L**ogiques pour la** S**pécification et** V**érification**

– Mémoire d'habilitation à diriger des recherches en Informatique –

Stéphane Demri

Laboratoire Spécification et Vérification
ENS Cachan, CNRS, INRIA
demri@lsv.ens-cachan.fr

22 juin 2007

Ce document est le mémoire d'habilitation à diriger des recherches en informatique soumis à l'Université Paris 7 - Denis Diderot. La soutenance a eu lieu le 19 juin 2007 à l'École Normale Supérieure de Cachan. Le jury est composé des membres suivants.

- Nicole Bidoit, professeure à l'Université de Paris 11 (examinatrice),
- Patrick Blackburn, directeur de recherche INRIA, LORIA, Nancy (examinateur),
- Ahmed Bouajjani, professeur à l'Université de Paris 7 (rapporteur),
- Philippe Schnoebelen, directeur de recherche CNRS, LSV, Cachan (examinateur),
- Moshe Vardi, professeur à l'Université de Rice, Houston (rapporteur),
- Igor Walukiewicz, directeur de recherche CNRS, LABRI, Bordeaux (rapporteur),
- Pierre Wolper, professeur à l'Université de Liège (examinateur).

Table des matières

Chapitre 1

Introduction

Ce mémoire présente une sélection de travaux effectués entre 1996 et 2006 avec une prédominance pour la période 2000–2006.

1.1 Sécurité des systèmes informatiques

L'omniprésence des systèmes informatiques aujourd'hui et leur émergence ces dernières décennies ont nécessité la création de nouveaux champs d'investigation scientifiques qui tiennent compte de la diversité de ces systèmes et des enjeux qu'ils représentent. Un système informatique a pour but de traiter, stocker, acheminer ou présenter de l'information. Ce que la société attend de ces systèmes est simplement qu'ils fonctionnent correctement (donc qu'ils remplissent les tâches pour lesquelles ils ont été conçus) et qu'ils assurent toutes les garanties de sécurité, une erreur pouvant être fatale. L'activité scientifique pour laquelle ces systèmes sont l'objet d'étude nécessite à la fois un travail de modélisation mais aussi l'élaboration de méthodes pour répondre aux questions les plus fondamentales. Une des difficultés est probablement de découvrir des lois générales qui résistent au rythme soutenu de l'évolution de ces systèmes. Par exemple, le développement des applications Web ou encore les intrusions hostiles dans les systèmes informatiques pourraient être régis par des lois communes dans un cadre uniforme, ce qui faciliterait alors leur analyse. Avec l'approche logique poursuivie dans ce travail et que nous détaillerons dans la suite, l'espoir est de pouvoir valider formellement les méthodes utilisées et de mesurer leurs limites, tout en résistant à l'épreuve du temps.

Comme les systèmes informatiques sont d'une diversité redoutable puisqu'ils comprennent pêle-mêle les systèmes logiciels, les cartes à puce, les protocoles cryptographiques, les systèmes embarqués, les applications Web, les systèmes d"imagerie médicale etc., la spécification de leur fonctionnement correct fait appel à des concepts très variés. La phase de modélisation est souvent l'occasion cependant de dégager les lignes communes. Dans un environnement hostile comme le Web, il s'agit par exemple de garantir l'intégrité et la confidentialité de l'information ou encore le respect de la vie privée, encore faut-il savoir ce que ces notions recouvrent exactement. Il arrive aussi que la vulnérabilité d'un système provienne

de la mauvaise programmation d'une application et alors il faut être capable de détecter les erreurs intrinsèques comme par exemple la possibilité d'écrire de données en dehors de la zone mémoire allouée. Une telle erreur peut entraîner l'arrêt inopiné d'un système critique provoquant des dégats considérables. Il faut aussi s'assurer que le système n'a pas d'erreur fonctionnelle ou qu'il garantisse une interaction constante avec les utilisateurs comme c'est le cas dans les systèmes d'exploitation. Evidemment cette diversité des systèmes et des propriétés à vérifier s'accompagnent d'une diversité des modélisations et des méthodes formelles de vérification.

1.2 Une méthode formelle de vérification

L'approche par model-checking de la vérification formelle de systèmes informatiques passe par une phase de modélisation durant laquelle le système et ses comportements sont modélisés en termes de structures mathématiques qui seront ensuite l'objet de l'analyse. Classiquement, un programme peut être représenté comme un graphe, éventuellement infini, dont les noeuds contiennent des informations sur l'état des registres par exemple. De même, le Web est souvent représenté comme un graphe fini dans lequel les transitions correspondent à des liens hypertextes et les noeuds à des pages. Une représentation graphique est aussi utilisée pour modéliser les états de connaissance d'agents dans un système distribué. Ainsi, la modélisation permet de définir une structure relationnelle (un graphe en général) qui représente certains aspects du système et de ses comportements. Cette représentation n'est en général pas complète puisqu'elle est souvent élaborée après diverses phases d'abstraction et de simplification.

Une fois le graphe défini, il s'agit de déterminer s'il vérifie certaines propriétés à l'aide de méthodes génériques. Par exemple, la question d'accessibilité entre deux noeuds du graphe est un problème central pour la vérification formelle de programmes et de nombreuses questions apparemment plus complexes se ramènent souvent à une série de questions d'accessibilité. L'introduction d'une logique pour vérifier ces propriétés permet à la fois d'avoir un langage formel de spécification qui ne souffre d'aucune ambiguïté et de développer des méthodes génériques pour une classe généralement infinie de propriétés à vérifier. De plus, et ce n'est pas le moindre des atouts, les techniques à mettre en place pour vérifier les propriétés exprimables dans la logique peuvent faire complétement abstraction du domaine d'application du système informatique initial, ce qui répond au souci de généralité souligné plus haut. Finalement, des questions plus générales que la simple vérification d'un système, à savoir l'implication de propriétés peuvent être aussi envisagées dans l'étude. Cette approche logique et formelle de la vérification garantit l'obtention de preuves mathématiques de correction lorsque cela est théoriquement possible, voir par exemple des applications pour les systèmes réactifs [Sch04], les protocoles cryptographiques [Gou02] ou encore les systèmes temps-réel [Lar05].

