Comme nous l'avons déjà cité dans le chapitre précédent, de nombreux travaux existant introduisent des logiques arborescentes pour vérifier des propriétés sur les entiers. Dans [DFGvD06] la décidabilité est obtenue grâce à des restrictions sur la relation de transition des modèles considérés. Les restriction que nous considérons ici portent sur le langage de contraintes et la logique mais la relation de transition des automates que nous considérons n'est pas contrainte. En ce qui concerne les automates à pile, ils ne peuvent simuler plusieurs compteurs donc les contraintes que nous vérifions sont plus riches que les contraintes régulières sur le contenu de la pile considérées dans [FWW97, EKS03]. La logique temporelle sous-jacente que nous considérons est cependant moins riche que dans ces travaux qui considèrent la totalité de CTL*.

La méthode que nous définissons dans ce chapitre utilise des résultats sur les systèmes bien structurés (voir [FS01, AČJT00]). Un intérêt de ces systèmes est que l'ensemble des prédécesseurs d'un ensemble d'états donné est calculable. La convergence de l'algorithme de calcul est assurée par un beau préordre sur l'ensemble des états du système. D'autres méthodes ont été introduites afin de calculer des points fixes plus complexes permettant d'exprimer des propriétés relatives à des problèmes de jeux ou de logiques temporelles [ABd03, BBS06].

La méthode que nous définissons ici peut être vue comme un nouvel exemple de calcul de point fixe dans une famille de systèmes bien structurés puisque les propriétés de CTL* s'expriment dans le μ-calcul.

8.2 Propriété des IPC*-automates

D'après le Lemme 7, toute contrainte de IPC* peut être transformée en contrainte de WIPC*. Nous nous restreignons donc au langage de contraintes WIPC* sans perdre de vue que ceci implique une explosion de la taille des formules. À partir de maintenant nous considérons des formules sous forme positive du fragment existentiel de CCTL*(WIPC*) que nous notons ECTL*(WIPC*). Nous commençons par définir quelques propriétés concernant les WIPC*-automates.

8.2.1 Rappels sur les beaux préordres

Les propriétés que nous définissons dans cette section sont liées aux résultats sur les systèmes bien structurés dont nous rappelons rapidement quelques définitions et résultats fondamentaux (pour plus d'informations, voir par exemple [FS01]).

Soit X un ensemble d'éléments quelconque. Un *beau préordre* sur X est une relation binaire $\preceq$ réflexive et transitive sur les éléments de X telle que pour chaque séquence infinie $x_0 \cdot x_1 \cdot x_2 \cdots$ d'éléments de X il existe deux indices i et j tels que $i < j$ et $x_i \preceq x_j$. Un ordre sur X est *antisymétrique* ssi pour tout $x, y \in X$, $x \preceq y$ et $y \preceq x$ implique $x = y$.

Un ensemble $U \subseteq X$ est *fermé par le haut* si pour tout $x \in X$ et $y \in U$ on a $y \preceq x$ implique $x \in U$. Il est facile de montrer que les ensembles fermés par le haut sont clos par intersection et unions.

Lemme 54 *L'intersection et l'union d'ensembles fermés par le haut est un ensemble fermé par le haut.*

Notons cependant que le complément d'un ensemble fermé par le haut n'est pas un ensemble fermé par le haut (c'est un ensemble fermé par le bas). Par exemple, $\{x \geq 2 \mid x \in \mathbb{N}\}$ est fermé par le haut par rapport à la relation $\geq$ mais pas son complément. La *fermeture par le haut* d'un ensemble $Y \subseteq X$, notée $\uparrow Y$, est l'ensemble $\{x \in X \mid \exists y \in Y \text{ tel que } y \preceq x\}$. Une *base* pour un ensemble fermé par le haut U est un sous-ensemble minimal $U_b \subseteq U$ dont la fermeture par le haut est égale à U, c'est-à-dire que $\uparrow U_b = U$ et pour tout $x, y \in U_b$ on a $x \preceq y$ implique $x = y$. Par la suite, nous utiliserons le résultat suivant concernant l'existence d'une base pour les ensemble fermés par le haut lorsque l'on considère un beau préordre antisymétrique.

Lemme 55 [FS01] *Étant donné un beau préordre antisymétrique $\preceq$ sur un ensemble X, tout ensemble fermé par le haut $U \subseteq X$ par rapport $\preceq$ a une unique base finie.*

Notre but dans le reste de ce chapitre est de définir un algorithme qui calcule l'ensemble $[\![\phi]\!]_{\mathcal{A}}$ à partir d'une formule ϕ de ECTL*(WIPC*) et d'un WIPC*-automate $\mathcal{A}$. Nous établissons dans cette section que cet ensemble est fermé par le haut par rapport à un beau préordre que nous définissons ci-dessous.

8.2.2 Systèmes bien structurés

La méthode que nous développons s'inspire de [Čer94] et repose sur la définition d'un beau préordre sur l'ensemble des états d'un WIPC*-automate. Nous étendons cette méthode en la combinant avec l'abstraction des modèles que nous avons définie dans la Section 4.2. Étant donné un ensemble de formules atomiques X de ECTL*(WIPC*), nous considérons les ressources syntaxiques suivantes.

- l est le plus grand entier tel qu'il existe un terme de la forme $\mathsf{X}^l x$ dans X.
- V est l'ensemble des variables utilisées dans X.
- Term est l'ensemble de termes $\{\mathsf{X}^i x : x \in V \text{ et } i \in \{0, \ldots, l\}\}$,
- C est l'ensemble des constantes utilisées dans X. Nous notons m et M respectivement le plus petit et plus grand élément de C.
- K est le plus petit commun multiple des entiers k tels qu'il existe une contrainte de périodicité utilisant la relation $\equiv_k$ dans X.

Sans perte de généralité, nous supposons que $K > 0$ et pour tout $m < d < M$ on a $d \in C$. Nous définissons la relation d'ordre $\preceq_X$ sur les valuations de la forme Term $\rightarrow \mathbb{Z}$ telle que $v \preceq_X v'$ ssi

- l'ordre des différents éléments est préservé,
  pour tout $a, b \in$ Term on a $v(a) \geq v(b)$ ssi $v'(a) \geq v'(b)$,
  pour tout $a \in$ Term et $d \in C$ on a $v(a) \geq d$ ssi $v'(a) \geq d$, et $d \geq v(a)$ ssi $d \geq v'(a)$
- les relations de congruence par rapport à K sont préservées,
  pour tout $a \in$ Term on a $v(a) \equiv_K v'(a)$,
- les valeurs affectées par v' sont plus "espacées" que celles de v,
  pour tout $a, b \in$ Term si $v(a) \geq v(b)$ alors $v'(a) - v'(b) \geq v(a) - v(b)$,
  pour tout $a \in$ Term et $d \in C$ si $v(a) \geq d$ alors $v'(a) - d \geq v(a) - d$,
  de même si $d \geq v(a)$ alors $d - v'(a) \geq d - v(a)$.

  Notons que ces conditions ne sont pas incompatibles avec le premier point qui impose que si $v(a) = v(b)$ alors $v'(a) = v'(b)$ (idem avec les constantes). En effet, toutes les inégalités sont larges.

D'après cette définition, si $v \preceq_X v'$ alors les propriétés suivantes sont vérifiées.

(O1) Pour tout terme $\mathsf{X}^i x \in$ Term tel que $v(\mathsf{X}^i x) > M$ on a $v'(\mathsf{X}^i x) \geq v(\mathsf{X}^i x)$. En effet, s'il en était autrement on aurait $v'(\mathsf{X}^i x) - M < v(\mathsf{X}^i x) - M$.

(O2) De même pour tout terme $\mathsf{X}^i x \in$ Term tel que $v(\mathsf{X}^i x) < m$ on a $v'(\mathsf{X}^i x) \leq v(\mathsf{X}^i x)$.

(O3) Enfin, pour tout terme $\mathsf{X}^i x \in$ Term tel qu'il existe $d \in C$ vérifiant $v(\mathsf{X}^i x) = d$ on a $v'(\mathsf{X}^i x) = v(\mathsf{X}^i x)$.

Pour résumer, les termes ayant une valeur supérieure à toutes les constantes croissent, les termes ayant une valeur inférieure aux constantes décroissent, et les termes qui sont égaux à une constante ne changent pas. De plus, d'après l'abstraction des valuations définie dans la Section 4.2, si $v \preceq_X v'$ alors $sv(v) = sv(v')$. Ceci est une conséquence de la préservation de l'ordre des valeurs affectées aux éléments de $\mathrm{Term} \uplus C$ et des relations de congruence modulo K. Nous pouvons donc établir le résultat suivant qui découle du Lemme 10 (II).

Lemme 56 *Pour toute paire de valuations v, v' de la forme* $\mathrm{Term} \to \mathbb{Z}$ *telle que $v \preceq_X v'$ et toute contrainte atomique α de* WIPC* *appartenant à X, on a $v \models \alpha$ ssi $v' \models \alpha$.*

Étant donnés une formule ϕ de ECTL*(WIPC*) et un WIPC*-automate $\mathcal{A}$, nous notons $\preceq_{\phi,\mathcal{A}}$ l'ordre construit par rapport à l'ensemble des contraintes atomiques de ECTL*(WIPC*) utilisées dans ϕ et dans la relation de transition de $\mathcal{A}$. Par extension, étant donnés deux chemins finis π et π' de longueur l nous notons $\pi \preceq_{\phi,\mathcal{A}} \pi'$ ssi $v_\pi \preceq_{\phi,\mathcal{A}} v_{\pi'}$ et pour tout $i \in \{0, \ldots, l\}$ l'état de contrôle à la position i est le même dans les deux chemins. Nous rappelons que pour tout chemin fini $\pi = \langle q_0, v_0 \rangle \cdot \langle q_1, v_1 \rangle \cdots$ dans $\mathcal{A}$, la valuation v_π est définie par $v_\pi(\mathsf{X}^i x) = v_i(x)$ pour tout $i \in \{0, \ldots, |\pi|\}$ et $x \in V$. Lorsque le contexte est clair, nous notons parfois la relation $\preceq_X$ simplement $\preceq$. L'ordre $\preceq_X$ vérifie la propriété suivante qui est une conséquence du même résultat pour l'ordre sur les n-uplets d'entiers (voir [Pet81]).

Lemme 57 *Pour tout ensemble X de formules atomiques de* ECTL*(WIPC*)*, l'ordre $\preceq_X$ est un beau préordre antisymétrique sur l'ensemble des valuations de la forme* $\mathrm{Term} \to \mathbb{Z}$*.*

Preuve : Pour toute valuation $v : \mathrm{Term} \to \mathbb{Z}$ nous définissons deux vecteurs :

- v_{mod} est le vecteur de dimension $|\mathrm{Term}|$ indexé par des termes tel que pour tout $\mathsf{X}^i x \in \mathrm{Term}$ on a $0 \leq v_{\mathrm{mod}}.(\mathsf{X}^i x) < K$ et $v_{\mathrm{mod}}.(\mathsf{X}^i x) \equiv_K v(\mathsf{X}^i x)$,
- v_{dif} est le vecteur de dimension $|\mathrm{Term} \times \mathrm{Term}| + |\mathrm{Term} \times C|$ indexé par des couples formés d'éléments de $\mathrm{Term} \cup C$ tel que pour tout $a \in \mathrm{Term}$ et $b \in \mathrm{Term} \cup C$ on a $v_{\mathrm{dif}}.(a, b) = |v(a) - v(b)|$.

Le vecteur v_{mod} représente les modulo des différentes valeurs affectées aux variables et v_{dif} les écarts entres ces valeurs. Par définition de $\preceq_X$, pour tout couple de valuations v, v' de la forme $\mathrm{Term} \to \mathbb{Z}$ on a $v \preceq Xv'$ ssi $v_{\mathrm{mod}} = v'_{\mathrm{mod}}$ et $v_{\mathrm{mod}} \preceq_N v'_{\mathrm{mod}}$ tel que $\preceq_N$ est l'ordre tel que :

$$v_{\mathrm{mod}} \preceq_N v'_{\mathrm{mod}} \text{ ssi pour tout } a \in \mathrm{Term} \text{ et } b \in \mathrm{Term} \cup C \text{ on a } v_{\mathrm{dif}}.(a, b) \leq v'_{\mathrm{dif}}.(a, b).$$

Nous pouvons alors facilement prouver que $\preceq_X$ est un beau préordre en utilisant les observations suivantes.

- le nombre de façons d'ordonner les éléments de l'ensemble fini $\mathrm{Term} \cup C$ est fini,
- le nombre de vecteurs distincts vérifiant la définition de v_{mod} est fini,
- $\preceq_N$ est un beau préordre sur les n-uplets d'entiers. □

Nous utilisons parfois la restriction de l'ordre $\preceq_X$ par rapport à un sous-ensemble de Term. Étant données deux valuations v, v' de la forme $\mathrm{Term}' \to \mathbb{Z}$ telles que $\mathrm{Term}' \subseteq \mathrm{Term}$ nous notons aussi $v \preceq_X v'$ si toutes les conditions requises par la définition de $\preceq_X$ sont vérifiées lorsque l'on considère seulement les éléments de Term' et C. Ceci nous permet entre autre d'utiliser la relation d'ordre sur les valuations de la forme $V \to \mathbb{Z}$ et par extension sur des états d'un WIPC*-automate. Par exemple, nous notons $\langle q, v\rangle \preceq_{\phi,\mathcal{A}} \langle q', v'\rangle$ ssi $q = q'$ et $v \preceq_{\phi,\mathcal{A}} v'$ par rapport à la restriction de $\preceq_{\phi,\mathcal{A}}$ aux valuations de la forme $V \to \mathbb{Z}$. Notons que pour tout ensemble $\mathrm{Term}' \subseteq \mathrm{Term}$ la restriction de la relation $\preceq_X$ est un beau préordre sur l'ensemble des valuations de la forme $\mathrm{Term}' \to \mathbb{Z}$. Une autre propriété de cette restriction est que pour toute paire de valuations v_1, v_2 de la forme $\mathrm{Term} \to \mathbb{Z}$ telle que $v_1 \preceq v_2$, les restrictions v'_1, v'_2 de v_1 et v_2 définies sur le même sous-ensemble de termes vérifient $v'_1 \preceq v'_2$ puisque dans l'ordre, les relations de périodicité et les écarts entres les valeurs affectées dans la restriction des valuations sont inchangés.

Nous terminons cette section en montrant que pour toute formule ϕ de ECTL*(WIPC*) et tout WIPC*-automate $\mathcal{A}$, l'ensemble $[\![\phi]\!]_{\mathcal{A}}$ est un ensemble fermé par le haut par rapport au beau préordre $\preceq_{\phi,\mathcal{A}}$. Nous établissons d'abord que $\mathcal{A}$ est un système bien structuré. Un système de transitions $S = \langle N, \to\rangle$ est *bien structuré* s'il existe un beau préordre $\preceq$ sur l'ensemble des sommets N tel que pour toute transition $n_1 \to n_2$ dans ce système et tout état $n'_1 \in N$ tel que $n_1 \preceq n'_1$ il existe un état $n'_2 \in N$ tel que $n_2 \preceq n'_2$ et $n'_1 \to n'_2$. La preuve que les WIPC*-automates sont des systèmes bien structurés découle d'un résultat de simulation plus fort énoncé ci-dessous.

Lemme 58 *Soit $\mathcal{A}$ un* WIPC**-automate, ϕ une formule de* CLTL(WIPC*) *et $\pi = \langle q_0, v_0\rangle \cdot \langle q_1, v_1\rangle \cdots$ un chemin infini dans $\mathcal{A}$. Pour toute valuation v'_0 telle que $v_0 \preceq_{\phi,\mathcal{A}} v'_0$, il existe un chemin infini $\pi' = \langle q_0, v'_0\rangle \cdot \langle q_1, v'_1\rangle \cdots$ tel que pour tout $i \geq 0$ on a $\pi(i)\cdots\pi(i+l) \preceq_{\phi,\mathcal{A}} \pi'(i)\cdots\pi'(i+l)$.*

Preuve : Soit $\langle q_0, v_0\rangle \xrightarrow{\alpha_0} \langle q_1, v_1\rangle \xrightarrow{\alpha_1} \cdots$ les transitions successives prises par π dans $\mathcal{A}$. D'après le Lemme 56, pour tout $i \in \mathbb{N}$ et toute valuation v' telle que $v_{\pi(i)\cdots\pi(i+l)} \preceq v'$ on a $v_{\pi(i)\cdots\pi(i+l)} \models \alpha_i$ ssi $v' \models \alpha_i$. Il nous suffit donc de montrer qu'à partir de v'_0 on peut construire une valuation v'_i pour chaque position $i > 0$ de π' tel que pour tout $i \geq 0$ on a $v_{\pi(i)\cdots\pi(i+l)} \preceq v_{\pi'(i)\cdots\pi'(i+l)}$. En effet, le chemin $\pi' = \langle q_0, v'_0\rangle \xrightarrow{\alpha_0} \langle q_1, v'_1\rangle \xrightarrow{\alpha_1} \cdots$ qui emprunte les même transitions que π est valide d'après la remarque ci-dessus.

Nous procédons par induction sur la position dans la séquence. Nous commençons donc par construire $l-1$ valuations $v'(1), \cdots, v'(l)$ telles que si $\pi'(i) = \langle q_i, v'_i\rangle$ pour tout $i \in \{1, \ldots, l\}$ alors $v_{\pi(0)\cdots\pi(l)} \preceq v_{\pi'(0)\cdots\pi'(l)}$. Comme la valuation v'_0 est déjà fixée, il nous faut donc compléter la valuation $v_{\pi'(0)\cdots\pi'(l)}$ de manière à ce qu'elle vérifie la propriété. Pour simplifier la présentation, nous notons $v = v_{\pi(0)\cdots\pi(l)}$, $v' = v_{\pi'(0)\cdots\pi'(l)}$ et conformément à la définition nous utilisons $v(\mathsf{X}^i x)$ plutôt que $v_i(x)$.

Nous construisons v' par induction. Pour le cas de base, on sait que les restrictions de v et v' aux éléments de V sont définies et vérifient la relation $\preceq$ puisque $v_0 \preceq v'_0$. Nous supposons maintenant que nous pouvons construire une affectation partielle v'_p pour un sous-ensemble de termes Term' vérifiant $v'_p(x) = v'_0(x)$ pour tout $x \in V$ et $v_p \preceq v'_p$ où v_p est la restriction de v aux éléments de Term'. Nous montrons qu'étant donné un terme $\mathsf{X}^i x \in \mathrm{Term} \setminus \mathrm{Term}'$, on peut définir une valeur $v'(\mathsf{X}^i x)$ telle que la valuation v'_p étendue avec la valeur définie pour $\mathsf{X}^i x$ vérifie la relation $\preceq$ par rapport à la restriction de v aux éléments de $\mathrm{Term}' \uplus \{\mathsf{X}^i x\}$.

- Si $v(\mathsf{X}^i x) > M$, nous considérons le terme de Term′ qui prend la plus grande valeur inférieure à $v(\mathsf{X}^i x)$ dans v, c'est-à-dire $\mathsf{X}^j y \in \text{Term}'$ tel que $v(\mathsf{X}^i y) = \max\{v_p(\mathsf{X}^k z) \mid v_p(\mathsf{X}^k z) \leq v(\mathsf{X}^i x)\}$. Si $v(\mathsf{X}^i y) \leq M$ alors nous posons $v'(\mathsf{X}^i x) = v(\mathsf{X}^i x)$, sinon

$$v'(\mathsf{X}^i x) = v(\mathsf{X}^i x) + (v'_p(\mathsf{X}^j y) - v_p(\mathsf{X}^j y)).$$

Nous montrons maintenant que la valuation v'_p étendue avec la valeur définie pour $\mathsf{X}^i x$ vérifie la relation $\preceq$ par rapport à la restriction correspondante de v. Nous commençons par le cas où $v(\mathsf{X}^i y) > M$.

– Pour montrer que les relations de congruences par rapport à K sont préservées, il suffit de remarquer que par définition $v'(\mathsf{X}^i x) - v'_p(\mathsf{X}^j y) = v(\mathsf{X}^i x) - v_p(\mathsf{X}^j y)$ et donc $v'(\mathsf{X}^i x) - v'_p(\mathsf{X}^j y) \equiv_K v(\mathsf{X}^i x) - v_p(\mathsf{X}^j y)$. Comme $v_p \preceq v'_p$ on a $v'_p(\mathsf{X}^j y) \equiv_K v_p(\mathsf{X}^j y)$, ce qui permet de déduire que $v'(\mathsf{X}^i x) \equiv_K v(\mathsf{X}^i x)$.

– Nous montrons maintenant que l'ordre des valeurs est préservé et que les écarts ne diminuent pas.

  Comme $v'(\mathsf{X}^i x) \geq v(\mathsf{X}^i x)$, cette propriété est directe pour tout terme $\mathsf{X}^k z \in \text{Term}'$ tel que $v_p(\mathsf{X}^k z) \leq M$ et toute constante $d \in C$ puisqu'on a $v_p(\mathsf{X}^k z) \geq v'_p(\mathsf{X}^k z)$ et que la valeur de d ne change pas.