La nature des propriétés à vérifier guide le choix de la logique à considérer. La logique classique du premier ordre qui a un problème de satisfaisabilité in-

décidable [Chu36] ou la logique du second-ordre qui a un problème de model-checking PSPACE-complet (avec une structure finie en entrée) sont souvent des candidates sérieuses mais il est bon de se demander si ces langages puissants sont vraiment adaptés aux problèmes particuliers. En effet, dans le but le réduire le coût algorithmique de la phase de vérification, on peut préférer une logique plus spécifique qui allie à la fois la généralité à un coût algorithmique moindre. Il s'agit ici de faire un bon compromis entre l'expressivité, la concision et la complexité, question classique au coeur de mon travail. Ainsi, pour quantifier sur des chemins dans un graphe il est préférable d'utiliser des logiques temporelles, voir par exemple [BBF+01] alors que pour quantifier seulement sur des noeuds successeurs, les logiques modales peuvent suffire [BdRV01] grâce à la notion de localité très présente dans leur sémantique.

Cette vision idyllique de la place des formalismes logiques pour la vérification des systèmes informatiques, confortée aussi par [HHI+01], s'oppose non seulement à la complexité intrinsèque de la description des systèmes mais aussi au coût algorithmique des problèmes de décision associés directement à la vérification. Ces questions de complexité jalonnent mon travail. De plus, lorsque le modèle à vérifier est infini, la nécessité d'avoir une représentation symbolique des ensembles infinis de configurations ajoute un paramètre supplémentaire dans le choix du formalisme logique.

Venons-en donc à des aspects plus concrets. Tout d'abord, une grande partie des chapitres 2 et 3 a pour fondement l'article [Pnu84] dans lequel A. Pnueli a introduit la logique temporelle comme langage de spécifications formelles pour la vérification de systèmes réactifs. Evidemment, les logiques temporelles existaient bien avant (voir par exemple les ouvrages de référence [Pri57, Pri67, Gab76, RU71, PF77]) mais n'étaient pas appliquées et conçues avec ce but déterminé. De même, les travaux de M. Vardi et P. Wolper [VW94] qui raffinent l'approche par automates de R. Büchi [Büc62] aux formules de la logique temporelle du temps linéaire LTL seront à la base de nombreuses de nos extensions. Le raffinement dont il est question ici concerne principalement l'obtention des bornes optimales de complexité de [SC85].

1.3 Contenu du document

Dans les chapitres 2–5, il est question de décidabilité, complexité algorithmique et expressivité avec pour motivations la vérification de systèmes modélisés comme des graphes finis et des systèmes à compteurs. Les aspects liés à la modélisation comme son adéquation avec les problèmes logiques seront peu abordés. Par contre, mon travail qui s'attaque aux fondements de l'approche logique suit deux directions principales: l'analyse de la complexité des problèmes de vérification et la détermination des limites pour la vérification automatique (décidabilité). De nouveaux formalismes logiques seront parfois aussi introduits. La description succincte de chaque chapitre est faite dans les sous-sections à venir en précisant les collaborateurs avec lesquels ces travaux ont été menés.

La section 1.4 présente les principales familles de travaux que j'ai menés et qui n'ont pas leur place dans les chapitres 2–5. Il s'agit de travaux couvrant principalement la période 1996–2002.

Le document ne contient pratiquement pas de preuve et le lecteur est renvoyé aux documents originaux. Lorsqu'un résultat est original, une preuve est alors fournie mais ce cas de figure demeure rare. Dans le document, l'attention du lecteur est attirée sur certains problèmes ouverts qui suscitent mon intérêt et j'espère celui du lecteur.

1.3.1 Varia autour de LTL

La logique temporelle du temps linéaire LTL [SC85] est un des formalismes les plus utilisés pour spécifier les comportements de systèmes réactifs. Le chapitre 2 traite principalement des réponses aux questions suivantes.

- Pour quels fragments de LTL les problèmes de model-checking et de satisfaisabilité permettent de réduire la complexité dans le pire des cas?
- Quel extension minimale de LTL avec un opérateur défini à partir d'un langage hors-contexte a un problème de model-checking indécidable?
- Quelle est la complexité du problème de model-checking pour LTL lorsque le système à vérifier est un produit de sous-systèmes dans le sens de Downey et Fellows avec le nombre de sous-systèmes comme paramètre?
- Quelles techniques avec automates pour LTL s'étendent au cas où les modèles sont de longueur supérieure à ω et permettent de caractériser la complexité?

Collaborateurs: François Laroussinie (LSV, Cachan), David Nowak (Tokyo University), Philippe Schnoebelen (LSV, Cachan)

1.3.2 Vérification de propriétés qualitatives et quantitatives

Le chapitre 3 s'intéresse principalement aux extensions de LTL où les variables propositionnelles sont raffinées en des contraintes interprétées pour un domaine concret (entiers, réels, chaînes de caractères, etc.). Dans ce chapitre de synthèse, les questions suivantes sont abordées.

- Quelles extensions produisent des problèmes indécidables?
- Comment étendre la technique de Vardi et Wolper pour tenir compte de ces nouvelles formules atomiques?
- Quand l'ajout de l'opérateur "freeze" conduit à des problèmes indécidables?
- Quels sont les fragments de LTL avec contraintes de Presburger qui soient décidables?

Collaborateurs: Deepak D'Souza (IISC, Bangalore), Régis Gascon (LSV, Cachan), Ranko Lazić (Warwick University), David Nowak (Tokyo University).

1.3.3 Contraintes de régularité

Le chapitre 4 est dédié à des formalismes logiques qui permettent d'exprimer des contraintes de régularité sur des graphes provenant de systèmes divers. Les questions suivantes sont abordées.

- Quel est le coût algorithmique des problèmes de contraintes de chemin pour les données semi-structurées?
- Quelles logiques avec contraintes de régularité et de Presburger sont dans PSPACE lorsque les modèles sont les structures arborescentes issues de documents XML?
- Pourquoi la gestion dynamique de politiques dans des logiques de la permission peut se traduire dans la logique PDL construite sur des termes de programmes réguliers?

Collaborateurs: Natasha Alechina (University of Nottinghman) Denis Lugiez (LIF, Marseille), Maarten de Rijke (University of Amsterdam)

1.3.4 Complexité des logiques modales grammaticales

Le chapitre 5 s'intéresse à la classe des logiques modales grammaticales qui capture de très nombreuses logiques présentes dans la littérature à divers titres (logiques terminologiques, logiques temporelles, logiques épistémiques etc.). Des réponses aux questions suivantes sont abordées dans ce chapitre de synthèse.

- Comment traduire une logique modale vers le fragment gardé GF de la logique classique alors que la classe des modèles de la logique n'est pas exprimable dans GF?
- Comment caractériser la complexité des logiques modales grammaticales en fonction des langages générés par les schémas d'axiomes?
- Pourquoi de nombreuses logiques modales admettent facilement des calculs analytiques?

Collaborateur: Hans de Nivelle (Max-Planck Institut für Informatik, Saarbrücken).

1.4 Autres travaux

Il y a des travaux menés durant la période 1996–2006 qui ne vont pas être présentés dans les chapitres suivants. Voici brièvement un aperçu de leur contenu qui est davantage orienté vers la démonstration automatique.

1.4.1 Procédures de décision avec des calculs analytiques

Les calculs par tableaux permettent souvent de concevoir des procédures de décision pour les logiques modales et temporelles [Gor99]. Pendant la période 1996–2000, je me suis intéressé à l'élaboration de calculs qui non seulement sont complets pour les logiques correspondantes (un minimum) mais qui permettent

aussi de définir des procédures de décision qui soient optimales par rapport à la complexité dans le pire de cas.

- Dans [Dem96b], un système de preuves par tableaux pour la logique modale de l'ailleurs [Seg81] a été défini et une borne supérieure de complexité NP a été établie. De plus, j'ai montré comment de nombreuses logiques multivaluées admettent une traduction simple vers cette logique [Dem00c].
- Dans [BD97], nous avons défini des calculs par tableaux avec décoration des formules pour des logiques enrichies de l'opérateur de différence [dR92]. Des procédures de décision en temps exponentiel ont été présentées (borne optimale dans le pire des cas) et les décorations sur les formules ont permis comme c'est souvent le cas pour ce type de calcul de contrôler l'application des règles d'inférence.

  Collaborateur: Philippe Balbiani (IRIT, Toulouse).

- Dans [Dem99b], des calculs de séquents complets pour des logiques temporelles hybrides ont été définis [Bla93]. Ces logiques admettent des nominaux ainsi que des opérateurs de passé et de futur. En fait, ces résultats s'appliquent à la large classe de logiques dont les modèles sont définissables dans la classe de formules Π_2 de la logique classique. Dans ces calculs, les nominaux sont utilisés comme des étiquettes ce qui permet d'internaliser la déduction, une propriété partagée avec [Bla00a]. Par exemple, les contraintes sur les relations d'accessibilité s'expriment simplement par des règles et nos définitions sont uniformes. Ce travail démontre aussi que l'obtention de la complétude n'est pas vraiment difficile (voir aussi [Tza99]).
- Dans [Dem02], j'ai défini des calculs de séquents pour la classe des logiques modales grammaticales linéaires à droite (cf. le chapitre 5) et établi une borne de complexité EXPTIME par analyse des calculs. De plus, une borne PSPACE a été obtenue par des moyens purement syntaxiques pour une large sous-classe de telles logiques.

1.4.2 Calculs de séquents généralisés

"Display Logic" (DL) [Bel82] est un cadre théorique général pour définir des calculs où les séquents sont des paires d'objets complexes avec des opérateurs structurels supplémentaires. Dans les calculs de séquents standard le seul opérateur structurel est la virgule. Des calculs de ce type pour les logiques modales usuelles ont été définies dans [Wan94, Wan98]. Par ailleurs, un article central [Kra96] caractérise les conditions pour qu'une logique modale puisse être capturée dans le cadre de Belnap/Wansing en transformant les axiomes modaux en règles structurelles. Les résultats dans [Kra96] établissent une réduction systématique des axiomes vers les règles et une propriété capitale des calculs obtenus est l'élimination de la règle de coupure.

De nombreuses logiques échappent cependant au cadre fixé dans [Kra96] et parmi elles les logiques modales du second ordre G (logique introduite par K.

Gödel [Göd33]) et Grz (logique introduite par A. Grzegorczyk [Grz69]). Ces formalismes sont connus pour admettre des interprétations remarquables dans l'arithmétique (voir par exemple [Sol76, Boo93]). Cela semble contredire le fait que (DL) soit un cadre plus général que celui des séquents standard. Cependant, nous avons montré dans [DG02b] qu'en affaiblissant lègèrement les conditions de [Kra96], on peut définir des calculs à la (DL) pour G et Grz. En fait, nous avons identifié une classe de logiques de la prouvabilité (incluant G et Grz) pour lesquelles une équivalence inattendue existe entre une propriété d'élimination des coupures dans ces calculs et la correction d'une traduction entre une logique de la prouvabilité et sa logique modale sous-jacente. Cela nous permet par exemple de définir des traductions entre Grz et S4.

Nous avons aussi défini des calculs dans (DL) pour des logiques temporelles hybrides [DG02a] et pour des logiques épistémiques [DG00c].

Collaborateur: Rajeev Goré (Australian National University, Canberra).

1.4.3 Logiques pour l'information incomplète

Les logiques pour l'information incomplète forment une classe de logiques modales avec des spécificités qui rendent quelquefois leur analyse difficile avec seulement les techniques usuelles des logiques modales [BdRV01]. La monographie [DO02] co-écrite avec E. Orłowska constitue la première tentative pour exposer les fondements de ces logiques dans un cadre uniforme (voir une analyse de ce livre dans [Jär06]). Elle contient aussi bien des résultats de complexité que la définition de systèmes de preuve complets. De plus, des résultats orginaux y sont présentés mais on y trouve aussi des résultats publiés par ailleurs sur des questions

- de décidabilité [Dem96c, Dem96a, DK98, Dem98],
- de complexité [Dem97b, Dem99a],
- de traduction vers des formalismes plus standard [DK98, DG00a, DG00b, Dem01b],
- d'axiomatisation [Dem97a, DO98, Dem99a].

Différents résultats de complexité pour ces logiques ne figurant pas dans cette monographie ont été publiés après la parution de [DO02].

- La logique SIM introduite dans [Kon98] est montrée EXPTIME-complète dans [DS02a] en utilisant des techniques à base d'automates de Büchi sur les arbres. Avec une technique analogue, une extension de la logique NIL introduite dans [DO07] est aussi montrée EXPTIME-complète.
- La logique DALLA [Gar86] et la logique LGM [Nak93] sont montrées PSPACE-complètes dans [Dem03] en établissant un résultat de complexité plus général pour une large classe de logiques multimodales et cela à partir d'un algorithme à la Ladner [Lad77] qui n'est pas sans rappeler les algorithmes de test de vacuité pour les automates d'arbres. D'autres logiques sont prouvées dans PSPACE dans [DS00].
- La logique NIL [Vak87] est prouvée PSPACE-complète dans [Dem00b].