  Considérons donc les termes de la forme $\mathsf{X}^k z \in \text{Term}'$ tels que $v_p(\mathsf{X}^k z) > M$. Si $v_p(\mathsf{X}^j y) \geq v_p(\mathsf{X}^k z)$ alors on obtient par transitivité que $v(\mathsf{X}^i x) \geq v_p(\mathsf{X}^k z)$. Comme $v_p \preceq v'_p$ on a aussi $v'_p(\mathsf{X}^j y) \geq v'_p(\mathsf{X}^k z)$ et $v_p(\mathsf{X}^j y) - v_p(\mathsf{X}^k z) \leq v'_p(\mathsf{X}^j y) - v'_p(\mathsf{X}^k z)$. Par transitivité, on obtient directement $v'(\mathsf{X}^i x) \geq v'_p(\mathsf{X}^k z)$. De plus $v_p(\mathsf{X}^j y) - v_p(\mathsf{X}^k z) \leq v'_p(\mathsf{X}^j y) - v'_p(\mathsf{X}^k z)$ implique que $v'_p(\mathsf{X}^k z) - v_p(\mathsf{X}^k z) \leq v'_p(\mathsf{X}^j y) - v_p(\mathsf{X}^j y)$ et comme par définition $v'(\mathsf{X}^i x) - v(\mathsf{X}^i x) = v'_p(\mathsf{X}^j y) - v_p(\mathsf{X}^j y)$ on a $v'_p(\mathsf{X}^k z) - v_p(\mathsf{X}^k z) \leq v'(\mathsf{X}^i x) - v(\mathsf{X}^i x)$. Donc on a bien $v(\mathsf{X}^i x) - v_p(\mathsf{X}^k z) \leq v'(\mathsf{X}^i x) - v'_p(\mathsf{X}^k z)$.

  Si $v(\mathsf{X}^i x) \geq v_p(\mathsf{X}^k z) \geq v_p(\mathsf{X}^j y)$ alors on a $v_p(\mathsf{X}^k z) = v_p(\mathsf{X}^j y)$ car par définition $v_p(\mathsf{X}^j y)$ est la plus grande valeur inférieure à $v(\mathsf{X}^i x)$. Comme $v_p \preceq v'_p$ on a aussi $v'_p(\mathsf{X}^j y) = v'_p(\mathsf{X}^k z)$. La propriété et donc respectée puisque par définition $v'(\mathsf{X}^i x) \geq v'_p(\mathsf{X}^j y)$ et $v'(\mathsf{X}^i x) - v(\mathsf{X}^i x) = v'_p(\mathsf{X}^j y) - v_p(\mathsf{X}^j y)$.

  Enfin si $v_p(\mathsf{X}^k z) \geq v(\mathsf{X}^i x) \geq v_p(\mathsf{X}^j y)$ alors en utilisant $v_p \preceq v'_p$ on a $v'_p(\mathsf{X}^k z) - v'_p(\mathsf{X}^j y) \geq v_p(\mathsf{X}^k z) - v_p(\mathsf{X}^j y)$ ce qui nous permet de déduire $v'_p(\mathsf{X}^k z) - v_p(\mathsf{X}^k z) \geq v'_p(\mathsf{X}^j y) - v_p(\mathsf{X}^j y)$. Comme $v'(\mathsf{X}^i x) - v(\mathsf{X}^i x) = v'_p(\mathsf{X}^j y) - v_p(\mathsf{X}^j y)$ on a donc $v'_p(\mathsf{X}^k z) - v_p(\mathsf{X}^k z) \geq v'(\mathsf{X}^i x) - v(\mathsf{X}^i x)$. Ceci nous permet de déduire que $v'_p(\mathsf{X}^k z) - v'(\mathsf{X}^i x) \geq v_p(\mathsf{X}^k z) - v(\mathsf{X}^i x)$ et que $v'_p(\mathsf{X}^k z) \geq v'(\mathsf{X}^i x)$ car $v_p(\mathsf{X}^k z) \geq v(\mathsf{X}^i x)$.

Si $v(\mathsf{X}^i y) \leq M$ alors pour tout terme $\mathsf{X}^k z \in \text{Term}'$ on a soit $v_p(\mathsf{X}^k z) \leq M < v(\mathsf{X}^i x)$ ou $M < v(\mathsf{X}^i x) \leq v_p(\mathsf{X}^k z)$. Dans le premier cas, on a $v'_p(\mathsf{X}^k z) \leq v_p(\mathsf{X}^k z)$ par les propriétés **(O2)** et **(O3)** de $\preceq$. Donc on a $v'_p(\mathsf{X}^k z) < v'(\mathsf{X}^i x)$ et $v(\mathsf{X}^i x) - v_p(\mathsf{X}^k z) \leq v'(\mathsf{X}^i x) - v'_p(\mathsf{X}^k z)$ car $v'(\mathsf{X}^i x) = v(\mathsf{X}^i x)$. La preuve pour les constantes est similaire à ce cas car pour tout $d \in C$ on a $d \leq M < v(\mathsf{X}^i x)$. Dans le deuxième cas, d'après **(O1)** on a $v_p(\mathsf{X}^k z) \leq v'_p(\mathsf{X}^k z)$. On obtient donc $v'(\mathsf{X}^i x) < v'_p(\mathsf{X}^k z)$ et $v_p(\mathsf{X}^k z) - v(\mathsf{X}^i x) \leq v'_p(\mathsf{X}^k z) - v'(\mathsf{X}^i x)$. La relation de périodicité par rapport à K est aussi préservée car $v'(\mathsf{X}^i x) = v(\mathsf{X}^i x)$.

- Si $v(\mathsf{X}^i x) < m$, la démonstration et similaire en considérant le terme $\mathsf{X}^j y \in \text{Term}'$ tel que $v(\mathsf{X}^j y) = \min\{v_p(\mathsf{X}^k z) \mid v_p(\mathsf{X}^k z) \geq v(\mathsf{X}^i x)\}$ et en posant $v'(\mathsf{X}^i x) = v(\mathsf{X}^i x) + (v'_p(\mathsf{X}^j y) - v_p(\mathsf{X}^j y))$ ou $v'(\mathsf{X}^i x) = v(\mathsf{X}^i x)$ si $v(\mathsf{X}^j y) \geq m$.

• Enfin, si $m \leq v(\mathsf{X}^i x) \leq M$ alors il existe $d \in C$ tel que $v(\mathsf{X}^i x) = d$. Pour satisfaire l'ordre $\preceq$, nous devons poser $v'(\mathsf{X}^i x) = d$ afin de préserver l'ordre entre les constantes et les termes. Les conditions requises sont vérifiées puisque elles le sont déjà dans v'_p par rapport à la constante d.

Par induction, la valuation v' construite en suivant cette méthode est telle que $v \preceq v'$.

Supposons maintenant que la propriété est vraie jusqu'à la position i. Nous montrons que l'on peut étendre la construction à la position $i+1$. Par définition de la restriction de $\preceq$, si $v_{\pi(i)\cdots\pi(i+l)} \preceq v_{\pi'(i)\cdots\pi'(i+l)}$ alors $v_{\pi(i+1)\cdots\pi(i+l)} \preceq v_{\pi'(i+1)\cdots\pi'(i+l)}$. Il faut donc à nouveau compléter la valuation $v_{\pi'(i)\cdots\pi'(i+l)}$ pour qu'elle vérifie la propriété. La même construction que pour le cas initial peut être réutilisée dans ce but. $\square$

Nous pouvons donc déduire que les WIPC*-automates sont des systèmes bien structurés. En effet dans le résultat précédent, le fait que pour tout $i \in \mathbb{N}$ on ait $\pi(i)\cdots\pi(i+l) \preceq \pi'(i)\cdots\pi'(i+l)$ implique en particulier que $\pi(i) \preceq \pi'(i)$ d'après la définition de la restriction de l'ordre $\preceq$.

Corollaire 15 *Les* WIPC**-automates sont des systèmes bien structurés.*

Ceci nous permet enfin de prouver le résultat attendu.

Lemme 59 *Étant donnés une formule ϕ de* ECTL*(WIPC*) *et un* IPC**-automate $\mathcal{A}$, l'ensemble $[\![\phi]\!]_\mathcal{A}$ est fermé par le haut par rapport à la relation $\preceq_{\phi,\mathcal{A}}$.*

Preuve : Nous procédons par induction sur la structure de ϕ. Si $\phi = \top$ alors la preuve est triviale puisque $[\![\phi]\!]_\mathcal{A}$ est l'ensemble des états de $\mathcal{A}$ qui est bien entendu fermé par le haut. Supposons maintenant que **(H1)** pour toute sous-formule d'état ϕ' de ϕ l'ensemble $[\![\phi']\!]_\mathcal{A}$ est fermé par le haut. Le cas des connecteurs booléens est direct en utilisant cette hypothèse d'induction.

Il reste le cas où ϕ est de la forme $E\psi$. Dans ce cas, si un état $\langle q_0, v_0\rangle$ satisfait ϕ alors il existe un chemin infini π à partir de la configuration $\langle q_0, v_0\rangle$ qui satisfait ψ. D'après le Lemme 58, pour tout état $\langle q_0, v'_0\rangle$ tel que $v_0 \preceq v'_0$ on peut construire un chemin infini π' ayant pour origine $\langle q_0, v'_0\rangle$ tel que pour tout $i \geq 0$ on a $\pi(i)\cdots\pi(i+l) \preceq \pi'(i)\cdots\pi'(i+l)$. Nous montrons par induction sur la structure de ψ que pour toute position $i \in \mathbb{N}$ et toute sous formule ψ' de ψ si π^i satisfait ψ' alors π'^i satisfait aussi ψ'.

– Si ψ' est une contrainte atomique, alors d'après le Lemme 56 la propriété est vérifiée puisque pour tout $i \geq 0$ on a $v_{\pi(i)\cdots\pi(i+l)} \preceq v_{\pi'(i)\cdots\pi'(i+l)}$.

– Si ψ' est une formule d'état alors nous pouvons conclure en utilisant **(H1)**. En effet pour toute position $i \in \mathbb{N}$, $\pi(i)\cdots\pi(i+l) \preceq \pi'(i)\cdots\pi'(i+l)$ implique en particulier $\pi(i) \preceq \pi'(i)$ et d'après **(H1)** $[\![\psi']\!]_\mathcal{A}$ est fermé par le haut.

– Supposons maintenant que **(H2)** pour toute sous formule de chemin de ψ est toute position $i \in \mathbb{N}$, si π^i satisfait ψ' alors π'^i satisfait ψ'.

  Les cas des opérateurs booléens $\wedge$ et $\vee$ sont directs par induction.

  Les cas des opérateurs temporels ne sont pas beaucoup plus compliqués puisqu'à chaque position, le même ensemble de sous-formules est vérifié.

Ceci établit donc que pour toute position i et toute sous-formule ψ' de ψ si π^i satisfait ψ' alors π'^i satisfait aussi ψ'. En particulier, pour la position 0, nous obtenons que si π satisfait ψ alors π' satisfait ψ. Ceci nous permet de conclure que $\langle q_0, v'_0 \rangle$ satisfait ϕ. $\square$

8.2.3 WIPC*-automates généralisés

Pour des raisons techniques, nous avons besoin par la suite de considérer une définition étendue des WIPC*-automates. Un WIPC*-automate généralisé est un WIPC*-automate dont les transitions peuvent être étiquetées par une combinaison booléenne de formules atomiques de CLTL(WIPC*) avec des termes quelconques et non plus uniquement des termes de la forme x ou $\mathsf{X}x$. Formellement la relation de transition d'un WIPC* automate généralisé est donc un sous-ensemble de $Q \times X \times Q$ où Q est l'ensemble des états de contrôle de l'automate et X est un ensemble fini de contraintes de CLTL(WIPC*).

La sémantique de ce modèle pour les exécutions infinies ne change pas. Il est cependant difficile de définir une transition sur un pas puisque les gardes peuvent porter sur plus d'états futurs. Nous n'avons de toute façon pas besoin de cette définition pour la suite. Nous rappelons donc juste qu'une exécution infinie est une séquence $\pi = \langle q_0, v_0 \rangle \xrightarrow{\alpha_0} \langle q_1, v_1 \rangle \xrightarrow{\alpha_1} \langle q_2, v_2 \rangle \xrightarrow{\alpha_2} \cdots$ telle que pour tout $i \geq 0$ la valuation v_{π^i} satisfait α_i. La condition d'acceptation est toujours une condition de Büchi.

La construction du beau préordre $\preceq_{\phi,\mathcal{A}}$ ne pose pas de problèmes particuliers puisque les contraintes sur les transitions d'un WIPC*-automate généralisé sont des contraintes atomiques de ECTL*(IPC*). De plus, comme les principaux arguments utilisés dans la preuve du Lemme 58 sont toujours valables pour les WIPC*-automates généralisés, celui-ci peut être étendu.

Lemme 60 *Soit $\mathcal{A}$ un* WIPC**-automate généralisé, ϕ une formule de* CLTL(WIPC*) *et $\pi = \langle q_0, v_0 \rangle \cdot \langle q_1, v_1 \rangle \cdots$ un chemin infini dans $\mathcal{A}$. Pour toute valuation v'_0 telle que $v_0 \preceq_{\phi,\mathcal{A}} v'_0$, il existe un chemin infini $\pi' = \langle q_0, v'_0 \rangle \cdot \langle q_1, v'_1 \rangle \cdots$ tel que pour tout $i \geq 0$ on a $\pi(i) \cdots \pi(i+l) \preceq_{\phi,\mathcal{A}} \pi'(i) \cdots \pi'(i+l)$.*

La même preuve que pour le Lemme 58 peut être utilisée. En effet, le Lemme 56 s'applique aussi dans le cas des gardes d'un WIPC*-automate généralisé par construction de l'ordre $\preceq_{\phi,\mathcal{A}}$. Nous pouvons alors procéder de la même manière que dans le cas non généralisé en utilisant la méthode permettant de compléter les valuations successives.

8.3 Systèmes d'inégalités

Nous avons souligné dans la section précédente le rapport entre la relation $\preceq$ et l'abstraction des valuations de la Section 4.2 en faisant remarquer que si $v \preceq v'$ alors $sv(v) = sv(v')$. Cependant l'implication inverse n'est pas vraie. Considérons par exemple les valuation $v = \langle x \leftarrow 1, y \leftarrow 3, z \leftarrow 7\rangle$ et $v' = \langle x \leftarrow 1, y \leftarrow 5, z \leftarrow 7\rangle$ et leurs abstractions par rapport aux ressources syntaxiques $l = 0$, $V = \{x, y, z\}$, $C = \emptyset$, et $K = 2$. On peut vérifier que $sv(v) = sv(v')$ par rapport à la définition de $\mathrm{SV}(X)$ dans la Section 4.2 puisque les deux valuations satisfont l'ordre $x < y < z$ et les contraintes de périodicité $x \equiv_2 y \equiv_2 z \equiv_2 1$. Cependant on n'a ni $v \preceq v'$ ni $v' \preceq v$ car $v(z)-v(y) > v'(z)-v'(y)$ et $v'(y)-v'(x) > v(y)-v(x)$. Nous avons donc besoin de définir une nouvelle représentation symbolique pour les ensembles de valuations fermés par le haut qui étend la représentation symbolique introduite dans le cas linéaire. La principale information que nous devons ajouter est une information quantitative sur l'écart entre les valeurs assignées par la valuation.

Étant donnés un ensemble de variables V_{S}, un ensemble de constantes C_{S} et un entier $K_{\mathrm{S}} > 0$, un système d'inégalités est une paire d'ensemble de contraintes $\mathrm{S} = \langle X_{\mathrm{dif}}, X_{\mathrm{mod}}\rangle$ telle que

- X_{dif} est un ensemble de contraintes différentielles de la forme $a - b \geq d$ tel que
  - $a \in V_{\mathrm{S}}$ et $b \in V_{\mathrm{S}} \uplus C_{\mathrm{S}}$ ou $a \in V_{\mathrm{S}} \uplus C_{\mathrm{S}}$ et $b \in V_{\mathrm{S}}$ (si $a, b \in C_{\mathrm{S}}$ la contrainte est inutile),
  - et $d \in \mathbb{N}$.
- X_{mod} est un ensemble de contraintes de périodicité tel que pour tout $x \in V_{\mathrm{S}}$ il existe exactement une contrainte de la forme $x \equiv_{K_{\mathrm{S}}} c$ où $c \in \{0, \ldots, K_{\mathrm{S}} - 1\}$.

Notons que la définition de X_{mod} est plus contraignante car on a exactement une contrainte par variable. Un système d'inégalités est cohérent ssi il existe une valuation $v : V_{\mathrm{S}} \to \mathbb{Z}$ telle que $v \models X_{\mathrm{dif}} \cup X_{\mathrm{mod}}$. Cette définition n'est pas minimale dans le sens où certaines informations peuvent être obsolètes. Par exemple, si l'ensemble de contraintes $X_{\mathrm{dif}} = \{x - y > 3, z - x > 2, z - y > 0\}$ fait partie d'un système d'inégalités S alors la contrainte $z - y > 0$ peut être remplacée par $z - y > 5$ sans affecter l'ensemble des valuations qui satisfont cet ensemble. Pour résoudre ce problème, nous définissons une forme normale pour les systèmes d'inégalités. Un système d'inégalités $\langle X_{\mathrm{dif}}, X_{\mathrm{mod}}\rangle$ est sous forme normale ssi

(NF1) pour chaque couple $a, b \in V_{\mathrm{S}} \cup C_{\mathrm{S}}$ il existe au plus une contrainte de la forme $a - b \geq d$ dans X_{dif},

(NF2) pour tout $a, b, c \in V_{\mathrm{S}} \cup C_{\mathrm{S}}$,

- si les contraintes $a - c \geq d_1$ et $c - b \geq d_2$ appartiennent à X_{dif} avec $a \in V_{\mathrm{S}}$ ou $b \in V_{\mathrm{S}}$,
- ou si $a - c = d_1$ et $c - b \geq d_2$ appartient à X_{dif} avec $a, c \in C_{\mathrm{S}}$ et $b \in V_{\mathrm{S}}$,
- ou encore si $a - c \geq d_1$ appartient à X_{dif} et $c - b = d_2$ avec $a \in V_{\mathrm{S}}$ et $b, c \in C_{\mathrm{S}}$,

alors il existe une contrainte dans X_{dif} de la forme $a - b \geq d$ telle que $d \geq d_1 + d_2$, (ces conditions expriment la transitivité)

(NF3) pour tout $a, b \in V_{\mathrm{S}} \cup C_{\mathrm{S}}$ tels qu'il existe une contrainte de la forme $a-b \geq d$ dans X_{dif}

- si $a, b \in V_{\mathrm{S}}$ alors il existe une contrainte $a \equiv_{K_{\mathrm{S}}} c_a$ appartenant à X_{mod} ssi il existe une contrainte $b \equiv_{K_{\mathrm{S}}} c_b$ appartenant à X_{mod} avec $c_a \equiv_{K_{\mathrm{S}}} c_b + d$.
- si $a \in C_{\mathrm{S}}$ alors il existe une contrainte $b \equiv_{K_{\mathrm{S}}} c_b$ dans X_{mod} telle que $a \equiv_{K_{\mathrm{S}}} c_b + d$.
- si $b \in C_{\mathrm{S}}$ alors il existe une contrainte $a \equiv_{K_{\mathrm{S}}} c_a$ telle que X_{mod} avec $c_a \equiv_{K_{\mathrm{S}}} b + d$.

(ces conditions expriment la compatibilité des contraintes de X_{mod} avec celles de X_{dif}).

L'ensemble de ces conditions exprime qu'aucune contrainte ne peut être ajoutée ou raffinée (dans le sens de l'exemple précédent) sans modifier l'ensemble des valuations qui satisfont le système.

De la même manière que pour les valuations symboliques, un système d'inégalités peut facilement être représenté par un graphe avec un étiquetage des sommets. Étant donné un système d'inégalités S, le graphe $G_{\mathrm{S}} = \langle N, \rightarrow, mod\rangle$ est défini par

- $N = \mathrm{Term} \uplus C$,
- pour toute contrainte de la forme $a - b \geq d$ dans S, il existe un arc entre a et b dont le poids est d que nous notons $b \xrightarrow{d} a$,
- pour toute constantes $d, d' \in C_{\mathrm{S}}$ telles que $d > d'$, il existe un arc $d' \xrightarrow{d-d'} d$,
- pour toute contrainte de la forme $a \equiv_K c$ dans S, le sommet a est étiqueté par la valeur $mod(a) = c$,
- pour toute constante d dans C_{S} le sommet d est étiqueté par la valeur $mod(d) = c$ telle que $c \equiv_{K_{\mathrm{S}}} d$ et $0 \leq c \leq K_{\mathrm{S}} - 1$.

Nous appelons le poids d'un chemin dans ce graphe la somme des poids de ces arcs. La Figure 8.1 montre la représentation graphique du système d'inégalités $\mathrm{S} = \langle X_{\mathrm{dif}}, X_{\mathrm{mod}}\rangle$ tel que

$$X_{\mathrm{dif}} = \left\{ \begin{array}{ll} x - \mathsf{X}x \geq 1 & \mathsf{X}x - \mathsf{X}^2x \geq 0, \\ \mathsf{X}x - \mathsf{X}^2x \geq 2 & x - y \geq 0, \\ x - \mathsf{X}y \geq 4 & \mathsf{X}y - y \geq 2, \\ 6 - \mathsf{X}y \geq 0 & \end{array} \right\} \text{ et } X_{\mathrm{mod}} = \left\{ \begin{array}{ll} x \equiv_2 0 & \mathsf{X}x \equiv_2 0, \\ \mathsf{X}^2x \equiv_2 0 & y \equiv_2 1, \\ \mathsf{X}x \equiv_2 0 & \mathsf{X}^2x \equiv_2 1 \end{array} \right\}$$

ainsi que la forme normale correspondant à ce système. Une valuation v satisfaisant S peut être vue comme un étiquetage des sommets de G_{S} tel que

- pour tout arc $n' \xrightarrow{d} n$ on a $v(n) - v(n') \geq d$,
- pour tout sommet n on a $v(n) \equiv_{K_{\mathrm{S}}} mod(n)$.