Collaborateurs: Dov Gabbay (King's College, London), Beata Konikowska (Institute of Computer Science, PAS, Varsovie) Ewa Orłowska (Institute of Telecommunications, Varsovie), Ulrike Sattler (University of Manchester), Jarosław Stepaniuk (Białystok University of Technology, Białystok), Dimiter Vakarelov (Sofia University).

Chapitre 2

Varia autour de LTL

Ce chapitre est construit sur la base de [DS02b, DLS06, DN07]. Il a pour objet l'étude de problèmes relatifs à la logique temporelle du temps linéaire LTL qui est très utilisée pour la spécification et vérification de programmes, voir par exemple [BBF$^+$01].

De façon schématique, je me suis intéressé aux problèmes suivants:

- complexité du model-checking et du problème de satisfaisabilité pour des fragments syntaxiques de LTL [DS02b],
- décidabilité d'extension de LTL avec un unique opérateur hors-contexte,
- complexité paramétrée du model-checking symbolique pour des logiques temporelles incluant LTL [DLS06],
- complexité d'extensions de LTL avec des modèles de longueur supérieure à ω [DN07].

2.1 Logique temporelle du temps linéaire LTL

Dans cette section, nous commençons par rappeler quelques définitions de base à propos de LTL avant d'aborder les questions que nous avons étudiées. Les formules de LTL sont construites à partir de la grammaire abstraite suivante:

$$\phi ::= \mathrm{p}_i \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \mathtt{X}\phi \mid \mathtt{F}\phi \mid \phi \mathtt{U} \psi$$

où $\mathrm{PROP} = \{\mathrm{p}_1, \mathrm{p}_2, \ldots\}$ est un ensemble infini dénombrable de variables propositionnelles. On note $\mathrm{LTL}_n^k(\mathrm{H}_1, \mathrm{H}_2, \ldots)$ le fragment de LTL restreint

- aux opérateurs temporels $\mathrm{H}_1, \mathrm{H}_2, \ldots$,
- aux formules de hauteur temporelle au plus $k \geq 0$,
- avec au plus $n \geq 1$ variables propositionnelles.

La temporelle hauteur d'une formule est le nombre maximal d'opérateurs temporels emboîtés. Par exemple $ht((\mathtt{XX}\mathrm{p}) \vee (\mathrm{p}\mathtt{U}\neg\mathrm{q})) = 2$. De même, $\mathrm{LTL}_\omega^2(\mathtt{F})$ dénote l'ensemble des formules de LTL de hauteur temporelle au plus 2 construites avec le seul opérateur temporel $\mathtt{F}$. $|\phi|$ dénote la taille de la formule ϕ vue comme une chaîne de caractères. Cette notation sera aussi utilisée pour les autres formalismes logiques.

Un modèle pour LTL est une séquence infinie $\sigma : \mathbb{N} \to \mathcal{P}(\mathrm{PROP})$, c'est-à-dire un mot infini de $\mathcal{P}(\mathrm{PROP})^\omega$. Étant donnés un modèle σ, $i \in \mathbb{N}$ et une formule ϕ, on définit par induction la relation de satisfaction:

- $\sigma, i \models \mathrm{p} \overset{\text{def}}{\Leftrightarrow} \mathrm{p} \in \sigma(i)$,
- $\sigma, i \models \neg\phi \overset{\text{def}}{\Leftrightarrow} \sigma, i \not\models \phi$ ($\vee$ et $\wedge$ ont leur sémantique usuelle),
- $\sigma, i \models \mathtt{X}\phi \overset{\text{def}}{\Leftrightarrow} \sigma, i+1 \models \phi$,
- $\sigma, i \models \mathtt{F}\phi \overset{\text{def}}{\Leftrightarrow}$ il existe $j \geq i$ tel que $\sigma, j \models \phi$,
- $\sigma, i \models \phi \mathtt{U} \psi \overset{\text{def}}{\Leftrightarrow}$ il existe $j \geq i$ tel que $\sigma, j \models \psi$ et pour $i \leq k < j$, on a $\sigma, k \models \phi$.

On note $\sigma \models \phi$ pour $\sigma, 0 \models \phi$. L'ensemble des modèles de ϕ, noté $\mathrm{Mod}(\phi)$, est $\{\sigma \in \mathcal{P}(\mathrm{PROP})^\omega : \sigma \models \phi\}$. Une formule ϕ est *satisfaisable* (pour LTL) $\overset{\text{def}}{\Leftrightarrow}$ $\mathrm{Mod}(\phi) \neq \emptyset$. Le problème de la satisfaisabilité pour LTL, noté SAT(LTL), est le suivant.

entrée: une formule ϕ de LTL,
question: est-ce qu'il existe un modèle σ tel que $\sigma \models \phi$?

Rappelons maintenant une version existentielle du problème de model-checking. Une structure de Kripke $\mathcal{M} = \langle S, R, V \rangle$ est composée

- d'un ensemble non-vide d'états S,
- d'une relation binaire $R \subseteq S \times S$,
- d'une fonction d'interprétation $V : S \to \mathcal{P}(\mathrm{PROP})$ (une valuation).

$\mathcal{M}$ est un graphe dont une interprétation du calcul propositionnel est associé à chaque état. Un chemin de $\mathcal{M}$ est une séquence $q_0 q_1 \ldots$ (finie ou infinie) telle que $q_i R q_{i+1}$ pour $i \geq 0$. On note $\mathrm{Chemins}(\mathcal{M}, q_0)$ l'ensemble des chemins infinis de $\mathcal{M}$ commençant par q_0. Par abus de langage, on note aussi $\mathrm{Chemins}(\mathcal{M}, q_0)$ l'ensemble des chemins infinis commençant par q_0 sur l'alphabet $\mathcal{P}(\mathrm{PROP})^\omega$. La première lettre de chacun de ces chemins est donc $V(q_0)$.

Le problème du model-checking pour LTL, noté MC$^\exists$(LTL), est le suivant:

entrée: une formule de LTL ϕ, une structure de Kripke finie et totale (pour $x \in S$, il existe $y \in S$ tel que $\langle x, y \rangle \in R$) $\mathcal{M}$ et $q_0 \in S$,
question: est-ce qu'il existe un chemin infini σ commençant par q_0 tel que $\sigma \models \phi$ (noté $\mathcal{M}, q_0 \models_\exists \phi$)?

Dans l'énoncé du problème ci-dessus, on suppose que le domaine de la fonction d'interprétation V dans $\mathcal{M}$ est restreint aux propositions atomiques apparaissant dans ϕ. La taille d'une structure finie $\langle S, R, V \rangle$ est $\mathrm{card}(S) + \mathrm{card}(R) + \Sigma_{x \in S} \mathrm{card}(V(x))$. On a $\mathcal{M}, q_0 \models_\exists \phi$ ssi $\mathrm{Chemins}(\mathcal{M}, q_0) \cap \mathrm{Mod}(\phi) \neq \emptyset$. Cette définition est duale de la définition habituellement utilisée en vérification où une quantification universelle est utilisée.

La proposition 2.1.1 énonce un résultat central entre les formules de LTL et les automates de Büchi qui permet d'obtenir des résultats de complexité optimaux. Cette approche par automate sera étendue dans la section 2.5 et dans le chapitre 3.

Proposition 2.1.1. [VW94] Pour chaque formule de LTL, il existe un automate de Büchi $\mathcal{A}_\phi$ tel que

1. $\mathrm{L}(\mathcal{A}_\phi) = \mathrm{Mod}(\phi)$,

2. $|\mathcal{A}_\phi|$ est en $2^{\mathcal{O}(|\phi|)}$,
3. $\mathcal{A}_\phi$ se calcule en espace polynomial en $|\phi|$.

Dans la proposition 2.1.1 ci-dessus, on se restreint aux variables propositionnelles apparaissant dans la formule ϕ.

Théorème 2.1.2. [SC85] Les problèmes de model-checking et satisfaisabilité pour LTL sont PSPACE-complets.

2.2 Complexité de fragments de LTL

Malgré la complexité théorique du problème de model-checking de LTL, en pratique la vérification avec LTL demeure possible. Dans [DS02b], nous avons considéré des fragments de LTL et analysé leur complexité pour les problèmes de model-checking et de satisfaisabilité. Ces fragments sont construits en restreignant le nombre de variables propositionnelles, la hauteur temporelle et les opérateurs temporels parmi `X`, `F`, `U` et l'opérateur plat $\mathtt{U}^p$. Ce dernier opérateur est en fait une restriction de `U` pour laquelle le premier argument est une formule du calcul propositionnel. Cet opérateur a été utilisé par exemple dans les travaux [Dam99, CC00].

La prise en compte de tels fragments prend tout son sens lorsque l'on sait que pour les applications pratiques, la hauteur temporelle des propriétés vérifiées dépasse rarement 3 (par exemple pour une propriété d'équité). La restriction à un nombre fini de variables est assez commune [Hal95, DG06] et permet en général de déceler des sauts de complexité lorsque ce n'est pas des sauts de décidabilité.

Les résultats que nous avons obtenus dans le papier [DS02b] sont reproduits dans la figure 2.1 avec des références aux résultats existant avant notre étude. Le symbole $\mathtt{U}^?$ est `U` ou $\mathtt{U}^p$. L'extension de ces résultats avec opérateurs de passé est faite dans [Mar03a].

Ce qui est marquant de prime abord dans ce résumé réside dans le fait que peu de fragments intéressants permettent vraiment de décroître la complexité. Par ailleurs, même si le tableau semble assez complet au regard de nos critères d'analyse (le nombre de variables propositionnelles, la hauteur temporelle et les opérateurs temporels), la question suivante reste ouverte.

Problème ouvert 1. Quelle est la complexité du model-checking pour les fragments $\mathrm{LTL}_1^\omega(\mathtt{F})$ et $\mathrm{LTL}_1^\omega(\mathtt{U})$? La NLOGSPACE-dureté et la borne supérieure PTIME sont déjà connues. Une question analogue se pose pour la satisfaisabilité. ◯

Il y a une autre question ouverte en relation avec le problème ouvert 1 (voir par exemple [DS02b, Problème ouvert 4.1]) d'une portée plus intéressante. Si un chemin est une séquence finie de valuations ou une séquence ultimement périodique de la forme uv^ω où u et v sont des mots finis, quel est le coût algorithmique pour vérifier si une formule de LTL est satisfaite sur un tel chemin? Le problème est clairement dans PTIME mais on ignore si le problème est dans NLOGSPACE ou PTIME-difficile.

	$n-1, k < \omega$	Model-checking	Satisfaisabilité
$\mathrm{L}(\ldots)$	$\mathrm{LTL}_n^0(\ldots)$	LOGSPACE	LOGSPACE
	$\mathrm{LTL}_\omega^0(\ldots)$	LOGSPACE [Lyn77]	NP-complet [Coo71]
$\mathrm{L}(\mathtt{F})$	$\mathrm{LTL}(\mathtt{F})$	NP-complet [SC85]	NP-complet [NO80]
	$\mathrm{LTL}_\omega^1(\mathtt{F})$	NP-complet	NP-complet
	$\mathrm{LTL}_2^\omega(\mathtt{F})$	NP-complet	NP-complet
	$\mathrm{LTL}_1^\omega(\mathtt{F})$	dans PTIME, NLOGSPACE-difficile	dans PTIME
	$\mathrm{LTL}_n^{k+1}(\mathtt{F})$	NLOGSPACE-complet	LOGSPACE
$\mathrm{L}(\mathtt{U}^?)$	$\mathrm{LTL}(\mathtt{U}^?)$	PSPACE-complet [SC85]	PSPACE-complet [SC85, HR83]
	$\mathrm{LTL}_\omega^2(\mathtt{U}^?)$	PSPACE-complet	PSPACE-complet
	$\mathrm{LTL}_\omega^1(\mathtt{U}^?)$	NP-complet	NP-complet
	$\mathrm{LTL}_2^\omega(\mathtt{U}^?)$	PSPACE-complet	PSPACE-complet
	$\mathrm{LTL}_1^\omega(\mathtt{U}^?)$	dans PTIME, NLOGSPACE-difficile	dans PTIME
	$\mathrm{LTL}_n^{1+k}(\mathtt{U}^?)$	NLOGSPACE-complet	LOGSPACE
$\mathrm{L}(\mathtt{X})$	$\mathrm{LTL}(\mathtt{X})$	NP-complet	NP-complet
	$\mathrm{LTL}_\omega^k(\mathtt{X})$	LOGSPACE	NP-complet
	$\mathrm{LTL}_1^\omega(\mathtt{X})$	NP-complet	NP-complet
	$\mathrm{LTL}_n^k(\mathtt{X})$	LOGSPACE	LOGSPACE
$\mathrm{L}(\mathtt{F}, \mathtt{X})$	$\mathrm{LTL}(\mathtt{F}, \mathtt{X})$	PSPACE-complet [SC85]	PSPACE-complet [SC85, HR83]
	$\mathrm{LTL}_\omega^{2+k}(\mathtt{F}, \mathtt{X})$	NP-complet	PSPACE-complet [Har85, Spa93a]
	$\mathrm{LTL}_\omega^1(\mathtt{F}, \mathtt{X})$	NP-complet	NP-complet
	$\mathrm{LTL}_1^\omega(\mathtt{F}, \mathtt{X})$	PSPACE-complet	PSPACE-complet
	$\mathrm{LTL}_n^{1+k}(\mathtt{F}, \mathtt{X})$	NLOGSPACE-complet	LOGSPACE
$\mathrm{L}(\mathtt{U}^?, \mathtt{X})$	$\mathrm{LTL}(\mathtt{U}^?, \mathtt{X})$	PSPACE-complet [SC85]	PSPACE-complet [SC85, HR83]
	$\mathrm{LTL}_\omega^2(\mathtt{U}^?, \mathtt{X})$	PSPACE-complet	PSPACE-complet [Har85, Spa93a]
	$\mathrm{LTL}_\omega^1(\mathtt{U}^?, \mathtt{X})$	NP-complet	NP-complet
	$\mathrm{LTL}_1^\omega(\mathtt{U}^?, \mathtt{X})$	PSPACE-complet	PSPACE-complet
	$\mathrm{LTL}_n^{1+k}(\mathtt{U}^?, \mathtt{X})$	NLOGSPACE-complet	LOGSPACE