Lemme 61 *Soit* $\mathrm{S} = \langle X_{\mathrm{dif}}, X_{\mathrm{mod}}\rangle$ *un système d'inégalités.*

(I) *On peut tester si* S *est cohérent en temps polynomial.*

(II) *Si* S *est cohérent alors il existe un système d'inégalités sous forme normale noté* $|\mathrm{S}|$ *tel que pour tout* $v : V_{\mathrm{S}} \rightarrow \mathbb{Z}$ *on a* $v \models \mathrm{S}$ *ssi* $v \models |\mathrm{S}|$.

Preuve : **(I)** La preuve est similaire à celle du Lemme 9 et à [Čer94, Lemme 5.5]. Nous n'avons cependant pas besoin de vérifier que l'ensemble X_{dif} est maximal. Un système d'inégalités S est cohérent ssi le graphe G_{S} vérifie les conditions suivantes.

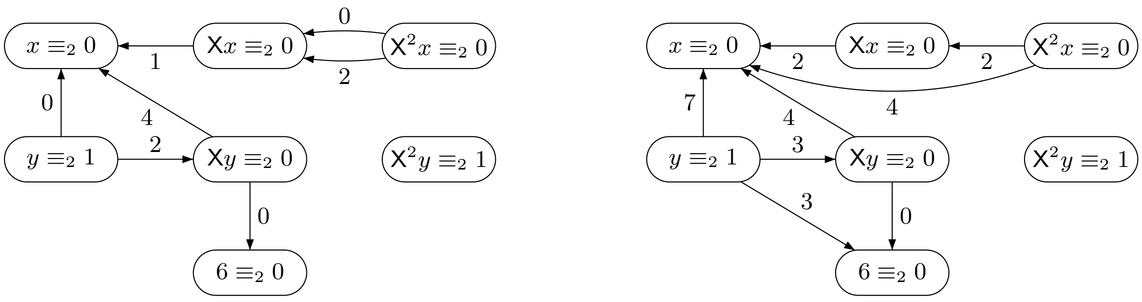


Figure 8.1: Représentation graphique d'un système d'inégalités et de sa forme normale

(SI1) Il n'existe pas de chemin $n_0 \xrightarrow{a_0} n_1 \xrightarrow{a_1} \dots \xrightarrow{a_{|\pi|-1}} n_{|\pi|}$ tel que $n_0 = n_{|\pi|}$ et $\sum_{i=0}^{|\pi|-1} a_i > 0$.

(SI2) Pour tout $d_1, d_2 \in C_{\mathrm{S}}$ tel que $d_1 \leq d_2$, il n'existe pas de chemin $n_0 \xrightarrow{a_0} n_1 \xrightarrow{a_1} \dots \xrightarrow{a_{|\pi|-1}} n_{|\pi|}$ où $n_0 = d_1$ et $n_{|\pi|} = d_2$ tel que $\sum_{i=0}^{|\pi|-1} a_i > d_2 - d_1$.

(SI3) Pour tous sommets n, n' de G_{S} tels que $n \xrightarrow{0} n'$ et $n' \xrightarrow{0} n$, on a $mod(n) \equiv_K mod(n')$.

(II) Nous définissons la normalisation de S par rapport à sa représentation graphique. Nous commençons par supprimer les arcs inutiles. Pour toute paire de sommets n_1, n_2 nous gardons uniquement l'arc ayant le poids maximal entre n_1 et n_2. Nous ajoutons ensuite tous les arcs qui peuvent être obtenus par transitivité ($n_1 \to n_2$ et $n_2 \to n_3$ impliquent $n_1 \to n_3$). Nous affectons pour le moment un poids égal à 0 à ces arcs. Après cette étape, il existe au plus un arc entre chaque paire de sommets du graphe donc le système d'inégalités correspondant satisfait **(NF1)**. Notons que cette transformation laisse l'ensemble des valuations satisfaisant le graphe inchangé.

Nous normalisons ensuite les poids par rapport aux contraintes de périodicité. Pour toute paire de sommets reliés par un arc $n_1 \xrightarrow{a} n_2$, nous affectons à l'arc un nouveau poids a' qui est le plus petit entier vérifiant $a' \geq a$ et $a' \equiv_{K_{\mathrm{S}}} mod(n) - mod(n')$. Cette opération n'ajoute pas d'arc et ne change pas l'ensemble des solutions puisque pour tout entier $a \leq b < a'$, si $n_1 - n_2 = b$ alors l'une des contraintes de périodicité n'est pas vérifiée car $b \not\equiv_{K_{\mathrm{S}}} mod(n) - mod(n')$. Notons que cette partie de la normalisation termine en temps linéaire par rapport à l'ensemble des arcs car comme on ne change pas la fonction mod, chaque arc est traité une seule fois. Lorsque cette opération est terminée, le système d'inégalités correspondant satisfait **(NF1)** et **(NF3)**.

Pour tout arc $n_1 \to n_2$ dans le graphe obtenu, nous mettons enfin à jour le poids en affectant le poids maximal des chemins entre n_1 et n_2. Comme le système d'inégalités est satisfaisable, ce poids est inférieur à ω car il n'existe pas de cycle ayant un poids strictement positif **(SI1)**. Ici encore, cette opération ne modifie pas l'ensemble des solutions. En effet, pour tout chemin entre n_1 et n_2 la différence entre la valeur de n_2 et celle de n_1 doit être supérieure à la somme des différences entre deux sommets consécutifs du chemin (par transitivité). Cette modification ne change pas non plus la satisfaction de la contrainte **(NF3)**. Comme le poids est mis à jour par rapport à un chemin fini π entre n_1 et n_2 et qu'avant la mise à jour le système satisfait **(NF3)**, pour tout $0 \leq i \leq |\pi| - 1$ on a $mod(\pi(i+1)) \equiv_{K_{\mathrm{S}}} mod(\pi(i)) + a_i$ car $\pi(i) \xrightarrow{a_i} \pi(i+1)$. Ceci implique par transitivité que

$mod(n_1) \equiv_{K_S} mod(n_2) + \Sigma_{i=0}^{|\pi|} a_i$ et $\Sigma_{i=0}^{|\pi|} a_i$ est bien le poids de π utilisé pour la mise à jour. Nous réitérons cette opération tant qu'il existe un poids qui peut être mis à jour. Pour s'assurer que cet partie termine, nous pouvons procéder par induction sur la distance maximale en nombre d'arcs entre deux sommets (en identifiant tous les sommets d'un même cycle de poids nul).

Une fois ces opérations effectuées, nous avons bien un système d'inégalités sous forme normale. De plus, l'ensemble des opérations que nous avons effectuées préserve le même ensemble de solutions pour le système obtenu. □

Nous appelons le système d'inégalités $|\mathrm{S}|$ défini ci-dessus la forme normale de S. Nous associons aux systèmes d'inégalités qui ne sont pas cohérents une forme normale particulière que nous notons $\mathrm{S}_{\mathrm{nil}}$ qui n'est pas satisfaisable. Un système d'inégalités sous forme normale vérifie la propriété suivante.

Lemme 62 *Soit* S *un système d'inégalités sous forme normale. Si* $\mathrm{S} \neq \mathrm{S}_{\mathrm{nil}}$ *alors toute affectation d'un sous-ensemble de variables* $V' \subseteq V_\mathrm{S}$ *qui satisfait toutes les contraintes de* S *impliquant uniquement les éléments de* V' *peut être étendu en valuation satisfaisant* S.

Preuve : Supposons que S soit un système d'inégalités sous forme normale et qu'il existe une affectation $v : V' \to \mathbb{Z}$ satisfaisant les conditions de l'énoncé. Nous raisonnons à nouveau par rapport à la représentation graphique de S. De ce point de vue, une telle affectation correspond à un étiquetage partiel des sommets qui vérifie les contraintes de poids sur les arcs et la fonction *mod*.

Soit $x \in V_\mathrm{S} \setminus V'$ une variable qui n'appartient pas au domaine de définition de v. Si nous montrons que l'étiquetage peut s'étendre à x alors par induction la propriété est vérifiée. Soit *val* l'étiquetage partiel des sommets qui affecte $v(n)$ à n si $n \in V'$ et la constante correspondant à n si $n \in C_\mathrm{S}$. Plusieurs cas se présentent :

1. S'il existe un arc entrant en x dans G_S alors nous affectons à x la valeur $v_x = \max\{val(n) + d \mid n \in V' \cup C_\mathrm{S} \text{ et } n \xrightarrow{d} x\}$.
2. Sinon, s'il existe un arc sortant de x dans G_S alors nous affectons à x la valeur $v_x = \min\{val(n) - d \mid n \in V' \cup C_\mathrm{S} \text{ et } x \xrightarrow{d} n\}$.
3. Enfin, s'il n'existe aucun arc ayant pour origine ou destination x, nous affectons à x n'importe quelle valeur v_x qui satisfait $mod(x)$.

Nous montrons maintenant que la valeur v_x affectée à x satisfait les contraintes du système. Nous commençons par la contrainte imposée par $mod(x)$.

– Dans les deux premiers cas, puisque le système d'inégalités est sous forme normale la valeur affectée satisfait la contrainte de périodicité représentée par $mod(x)$ d'après **(NF3)**. En effet, **(NF3)** implique que pour tout $n \in V$, si $n \xrightarrow{d} x$ alors $x \equiv_{K_\mathrm{S}} val(n)+d$ et si $x \xrightarrow{d} n$ alors $val(n) \equiv_{K_\mathrm{S}} x + d$.
– Pour le dernier cas, la satisfaction de la contrainte de périodicité représentée par $mod(x)$ est explicitée dans la définition.

Pour ce qui est des contraintes représentées par les arcs entre x et les sommets qui ont déjà une valeur (c'est-à-dire les sommets de $V' \cup C_{\mathrm{S}}$), nous avons les cas suivants.

– Dans le premier cas, pour tout sommet n tel que $n \xrightarrow{d} x$ on a $v_x \geq val(n) + d$ car par définition $v_x = \max\{val(n) + d \mid n \in V' \cup C_{\mathrm{S}} \text{ et } n \xrightarrow{d} x\}$. Nous pouvons donc déduire que $v_x - val(n) \geq d$ et donc la contrainte est respectée.

  Pour tout sommet n tel que $x \xrightarrow{d} n$ nous procédons de la façon suivante. Par définition, il existe $n' \in V'$ tel que $v_x = val(n') + d'$ et $n' \xrightarrow{d'} x$. Comme le système d'inégalités est sous forme normale, il vérifie **(NF2)** et donc on a $n' \xrightarrow{d+d'} n$. Comme n et n' appartiennent tous les deux à $V' \cup C_{\mathrm{S}}$ nous savons par définition de val que $val(n) - val(n') \geq d+d'$, ce qui nous permet de déduire $val(n) - v_x \geq d$ puisque $val(n') = v_x - d'$.

– Le deuxième cas peut être traité de la même manière. Il est même plus court car dans ce cas il n'existe pas d'arc sortant de x.

– Le cas restant est trivial puisqu'il n'existe pas d'arcs contraignant la valeur v_x. □

Contrairement aux valuations symboliques que nous avons utilisées jusqu'à présent, les systèmes d'inégalités ne forment pas une partition finie de l'ensemble des valuations. Notre but ici n'est pas de définir une abstraction de l'ensemble des modèles ayant pour élément de base un ensemble fini de valuations symboliques, mais un formalisme permettant de représenter de manière finie les ensembles d'états qui satisfont une formule donnée. Pour pouvoir construire une telle représentation, nous définissons des opérations élémentaires qui nous permettent de manipuler plus facilement les systèmes d'inégalités dans la suite.

Considérons deux systèmes d'inégalités $\mathrm{S} = \langle X_{\mathrm{dif}}, X_{\mathrm{mod}} \rangle$ et $\mathrm{S}' = \langle X'_{\mathrm{dif}}, X'_{\mathrm{mod}} \rangle$ *dont les contraintes de périodicité sont construites par rapport au même entier K*. Sans perte de généralité, nous pouvons aussi supposé que les ensembles de variables et de constantes utilisés dans ces deux systèmes d'inégalités sont identiques. Tous les systèmes d'inégalités que nous manipulons par la suite vérifient cette propriété. Nous n'avons donc pas besoin de définition générale des opérations suivantes.

– L'*intersection* $\mathrm{S} \cap \mathrm{S}'$ est la forme normale de $\langle X^{\cap}_{\mathrm{dif}}, X^{\cap}_{\mathrm{mod}} \rangle$ tel que $X^{\cap}_{\mathrm{dif}} = X_{\mathrm{dif}} \cup X'_{\mathrm{dif}}$ et $X^{\cap}_{\mathrm{mod}} = X_{\mathrm{mod}} = X'_{\mathrm{mod}}$. Notons que si $X_{\mathrm{mod}} \neq X'_{\mathrm{mod}}$ alors le système $\mathrm{S} \cap \mathrm{S}'$ est incohérent (donc $\mathrm{S} \cap \mathrm{S}' = \mathrm{S}_{\mathrm{nil}}$).

– La *projection* de S sur un sous-ensemble X de V_{S}, notée $\mathrm{S}_{|X}$, est la restriction de la forme normale de S aux contraintes qui utilisent uniquement des variables de X.

La preuve du résultat suivant est directe en considérant ces définitions ainsi que les résultats qui précèdent.

Lemme 63 *Soit $\mathrm{S} = \langle X^1_{\mathrm{dif}}, X^1_{\mathrm{mod}} \rangle$ et $\mathrm{S}' = \langle X'_{\mathrm{dif}}, X'_{\mathrm{mod}} \rangle$ deux systèmes d'inégalités dont les contraintes de périodicité sont construites par rapport au même entier K.*

1. *Pour toute valuation $v : \mathrm{Term} \to \mathbb{Z}$, on a $v \models \mathrm{S} \cap \mathrm{S}'$ ssi $v \models \mathrm{S}$ et $v \models \mathrm{S}'$*
2. *Pour toute valuation $v : X \to \mathbb{Z}$ où $X \subseteq V_{\mathrm{S}}$ on a $v \models \mathrm{S}_{|X}$ ssi il existe une valuation $v' : V_{\mathrm{S}} \to \mathbb{Z}$ telle que $v' \models \mathrm{S}$ et $v'(a) = v(a)$ pour tout $a \in X$.*

En effet, la normalisation d'un système d'inégalités préserve l'ensemble des solutions d'après le Lemme 61 (II). Pour la projection, la propriété est une conséquence du Lemme 62 qui permet de compléter une valuation satisfaisant une partie de la forme normale de S. Enfin, il est possible de définir un ordre partiel sur les systèmes d'inégalités. Nous définissons l'ordre $\sqsubseteq$ sur les systèmes d'inégalités tel que $\mathrm{S} \sqsubseteq \mathrm{S}'$ ssi

- $V_{\mathrm{S}} = V_{\mathrm{S}'}$, $C_{\mathrm{S}} = C_{\mathrm{S}'}$ et $K_{\mathrm{S}} = K_{\mathrm{S}'}$,
- pour toute contrainte $a - b \geq c$ de X_{dif} il existe une contrainte de la forme $a - b \geq c'$ dans X'_{dif} telle que $c' \geq c$,
- $X_{\mathrm{mod}} = X'_{\mathrm{mod}}$.

Notons que par définition, si $\mathrm{S} \sqsubseteq \mathrm{S}'$ alors toute valuation qui satisfait S' satisfait S. Nous pouvons établir la propriété suivante sur cet ordre.

Lemme 64 *Étant donnés un ensemble de variables V, de constantes C et un entier positif K, l'ordre $\sqsubseteq$ est un beau préordre antisymétrique sur l'ensemble des systèmes d'inégalités construits par rapport à V, C et K.*

La preuve est aussi une conséquence de l'ordre sur les n-uplets d'entiers où $n = |V \uplus C|$. En effet, les contraintes de X_{dif} peuvent être représentées par un vecteur en affectant une position arbitraire à chaque paire d'éléments de $V \uplus C$.

Nous utilisons les systèmes d'inégalités par la suite pour représenter des ensembles d'états ou de transitions d'un WIPC*-automates. Soit $\mathcal{A}$ un WIPC*-automate et ϕ une formule de ECTL*(IPC*). Nous considérons les éléments l, K, V, C et Term définis précédemment. Un système d'inégalités est *local* par rapport à $\mathcal{A}$ et ϕ si $V_{\mathrm{S}} = V$, $C_{\mathrm{S}} = C$ et $K = K_{\mathrm{S}}$. Un ensemble d'états X de $\mathcal{A}$ est *représenté* par une famille d'ensembles finis de systèmes d'inégalités locaux $(\mathcal{S}_q)_{q \in Q}$ ssi on a pour toute configuration $\langle q, v\rangle$ de $\mathcal{A}$

$$\langle q, v\rangle \in X \text{ ssi il existe un système d'inégalités } \mathrm{S} \in \mathcal{S}_q \text{ tel que } v \models \mathrm{S}.$$

Par définition du beau préordre $\preceq_{\phi,\mathcal{A}}$, les systèmes d'inégalités permettent de représenter les ensembles fermés par le haut.

Lemme 65 *Soit U un ensemble d'états d'un* WIPC**-automate $\mathcal{A}$ et ϕ une formule de* ECTL*(WIPC*). *Il existe une représentation de U par une famille d'ensembles finis de systèmes d'inégalités locaux ssi U est un ensemble fermé par le haut par rapport à l'ordre $\preceq_{\phi,\mathcal{A}}$.*

Preuve : Supposons que U soit un ensemble fermé par le haut. Comme la relation $\preceq$ est antisymétrique, il existe d'après le Lemme 55 une base finie $\{\langle q_1, v_1\rangle, \ldots, \langle q_n, v_n\rangle\}$ telle que $U = \bigcup_{i=1}^{n} \uparrow \{\langle q_i, v_i\rangle\}$. Il nous suffit donc de montrer qu'un ensemble de la forme $\uparrow \{\langle q_i, v_i\rangle\}$ peut être représenté par un système d'inégalités. Le système d'inégalités $\mathrm{S}_{\uparrow\{\langle q_i, v_i\rangle\}}$ est défini par

- pour tout $x, y \in V$ tel que $v_i(x) - v_i(y) \geq 0$, la contrainte $x - y \geq d$ telle que $d = v_i(x) - v_i(y)$ appartient à $\mathrm{S}_{\uparrow\{\langle q_i, v_i\rangle\}}$,

- pour tout $x \in V$ et $d \in C$ tel que $v_i(x) - d \geq 0$, la contrainte $x - d \geq d'$ telle que $d' = v_i(x) - d$ appartient à $\mathrm{S}_{\uparrow\{\langle q_i, v_i\rangle\}}$,
- pour tout $x \in V$ et $d \in C$ tel que $d - v_i(x) \geq 0$, la contrainte $d - x \geq d'$ telle que $d' = d - v_i(x)$ appartient à $\mathrm{S}_{\uparrow\{\langle q_i, v_i\rangle\}}$,
- pour tout $x \in V$, la contrainte $x \equiv_K c$ telle que $c \equiv_K v_i(x)$ et $0 \leq c \leq K-1$ appartient à $\mathrm{S}_{\uparrow\{\langle q_i, v_i\rangle\}}$.

Par définition de l'ordre $\preceq$ il est évident que toute valuation v_i' vérifie $v_i \preceq v_i'$ ssi $v_i' \models \mathrm{S}_{\uparrow\{\langle q_i, v_i\rangle\}}$.

Réciproquement, supposons que v satisfait un système d'inégalités S. Pour toute valuation v' telle que $v \preceq v'$ il est facile de vérifier qu'on a bien $v' \models \mathrm{S}$. En effet, la relation d'ordre préserve les contraintes de congruence par rapport à K et l'écart entre les valeurs affectées augmente. En conséquence, si par exemple la contrainte $x - y \geq d$ est satisfaite par v alors comme $v'(x) - v'(y) \geq v(x) - v(y)$ par définition de $\preceq$ on a bien $v' \models x - y \geq d$.

Donc tout ensemble de systèmes d'inégalités locaux représente un ensemble de valuations de la forme $V \rightarrow \mathbb{Z}$ fermé par le haut. Nous pouvons donc en déduire que tout ensemble d'états représenté par une famille d'ensembles de systèmes d'inégalités locaux est un ensemble fermé par le haut. $\square$

8.4 Algorithme symbolique de model-checking

8.4.1 Idée générale de l'algorithme

Les résultats des Lemmes 59 et 65 impliquent qu'il est possible de représenter un ensemble de la forme $[\![\phi]\!]_{\mathcal{A}}$ grâce à une famille de systèmes d'inégalités. Notre but dans cette section est donc de définir une méthode de construction pour cette représentation à partir d'une formule ϕ de ECTL*(WIPC*) et d'un WIPC*-automate $\mathcal{A}$. Nous considérons un WIPC*-automate $\mathcal{A} = \langle Q, I, F, \delta\rangle$ et une formule ϕ de ECTL*(IPC*) ainsi que les ressources syntaxiques l, K, V, C et Term utilisés dans $\mathcal{A}$ et ϕ (voir la Section 8.3).

Pour construire la représentation de $[\![\phi]\!]_{\mathcal{A}}$, nous allons procéder par induction sur la structure de ϕ. Le cas de base $\phi \equiv \top$ est évident et l'étape d'induction pour les connecteurs Booléens peut facilement être prouvée en procédant à des unions d'ensembles de systèmes d'inégalités et des intersections de systèmes d'inégalités (les détails seront donnés plus tard). Le cas le plus difficile concerne les formules de la forme $E\psi$. Comme nous procédons par induction, nous supposons que pour toute sous-formule d'état ϕ' de ψ nous connaissons une représentation de $[\![\phi']\!]_{\mathcal{A}}$.

Nous réduisons la construction de la représentation de $[\![\phi]\!]_{\mathcal{A}}$ dans le cas $\phi = E\psi$ à la construction de la représentation d'un ensemble d'états à partir desquels il existe un chemin vérifiant un certain nombre de propriétés. Étant donnés un ensemble d'états de $\mathcal{A}$ fermé par le haut U et un sous-ensemble d'états de contrôle F de $\mathcal{A}$, nous notons $[\![\pi_U^F]\!]_{\mathcal{A}}$ l'ensemble des états de $\mathcal{A}$ à partir desquels il existe un chemin infini tel que :

- tous les états du chemin appartiennent à U,
- l'ensemble d'états de contrôle F est visité infiniment souvent.

Intuitivement, la construction de la représentation de $[\![\phi]\!]_{\mathcal{A}}$ peut être réduite à la construction d'un ensemble d'états de la forme $[\![\pi_U^F]\!]_{\mathcal{A}}$ en utilisant la construction d'automate qui reconnaît les modèles d'une formule de LTL. L'ensemble F correspond à l'ensemble des états finaux de cette construction et U représente les différents états pour lesquels un ensemble de sous-formules d'états défini par la construction est satisfait afin de pouvoir considérer ces sous-formules comme des propositions. Ainsi la représentation de U peut être calculée à l'avance à partir des représentations des ensembles $[\![\phi']\!]_{\mathcal{A}}$ tels que ϕ' est une sous-formule d'état de ψ d'après notre hypothèse d'induction. Ces différents éléments sont développés dans la preuve du résultat ci-dessous.

Lemme 66 *Soit $\mathcal{A}$ un* WIPC**-automate et ϕ une formule de* ECTL*(WIPC*) *de la forme $E\psi$ pour laquelle la représentation de $[\![\phi']\!]_{\mathcal{A}}$ est disponible pour toute sous-formule d'état ϕ'. La construction d'une représentation de $[\![\phi]\!]_{\mathcal{A}}$ se réduit à la construction d'un ensemble de la forme $[\![\pi_U^F]\!]_{\mathcal{A}'}$ où $\mathcal{A}'$ est un* WIPC**-automate généralisé et la représentation de U est disponible.*

Preuve : Considérons une formule de ECTL*(WIPC*) de la forme $E\psi$. La réduction utilise une adaptation de la construction d'automate de LTL pour ψ en considèrant les contraintes atomiques et les sous-formules d'état maximales de ϕ comme des propositions. Une sous-formule maximale ϕ' est une sous-formule d'état stricte de ϕ telle qu'il n'existe pas de sous-formule d'état stricte ϕ'' de ϕ dont ϕ' est une sous-formule stricte. Nous supposons que la représentation de $[\![\phi']\!]_{\mathcal{A}}$ est disponible pour toute sous-formule maximale ϕ'. Nous considérons aussi sans perte de généralité que les différents systèmes d'inégalités qui composent ces représentations sont construits par rapport aux mêmes ressources V, C et K qui sont celles utilisées dans ϕ et $\mathcal{A}$.

Nous définissons la clôture standard de ψ pour les formules sous forme positives en considérant les contraintes atomiques et les sous-formules d'états comme des propositions (voir la construction du Lemme 49 pour le cas $E\psi$). Nous construisons un automate intermédiaire à partir duquel $\mathcal{A}'$ peut être obtenu. L'ensemble des états de cet automate est $\text{Atom}(\psi) \times Q$ où $\text{Atom}(\psi)$ est l'ensemble des atomes de ψ, c'est-à-dire l'ensemble des sous-ensembles maximalement cohérents de la clôture de ψ, et Q est l'ensemble des états de $\mathcal{A}$. L'ensemble des états initiaux est composé des états de la forme $\langle At, q\rangle$ tel que $\psi \in At$ et $q \in Q$. La relation de transition de ce WIPC*-automate généralisé combine la relation de transition de l'automate de LTL [VW94] et celle de $\mathcal{A}$, on a

$$\langle At, q\rangle \xrightarrow{\alpha\wedge\alpha_{At}} \langle At', q'\rangle$$

ssi

- pour toute formule $\mathsf{X}\phi'$ appartenant à la clôture de ψ, $\mathsf{X}\phi'$ appartient à At ssi ϕ' appartient à At',
- $q \xrightarrow{\alpha} q'$ est une transition de $\mathcal{A}$,
- α_{At} est la conjonction des formules atomiques de ECTL*(WIPC*) qui appartiennent à l'atome At.

La condition d'acceptation est spécifiée par une condition de Büchi généralisée. Considérons l'ensemble $\{\phi_1 \mathsf{U} \phi'_1, \ldots, \phi_n \mathsf{U} \phi'_n\}$ des formules construites avec l'opérateur U dans la clôture de ψ. Si cet ensemble est vide alors $F = \{\langle At, q\rangle \mid q$ est un état final de $\mathcal{A}$ et $At \in \mathrm{Atom}(\psi)\}$, sinon $F = \{F_1, \ldots, F_n\}$ tel que $F_i = \{\langle At, q\rangle \mid q$ est un état final de $\mathcal{A}$ et $\phi_i \mathsf{U} \phi'_i \notin At$ ou $\phi'_i \in At\}$.

Pour obtenir l'automate $\mathcal{A}'$ nous utilisons la construction classique qui permet de transformer l'automate ci-dessus en automate avec une condition d'acceptation de Büchi standard. Comme cette construction introduit des copies des états de l'automate initial, nous notons l'ensemble des états de $\mathcal{A}'$ avec des éléments de $\mathrm{Atom}(\psi) \times Q \times \{1, \ldots, n\}$ (l'ensemble des états initiaux et finaux est adapté en fonction).

Par construction, s'il existe une exécution acceptante à partir d'un état initial q'_0 pour $\mathcal{A}'$ telle qu'à chaque position les formules d'état appartenant à l'atome associé à l'état de contrôle courant sont vérifiées alors la formule ψ est satisfaite par un état initial q'_0. L'existence d'une telle exécution correspond à tester si l'état initial q'_0 appartient à $[\![\pi_U^F]\!]_{\mathcal{A}'}$ tel que F est l'ensemble des états finaux de $\mathcal{A}'$ et U est égal à :

$$\bigcup_{\langle At, q, i\rangle \in \mathcal{A}'} \{\langle\langle At, q, i\rangle, v\rangle \mid \text{ pour tout } E\psi' \in At \text{ on a } \langle q, v\rangle \in [\![E\psi']\!]_{\mathcal{A}}\}.$$

D'après l'hypothèse d'induction, la représentation de cet ensemble peut être construite. En effet, la représentation de l'ensemble

$$\{\langle\langle At, q, i\rangle, v\rangle \mid \text{ pour tout } E\psi' \in At \text{ on a } \langle q, v\rangle \in [\![E\psi']\!]_{\mathcal{A}}\}$$

est l'ensemble composé des systèmes d'inégalités de la forme

$$\bigcap_{E\psi' \in At} \mathrm{S}_{E\psi'} \text{ tel que } \mathrm{S}_{E\psi'} \in \mathcal{S}_{q, E\psi'}$$

où $\mathcal{S}_{q,E\psi'}$ est la représentation de l'ensemble des états de la forme $\langle q, v\rangle$ appartenant à $[\![E\psi']\!]_{\mathcal{A}}$. Notons que comme la construction de $\mathcal{A}'$ n'introduit pas de nouvelle ressource syntaxique qui n'est pas déjà présente dans $\mathcal{A}$ ou ϕ, tous ces systèmes sont locaux par rapport à l'automate $\mathcal{A}'$.

Nous montrons maintenant que l'ensemble $[\![E\psi]\!]_{\mathcal{A}}$ est la projection de l'ensemble des états initiaux de $\mathcal{A}'$ appartenant à $[\![\pi_U^F]\!]_{\mathcal{A}'}$ sur les états de $\mathcal{A}$, c'est-à-dire que

$$\langle q, v\rangle \in [\![E\psi]\!]_{\phi} \text{ ssi il existe un état de la forme } \langle\langle At, q, j\rangle, v\rangle \in [\![\pi_U^F]\!]_{\mathcal{A}'} \text{ tel que } \psi \in At.$$

Si $\langle q, v\rangle \in [\![E\psi]\!]_{\mathcal{A}}$ alors il existe un chemin π dans $\mathcal{A}$ tel que $\pi \models \psi$. On peut facilement montrer que pour tout $i \in \mathbb{N}$, il existe un unique atome At_i tel que $\pi^i \models At_i$ car l'ensemble des atomes définit une partition des modèles linéaires (les séquences de la forme $\mathbb{N} \to (V \to \mathbb{Z})$). Par construction de $\mathcal{A}'$ qui est une adaptation de la construction de l'automate pour LTL combinée avec la relation de transition de $\mathcal{A}$ il existe une exécution acceptante $\langle\langle At_0, q_0, j_0\rangle, v_0\rangle \to \langle\langle At_1, q_1, j_1\rangle, v_1\rangle \to \cdots$ telle que pour tout $i \in \mathbb{N}$ on a $\pi(i) = \langle q_i, v_i\rangle$. Cette exécution acceptante visite donc infiniment souvent un état final de F. Il reste à montrer que cette exécution visite seulement des états de U. Étant donné que pour toute position dans l'exécution on a $\pi^i \models At_i$, on a en particulier $\pi(i) \models E\psi'$ pour toute formule $E\psi'$ appartenant à At_i. Comme U représente l'ensemble des états de la forme $\langle\langle At, q, j\rangle, v\rangle$ pour lesquels toutes les formules d'états maximales appartenant à At sont vérifiées on a bien $\pi(i) \in U$.

Réciproquement considérons un état $\langle\langle At_0, q_0, j_0\rangle, v_0\rangle \in [\![\pi_U^F]\!]_{\mathcal{A}'}$ tel que $\psi \in At_0$. Il existe donc un chemin de la forme $\langle\langle At_0, q_0, j_0\rangle, v_0\rangle \to \langle\langle At_1, q_1, j_1\rangle, v_1\rangle \to \cdots$ dans $\mathcal{A}'$ qui passe uniquement par des états de U et visite infiniment souvent un état de F. Nous montrons que le chemin $\pi = \langle q_0, v_0\rangle \to \langle q_1, v_1\rangle \to \cdots$ dans $\mathcal{A}$ vérifie la propriété suivante pour tout $i \in \mathbb{N}$:

$$\text{pour toute sous-formule } \psi' \text{ de } \psi \text{ dans } At_i \text{ on a } \pi^i \models \psi'.$$

Notons que le chemin π est valide dans $\mathcal{A}$ car la relation de transition de $\mathcal{A}'$ utilise celle de $\mathcal{A}$. Nous procédons par induction sur la structure de ψ'.

- Si ψ' est une contrainte atomique α alors $\pi^i \models \alpha$ par définition de la relation de transition de $\mathcal{A}'$.
- Si ψ' est de la forme $E\psi''$ alors comme $\pi(i)$ appartient à U on a bien $\pi(i) \models E\psi''$.
- L'étape d'induction pour les connecteurs $\wedge, \vee$ et les opérateurs temporels peut ensuite être traitée de façon standard par rapport à la preuve classique pour LTL. Le cas de l'opérateur U peut facilement être déduit en utilisant la définition des états finaux qui correspond à celle de l'automate de LTL (voir [VW94]).

Ceci prouve donc que $\pi \models \psi$ puisque $\psi \in At_0$.

Ainsi, la représentation de $[\![E\psi]\!]_{\mathcal{A}}$ peut facilement être obtenue à partir de la représentation de $[\![\pi_U^F]\!]_{\mathcal{A}'}$ en faisant l'union des différents ensembles de systèmes d'inégalités de cette représentation qui représentent des ensembles d'états initiaux ayant la même image par projection sur les états de contrôle de $\mathcal{A}$. □

8.4.2 Construction pour le cas $E\psi$

À partir de maintenant, nous considérons donc un WIPC*-automate généralisé $\mathcal{A}$ ayant un ensemble d'états de contrôle Q. Les ensembles d'états initiaux et finaux ne sont pas importants pour la construction qui suit. Nous considérons aussi les ensembles V, C ainsi que les entiers K et l définis précédemment par rapport à l'ensemble des contraintes atomiques dans la relation de transition de $\mathcal{A}$. Notons que la relation de transition de l'automate construit dans le Lemme 66 tient aussi compte des ressources syntaxiques de la formule ϕ que nous voulons vérifiée. Nous consacrons la suite à la construction d'un ensemble de la forme $[\![\pi_U^F]\!]_{\mathcal{A}}$ dans $\mathcal{A}$, étant donnés un ensemble $F \subseteq Q$ et une représentation de l'ensemble d'états U.

Pour construire cette représentation, nous avons besoin dans un premier temps de caractériser la relation d'accessibilité dans $\mathcal{A}$ à l'aide de systèmes d'inégalités. Pour ce faire nous devons considérer des systèmes d'inégalités particuliers. Soit V' l'ensemble de variables obtenus en primant les variables de V. Un *système d'inégalités transitionnel* pour $\mathcal{A}$ est un système d'inégalités construit par rapport aux éléments de $V \uplus V' \uplus C$. Une paire d'états $\langle q, v\rangle, \langle q', v'\rangle$ de $\mathcal{A}$ satisfait un système transitionnel S_t ssi la valuation qui pour tout $x \in V$ affecte la valeur $v(x)$ à x et pour tout $x' \in V'$ affecte la valeur $v'(x)$ à x' satisfait toutes les contraintes de S_t. Étant donnés deux états de contrôles q et q', un ensemble de systèmes d'inégalités transitionnels caractérise une relation d'accessibilité de la forme $q \to^* q'$ si un couple de valuations v, v' satisfait un système de cet ensemble ssi il existe une exécution finie dans $\mathcal{A}$ entre $\langle q, v\rangle$ et $\langle q', v'\rangle$. Nous appelons cet ensemble de systèmes d'inégalités la *représentation de la relation d'accessibilité* $q \to^* q'$. Nous considérons des relations d'accessibilité un peu plus complexes par la suite. Étant donnés un ensemble d'états de $\mathcal{A}$ fermé par le haut U et une séquence d'états de contrôle π entre deux états de contrôle q et q', nous notons

- $q \to_U^* q'$ ssi il existe une exécution finie visitant uniquement des états de U débutant à l'état de contrôle q et terminant dans l'état de contrôle q',
- $q \xrightarrow{\pi}_U q'$ ssi il existe une exécution finie visitant uniquement des états de U et suivant le chemin décrit par la séquence d'états de contrôle π entre les états de contrôle q et q'.

Nous montrons plus bas qu'étant donnés q, q' et un chemin π entre ces deux états de contrôle, ces relations d'accessibilité peuvent aussi être caractérisée par un ensemble de systèmes d'inégalités transitionnels de telle façon que si un couple de valuations v, v' satisfait un élément de la représentation alors $\langle q, v\rangle \to_U^* \langle q', v'\rangle$ (respectivement $\langle q, v\rangle \xrightarrow{\pi}_U \langle q', v'\rangle$).

L'opération de composition de systèmes d'inégalités transitionnels $\mathrm{S}\cdot\mathrm{S}'$ définie ci-dessous nous sera utile pour construire ces caractérisations.

1. Soit $V'' = \{x'' \mid x \in V\}$ l'ensemble de variables obtenues en primant deux fois les variables de V. On met d'abord en correspondance les variables d'arrivée de S avec les variables de départ de S' en renommant les variables de la façon suivante :
   - on substitue chaque variable $x' \in V'$ dans S en $x'' \in V''$,
   - on substitue chaque variable $x \in V$ dans S' en $x'' \in V''$.
2. Le système $\mathrm{S} \cdot \mathrm{S}'$ est la projection sur $V \uplus V'$ de l'intersection des systèmes obtenus après renommage.

Par construction il est évident que si $\langle q, v\rangle, \langle q'', v''\rangle$ satisfait S et $\langle q'', v''\rangle, \langle q', v'\rangle$ satisfait S' alors $\langle q, v\rangle, \langle q', v'\rangle$ satisfait $\mathrm{S} \cdot \mathrm{S}'$. Intuitivement, cette propriété nous permet de calculer la représentation d'une relation d'accessibilité obtenue en composant deux relations d'accessibilité.

Lemme 67
(I) *Étant donnés un chemin fini π de q vers q' et la représentation d'un ensemble d'états U fermé par le haut, on peut construire une représentation de $q \xrightarrow{\pi}_U q'$.*
(II) *Étant donnés deux états de contrôles q est q' et la représentation d'un ensemble d'états U fermé par le haut, on peut construire une représentation de $q \to_U^* q'$.*

Preuve : **(I)** Nous considérons $|\pi| + l + 1$ copies de l'ensemble des variables V que nous notons V^i pour tout $i \in \{0, \ldots, |\pi|+l\}$. Nous avons $\langle x, i\rangle \in V^i$ pour tout $i \in \{0, \ldots, |\pi|+l\}$ ssi $x \in V$. La représentation de U est donnée par une famille d'ensembles de systèmes d'inégalités locaux $(\mathcal{S}_q^U)_{q\in Q}$.

Les disjonctions de contraintes atomiques sur les transitions de l'automate posent problème car elles sous-entende un choix non-déterministe dans l'exécution. Nous supposons donc que les contraintes sur les transitions de l'automate sont des conjonctions de contraintes de la forme $\mathsf{X}^i x \sim \mathsf{X}^j y$, $\mathsf{X}^i x \sim d$, $d \sim \mathsf{X}^i x$ ou $\mathsf{X}^i x \equiv_K c$ pour $\sim \in \{>, \geq\}$. Nous pouvons toujours nous ramener à ce cas de la manière suivante.

- D'abord il est clair que toutes les relations de WIPC* qui ne sont pas de la forme $x \equiv_K c$, $x \sim y$, $d \sim x$, $x \sim d$ ou $x \equiv_K c$ tel que $\sim \in \{>, \geq\}$ peuvent s'exprimer par des conjonctions ou disjonctions de relations de ce type. Nous rappelons que K est le ppcm des entier k apparaissant dans les relations de périodicité $\equiv_k$. La négation peut aussi être supprimée de la même manière.
- Ensuite, les formules obtenues peuvent être mises sous forme normale disjonctive.
- Enfin pour chaque transition $q \xrightarrow{\alpha} q'$ dans l'automate résultant, on décompose la transition de manière à avoir une transition $q \xrightarrow{\alpha'} q'$ pour chaque conjonction α' qui est un disjoint de α (α est sous forme DNF). Cette construction préserve le graphe des configurations car chaque disjoint peut être vu comme un choix non-déterministe.

Maintenant que toutes les transitions dans $\mathcal{A}$ sont étiquetées par des conjonctions, nous considérons un chemin particulier $\pi_r = q_0 \xrightarrow{\alpha_0} \cdots \xrightarrow{|\pi|-1} q_{|\pi|}$ suivant les états de contrôle définis par la séquence π. Soit $\mathcal{S}$ l'ensemble de systèmes d'inégalités tel que $\mathrm{S} \in \mathcal{S}$ ssi les conditions suivantes sont vérifiées :

(1) pour tout $i \in \{0, \ldots, |\pi|-1\}$ la comparaison $\mathsf{X}^j x > \mathsf{X}^{j'} y$ (resp. $\mathsf{X}^j x > d$ ou $d > \mathsf{X}^j x$) est un conjoint de α_i ssi $\langle x, i+j\rangle - \langle y, i+j'\rangle \geq 1$ (resp. $\langle x, i+j\rangle - d \geq 1$ ou $d - \langle x, i+j\rangle \geq 1$) est une contrainte de S,

(2) pour tout $i \in \{0, \ldots, |\pi|-1\}$ la comparaison $\mathsf{X}^j x \geq \mathsf{X}^{j'} y$ (resp. $\mathsf{X}^j x \geq d$ ou $d \geq \mathsf{X}^j x$) est un conjoint de α_i ssi $\langle x, i+j\rangle - \langle y, i+j'\rangle \geq 0$ (resp. $\langle x, i+j\rangle - d \geq 0$ ou $d - \langle x, i+j\rangle \geq 0$) est une contrainte de S,

(3) pour tout $i \in \{0, \ldots, |\pi|-1\}$ si la contrainte de périodicité $\mathsf{X}^j x \equiv_K c$ est un conjoint de α^i alors $\langle x, i+j\rangle \equiv_K c$ appartient à S

Ces conditions expriment les contraintes liées aux transitions franchies. Nous devons aussi tenir compte des contraintes imposées par l'ensemble U donc,

(4) pour tout $i \in \{0, \ldots, |\pi|\}$, il existe un système d'inégalités $\mathrm{S}' \in \mathcal{S}_{q_i}^U$ tel que

- $x - y \geq d$ appartient à S$'$ ssi $\langle x, i\rangle - \langle y, i\rangle \geq d$ appartient à S,
- $x - d' \geq d$ appartient à S$'$ ssi $\langle x, i\rangle - d' \geq d$ appartient à S,
- $d' - y \geq d$ appartient à S$'$ ssi $d' - \langle y, i\rangle \geq d$ appartient à S,
- $x \equiv_K c$ appartient à S$'$ ssi $\langle x, i\rangle \equiv_K c$ appartient à S.

Notons que les systèmes obtenus ne sont pas sous forme normale. En effet, on peut avoir par exemple une contrainte $x - y \geq 1$ issue du point **(1)** et une autre de la forme $x - y \geq d$ avec $d \neq 1$ issue du point **(4)** dans le même système d'inégalité.

Considérons une séquence $\langle q_0, v_0\rangle \cdot \langle q_1, v_1\rangle \cdots \langle q_{|\pi|}, v_{|\pi|}\rangle$ suivant les états de contrôle de π. Par construction, la valuation v telle que $v(\langle x, i\rangle) = v_i(x)$ pour tout $x \in V$ et $i \in \{0, \ldots, |\pi|\}$ satisfait S ssi $\langle q_0, v_0\rangle \cdot \langle q_1, v_1\rangle \cdots \langle q_{|\pi|}, v_{|\pi|}\rangle$ est une sous-exécution finie valide dans $\mathcal{A}$ qui suit π_r et visite uniquement des états de U. Le fait que l'exécution soit valide découle des points **(1)** à **(3)** et la condition **(4)** impose que les états visités appartiennent à U.