FIG. 2.1 – *Complexité des fragments de LTL*

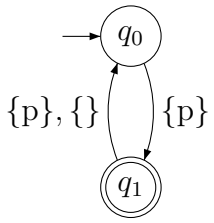


FIG. 2.2 – *Automate de Büchi sur l'alphabet* $\{\{\mathrm{p}\}, \{\}\}$.

Problème ouvert 2. Quelle est la complexité du model-checking d'un chemin avec LTL? ◯

La complexité du problème du model-checking de chemins pour différents modes de représentation des chemins (pas simplement en extension comme dans le problème ouvert 2) et pour différentes logiques (pas seulement LTL) a été étudiée dans [MS03] (voir aussi [MS06]). Une question analogue est posée dans [MW04] pour un fragment de LTL qui par ailleurs est dans NP malgré la présence des opérateurs `X` et `F`.

2.3 LTL avec opérateurs temporels hors-contexte

D'après le théorème de Kamp [Kam68], LTL avec passé possède le même pouvoir d'expression que la logique du premier ordre à un successeur. Cependant, il existe des propriétés assez simples que nous aimerions exprimer qui ne peuvent l'être avec LTL.

Proposition 2.3.1. [Wol83] Il n'existe pas de formule de LTL ϕ construite sur la seule variable propositionnelle p telle que $\mathrm{Mod}(\phi)$ est l'ensemble des structures de LTL pour lesquelles p est vérifiée dans au moins tous les états pairs.

C'est pourquoi dans [Wol83], une extension de LTL a été définie en ajoutant des opérateurs temporels à l'aide d'automates finis, en fait à l'aide de langages réguliers de mots finis définis par des grammaires linéaires à droite. Un automate de Büchi peut cependant facilement reconnaître un tel langage, voir la figure 2.2.

Rappelons brièvement comment l'extension ETL_f de LTL est définie. Soit $\mathcal{A} = \langle \Sigma, S, S_0, \rho, F \rangle$ un automate fini dont les lettres de Σ sont linéairement ordonnées, par exemple avec $a_1 < \ldots < a_k$. Les formules de ETL_f sont obtenues à partir de celle de LTL en autorisant les formules de la forme $\mathcal{A}(\phi_1, \ldots, \phi_k)$ pour tous les automates $\mathcal{A}$ avec un alphabet de k lettres. La relation de satisfaction est étendue ainsi:

- $\sigma, i \models \mathcal{A}(\phi_1, \ldots, \phi_k) \stackrel{\text{def}}{\Leftrightarrow}$
  - soit $S_0 \cap F \neq \emptyset$ ($\epsilon \in \mathrm{L}(\mathcal{A})$)

– soit il existe un mot fini $b_1 b_2 \dots b_n \in \mathrm{L}(\mathcal{A})$ tel que pour $0 \leq i' < n$, si $b_{i'+1} = a_j$ alors $\sigma, i + i' \models \phi_j$.

Si $S_0 \cap F \neq \emptyset$, alors $\mathcal{A}(\phi_1, \dots, \phi_k)$ est équivalent à $\top$.

L'opérateur "Until" $\mathtt{U}$ se code facilement avec un automate $\mathcal{A}_{\mathtt{U}}$ tel que $\mathrm{L}(\mathcal{A}_{\mathtt{U}}) = a_1^* a_2$ avec $a_1 < a_2$ et $\phi_1 \mathtt{U} \phi_2$ est précisément $\mathcal{A}_{\mathtt{U}}(\phi_1, \phi_2)$.

Proposition 2.3.2. [VW94] ETL_f a le même pouvoir d'expression que les automates de Büchi, c'est-à-dire, un langage L de mots infinis est reconnu par un automate de Büchi si et seulement si il existe une formule de ETL_f telle que $\mathrm{Mod}(\phi) = \mathrm{L}$.

Ainsi, la classe de langages définis par des formules de ETL_f est égale à la classe des langages

- reconnus par des automates de Büchi (proposition 2.3.2),
- définis par des formules de la logique monadique du second-ordre à un successeur (S1S),
- ω-réguliers,
- définis par des formules de l'extension de LTL avec quantification sur les variables propositionnelles,
- définis avec variante LTL où l'opérateur until est indicé par des langages réguliers de mots finis [HT99],
- définis par des formules de l'extension de LTL avec opérateur de point fixe [Var88].

ETL_f est une extension puissante de LTL mais l'équivalence d'expressivité des formalismes cités ci-dessus ne présume pas de leur concision. En effet, le problème de vacuité pour les automates de Büchi est NLOGSPACE-complet tandis que le problème de satisfaisabilité pour ETL_f est plus complexe mais de complexité comparable à celle de LTL.

Théorème 2.3.3. [VW94] $\mathrm{MC}^{\exists}(\mathrm{ETL}_f)$ et $\mathrm{SAT}(\mathrm{ETL}_f)$ sont PSPACE-complets.