En utilisant le Lemme 63 (II), nous pouvons en déduire que la valuation v telle que $v(\langle x, i\rangle) = v_i(x)$ pour tout $x \in V$ et $i \in \{0, |\pi|\}$ satisfait la restriction de S aux variables de $V^0 \uplus V^{|\pi|}$ ssi il existe une valuation qui étend v et satisfait S, ce qui implique l'existence d'une sous-exécution finie valide $\langle q_0, v_0\rangle \cdot \langle q_1, v_1\rangle \cdots \langle q_{|\pi|}, v_{|\pi|}\rangle$ dans $\mathcal{A}$ qui suit π_r et visite uniquement des états de U. Nous pouvons donc en conclure que la relation d'accessibilité en suivant le chemin π_r est caractérisée par l'ensemble $\mathcal{S}_{\pi_r}$ composé des systèmes d'inégalités transitionnels obtenus par projection des systèmes de $\mathcal{S}$ sur les variables de $V^0 \uplus V^{|\pi|}$ et renommage des variables de V^0 en V et $V^{|\pi|}$ en V'.

Comme π décrit l'ensemble des chemins de la forme π_r considérés ci-dessus, la caractérisation de $q \xrightarrow{\pi}_U q'$ est l'union de toutes les représentations obtenues à partir des différents chemins π_r.

(II) Pour construire la représentation d'une relation de la forme $q \rightarrow_U^* q'$ nous procédons par induction sur la longueur des chemins entre q et q'. Pour toute paire d'états de contrôles $q, q' \in Q$ nous posons

- $\mathcal{S}^1_{q \rightarrow_U^* q'} = \mathcal{S}_{q \rightarrow_U q'}$ où $\mathcal{S}_{q \rightarrow_U q'}$ est la représentation de la relation d'accessibilité de q à q' sur une seule transition impliquant uniquement des états de U. Cette représentation peut facilement être construite car c'est en fait la représentation de la relation $q \xrightarrow{\pi}{}^*_U q'$ telle que $\pi = q \cdot q'$.
- $\mathcal{S}^{i+1}_{q \rightarrow_U^* q'} = \mathcal{S}^i_{q \rightarrow_U^* q'} \cup \bigcup_{q'' \in Q} (\mathcal{S}^i_{q \rightarrow_U^* q''} \cdot \mathcal{S}^1_{q'' \rightarrow_U^* q'})$.

Par définition de la composition de systèmes d'inégalités, pour tout $i \geq 1$ l'ensemble $\mathcal{S}^i_{q \rightarrow_U^* q'}$ caractérise la relation de transition entre q et q' par rapport aux chemins de longueur inférieure à i et visitant uniquement des états de U. Bien que l'ensemble des chemins entre deux états de contrôle q et q' peut être infini, nous montrons que pour toute paire d'états de contrôle q et q' la construction de $\mathcal{S}^i_{q \rightarrow_U^* q'}$ se stabilise. Nous considérons que la construction se stabilise s'il existe une position $i > 0$ telle que pour tout $j \geq i$ et toute paire d'états $\langle q, v\rangle, \langle q', v'\rangle$ de $\mathcal{A}$ on a $\langle q, v\rangle, \langle q', v'\rangle \models \mathcal{S}^i_{q \rightarrow_U^* q'}$ ssi $\langle q, v\rangle, \langle q', v'\rangle \models \mathcal{S}^j_{q \rightarrow_U^* q'}$.

Pour cela nous définissons un ordre sur les ensembles de systèmes d'inégalités qui utilise le beau préordre $\sqsubseteq$ sur les systèmes d'inégalités. Étant donnés deux ensembles de systèmes d'inégalités $\mathcal{S}$ et $\mathcal{S}'$, nous notons $\mathcal{S} \sqsubseteq \mathcal{S}'$ ssi pour tout système d'inégalités S dans $\mathcal{S}$ il existe un système d'inégalités S$'$ dans $\mathcal{S}'$ tel que S$' \sqsubseteq$ S (attention, l'ordre des éléments change). Notons que cet ordre est réflexif et transitif. Cette définition implique que tout élément appartenant à l'ensemble représenté par $\mathcal{S}$ appartient aussi à l'ensemble représenté par $\mathcal{S}'$, en conséquence de l'ordre sur les systèmes d'inégalités. Donc si on a $\mathcal{S}' \sqsubseteq \mathcal{S}$ et $\mathcal{S} \sqsubseteq \mathcal{S}'$ alors $\mathcal{S}$ et $\mathcal{S}'$ représentent le même ensemble. De plus, en analysant la définition on remarque que si $\mathcal{S} \subseteq \mathcal{S}'$ alors $\mathcal{S} \sqsubseteq \mathcal{S}'$.

Nous considérons une paire d'états de contrôle $q, q' \in Q$. Pour la suite nous notons $\mathcal{S}_i$ au lieu de $\mathcal{S}^i_{q \rightarrow^*_U q'}$ pour tout $i \geq 0$. D'après la construction, nous avons $\mathcal{S}_i \subseteq \mathcal{S}_j$ pour tout $i < j$, par conséquent on a aussi $\mathcal{S}_i \sqsubseteq \mathcal{S}_j$ pour tout $i < j$. Donc s'il existe $i > 0$ tel que pour tout $j > i$ on a $\mathcal{S}_j \sqsubseteq \mathcal{S}_i$ alors la séquence se stabilise. Nous procédons par l'absurde. Si nous supposons qu'il n'existe pas une telle position, alors nous pouvons construire une séquence infinie

$$\mathcal{S}_{i_0} \sqsubseteq \mathcal{S}_{i_1} \sqsubseteq \mathcal{S}_{i_2} \sqsubseteq \mathcal{S}_{i_3} \cdots$$

telle que pour tout $j \geq 0$ on a $i_j < i_{j+1}$ et $\mathcal{S}_{i_{j+1}} \not\sqsubseteq \mathcal{S}_{i_j}$. Pour tout $j' < j$ on a aussi $\mathcal{S}_{i_j} \not\sqsubseteq \mathcal{S}_{i_{j'}}$ sinon par transitivité $\mathcal{S}_{i_{j'+1}} \sqsubseteq \mathcal{S}_{i_{j'}}$.

Nous montrons que cette séquence vérifie la propriété suivante : pour tout $j > 0$ il existe un système d'inégalités $\mathrm{S}_j \in \mathcal{S}_{i_j}$ tel que pour tout $j' < j$ et $\mathrm{S}_{j'} \in \mathcal{S}_{i_{j'}}$ on a $\mathrm{S}_{j'} \not\sqsubseteq \mathrm{S}_j$. Nous procédons à nouveau par l'absurde. Supposons que pour tout $\mathrm{S}_j \in \mathcal{S}_{i_j}$ il existe $j' < j$ et $\mathrm{S}_{j'} \in \mathcal{S}_{i_{j'}}$ tel que $\mathrm{S}_{j'} \sqsubseteq \mathrm{S}_j$. Comme $i_{j'} \leq i_{j-1}$, on a $\mathcal{S}_{i_{j'}} \sqsubseteq \mathcal{S}_{i_{j-1}}$ et donc il existe $\mathrm{S}_{j-1} \in \mathcal{S}_{i_{j-1}}$ tel que $\mathrm{S}_{j-1} \sqsubseteq \mathrm{S}_{j'}$. Ceci implique que pour tout $\mathrm{S}_j \in \mathcal{S}_{i_j}$ il existe $\mathrm{S}_{j-1} \in \mathcal{S}_{i_{j-1}}$ tel que $\mathrm{S}_{j-1} \sqsubseteq \mathrm{S}_j$. Donc on a $\mathcal{S}_j \sqsubseteq \mathcal{S}_{j-1}$ ce qui nous donne une contradiction.

Le résultat ci-dessus implique que nous pouvons construire une séquence infinie $\mathrm{S}_0 \cdot \mathrm{S}_1 \cdots$ de systèmes d'inégalités tels que pour tout $i, j \geq 0$, si $i < j$ alors on a $\mathrm{S}_i \not\sqsubseteq \mathrm{S}_j$. Nous obtenons donc une contradiction car $\sqsubseteq$ est un beau préordre sur les systèmes d'inégalités.

Donc, pour toute paire d'états de contrôle q et q' la construction se stabilise. Ce qui signifie qu'il existe une position i telle que pour toute paire $q, q' \in Q$ on a $\mathcal{S}^{i+1}_{q \rightarrow^*_U q'} = \mathcal{S}^i_{q \rightarrow^*_U q'}$. Lorsque cette position est atteinte on sait que l'on a atteint le point fixe de l'algorithme. On peut alors arrêter la construction et conclure que $\mathcal{S}^i_{q \rightarrow^*_U q'}$ caractérise la relation $q \rightarrow^*_U q'$. $\square$

Un corollaire de ce résultat est qu'étant donnés deux ensembles d'états X et U ainsi que leurs représentations on peut construire une représentation de l'ensemble des états à partir desquels on peut atteindre un état de X en visitant uniquement des états de U. Nous notons cet ensemble $\mathrm{Pre}^*_U(X)$.

Corollaire 16 *Étant donnés deux ensembles d'états X et U et leurs représentations, on peut construire une représentation de l'ensemble*

$$\mathrm{Pre}^*_U(X) = \{\langle q, v\rangle \mid \langle q, v\rangle \rightarrow^*_U \langle q, v'\rangle \textit{ tel que } \langle q', v'\rangle \in X\}.$$

Preuve : Supposons que l'ensemble X soit représenté par une famille d'ensembles de systèmes d'inégalités $(\mathcal{S}^X_q)_{q \in Q}$. À partir de la représentation de U, on sait d'après le Lemme 67 qu'on peut construire une représentation de $q \rightarrow^*_U q'$ pour toute paire d'états q, q'. La représentation de l'ensemble $\mathrm{Pre}^*_U(X)$ est définie par la famille $(\mathcal{S}_q)_{q \in Q}$ telle que pour tout $q \in Q$ on a

$$\mathcal{S}_q = \{(\mathrm{S}_1 \cap \mathrm{S}'_2)_{|V} \mid \mathrm{S}_1 \in \mathcal{S}_{q \rightarrow^*_U q'} \text{ et } \mathrm{S}_2 \in \mathcal{S}^X_{q'}\}$$

où nous notons S'_2 le système d'inégalités obtenu à partir de S_2 en primant les variables, afin de faire coïncider ce système d'inégalités avec l'ensemble d'arrivée du système d'inégalités transitionnel S_1. En utilisant le résultat du Lemme 63 (I) sur l'intersection de systèmes d'inégalités, il est facile de montrer que $\langle q, v\rangle \in \mathrm{Pre}_U^*(X)$ ssi il existe un système d'inégalités $S \in \mathcal{S}_q$ tel que $\langle q, v\rangle$ satisfait S. $\square$

Nous caractérisons maintenant l'ensemble d'états $[\![\pi_U^F]\!]_{\mathcal{A}}$ en utilisant les définitions ci-dessus. Pour tout $i \in \mathbb{N}$ nous notons $(\pi)^i$ la séquence d'états de contrôles obtenue en concaténant i fois le cycle π. Pour tout cycle π, nous définissons les ensembles :

– $X_\pi^{\preceq} = \{\langle q, v\rangle \mid \langle q, v\rangle \xrightarrow{\pi}_U \langle q, v'\rangle \text{ et } v \preceq v'\}$ et

– $X_\pi^{\infty} = \{\langle q, v\rangle \mid \langle q, v\rangle \xrightarrow{(\pi)^\omega}_U\}$.

Nous introduisons la notation $\langle q, v\rangle \xrightarrow{(\pi)^\omega}_U$ pour exprimer qu'il existe une exécution infinie à partir de l'état $\langle q, v\rangle$ qui répète infiniment le cycle π et visite uniquement des états appartenant à U. L'ensemble $[\![\pi_U^F]\!]_{\mathcal{A}}$ peut être caractérisé de la manière suivante.

Lemme 68 *Pour tout état $\langle q, v\rangle$ d'un* WIPC**-automate généralisé $\mathcal{A}$, on a $\langle q, v\rangle \in [\![\pi_U^F]\!]_{\mathcal{A}}$ ssi il existe un ensemble X tel que $\langle q, v\rangle \in \mathrm{Pre}_U^*(X)$ et $X_\pi^{\preceq} \subseteq X \subseteq X_\pi^{\infty}$ où π est un cycle ayant pour origine un état de contrôle $q_f \in F$.*

Preuve : Considérons un état $\langle q_0, v_0\rangle$ appartenant à $[\![\pi_U^F]\!]_{\mathcal{A}}$. Par définition de $[\![\pi_U^F]\!]_{\mathcal{A}}$ il existe un état final $q_f \in F$ et une exécution à partir de $\langle q_0, v_0\rangle$ de la forme :

$$\langle q_0, v_0\rangle \rightarrow_U^* \langle q_f, v_1\rangle \rightarrow_U^* \langle q_f, v_2\rangle \rightarrow_U^* \langle q_f, v_3\rangle \cdots$$

Comme l'ordre $\preceq_{\phi,\mathcal{A}}$ est un beau préordre sur les états de $\mathcal{A}$ (Lemme 57), il existe donc i et j tels que $0 < i < j$ et $v_i \preceq_{\phi,\mathcal{A}} v_j$. Par conséquent, nous avons $\langle q_f, v_i\rangle \in X_\pi^{\preceq}$ tel que π est la projection du sous chemin entre $\langle q_f, v_i\rangle$ et $\langle q_f, v_j\rangle$ sur les états de contrôle de $\mathcal{A}$. Le chemin π est donc bien un cycle ayant pour origine un état final. De plus, pour tout ensemble X tel que $X_\pi^{\preceq} \subseteq X$ on a $\langle q_f, v_i\rangle \in X$. Puisqu'il existe un chemin entre $\langle q_0, v_0\rangle$ et $\langle q_f, v_i\rangle$ passant uniquement par des états de U, on a bien $\langle q_0, v_0\rangle \in \mathrm{Pre}_U^*(X)$.

Réciproquement, soit π un cycle ayant pour origine un état de contrôle $q_f \in F$ et un ensemble X vérifiant $X_\pi^{\preceq} \subseteq X \subseteq X_\pi^{\infty}$. Par définition, si $\langle q_0, v_0\rangle \in \mathrm{Pre}_U^*(X)$ alors il existe un état $\langle q, v\rangle$ tel que $\langle q_0, v_0\rangle \rightarrow_U^* \langle q, v\rangle$ et $\langle q, v\rangle \in X$. Or, puisque $X \subseteq X_\pi^{\infty}$ on a aussi $\langle q, v\rangle \in X_\pi^{\infty}$. Cela signifie que $q = q_f$ et qu'à partir de $\langle q_f, v\rangle$ il existe une exécution infinie répétant la boucle π dans $\mathcal{A}$ et visitant uniquement des états de l'ensemble U. Il est évident que cette exécution visite infiniment souvent q_f. Il existe donc une exécution de la forme

$$\langle q_0, v_0\rangle \rightarrow_U^* \langle q_f, v\rangle \xrightarrow{(\pi)^\omega}_U$$

qui témoigne que $\langle q_0, v_0\rangle \in [\![\pi_U^F]\!]_{\mathcal{A}}$. $\square$

Il nous faut donc maintenant montrer qu'étant donné un cycle π, on peut construire une représentation d'un ensemble vérifiant les conditions du Lemme 68.

Lemme 69 *Étant donné un cycle π et la représentation d'un ensemble U fermé par le haut, on peut construire la représentation d'un ensemble X tel que $X_\pi^\preceq \subseteq X \subseteq X_\pi^\infty$.*

Nous consacrons la dernière section de ce chapitre à la preuve constructive de ce résultat (voir Section 8.5). Nous avons maintenant tous les éléments nécessaires pour établir le principal résultat technique de cette section.

Lemme 70 *Étant donnés un ensemble d'états de contrôle F et une représentation d'un ensemble U fermé par le haut on peut construire une représentation de l'ensemble $[\![\pi_U^F]\!]_\mathcal{A}$.*

Preuve : Pour chaque $q_f \in F$ nous pouvons construire une représentation de la relation $q_f \rightarrow_U^* q_f$. D'après la preuve du Lemme 67 (II), cette construction termine après un nombre i d'itérations et la représentation obtenue correspond en particulier à la relation d'accessibilité par des chemins de longueur inférieure à i. Nous pouvons donc déduire qu'il existe un nombre fini de chemins représentatifs $\{\pi_1^f, \dots, \pi_{n_f}^f\}$ de longueur inférieure à i pour la relation $q_f \rightarrow_U^* q_f$ dans le sens où la représentation $\mathcal{S}_{q \rightarrow_U^* q'}$ de la relation $q_f \rightarrow_U^* q_f$ est égale à

$$\mathcal{S}^i_{q_f \rightarrow_U^* q_f} = \bigcup_{0 \leq j \leq n_f} \mathcal{S}_{\pi_j^f}$$

où $\mathcal{S}_{\pi_j^f}$ est la représentation de $q_f \xrightarrow{\pi_j^f}_U q_f$.

Pour chacun de ces cycles π_j^f, nous pouvons construire d'après le Lemme 69 la représentation $\mathcal{S}_j^f$ d'un ensemble d'états X_j^f tel que $X^\preceq_{\pi_j^f} \subseteq X_j^f \subseteq X^\infty_{\pi_j^f}$. Ainsi nous pouvons construire la représentation d'un ensemble X comme étant la famille $(\mathcal{S}_{q_f})_{q \in F}$ telle que pour tout $q_f \in F$ on a $\mathcal{S}_{q_f} = \bigcup_{0 \leq j \leq n_f} \mathcal{S}_j^f$.

Nous montrons maintenant que $\mathrm{Pre}_U^*(X) = [\![\pi_U^F]\!]_\mathcal{A}$. Si un état $\langle q, v\rangle$ appartient à $\mathrm{Pre}_U^*(X)$ alors il existe un état de contrôle $q_f \in F$ et une valuation v' telle que $\langle q, v\rangle \rightarrow_U^* \langle q_f, v'\rangle$ et $\langle q_f, v'\rangle$ appartient à X. D'après la représentation de X, il existe un système d'inégalités S appartenant à un ensemble de la forme $\mathcal{S}_i^f$ tel que $\langle q_f, v'\rangle \models$ S. Donc l'état $\langle q, v\rangle$ appartient à X_i^f et nous pouvons utiliser le Lemme 68 pour conclure.

Réciproquement, si un état $\langle q, v\rangle$ appartient à $[\![\pi_U^F]\!]_\mathcal{A}$ alors il existe un chemin π de la forme

$$\langle q_0, v_0\rangle \rightarrow_U^* \langle q_f, v_1\rangle \rightarrow_U^* \langle q_f, v_2\rangle \rightarrow_U^* \langle q_f, v_3\rangle \cdots$$

Nous reprenons alors les arguments du Lemme 68. D'abord, il existe $i, j > 0$ tels que $i < j$ et $v_i \preceq v_j$ puisque $\preceq$ est un beau préordre. De plus, comme $\bigcup_{0 \leq j \leq n_f} \mathcal{S}_{\pi_j^f}$ caractérise la relation de transition $q_f \rightarrow_U^* q_f$ il existe un chemin π_k^f tel que $\langle q_f, v_i\rangle, \langle q_f, v_j\rangle \models \mathcal{S}_{\pi_k^f}$. Par conséquent la relation $\langle q_f, v_i\rangle \xrightarrow{\pi_k^f}_U \langle q_f, v_j\rangle$ et valide et donc $\langle q_f, v_i\rangle \in X^\preceq_{\pi_k^f}$ puisque $v_i \preceq v_j$. Comme $X^\preceq_{\pi_k^f} \subseteq X_k^f$, l'état $\langle q_f, v_i\rangle$ satisfait aussi $\mathcal{S}_k^f$ ce qui implique que l'on a bien $\langle q_0, v_0\rangle \in \mathrm{Pre}_U^*(X)$.

Comme nous pouvons construire une représentation de X et que la représentation de U est disponible, il est aussi possible de construire une représentation de l'ensemble $\mathrm{Pre}^*_U(X)$ d'après le Corollaire 16. Puisque $\mathrm{Pre}^*_U(X) = [\![\pi^F_U]\!]_{\mathcal{A}}$ cette représentation est aussi une représentation de $[\![\pi^F_U]\!]_{\mathcal{A}}$. □

8.4.3 Construction de $[\![\phi]\!]_{\mathcal{A}}$

En utilisant, les Lemmes 66 et 70 nous pouvons alors démontrer le résultat attendu.

Théorème 41 *Étant donnés un* WIPC**-automate $\mathcal{A}$ et une formule ϕ de* ECTL*(WIPC*)*, on peut construire une famille d'ensembles de systèmes d'inégalités représentant $[\![\phi]\!]_{\mathcal{A}}$.*

Preuve : Nous procédons par induction sur la structure de la formule ϕ. Si $\phi = \top$ alors la représentation de $[\![\phi]\!]_{\mathcal{A}}$ est la famille d'ensembles de systèmes d'inégalités $(\mathcal{S}_q)_{q\in Q}$ telle que pour tout $q \in Q$, $\mathcal{S}_q$ est l'ensemble des systèmes d'inégalités construits par rapport à $\mathcal{A}$ et ϕ de la forme $\langle X_{\mathrm{dif}}, X_{\mathrm{mod}}\rangle$ tel que $X_{\mathrm{dif}} = \emptyset$. Ainsi pour tout état $\langle q, v\rangle$ il existe $\mathrm{S} \in \mathcal{S}_q$ tel que $\langle q, v\rangle \models \mathrm{S}$ (il suffit juste de choisir le système d'inégalités ayant le bon ensemble de contraintes de périodicité).

Supposons maintenant que pour toute sous-formule d'état ϕ' de ϕ on dispose de la représentation $(\mathcal{S}^{\phi'}_q)_{q\in Q}$ de $[\![\phi']\!]_{G_{\mathcal{A}}}$.

La représentation $(\mathcal{S}_q)_{q\in Q}$ de $[\![\phi_1 \wedge \phi_2]\!]_{\mathcal{A}}$ est définie par

$$\mathcal{S}_q = \{\mathrm{S}^1_q \cap \mathrm{S}^2_q \mid \mathrm{S}^1_q \in \mathcal{S}^{\phi_1}_q \text{ et } \mathrm{S}^2_q \in \mathcal{S}^{\phi_2}_q\}$$

pour tout $q \in Q$. La représentation de $[\![\phi_1 \vee \phi_2]\!]_{\mathcal{A}}$ est la famille $(\mathcal{S}_q)_{q\in Q}$ telle que pour tout $q \in Q$ on a

$$\mathcal{S}_q = \mathcal{S}^{\phi_1}_q \cup \mathcal{S}^{\phi_2}_q.$$

Enfin pour le cas où ϕ est de la forme $E\psi$, on peut utiliser le Lemme 66 pour réduire le problème à la construction de la représentation d'un ensemble de la forme $[\![\pi^F_U]\!]_{\mathcal{A}'}$ où la représentation de U est disponible par hypothèse d'induction. D'après le Lemme 70, la représentation de cet ensemble peut être construite. □

Nous pouvons donc étendre le résultat de décidabilité établit dans la Section 4.4 pour le problème du model-checking de CLTL(IPC*).

Théorème 42 *Le problème du model-checking de* ECTL*(IPC*) *est décidable.*

Preuve : D'après le Lemme 7, ce problème se réduit au model-checking de ECTL*(WIPC*) avec un explosion exponentielle de la taille des contraintes de la formule et l'automate. Le résultat du Théorème 41 établit que l'on peut alors construire une représentation, sous la forme d'une famille de systèmes d'inégalités, de l'ensemble des états qui satisfont la formule. Comme tester si un état de l'automate satisfait un système d'inégalités dans un ensemble donné est décidable, le problème du model-checking est aussi décidable. $\square$

Bien sûr, comme la négation d'une formule de ECTL*(IPC*) est une formule du fragment universel ACTL*(IPC*) et vice-versa, nous obtenons le corollaire suivant.

Corollaire 17 *Le problème de model-checking de* ACTL*(IPC*) *est décidable.*

La complexité de la procédure que nous utilisons est difficile à établir à cause des conditions de terminaisons utilisées dans l'algorithme qui reposent sur des beaux préordres (en particulier dans le Lemme 67).

8.5 Preuve du Lemme 69

Nous terminons par la preuve du Lemme 69 dont nous rappelons l'énoncé ci-dessous :

Étant donné un cycle π et la représentation d'un ensemble U fermé par le haut, on peut construire la représentation d'un ensemble X tel que $X_\pi^{\preceq} \subseteq X \subseteq X_\pi^{\infty}$.

Soit π un cycle ayant pour origine un état de contrôle q et $\mathcal{S}_\pi$ la représentation de la relation $q \xrightarrow{\pi}_U q$, c'est-à-dire la représentation de l'ensemble des couples d'états $\langle q, v\rangle, \langle q, v'\rangle$ tels que l'on peut aller de $\langle q, v\rangle$ à $\langle q, v'\rangle$ en suivant π et en visitant unique ment des états de U. Pour tout système d'inégalités transitionnel $\mathrm{S}_\pi \in \mathcal{S}_\pi$ on peut construire un système d'inégalités local S'_π de façons à ce que l'ensemble $\mathcal{S}'_\pi$ composé des systèmes d'inégalités locaux S'_π obtenus à partir de l'ensemble des systèmes S_π qui composent $\mathcal{S}_\pi$ vérifie la propriété suivante

$$X_\pi^{\preceq} \subseteq \{\langle q, v\rangle \mid \text{ il existe } \mathrm{S}'_\pi \in \mathcal{S}'_\pi \text{ tel que } v \models \mathrm{S}'_\pi\} \subseteq X_\pi^{\infty}.$$

L'idée de la construction est qu'à partir de chaque système d'inégalités transitionnel $\mathrm{S}_\pi \in \mathcal{S}_\pi$, nous allons construire un système d'inégalités local S'_π tel que :

- pour tout couple de valuations v, v' tel que $v \preceq v'$ et $v, v' \models \mathrm{S}_\pi$ on a $v \models \mathrm{S}'_\pi$ et
- pour toute valuation v telle que $v \models \mathrm{S}'_\pi$ il existe une séquence infinie de valuations $v_0, v_1, \cdots$ telle que $v_0 = v$ et $v_i, v_{i+1} \models \mathrm{S}_\pi$.

Nous identifions dans notre construction un système symbolique avec sa représentation graphique, pour des raisons de présentation. Dans la représentation symbolique d'un système d'inégalités transitionnel S, nous introduisons les relations suivantes

- $a \,\mathrm{r}\, b$ ssi $b \rightarrow a$ ou $b' \rightarrow a'$ dans S,
- $a \,\mathrm{r}_d\, b$ ssi $b \rightarrow a'$ dans S,
- $a \,\mathrm{r}_g\, b$ ssi $b' \rightarrow a$ dans S.

Dans cette définition, nous identifions a et a' pour tout $a \in C$. Nous utilisons aussi les notations suivantes :

- $a(\mathrm{r}_1 \cup \mathrm{r}_2)b$ ssi $a \,\mathrm{r}_1\, b$ ou $a \,\mathrm{r}_2\, b$,

– $a\,\mathrm{r}_1^*\,b$ s'il existe un nombre fini de sommets $a_1, \ldots, a_s$ tel que $a_1 = a$, $a_s = b$ et $a_1\,\mathrm{r}_1 \ldots \mathrm{r}_1\,a_s$.

Nous définissons à partir de ces relations deux types de variables dans les systèmes d'inégalités transitionnels :

– une *variable croissante* $x \in V$ dans S est une variable telle que
  – $x\,\mathrm{r}^*\,a_0\,\mathrm{r}_d\,a_1\,(\mathrm{r} \cup \mathrm{r}_d)^*\,x$
  – ou bien $x\,(\mathrm{r} \cup \mathrm{r}_d \cup \mathrm{r}_g)\,y$ où y est une variable croissante,
– une *variable décroissante* $x \in V$ dans S est une variable telle que
  – $x\,\mathrm{r}^*\,a_0\,\mathrm{r}_g\,a_1\,(\mathrm{r} \cup \mathrm{r}_g)^*\,x$
  – ou bien $y\,(\mathrm{r} \cup \mathrm{r}_d \cup \mathrm{r}_g)\,x$ où y est une variable décroissante,

Par convention, les constantes sont toujours considérées comme des variables à la fois croissantes et décroissantes. La propriété qui nous intéresse sur ces variables est la suivante.

Proposition 1 *Pour toute variable croissante (respectivement décroissante) x dans un système d'inégalités transitionnel* S *et toute paire $v, v' \models$* S *telle que $v \preceq v'$, on a $v(x) \leq v'(x)$ (respectivement $v(x) \geq v'(x)$).*

Preuve : Considérons un couple de valuations v, v' tel que $v, v' \models$ S et $v \preceq v'$. Par définition de l'ordre $\preceq$, nous avons $v(a) \leq v(b)$ ssi $v'(a) \leq v'(b)$ pour tout $a, b \in V \cup C$. Ceci implique que si $a\,\mathrm{r}^*\,b$ alors $v(b) \leq v(a)$ et $v'(b) \leq v'(a)$. De plus si $v, v' \models$ S et $a\,\mathrm{r}_d\,b$ alors $v(b) \leq v'(a)$ par définition de la satisfaction d'un système d'inégalités transitionnel.

Nous procédons par l'absurde et supposons qu'une variable croissante x vérifie $v(x) > v'(x)$. Nous montrons par induction que pour tout $i \geq 0$ s'il existe une suite de relations de la forme

$$x\,\mathrm{r}^*\,a_0\,\mathrm{r}_d\,b_0 \cdots b_{i-1}\,\mathrm{r}^*\,a_i\,\mathrm{r}_d\,b_i$$

alors on a $v(b_i) < v(x)$ et $v'(b_i) < v'(x)$. Pour $i = 0$, nous utilisons les propriétés précédentes pour montrer que $v(b_0) \leq v'(a_0) \leq v'(x)$ car $x\,\mathrm{r}^*\,a_0$ et $a_0\,\mathrm{r}_d\,b_0$. Comme $v(x) > v'(x)$ nous avons bien $v(b_0) < v(x)$ et comme $v \preceq v'$ ceci implique que $v'(b_0) < v'(x)$.

Par induction, supposons que $v'(b_{i-1}) < v'(x)$ et $v(b_{i-1}) < v(x)$. Comme $b_{i-1}\,\mathrm{r}^*\,a_i\,\mathrm{r}_d\,b_i$ nous avons $v(b_i) \leq v'(a_i) \leq v'(b_{i-1})$. Par hypothèse d'induction on a $v'(b_{i-1}) < v'(x)$ donc on obtient $v(b_i) < v'(x)$. Puisque $v'(x) < v(x)$ et $v \preceq v'$ nous avons bien $v(x) < v(b_i)$ et $v'(x) < v'(b_i)$.

Par définition si x est une variable croissante alors il existe une suite de relation de la forme

$$x\,\mathrm{r}^*\,a_0\,\mathrm{r}_d\,b_0 \cdots b_{i-1}\,\mathrm{r}^*\,a_i\,\mathrm{r}_d\,b_s$$

telle que que $b_s = x$. D'après le résultat précédent, nous obtenons une contradiction car dans ce cas $v(x) < v(b_s)$ et $b_s = x$. Ceci implique que $v(x) \leq v'(x)$.

Si y est une variable croissante et $x\,(\mathrm{r} \cup \mathrm{r}_d \cup \mathrm{r}_g)\,y$ alors nous supposons que $y < y'$.

– si $x\,\mathrm{r}\,y$ alors $v(y) \leq v(x)$ et $v'(y) \leq v'(x)$. Par définition de $\preceq$ ceci implique que $v'(y) - v(y) \leq v'(x) - v(x)$ et comme $v'(y) \geq v(y)$ ceci nous donne $v'(x) \geq v(x)$.

– si $x \, \mathrm{r}_d \, y$ alors $v'(y) \leq v(x)$ et par transitivité nous obtenons $y \leq x$ ce qui nous ramène au cas précédent.
– si $x \, \mathrm{r}_g \, y$ alors $v(y) \leq v'(x)$. Par l'absurde si nous supposons que $v(x) > v'(x)$ alors par transitivité nous obtenons $v(x) > v(y)$ ce qui nous ramène au premier cas. Nous avons donc une contradiction puisque dans ce cas $0 \leq v'(y) - v(y) \leq v(x') - v(x)$.

La preuve pour les variables décroissantes est similaire. □

La construction de S'_π se divise en plusieurs étapes. Nous montrons d'abord le résultat suivant.

Proposition 2 *À partir de S_π, on peut construire un système d'inégalités sous forme normale $T^*(\mathrm{S}_\pi)$ vérifiant les propriétés suivantes :*

(a) $\{(\langle q,v\rangle,\langle q,v'\rangle) \mid v,v' \models \mathrm{S}_\pi$ *et* $v \preceq v'\} \subseteq \{(\langle q,v\rangle,\langle q,v'\rangle) \mid v,v' \models T^*(\mathrm{S}_\pi)\}$,

(b) $\{(\langle q,v\rangle,\langle q,v'\rangle) \mid v,v' \models T^*(\mathrm{S}_\pi)\} \subseteq \{(\langle q,v\rangle,\langle q,v'\rangle) \mid v,v' \models \mathrm{S}_\pi\}$,

(c) pour toute variable croissante $x \in V$ dans $T^*(\mathrm{S}_\pi)$ on a $x \rightarrow x'$,
pour toute variable décroissante $x \in V$ dans $T^*(\mathrm{S}_\pi)$ on a $x' \rightarrow x$,
pour tout $a,b \in V \cup C$ on a $a \rightarrow b$ ssi $a' \rightarrow b'$ dans $T^*(\mathrm{S}_\pi)$.

Lorsqu'un système d'inégalités vérifie la condition **(c)**, nous disons qu'il est *complet*. L'idée derrière ce résultat est que l'on peut compléter le système d'inégalités sans éliminer de solutions de S_π de la forme v, v' telle que $v \preceq v'$ (et sans rajouter de solutions).

Preuve : Soit $T(\mathrm{S}_\pi)$ la forme normale du système d'inégalités obtenu à partir de S_π en ajoutant les arcs suivants s'ils n'existent pas déjà.

– Pour tout $x \in V$, si x est croissante dans S_π alors on ajoute un arc $x \xrightarrow{0} x'$.
– Pour tout $x \in V$, si x est décroissante dans S_π alors on ajoute un arc $x' \xrightarrow{0} x$.
– Pour tout $a,b \in V \cup C$ tel qu'il existe un arc $a \rightarrow b$ on ajoute un arc $a' \xrightarrow{0} b'$.
– Pour tout $a',b' \in V' \cup C$ tel qu'il existe un arc $a' \rightarrow b'$ on ajoute un arc $a \xrightarrow{0} b$.

Par construction, si $v,v' \models T(\mathrm{S}_\pi)$ alors $v,v' \models \mathrm{S}_\pi$ car les arcs ajoutés impliquent des contraintes supplémentaires et la normalisation ne changent pas l'ensemble des solutions. De plus, si $v,v' \models \mathrm{S}_\pi$ et $v \preceq v'$ alors $v,v' \models T(\mathrm{S}_\pi)$. En effet, supposons que $v,v' \models \mathrm{S}_\pi$ et $v \preceq v'$. Nous montrons que le couple v,v' satisfait le système obtenu avant la normalisation puisque cette opération ne change pas l'ensemble des solutions. Les contraintes induites par les arcs déjà présents dans S_π et les contraintes de périodicité sont satisfaites par hypothèse. Pour les arcs ajoutés, plusieurs cas se présentent :

– si on a rajouté un arc $x \xrightarrow{0} x'$ car x est croissante, alors comme x est croissante d'après la Proposition 1 on a bien $v'(x) - v(x) \geq 0$ car $v \preceq v'$,
– de même, si on a rajouté un arc $x' \xrightarrow{0} x$ car x est décroissante on a bien $v(x) - v'(x) \geq 0$,
– enfin, pour tout arc $a \xrightarrow{0} b$ ou $a' \xrightarrow{0} b'$ rajouté à cause de l'un des deux derniers points de la construction, la contrainte induite est aussi satisfaite car par propriété de $\preceq$,

l'ordre des éléments de $V \uplus C$ est le même que celui de $V' \uplus C$.

Pour tout $i \in \mathbb{N}$, nous définissons $T^i(\mathrm{S}_\pi)$ tel que $T^0(\mathrm{S}_\pi) = T(\mathrm{S}_\pi)$ et $T^{i+1}(\mathrm{S}_\pi) = T(T^i(\mathrm{S}_\pi))$. Il existe une position $i \geq 0$ telle que $T^{i+1}(\mathrm{S}_\pi) = T^i(\mathrm{S}_\pi)$. En effet, le nombre d'arcs possibles dans un système d'inégalités normalisé est borné et les opérations que nous effectuons ne suppriment jamais d'arc. Il existe donc un entier i tel que l'on ne peut plus rajouter d'arcs à $T^i(\mathrm{S}_\pi)$. Nous posons alors $T^*(\mathrm{S}_\pi) = T^i(\mathrm{S}_\pi)$.

Par construction, $T^*(\mathrm{S}_\pi)$ est complet. De plus, pour tout $i \geq 0$ si $v, v' \models T^{i+1}(\mathrm{S}_\pi)$ alors $v, v' \models T^i(\mathrm{S}_\pi)$. De même, pour tout $i \geq 0$ si $v, v' \models T^i(\mathrm{S}_\pi)$ et $v \preceq v'$ alors $v, v' \models T^{i+1}(\mathrm{S}_\pi)$. Par induction, nous obtenons que si $v, v' \models T^*(\mathrm{S}_\pi)$ alors $v, v' \models \mathrm{S}_\pi$ et si $v, v' \models \mathrm{S}_\pi$ et $v \preceq v'$ alors $v, v' \models T^*(\mathrm{S}_\pi)$. Les propriétés **(a)** et **(b)** sont donc bien vérifiées. $\square$

Pour obtenir une représentation d'un ensemble d'états à partir desquels il existe un chemin répétant infiniment la boucle π, nous allons composer itérativement la représentation de $T^*(\mathrm{S}_\pi)$ avec elle-même. À partir de $T^*(\mathrm{S}_\pi)$, nous construisons inductivement le système d'inégalités S_π^i pour tout $i \in \mathbb{N}$ de la façon suivante.

- $\mathrm{S}_\pi^0 = T^*(\mathrm{S}_\pi)$,
- S_π^{i+1} est la forme normale de $\mathrm{S}_\pi^i \cdot T^*(\mathrm{S}_\pi)$.

Nous définissons ci-dessous une condition d'arrêt pour cette étape de la construction. Nous arrêtons lorsque le graphe de la forme S_π^i obtenu satisfait certaines conditions qui nous seront utiles pour la dernière étape de la preuve.

Proposition 3 *Il existe un indice $0 \leq i_f \leq |V|$ tel que $\mathrm{S}_\pi^{i_f}$ vérifie les propriétés suivantes.*

(i) $\mathrm{S}_\pi^{i_f}$ est complet.

(ii) Il n'existe pas d'arc de la forme $b \to a'$ ou $c' \to a'$ tel que a n'est pas une variable croissante et c est une variable croissante dans $T^*(\mathrm{S}_\pi)$.

(iii) Il n'existe pas d'arc de la forme $a' \to b$ ou $a' \to c'$ tel que a n'est pas une variable décroissante et c est une variable décroissante dans $T^*(\mathrm{S}_\pi)$.

Preuve : La propriété **(i)** est directe d'après la construction de $T^*(\mathrm{S}_\pi)$. On peut facilement montrer par induction que pour tout $i \geq 0$ toute variable croissante x on a un arc $x \to x'$ dans S_π^i. Le cas de base est une conséquence de la Proposition 2. De plus si par on suppose qu'il existe un arc $x \to x'$ dans S_π^i alors comme il existe aussi un arc $x \to x'$ dans $T^*(\mathrm{S}_\pi)$ on a bien $x \to x'$ dans $\mathrm{S}_\pi^i \cdot T^*(\mathrm{S}_\pi)$ par transitivité. Les autres conditions de complétude peuvent être montrée de la même manière.

Nous montrons maintenant que la construction vérifie **(ii)**. Nous montrons d'abord que pour tout $i \in \mathbb{N}$ tous les arcs ayant pour destination a' dans S_π^i tel que a n'est pas croissante dans $T^*(\mathrm{S}_\pi)$ ont une origine une variable qui n'est pas une variable croissante (primée ou non). Pour le cas $i = 0$, on a $\mathrm{S}_\pi^0 = T^*(\mathrm{S}_\pi)$. Les deux cas possibles sont des arcs de la forme $b \to a'$ et $b' \to a'$. Dans les deux cas, si b est une variable croissante, on obtient une contradiction car on a respectivement $a \, \mathrm{r}_d \, b$ et $a \, \mathrm{r} \, b$ selon le cas, ce qui implique que a devrait être croissante. Pour l'étape d'induction, supposons que la propriété soit vraie pour tout S_π^i. Par construction, S_π^{i+1} est la forme normale de $S_\pi^i \cdot T^*(S_\pi)$ et par définition S_π^i et $T^*(S_\pi)$ sont sous forme normale. Ceci implique qu'on a un arc de la forme $b \to a'$ dans S_π^{i+1} ssi il existe un sommet c tel que $b \to c'$ appartient à S_π^i et $c \to a'$ appartient à $T^*(S_\pi)$. La variable c n'est pas croissante dans $T^*(S_\pi)$ sinon on obtient que a est croissante. D'après l'hypothèse d'induction, b n'est donc pas croissante non plus car l'arc $b \to c'$ a pour destination une variable qui n'est pas croissante. Pour le cas où on a un arc de la forme $b' \to a'$ dans S_π^{i+1} alors la propriété est directe car par définition de la composition, on doit avoir $b' \to a'$ dans $T^*(S_\pi)$.

Il nous reste donc à montrer qu'il existe un entier $0 \leq i_f \leq |V|$ tel que $S_\pi^{i_f}$ ne contient pas d'arc de la forme $b \to a'$ tel que a et b ne sont pas des variables croissantes. Remarquons que si a et b ne sont pas des variables croissantes dans $T^*(S_\pi)$, il n'existe pas de séquence de la forme $a_1 \,\mathrm{r}_d\, a_2 \,\mathrm{r}_d \cdots \mathrm{r}_d\, a_s$ telle que $a_1 = a$, $b_s = b$ et $s > V$. Sinon, il existe deux indices j et j' tels que $a_j = a_{j'}$ et nous obtenons que a_j est croissante par définition. Nous obtenons alors une contradiction puisque par transitivité ceci implique que a est croissante.