De même, la satisfaisabilité pour LTL avec opérateurs de point fixe est PSPACE-complet [Var88]. Par contre, le problème de la satisfaisabilité pour S1S et pour LTL avec quantification sur les variables propositionnelles sont non-élémentaires (la complexité en temps n'est pas une tour d'exponentielles de hauteur fixe) [Mey73]. S1S apparaît comme le langage le plus concis pour décrire les langages de mots infinis ω-réguliers.

Il est facile d'étendre la définition de ETL_f en remplaçant les formules de la forme $\mathcal{A}(\phi_1, \dots, \phi_n)$ par des formules de la forme $\mathrm{L}(\phi_1, \dots, \phi_n)$ où L est un langage de mots finis sur un alphabet $\Sigma = \{a_1, \dots, a_n\}$, langage spécifié dans un formalisme choisi. Pour une classe de langages $\mathcal{C}$, on note LTL + $\mathcal{C}$ l'extension de LTL aux formules de la forme $\mathrm{L}(\phi_1, \dots, \phi_n)$ pour $\mathrm{L} \in \mathcal{C}$. Évidemment, ETL_f est précisément LTL + REG où REG est la classe des langages réguliers de mots finis représentés par des automates finis. Il est naturel de vouloir étudier LTL + HC où HC est la classe des langages hors-contexte représentés par des grammaires

hors-contexte. Comme de nombreux problèmes sont indécidables pour la classe des grammaires hors-contexte, il n'est pas très étonnant d'avoir le résultat suivant.

Proposition 2.3.4. SAT(LTL + HC) est indécidable.

En fait, l'indécidabilité est obtenue en codant le problème de l´égalité de langage entre grammaires hors-contexte, qui est connu pour être un problème indécidable. La proposition 2.3.4 qui implique aussi que MC$^{\exists}$(LTL + HC) est indécidable, est intéressante mais finalement elle repose sur le fait que LTL + HC peut assez facilement exprimer l'équivalence entre grammaires hors-contexte et pas tellement sur les relations entre les opérateurs temporels usuels $\mathtt{X}$ et $\mathtt{G}$ et les formules de la forme $\mathrm{L}(\phi_1, \ldots, \phi_n)$. Dans la suite nous allons montrer qu'il existe un langage hors-contexte L tel que SAT(LTL + $\{\mathrm{L}\}$) et MC$^{\exists}$(LTL + $\{\mathrm{L}\}$) sont aussi indécidables. Évidemment, la preuve de la proposition 2.3.4 ne serait pas réutilisable de même que toute preuve qui réduise un problème indécidable pour une classe infinie de grammaires hors-contexte.

Soit L_0 le langage $\{a_1^n \cdot a_2 \cdot a_1^{n-1} \cdot a_3 : n \geq 1\}$ et L_1 le langage $\{a_1^n \cdot a_2 \cdot a_1^n \cdot a_3 : n \geq 0\}$. On peut vérifier la validité dans LTL + $\{\mathrm{L}_0, \mathrm{L}_1\}$ de la formule suivante:

$$\mathrm{L}_1(\mathrm{p}, \mathrm{q}, \mathrm{r}) \Leftrightarrow (\mathrm{q} \wedge \mathtt{X}\mathrm{r}) \vee \mathrm{L}_0(\mathrm{p}, \mathrm{q}, \mathrm{p} \wedge \mathtt{X}\mathrm{r}).$$

Par conséquent, SAT(LTL + $\{\mathrm{L}_0, \mathrm{L}_1\}$) est indécidable ssi SAT(LTL + $\{\mathrm{L}_0\}$) est indécidable. On peut aussi montrer que l'on peut définir $\mathtt{F}$ avec L_1: $\mathtt{F}\phi$ est équivalent à $\mathrm{L}_1(\top, \phi, \top)$. Ainsi, c'est bien l'indécidabilité de SAT(LTL($\mathtt{X}$) + $\{\mathrm{L}_0\}$) et de MC$^{\exists}$(LTL + $\{\mathrm{L}_0\}$) qui va être établie ci-dessous.

Proposition 2.3.5. SAT(LTL($\mathtt{X}$) + $\{\mathrm{L}_0\}$) est Σ_1^1-complet.

La borne supérieure Σ_1^1 est assez immédiate.

Preuve: Un jeu de dominos $Dom = \langle C, D, Coul \rangle$ est un triplet composé

- d'un ensemble fini C de couleurs,
- d'un ensemble fini D de dominos,
- d'une fonction $Coul : D \times \{haut, bas, droite, gauche\} \rightarrow C$ qui associe une couleur à chaque côté de chaque domino.

On dit que Dom peut couvrir une surface $S \subseteq \mathbb{N} \times \mathbb{N}$ ssi il existe une façon de couvrir S avec les éléments de Dom en préservant les contraintes de motifs: seuls des côtés de même couleur peuvent être côte à côte (horizontalement, verticalement et sans possiblité de faire tourner les dominos). Le problème DOMREC défini ci-dessous est Σ_1^1-complet [Har85].

entrée: un jeu de dominos et une couleur c.
question: peut-on paver $\mathbb{N} \times \mathbb{N}$ avec la couleur c présente en un nombre infini de positions?

Nous allons réduire DOMREC à SAT(LTL + $\{\mathrm{L}_0, \mathrm{L}_1\}$) sachant que $\mathrm{L}_1(\cdot)$ et $\mathtt{F}$ peuvent être définis à partir de $\mathrm{L}_0(\cdot)$.

Soient $Dom = \langle C, D, Coul \rangle$ un jeu de dominos et une couleur $c \in C$ formant une instance de DOMREC avec $C = \{1, \ldots, n\}$, $D = \{d_1, \ldots, d_m\}$, et $c = 1$. Nous utilisons les variables propositionnelles suivantes.

- in est une variable propositionnelle qui est vraie dans un état codant une position de $\mathbb{N}^2$. En effet, il y aura des états du modèle qui ne correspondent à aucune position. Pour faciliter la présentation, on introduit aussi la variable out interprétée comme la négation de in.
- Pour $1 \leq j \leq m$, on considère la variable $\boxed{\text{j}}$ avec pour interprétation souhaitée: "la position de $\mathbb{N}^2$ associée à l'état courant est occupée par un dominos de type d_j".
- Pour $1 \leq i \leq n$, on utilise les variables $haut_i$, bas_i, $gauche_i$, $droite_i$. Par exemple, $haut_1$ est vraie quand le domino sur la position associée à l'état courant a la couleur 1 en haut.