Nous montrons maintenant que pour tout $0 \leq i \leq |V|$, il existe un arc $b \to a'$ dans S_π^i tel que a et b ne sont pas des variables croissantes ssi il existe une séquence de la forme $a_1 \,\mathrm{r}_d\, a_2 \,\mathrm{r}_d \cdots \mathrm{r}_d\, a_s$ telle que $a_1 = a$, $a_s = b$ et $s = i + 1$. Cette propriété est évidente pour le cas où $i = 0$ car $S_\pi^0 = T^*(S_\pi)$. Pour l'étape d'induction, nous rappelons que S_π^{i+1} est la forme normale de $S_\pi^i \cdot T^*(S_\pi)$ et S_π^i, $T^*(S_\pi)$ sont eux-mêmes sous forme normale. Ainsi on a un arc $b \to a'$ ssi il existe c tel que $b \to c'$ appartient à S_π^i et $c \to a'$ appartient à $T^*(S_\pi)$. Nous pouvons facilement en déduire qu'il existe une séquence $a \,\mathrm{r}_d\, c \,\mathrm{r}_d \cdots \mathrm{r}_d\, b$ où le nombre d'éléments de $c \,\mathrm{r}_d \cdots \mathrm{r}_d\, b$ est $i+1$ par hypothèse d'induction. Réciproquement s'il existe une séquence $a \,\mathrm{r}_d\, c \cdots \mathrm{r}_d\, b$ de longueur $i + 2$ alors il existe un arc $b \to c'$ dans S_π^i et un autre $c \to a'$ dans $T^*(S_\pi)$. Par construction de S_π^{i+1} on a donc un arc $b \to a'$. La propriété est donc bien vérifiée.

Ces résultats impliquent que si on prend i_f supérieur à la taille de la plus grande séquence $a \,\mathrm{r}_d \cdots \mathrm{r}_d\, b$ où a, b sont non croissantes alors tous les arcs ayant pour destination a' tel que a est non croissante ont pour origine un sommet de la forme b' tel que b est non croissante. Comme la longueur maximale d'une séquence $a \,\mathrm{r}_d \cdots \mathrm{r}_d\, b$ est bornée par $|V|$, la propriété est bien vérifiée.

La preuve du point **(iii)** est similaire. □

Nous notons S''_π le système d'inégalités transitionnel obtenu de la façon suivante. S'il existe un indice $0 \leq i \leq i_f$ tel que le système S_π^i contient une variable a pour laquelle $mod(a) \neq mod(a')$ alors $S_\pi^i = S_{\mathrm{nil}}$ (par définition de la composition) et dans ce cas nous posons $S''_\pi = S_{\mathrm{nil}}$. Sinon S''_π est obtenu à partir de $S_\pi^{i_f}$ en effectuant la transformation suivante : s'il existe un arc $a \xrightarrow{d} b$ alors on remplace l'arc $a' \xrightarrow{d'} b'$ par $a' \xrightarrow{\max(d,d')} b'$. Nous supposons pour la suite que $S''_\pi \neq S_{\mathrm{nil}}$.

Proposition 4 *Le système d'inégalités S''_π vérifie les propriétés ci-dessous.*

(a) Si $v, v' \models (T^*(S_\pi))^{i_f}$ et $v \preceq v'$ alors $v, v' \models S''_\pi$.

(b) Si $v, v' \models S''_\pi$ alors $\langle q, v\rangle \xrightarrow{(\pi)^{i_f}} \langle q, v'\rangle$.

(c) pour tout $v \models (S''_\pi)_{|V}$, il existe deux valuations v_1 et v_2 telles que $v_1 \preceq v_2$, $v, v_1 \models S''_\pi$ et $v_1, v_2 \models S''_\pi$.

Avant de prouver ce résultat nous voulons souligner quelques conséquences de celui-ci.

- D'après **(a)**, tout couple de valuations v, v' tel que $v \preceq v'$ qui satisfait $(T^*(S_\pi))^{i_f}$ satisfait S''_π. Nous verrons plus tard que pour tout $0 \leq i \leq i_f$, s'il existe une séquence finie de valuations $v_0, v_1, \ldots v_i$ telle que $v_j, v_{j+1} \models T^*(S_\pi)$ pour tout $0 \leq j < i$ alors $v_0, v_i \models (T^*(S_\pi))^i$ afin de relier ce résultat avec la Proposition 2(a).
- En utilisant le Lemme 60 et les propriétés **(b)** et **(c)**, on peut déduire l'existence d'un chemin infini répétant infiniment la boucle π à partir de $\langle q, v\rangle$ tel que $v \models (S''_\pi)_{|V}$.

Preuve : Nous commençons par prouver la propriété **(a)**. Soit v, v' un couple de valuations tel que $v, v' \models (T^*(S_\pi))^{i_f}$ et $v \preceq v'$. Par l'absurde, supposons que $v, v' \not\models S''_\pi$. Alors il existe deux sommets a' et b' tels que $b' \xrightarrow{d_m} a'$ et $v'(a) - v'(b) < d_m$. Comme par construction $d_m = \max(d, d')$ tel que $b \xrightarrow{d} a$ et $b' \xrightarrow{d'} a'$ dans $(T^*(S_\pi))^{i_f}$ on a $v(a) - v(b) \geq d$ et $v'(a) - v'(b) \geq d'$. Comme $v \preceq v'$ et $v(a) - v(b) \geq 0$ on a $v'(a) - v'(b) \geq v(a) - v(b) \geq d$ ce qui implique que $v'(a) - v'(b) \geq \max(d, d')$ car $v'(a) - v'(b)$ est aussi supérieur à d'. Nous obtenons donc une contradiction.

D'après la construction S''_π, la propriété **(b)** peut facilement être déduite de la Proposition 2(b) et de la définition de l'opération de composition de systèmes d'inégalités transitionnels.

La dernière propriété est plus compliquée à prouver. D'abord, notons que dans la construction de S''_π on n'ajoute pas d'arc à S^i_π et les différents poids des arcs augmentent donc S''_π est complet et hérite des propriétés **(ii)** et **(iii)** de $S^{i_f}_\pi$ (voir Proposition 3).

Si $v \models (S''_\pi)_{|V}$ alors il existe une valuation v' telle que $v, v' \models S''_\pi$ Nous définissons la valuation v_1 de la façon suivante :

$$v_1(a) = \begin{cases} v'(a) + N_1 & \text{si } a \text{ est croissante et pas décroissante} \\ v'(a) - N_1 & \text{si } a \text{ est décroissante et pas croissante} \\ v'(a) & \text{sinon} \end{cases}$$

où N_1 est un multiple de K vérifiant

1. $N_1 > \max\{|v'(b) - v'(a)| : a, b \in V \cup C\}$,
2. $v'(a) + N_1 > M$ pour tout $a \in V \cup C$,
3. $v'(a) - N_1 < m$ pour tout $a \in V \cup C$.

Notons que par définition de N_1, si b n'est ni croissante ni décroissante ou à l'inverse les deux à la fois alors

- pour tout $a \in V$ qui est croissante et pas décroissante $v_1(a) > v_1(b)$,
- pour tout $a \in V$ qui est décroissante et pas croissante $v_1(b) > v_1(a)$.

Le fait que $v, v_1 \models \mathrm{S}''_\pi$ peut être montrer par analyse de cas.

- les contraintes de périodicité sont vérifiées car N_1 est un multiple de K et $v, v' \models \mathrm{S}''_\pi$.
- les contraintes induites par les arcs de la forme $b \xrightarrow{d} a$ sont satisfaites car $v \models (\mathrm{S}''_\pi)_{|V}$.
- s'il existe un arc de la forme $b' \xrightarrow{d} a$ alors d'après **(iii)** b est une variable décroissante et donc $v_1(b) \leq v'(b)$. Or comme $v, v' \models \mathrm{S}''_\pi$ on a $v(a) - v'(b) \geq d$ ce qui implique que $v(a) - v_1(b) \geq d$ est bien vérifié.
- s'il existe un arc de la forme $b \xrightarrow{d} a'$ alors d'après **(ii)** a est une variable croissante et donc $v_1(a) \geq v'(a)$. Or comme $v, v' \models \mathrm{S}''_\pi$ on a $v(a) - v'(b) \geq d$ ce qui implique que $v_1(a) - v(b) \geq d$ est bien vérifié.
- enfin pour tout arc de la forme $b' \xrightarrow{d} a'$ on a $v'(a) - v'(b) \geq d$ car $v, v' \models \mathrm{S}''_\pi$. Si a est croissante et pas décroissante alors par **(ii)** b est aussi croissante. Ainsi $v_1(a) = v'(a) + N_1$ et $v_1(b) = v'(b) + N_1$ et donc $v_1(a) - v_1(b) \geq d$. Les autre cas peuvent être traités de la même manière.

De plus, nous avons $v_1 \models (\mathrm{S}''_\pi)_{|V}$. On sait déjà que $v_1 \models (\mathrm{S}''_\pi)_{|V'}$ d'après le résultat ci-dessus. Or, comme par construction de S''_π on a $mod(a) = mod(a')$ et $a \xrightarrow{d} b$ implique $a' \xrightarrow{d'} b'$ tel que $d' \geq d$ pour tout $a, b \in V \cup C$ la propriété peut facilement être déduite.

Il nous reste à montrer que l'on peut construire à partir de v_1 une valuation v_2 telle que $v_1, v_2 \models \mathrm{S}''_\pi$ et $v_1 \preceq v_2$. Nous posons N_2 comme étant un multiple de K vérifiant :

$$\begin{aligned} N_2 = \max\big(&\max\{|v'(a) - v'(b)| + d \mid b' \xrightarrow{d} a \text{ appartient à } \mathrm{S}''_\pi\}, \\ &\max\{|v'(a) - v'(b)| + d \mid b \xrightarrow{d} a' \text{ appartient à } \mathrm{S}''_\pi\}\big) \end{aligned}$$

et nous définissons v_2 par

$$v_2(a) = \begin{cases} v_1(a) + N_2 & \text{si } a \text{ est croissante et pas décroissante} \\ v_1(a) - N_2 & \text{si } a \text{ est décroissante et pas croissante} \\ v_1(a) & \text{sinon} \end{cases}$$

La preuve que $v_1, v_2 \models \mathrm{S}''_\pi$ est très semblable à celle de $v, v_1 \models \mathrm{S}''_\pi$:

- les cas des contraintes modulo et des arcs de la forme $a \to b$ ou $a' \to b'$ sont similaires en considérant que $v_1 \models (\mathrm{S}''_\pi)_{|V}$ et $v_1 \models (\mathrm{S}''_\pi)_{|V'}$ d'après les résultats qui précèdent.
- s'il existe un arc de la forme $b' \xrightarrow{d} a$ alors d'après **(iii)** b est une variable décroissante et donc $v_2(b) \geq v_1(b) - N_2$. Par définition de la valeur N_2 nous obtenons que $v_2(b) \geq v_1(b) - |v'(a) - v'(b)| + d$. Donc $v_1(a) - v_2(b) \geq v_1(a) - v_1(b) - |v'(a) - v'(b)| + d \geq d$. La paire v_1, v_2 vérifie donc la contrainte induite par cet arc.
- le cas des arcs de la forme $b \xrightarrow{d} a'$ peut être traité de la même façon en utilisant le fait que d'après **(ii)** a est une variable croissante.

Enfin, nous montrons que $v_1 \preceq v_2$. Cette propriété découle directement des observations suivantes sur la définition de v_1 et v_2 :

– pour tout $a \in V$ qui est croissante et pas décroissante on a $v_2(a) \geq v_1(a) \geq M$,

– pour tout $a \in V$ qui est décroissante et pas croissante on a $v_2(a) \leq v_1(a) \leq m$,

– pour les autres éléments $a \in V \cup C$ on a $v_2(a) = v_1(a)$ et

  – pour tout $a \in V$ qui est croissante et pas décroissante $v_1(a) > v_1(b)$

  – pour tout $a \in V$ qui est décroissante et pas croissante $v_1(b) > v_1(a)$.

On peut facilement vérifier en utilisant ces propriétés que par définition de v_2, pour tout $a, b \in V \cup C$ on a $v_1(a) - v_1(b) \geq 0$ implique $v_2(a) - v_2(b) \geq v_1(a) - v_1(b)$. □

Nous posons $\mathrm{S}'_\pi = (\mathrm{S}''_\pi)_{|V}$ et montrons maintenant que

$$X_\pi^{\preceq} \subseteq \{\langle q, v\rangle \mid \text{ il existe } \mathrm{S}'_\pi \in \mathcal{S}'_\pi \text{ tel que } v \models \mathrm{S}'_\pi\} \subseteq X_\pi^\infty$$

où $\mathcal{S}'_\pi$ est l'ensemble composé des systèmes d'inégalités locaux S'_π obtenus à partir des systèmes S_π de $\mathcal{S}_\pi$ en utilisant la transformation ci-dessus.

Si $\langle q_0, v_0\rangle \in X_\pi^{\preceq}$ alors il existe un chemin infini de la forme $\langle q, v_0\rangle \xrightarrow{\pi}_U \langle q, v_1\rangle \xrightarrow{\pi}_U \langle q, v_2\rangle \xrightarrow{\pi}_U \cdots$ tel que pour tout $i \geq 0$ on a $v_i \preceq v_{i+1}$. Ceci est une conséquence du résultat de simulation dans les WIPC*-automates généralisés établi par le Lemme 60. En effet, d'après ce résultat pour tout chemin π_v à partir de $\langle q, v\rangle$ et toute valuation v' telle que $v \preceq v'$ il existe un chemin $\pi_{v'}$ à partir de $\langle q, v'\rangle$ qui suit exactement les mêmes transitions tel que $\pi_v(i) \preceq \pi_{v'}(i)$ pour tout $i \in \mathbb{N}$. Donc si $\langle q, v_0\rangle \xrightarrow{\pi}_U \langle q, v_1\rangle$ et $v_0 \preceq v_1$ alors il existe v_2 tel que $\langle q, v_1\rangle \xrightarrow{\pi}_U \langle q, v_2\rangle$ et $v_1 \preceq v_2$ et ainsi de suite. On peut donc répéter à chaque fois la même séquence de transitions en satisfaisant tout le temps les mêmes ensembles de contraintes par propriété de $\preceq$. En conséquence, il existe $\mathrm{S}_\pi \in \mathcal{S}_\pi$ tel que pour tout $i \geq 0$ on a $v_i, v_{i+1} \models \mathrm{S}_\pi$ (voir la preuve du Lemme 67 sur la construction des représentations pour les relations d'accessibilité). D'après la Proposition 2(a), pour tout $i \geq 0$ nous avons donc $v_i, v_{i+1} \models T^*(\mathrm{S}_\pi)$ et par définition de l'opération de composition des systèmes d'inégalités $v_0, v_{i_f} \models (T^*(\mathrm{S}_\pi))^{i_f}$. Comme par transitivité $v_0 \preceq v_{i_f}$, le système S''_π construit à partir de $(T^*(\mathrm{S}_\pi))^{i_f}$ est différent de $\mathrm{S}_{\mathrm{nil}}$ car par définition de $\preceq$ on doit avoir $mod(a) = mod(a')$ pour tour $a \in V \cup C$. De plus, on a $v_0, v_{i_f} \models \mathrm{S}''_\pi$ d'après la Proposition 4(a). Donc $v_0 \models \mathrm{S}'_\pi$ pour un système d'inégalités $\mathrm{S}'_\pi \in \mathcal{S}'_\pi$.

Supposons maintenant que $\langle q_0, v_0\rangle \in \{\langle q, v\rangle \mid \text{ il existe } \mathrm{S}'_\pi \in \mathcal{S}'_\pi \text{ tel que } v \models \mathrm{S}'_\pi\}$. Il existe donc un système d'inégalités S_π dans $\mathcal{S}_\pi$ utilisé pour la construction du système d'inégalités S'_π tel que $v_0 \models \mathrm{S}'_\pi$. Par construction, S'_π est la projection sur l'ensemble des variables d'un système d'inégalités transitionnel S''_π vérifiant les propriétés énoncées dans la Proposition 4. D'après la condition (c) de la Proposition 4, il existe deux valuations v et v' telles que $v' \preceq v''$, $v_0, v \models \mathrm{S}''_\pi$ et $v, v' \models \mathrm{S}''_\pi$. En utilisant la propriété (b) de la Proposition 4 nous obtenons donc qu'il existe un chemin $\langle q, v_0\rangle \xrightarrow{(\pi)^{i_f}}_U \langle q, v\rangle \xrightarrow{(\pi)^{i_f}}_U \langle q, v'\rangle$. En itérant le Lemme 60 sur la partie $\langle q, v\rangle \xrightarrow{(\pi)^{i_f}}_U \langle q, v'\rangle$ (car $v \preceq v'$) nous obtenons que $\langle q, v_0\rangle \xrightarrow{(\pi)^\omega}_U$ et donc $\langle q, v_0\rangle \in X_\pi^\infty$. □

Synthèse III

Nous avons considéré dans cette dernière partie les extensions des logiques temporelles arborescentes avec des domaines concrets. En plus de nos motivations générales liées à la spécification de propriétés quantitatives sur les modèles des systèmes informatiques, ces extensions permettent la généralisation et le raffinement de certains résultats et certaines approches du cas linéaire.

Nous avons défini d'abord une méthode à base d'automates avec une abstraction des modèles arborescents qui généralise les approches utilisées dans les Parties II et III. Cette méthode introduit une nouvelle représentation symbolique et une nouvelle construction d'automate. Les automates que nous manipulons dans ce cadre sont des automates d'arbres alternants. Nous avons ensuite appliqué cette méthode pour prouver la décidabilité des problèmes de la satisfaisabilité et du model-checking pour une famille de logiques de la forme $\mathrm{CCTL}^\star(\mathcal{D})$ qui vérifient une propriété générale d'abstraction des modèles. Cette propriété permet de vérifier facilement qu'un modèle symbolique de $\mathrm{CCTL}^\star(\mathcal{D})$ correspond à un modèle concret à l'aide d'un automate d'arbres. Ces résultats raffinent plusieurs résultats concernant divers fragments linéaires de la forme $\mathrm{CLTL}(\mathcal{D})$ introduits dans la littérature qui vérifie cette même propriété [BC02, DD07, Dem04].

Cependant les domaines concrets qui manipulent des entiers utilisés dans la Partie II ne font pas partie de cette famille. Nous avons donc utilisé une autre méthode reposant sur la définition d'un beau préordre afin de construire l'ensemble des états d'un $\mathrm{WIPC}^\star$-automate qui satisfont une formule du fragment existentiel $\mathrm{ECTL}^\star(\mathrm{WIPC}^\star)$ de la logique $\mathrm{CCTL}^\star(\mathrm{WIPC}^\star)$. Nous rappelons que l'ensemble des formules atomiques de ce langage est composé de contraintes de périodicité et de comparaisons entre les entiers. Nous pouvons déduire de cette construction la décidabilité du problème du model-checking de $\mathrm{ECTL}^\star(\mathrm{WIPC}^\star)$ et plus généralement de $\mathrm{ECTL}^\star(\mathrm{IPC}^\star)$. Ce résultat raffine donc une partie du résultat de décidabilité pour le problème du model-checking de $\mathrm{CLTL}(\mathrm{IPC}^\star)$ démontré dans le Chapitre 4.

Plusieurs problèmes restent ouverts. Tout d'abord la complexité du problème du model-checking pour le fragment existentiel de la logique $\mathrm{CCTL}^\star(\mathrm{WIPC}^\star)$ nécessiterait une analyse fastidieuse de l'algorithme que nous avons défini dans le Chapitre 8. La borne inférieure de complexité pour ce problème n'est pas non plus connue, ce qui n'exclut pas l'existence d'une méthode plus efficace en terme de complexité. Notons cependant que la méthode que nous avons utilisée résout un problème plus général que celui du model-checking car nous construisons une représentation de l'ensemble des états vérifiant la formule en entrée.

De plus, la décidabilité du problème de la satisfaisabilité de $\mathrm{CCTL}^\star(\mathrm{WIPC}^\star)$ ne peut pas être déduite des résultats de ce chapitre. Il est possible que la méthode à base d'automate d'arbres alternant que nous avons définie permette de résoudre ce problème. Plus généralement, nous conjecturons que cette méthode à base d'automate peut être adaptée à plusieurs domaines concrets $\mathcal{D}$ tels que $\mathrm{CCTL}^\star(\mathcal{D})$ n'appartient pas à la famille de logiques traitées dans le Chapitre 7. Ceci nécessite essentiellement de caractériser à l'aide d'un automate l'ensemble des modèles symboliques de $\mathrm{CCTL}^\star(\mathcal{D})$ qui correspondent à un modèle concret.

Conclusion

Nous avons défini un cadre général pour l'extension des logiques temporelles avec des contraintes atomiques induites par des domaines concrets. Cette famille de langages permet d'exprimer des propriétés quantitatives sur des représentations symboliques des systèmes informatiques, en particulier sur des contraintes mettant en relation des compteurs lorsque les variables que l'on manipule prennent des valeurs entières. Nous avons démontré plusieurs résultats de décidabilité et de complexité pour les problèmes de satisfaisabilité et de model-checking liés à ces logiques en utilisant principalement des méthodes basées sur des traductions vers des automates.

Les différents résultats que nous avons démontrés peuvent se classer dans trois principales catégories qui correspondent aux différentes parties de ce document.

- les logiques temporelles linéaires étendues avec des contraintes sur les entiers,
- les logiques temporelles linéaires étendues avec des mécanismes de stockage des valeurs,
- les logiques temporelles arborescentes étendues sur les domaines concrets.

En ce qui concerne les logiques temporelles linéaires étendues avec des contraintes sur les entiers, nous avons établi une classification de différents fragments restreignant principalement

- l'ensemble des contraintes utilisées,
- le nombre de variables des formules,
- et la longueur temporelle des formules.

Il apparaît que la cause prédominante de l'indécidabilité pour cette famille de logiques est la présence de contraintes de la forme $x = y + 1$. Cette remarque n'est pas très surprenante mais ce qui l'est plus est la difficulté de retrouver des fragments décidables en présence de telles contraintes ou plus généralement de mécanisme de comptage dans la logique. Nous avons établi pour de telles logiques les limites entre décidabilité et indécidabilité. Le seul fragment décidable à la fois pour les problèmes de model-checking et de satisfaisabilité en présence de mécanisme de comptage est le fragment composé des formules ayant une longueur temporelle de un construite sur une seule variable. Ces problèmes sont même PSPACE-complets comme le fragment de LTL correspondant. Cependant, en adaptant la méthode utilisée pour prouver la décidabilité, nous obtenons que le model-checking du fragment sans quantificateur de l'arithmétique de Presburger sur les automates à un compteur est aussi PSPACE-complet. À l'inverse, en l'absence de tels mécanismes de comptage, on peut beaucoup plus facilement définir des extensions décidables de LTL. Ainsi, nous avons montré que l'extension de LTL avec un large ensemble de contraintes sur les entiers qui ne contient pas de contraintes de la forme $x = y+1$, que nous appelons ensemble de contraintes qualitatives, a des problèmes de satisfaisabilité et de model-checking PSPACE-complets.

Dans une autre partie nous avons considéré une extension de LTL avec des mécanismes de stockage des valeurs qui permet d'exprimer des contraintes de répétition d'une valeur. Les résultats de décidabilité que nous avons démontrés complètent les résultats connus sur de telles logiques. Nous établissons aussi un autre type de relation entre ce formalisme et les systèmes à compteurs car la technique que nous utilisons repose sur une réduction vers un problème pour les réseaux de Petri.

Enfin dans la dernière partie nous avons considéré les extensions des logiques arborescentes sur les domaines concrets. Nous avons défini une extension des approches à base d'automates de mots utilisées dans le cas linéaire et appliquée celle-ci à une famille générale de domaines concrets. Nous obtenons la décidabilité des problèmes de la satisfaisabilité et du model-checking pour cette sous-famille de logiques qui permettent de vérifier qu'un modèle concret correspond à un modèle abstrait en effectuant des tests locaux. Puis nous avons utilisé une approche constructive pour prouver la décidabilité du problème de model-checking pour le fragment existentiel de $\mathrm{CTL}^{\star}$ étendu avec le langage de contraintes qualitatives utilisé dans la partie sur les logiques temporelles linéaires, généralisant ainsi une partie des résultats de ce chapitre.

Nous avons tout au long de ce document discuté de plusieurs extensions possibles pour les différents résultats que nous avons établi. Nous rappelons pour conclure quelques unes des pistes les plus prometteuses.

Concernant les domaines concrets, la perspective la plus réaliste pour le moment est de généraliser une partie des résultats que nous avons établis pour les entiers sur les chaînes de caractères. Les domaines concrets ayant pour domaine d'interprétation les chaînes de caractères apportent quelques difficultés supplémentaires telle que le fait que les ordres considérés soient partiels contrairement aux domaines sur les entiers que nous avons considérés. Les techniques utilisées pour les entiers ne peuvent donc pas s'adapter telle qu'elle aux chaînes de caractères mais il est possible qu'elle puissent être étendue pour traiter ces problèmes.

Une autre direction intéressante concerne les extensions arborescentes. Certains travaux présentés ici peuvent être vus comme des travaux préliminaires dans le sens où il est raisonnable de penser que les techniques utilisées peuvent être adaptées pour d'autres extensions de logiques arborescentes sur des domaines concrets. En particulier, la méthode définie dans le Chapitre 7 peut s'appliquer aux domaines concrets manipulant des entiers à condition de trouver une caractérisation des modèles symboliques arborescents satisfaisables similaire à celle du Chapitre 4. Une autre perspective dans cette direction serait de généraliser les résultats sur les logiques avec mécanismes de stockage pour les logiques temporelles arborescentes.

Enfin, il existe de nombreuses autres perspectives plus générales concernant les restrictions que nous avons considérées tout au long de cette thèse. Dans la mesure du possible, nous avons essayé de garder tout le pouvoir expressif du langage logique et des modèles en restreignant uniquement les contraintes et les ressources syntaxiques des formules. Il est possible d'envisager le problème d'une autre façon en considérant un langage de contraintes le plus large possible et restreignant le langage logique ou les modèles considérés. Ce genre de restrictions laisse la place à de nombreux autres travaux.

Bibliographie

[ABd03] P. Abdulla, A. Bouajjani, and J. d'Orso. Deciding monotonic games. In *CSL'03*, volume 2803 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2003.

[AČJT00] P. Abdulla, K. Čerāns, B. Jonsson, and Y. Tsay. Algorithmic analysis of programs with well quasi-ordered domains. *Information and Computation*, 160 :109–127, 2000.

[AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126 :183–235, 1994.

[AH94] R. Alur and Th. Henzinger. A really temporal logic. *Journal of the ACM*, 41(1) :181–204, 1994.

[BB90] J. Berstel and L. Boasson. Context-free languages. In *Handbook of Theoretical Computer Science, Volume B, Formal models and semantics*, pages 59–102. Elsevier, 1990.

[BBFS98] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati. An access control model supporting periodicity constraints and temporal reasoning. *ACM Transactions on Databases Systems*, 23(3) :231–285, 1998.

[BBH+06] A. Bouajjani, M. Bozga, P. Habermehl, R. Iosif, P. Moro, and T. Vojnar. Programs with lists are counter automata. In *CAV'06*, volume 4144 of *Lecture Notes in Computer Science*, pages 517–531. Springer, 2006.

[BBS06] C. Baier, N. Bertrand, and Ph. Schnoebelen. On computing fixpoints in well-structured regular model-checking, with applications to lossy channel systems. In *LPAR'06*, volume 4246 of *Lecture Notes in Computer Science*, pages 347–361. Springer, 2006.

[BC02] Ph. Balbiani and J.F. Condotta. Computational complexity of propositional linear temporal logics based on qualitative spatial or temporal reasoning. In *FroCoS'02*, volume 2309 of *Lecture Notes in Artificial Intelligence*, pages 162–173. Springer, 2002.

[BDG88] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* Springer, 2nd edition, 1988.

[BDL07] R. Brochenin, S. Demri, and É. Lozes. Reasoning about sequences of memory states. In *LFCS'07*, volume 4514 of *Lecture Notes in Computer Science*, pages 100–114. Springer, 2007.

[BDR03] V. Bruyère, E. Dall'Olio, and J.F. Raskin. Durations, parametric model-checking in timed automata with Presburger arithmetic. In *STACS'03*, volume 2607 of *Lecture Notes in Computer Science*, pages 687–698. Springer, 2003.

[BEH95] A. Bouajjani, R. Echahed, and P. Habermehl. On the verification problem of nonregular properties for nonregular processes. In *LICS'95*, pages 123–133. IEEE, 1995.

[BEM97] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata : application to model-checking. In *CONCUR'97*, volume 1243 of *Lecture Notes in Computer Science*, pages 135–150. Springer, 1997.

[BFLP03] S. Bardin, A. Finkel, J. Leroux, and L. Petrucci. FAST : Fast Acceleration of Symbolic Transition systems. In *CAV'03*, volume 2725 of *Lecture Notes in Computer Science*, pages 118–121. Springer, 2003.

[BFLS06] S. Bardin, A. Finkel, E. Lozes, and A. Sangnier. From pointer systems to counter systems using shape analysis. In *AVIS'06*, 2006.

[BG06] L. Bozzelli and R. Gascon. Branching-time temporal logic extended with qualitative Presburger constraints. In *LPAR'06*, volume 4246 of *Lecture Notes in Artificial Intelligence*, pages 197–211. Springer, 2006.

[BH91] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *IJCAI'91*, volume 1, pages 452–457, 1991.

[BH96] A. Bouajjani and P. Habermehl. Constrained properties, semilinear sets, and Petri nets. In *CONCUR'96*, volume 1119 of *Lecture Notes in Computer Science*, pages 481–497. Springer, 1996.

[BHM03] A. Bouajjani, P. Habermehl, and R. Mayr. Automatic verification of recursive procedures with one integer parameter. *Theoretical Computer Science*, 295(1-3) :85–106, 2003.

[BMS+06] M. Bojańczyk, A. Muscholl, Th. Schwentick, L. Segoufin, and C. David. Two-variable logic on words with data. In *LICS'06*, pages 7–16. IEEE, 2006.

[Boi98] B. Boigelot. *Symbolic methods for exploring infinite state spaces*. PhD thesis, Université de Liège, 1998.

[BS95] P. Blackburn and J. Seligman. Hybrid languages. *Journal of Logic, Language, and Information*, 4(3) :251–272, 1995.

[Büc62] J. R. Büchi. On a decision method in restricted second-order arithmetic. In *Proceedings of the International Congress on logic, Math, and Philosophy of Science (1960)*. Standford University Press, 1962.

[CBGG97] A. Cohn, B. Bennett, J. Gooday, and N. Gotts. RCC : a calculus for region based qualitative spatial reasoning. *GeoInformatica*, 1 :275–316, 1997.

[CC00] H. Comon and V. Cortier. Flatness is not a weakness. In *CSL'00*, volume 1862 of *Lecture Notes in Computer Science*, pages 262–276. Springer, 2000.

[CDG+97] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on : `http://www.grappa.univ-lille3.fr/tata`, 1997. release October, 1rst 2002.

[CE81] E. Clarke and E. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1981.

[CEP95] A. Cheng, J. Esparza, and J. Palsberg. Complexity results for 1-safe nets. *Theoretical Computer Science*, 147(1&2) :117–136, 1995.

[Čer94] K. Čerāns. Deciding properties of integral relational automata. In *ICALP'94*, volume 820 of *Lecture Notes in Computer Science*, pages 35–46. Springer, 1994.

[CES86] E. Clarke, E. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2) :244–263, 1986.

[CGL94] E. Clarke, O. Grumberg, and D. Long. Model-checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5) :1512–1542, 1994.

[CGP99] E. Clarke, O. Grumberg, and D. Peled. *Model checking*. MIT Press, 1999.

[CJ98] H. Comon and Y. Jurski. Multiple counters automata, safety analysis and Presburger arithmetic. In *CAV'98*, volume 1427 of *Lecture Notes in Computer Science*, pages 268–279. Springer, 1998.

[CKS81] A. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *Journal of the ACM*, 28(1) :114–133, 1981.

[CM77] A. Chandra and P. Merlin. Optimal implementation of conjunctive queries in relational databases. In *STOC'77*, pages 77–90. ACM, 1977.

[CR04] C. Chitic and D. Rosu. On validation of XML streams using finite state machines. Technical Report CSRG-489, University of Toronto, 2004.

[DD07] S. Demri and D. D'Souza. An automata-theoretic approach to constraint LTL. *Information and Computation*, 205(3) :380–415, 2007.

[DDG07] S. Demri, D. D'Souza, and R. Gascon. Decidable temporal logic with repeating values. In *LFCS'07*, volume 4514 of *Lecture Notes in Computer Science*, pages 180–194. Springer, 2007.

[DE95] J. Desel and J. Esparza. *Free choice Petri nets*. Cambridge University Press, 1995.

[Dem04] S. Demri. LTL over integer periodicity constraints (extended abstract). In *FOSSACS'04*, volume 2987 of *Lecture Notes in Computer Science*, pages 121–135. Springer, 2004.

[DFGvD06] S. Demri, A. Finkel, V. Goranko, and G. van Drimmelen. Towards a model-checker for counter systems. In *ATVA'06*, volume 4218 of *Lecture Notes in Computer Science*, pages 493–507. Springer, 2006.

[DG05] S. Demri and R. Gascon. Verification of qualitative $\mathbb{Z}$-constraints. In *CONCUR'05*, volume 3653 of *Lecture Notes in Computer Science*, pages 518–532. Springer, 2005.

[DG07] S. Demri and R. Gascon. The effects of bounding syntactic resources on Presburger LTL (extended abstract). In *TIME'07*, pages 94–104. IEEE, 2007.

[DL06] S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. In *LICS'06*, pages 17–26. IEEE, 2006.

[DLN07] S. Demri, R. Lazić, and D. Nowak. On the freeze quantifier in constraint LTL : decidability and complexity. *Information and Computation*, 205(1) :2–24, 2007.

[DPK03] Z. Dang, P. San Pietro, and R. Kemmerer. Presburger liveness verification of discrete timed automata. *Theoretical Computer Science*, 299 :413–438, 2003.

[DS02] S. Demri and Ph. Schnoebelen. The complexity of propositional linear temporal logics in simple cases. *Information and Computation*, 174(1) :84–103, 2002.

[EH83] E. Allen Emerson and J. Halpern. "Sometimes" and "Not Never" revisited : On branching versus linear time. In *POPL'83*, pages 127–140. ACM, 1983.

[EKS03] J. Esparza, A. Kučera, and S. Schwoon. Model-checking LTL with regular valuations for pushdown systems. *Information and Computation*, 186(2) :355–376, 2003.

[EL87] E. Emerson and C.-L. Lei. Modalities for model checking : Branching time logic strikes back. *Science of Computer Programming*, 8(3) :275–306, 1987.

[ES84] E. Allen Emerson and A. Prasad Sistla. Deciding full branching time logic. *Information and Computation*, 61(3) :175–201, 1984.

[Fit02] M. Fitting. Modal logic between propositional and first-order. *Journal of Logic and Computation*, 12(6) :1017–1026, 2002.

[FL02] A. Finkel and J. Leroux. How to compose Presburger accelerations : Applications to broadcast protocols. In *FST&TCS'02*, volume 2256 of *Lecture Notes in Computer Science*, pages 145–156. Springer, 2002.

[FS00] A. Finkel and G. Sutre. Decidability of reachability problems for classes of two counters automata. In *STACS'00*, volume 2256 of *Lecture Notes in Computer Science*, pages 346–357. Springer, 2000.

[FS01] A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere ! *Theoretical Computer Science*, 256(1-2) :63–92, 2001.

[FWW97] A. Finkel, B. Willems, and P. Wolper. A direct symbolic approach to model-checking pushdown systems (extended abstract). In *INFINITY'97*, volume 9 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 1997.

[Gab89] D. Gabbay. The declarative past and imperative future : Executable temporal logic for interactive systems. In *Temporal Logic in Specification*, volume 398 of *Lecture Notes in Computer Science*, pages 409–448. Springer, 1989.

[Gas05] R. Gascon. Verifying qualitative and quantitative properties with LTL over concrete domains. In *Proceedings of the 4th Workshop on Methods for Modalities (M4M'05)*, volume 194 of *Informatik Bericht*, pages 54–61. Humboldt Universität zu Berlin, 2005.

[GK03] P. Gastin and D. Kuske. Satisfiability and model checking for MSO-definable temporal logics are in PSPACE. In *CONCUR'03*, volume 2761 of *Lecture Notes in Computer Science*, pages 222–236. Springer, 2003.

[GKK+03] D. Gabelaia, R. Kontchakov, A. Kurucz, F. Wolter, and M. Zakharyaschev. On the computational complexity of spatio-temporal logics. In *FLAIRS'03*, pages 460–464. AAAI, 2003.

[Gor96] V. Goranko. Hierarchies of modal and temporal logics with references pointers. *Journal of Logic, Language, and Information*, 5 :1–24, 1996.

[GPSS80] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal basis of fairness. In *POPL'80*, pages 163–173. ACM, 1980.

[Hal95] J. Halpern. The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artificial Intelligence*, 75(2) :361–372, 1995.

[Iba78] O. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM*, 25(1) :116–133, 1978.

[ID01] O. Ibarra and Z. Dang. On removing the stack from reachability constructions. In *ISAAC'01*, volume 2223 of *Lecture Notes in Computer Science*, pages 244–256. Springer, 2001.

[ISD+00] O. Ibarra, J. Su, Z. Dang, T. Bultan, and A. Kemmerer. Counter machines : decidable properties and applications to verification problems. In *MFCS'00*, volume 1893 of *Lecture Notes in Computer Science*, pages 426–435. Springer, 2000.

[Jan90] P. Jančar. Decidability of a temporal logic problem for Petri nets. *Theoretical Computer Science*, 74(1) :71–93, 1990.

[JKMS04] P. Jančar, A. Kučera, F. Moller, and Z. Sawa. DP lower bounds for equivalence-checking and model-checking of one-counter automata. *Information and Computation*, 188(1) :1–19, 2004.

[Kam68] H. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, UCLA, 1968.

[Kir02] D. Kirsten. Alternating tree automata and parity games. In *Automata, Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*, pages 153–167. Springer, 2002.

[Kos82] R. Kosaraju. Decidability of reachability in vector addition systems. In *STOC'82*, pages 267–281. ACM, 1982.

[Kuč00] A. Kučera. Efficient verification algorithms for one-counter processes. In *ICALP'00*, volume 1853 of *Lecture Notes in Computer Science*, pages 317–328. Springer, 2000.

[KV06] O. Kupferman and M.Y. Vardi. Memoryful branching-time logics. In *LICS'06*, pages 265–274. IEEE, 2006.

[KVW00] O. Kupferman, M. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model-checking. *Journal of the ACM*, 47(2) :312–360, 2000.

[Laz06] R. Lazić. Safely freezing LTL. In *FST&TCS'06*, volume 4337 of *Lecture Notes in Computer Science*, pages 381–392. Springer, 2006.

[LLT05] P. Lafourcade, D. Lugiez, and R. Treinen. Intruder deduction for AC-like equational theories with homomorphisms. In *RTA'05*, volume 3467 of *Lecture Notes in Computer Science*, pages 308–322. Springer, 2005.

[LM01] U. Dal Lago and A. Montanari. Calendars, time granularities, and automata. In *Int. Symposium on Spatial and Temporal Databases*, volume 2121 of *Lecture Notes in Computer Science*, pages 279–298. Springer, 2001.

[LM05] C. Lutz and M. Milicic. A tableau algorithm for description logics with concrete domains and GCIs. In *TABLEAUX'05*, volume 3702 of *Lecture Notes in Computer Science*, pages 201–216. Springer, 2005.

[LMS02] F. Laroussinie, N. Markey, and Ph. Schnoebelen. Temporal logic with forgettable past. In *LICS'02*, pages 383–392. IEEE, 2002.

[LMS04] F. Laroussinie, N. Markey, and Ph. Schnoebelen. Model-checking timed automata with one or two clocks. In *CONCUR'04*, volume 3170 of *Lecture Notes in Computer Science*, pages 387–401. Springer, 2004.

[LP05] A. Lisitsa and I. Potapov. Temporal logic with predicate λ-abstraction. In *TIME'05*, pages 147–155. IEEE, 2005.

[LPZ85] O. Lichtenstein, A. Pnueli, and L. Zuck. The glory of the past. In *Logic of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 196–218. Springer, 1985.

[Lut04] C. Lutz. NEXPTIME-complete description logics with concrete domains. *ACM Transactions on Computational Logic*, 5(4) :669–705, 2004.

[LW05] S. Lasota and I. Walukiewicz. Alternating timed automata. In *FOSSACS'05*, volume 3441 of *Lecture Notes in Computer Science*, pages 250–265. Springer, 2005.

[Min67] M. Minsky. *Computation, Finite and Infinite Machines*. Prentice Hall, 1967.

[MOS05] M. Müller-Olm and H. Seidl. Analysis of modular arithmetic. In *ESOP'05*, volume 3444 of *Lecture Notes in Computer Science*, pages 46–60. Springer, 2005.

[OW04] J. Ouaknine and J. Worrell. On the language inclusion problem for timed automata : Closing a decidability gap. In *LICS'04*, pages 54–63. IEEE, 2004.

[Pap94] C. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[Pet81] J.L. Peterson. *Petri Net Theory and the Modelling of Systems*. Prentice-Hall, 1981.

[Pnu77] A. Pnueli. The temporal logic of programs. In *FOCS'77*, pages 46–57. IEEE, 1977.

[Pre29] M. Presburger. Über die vollständigkeit eines gewissen systems der arithmetik ganzer zahlen, in welchem die addition als einzige operation hervortritt. *Comptes Rendus du Ier congrès de Mathématiciens des Pays Slaves*, pages 92–101, 1929.

[Reu90] C. Reutenauer. *The mathematics of Petri nets*. Prentice-Hall, 1990.

[Saf88] S. Safra. On the complexity of ω-automata. pages 319–327. IEEE, 1988.

[Sav70] W. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2) :177–192, 1970.

[SB05] S. Seshia and R. Bryant. Deciding quantifier-free presburger formulas using parameterized solution bounds. *Logical Methods in Computer Science*, 1(2 :6) :1–26, 2005.

[SC85] A. Sistla and E. Clarke. The complexity of propositional linear temporal logic. *Journal of the ACM*, 32(3) :733–74, 1985.

[Seg06] L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *CSL'06*, volume 4207 of *Lecture Notes in Computer Science*, pages 41–57. Springer, 2006.

[Ser06] O. Serre. Parity games played on transition graphs of one-counter processes. In *FOSSACS'06*, volume 3921 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 2006.

[Tho90] W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Volume B : Formal Models and Sematics (B)*, pages 133–192. 1990.

[Var88] M. Vardi. A temporal fixpoint calculus. In *POPL'88*, pages 250–259. ACM, 1988.

[VW86] M. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *LICS'86*, pages 332–344. IEEE, 1986.

[VW94] M. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115 :1–37, 1994.

[Wal01] I. Walukiewicz. Pushdown processes : games and model-checking. *Information and Computation*, 164(2) :234–263, 2001.

[Wil99] T. Wilke. CTL$^+$ is exponentially more succinct than CTL. In *FSTTCS'99*, volume 1738 of *Lecture Notes in Computer Science*, pages 110–121. Springer, 1999.

[Wol83] P. Wolper. Temporal logic can be more expressive. *Information and Computation*, 56 :72–99, 1983.

[WTT04] M. Wakatsuki, K. Teraguchi, and E. Tomita. Polynomial time identification of strict deterministic restricted one-counter automata in some class from positive data. In *ICGI'04*, volume 3264 of *Lecture Notes in Artificial Intelligence*, pages 260–272. Springer, 2004.

[WZ00] F. Wolter and M. Zakharyaschev. Spatio-temporal representation and reasoning based on RCC-8. In *KR'00*, pages 3–14, 2000.