Chaque état représentant une position de $\mathbb{N}^2$ est occupé par un unique domino:

$$\mathtt{G}(in \Rightarrow \bigvee_{j=1}^{m} (\boxed{\text{j}} \wedge \bigwedge_{j'=1, j' \neq j}^{m} \neg \boxed{\text{j}'}))$$

Les variables propositionnelles relatives aux couleurs sont compatibles avec la définition des types de dominos:

$$\mathtt{G}(in \Rightarrow \bigwedge_{j=1}^{m} \boxed{\text{j}} \implies$$

$$\bigwedge_{s \in \{haut, bas, droite, gauche\}} s_{Coul(d_j, s)} \wedge \bigwedge_{1 \leq j' \neq Coul(d_j, s) \leq n} \neg s_{j'})$$

On écrit PAVAGE la conjonction de formules ci-dessus. A présent, nous allons définir les états du modèle qui correspondent à des positions de $\mathbb{N}^2$. Soit SERPENT la conjonction des formules ci-dessous:

- $\mathtt{G}(in \Leftrightarrow \neg out)$,
- $in \wedge \mathtt{X}out \wedge \mathtt{XX}in \wedge \mathtt{XXX}in \wedge \mathtt{XXXX}out$,
- $\mathtt{G}(out \Rightarrow \mathtt{XL}_1(in, out, in \wedge \mathtt{X}out))$.

La seule structure construite sur $\{in, out\}$ vérifiant la formule SERPENT est la suivante:

$$\{in\} \cdot \{out\} \cdot \{in\}^2 \cdot \{out\} \cdot \{in\}^3 \cdot \{out\} \cdot \{in\}^4 \ldots$$

Cette séquence fait référence au chemin présenté dans la figure 2.3 (la partie grisée est dans $\mathbb{N}^2$). La réelle difficulté dans la preuve n'est finalement pas de concevoir un chemin à travers $\mathbb{N}^2$ mais plutôt de définir un chemin pour lequel il soit facile et possible d'accéder aux voisins de droite ou du dessus.

Pour chaque état vérifiant in, nous avons besoin de nous rappeler s'il apparaît dans une séquence de in ascendante ou descendante. Ce critère sera utile pour accéder aux voisins de droite ou d'en haut. C'est pourquoi nous introduisons

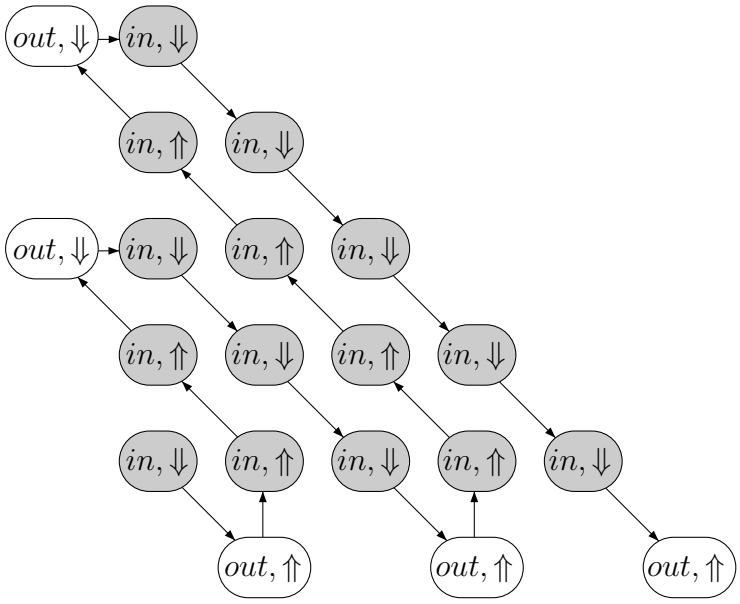


FIG. 2.3 – *Le parcours dans* $\mathbb{N}^2$

aussi les variables $\Uparrow$ et $\Downarrow$. Cette dernière variable est redondante mais facilite la présentation. Soit DIRECTION la conjonction suivante de formules:

- $\mathtt{G}(\Uparrow \Leftrightarrow \neg \Downarrow)$,
- $\Downarrow \wedge \mathtt{X} \Uparrow$,
- $\mathtt{G}(in \wedge \mathtt{X}in \wedge \Uparrow \Rightarrow \mathtt{X} \Uparrow)$ (“nous restons sur une chaîne ascendante”),
- $\mathtt{G}(in \wedge \mathtt{X}in \wedge \Downarrow \Rightarrow \mathtt{X} \Downarrow)$ (“nous restons sur une chaîne descendante”),
- $\mathtt{G}(in \wedge \mathtt{X}out \wedge \Uparrow \Rightarrow (\mathtt{X} \Downarrow \wedge \mathtt{XX} \Downarrow))$ (“nous passons d’une chaîne ascendante à descendante”),
- $\mathtt{G}(in \wedge \mathtt{X}out \wedge \Downarrow \Rightarrow (\mathtt{X} \Uparrow \wedge \mathtt{XX} \Uparrow))$ (“nous passons d’une chaîne descendante à ascendante”).

La seule structure construite sur $\{in, out, \Uparrow, \Downarrow\}$ qui satisfasse $\mathrm{SERPENT} \wedge \mathrm{DIRECTION}$ est:

$$\{in, \Downarrow\}\{out, \Uparrow\} \cdot \{in, \Uparrow\}^2 \cdot \{out, \Downarrow\} \cdot \{in, \Downarrow\}^3 \cdot \{out, \Uparrow\} \cdot \{in, \Uparrow\}^4 \ldots$$

Cette structure encode le chemin à travers $\mathbb{N}^2$ décrit dans la figure 2.3.
Le chemin permet d’accéder aux voisins adjacents de la façon suivante:

- dans un état $\{in, \Uparrow\}$, le voisin de droite est accédé avec L_1,
- dans un état $\{in, \Uparrow\}$, le voisin d’en haut est accédé avec L_0,
- dans un état $\{in, \Downarrow\}$, le voisin de droite est accédé avec L_0,
- dans un état $\{in, \Downarrow\}$, le voisin d’en haut est accédé avec L_1.

Soit CONTRAINTES la conjonction suivante de formules qui exprime les contraintes de couleur pour les positions adjacentes:

- $\mathtt{G}(in \wedge \Uparrow \Rightarrow (\bigwedge_{1 \leq i \leq n} droite_i \Rightarrow \mathrm{L}_1(in, out, gauche_i)))$,

– $\mathtt{G}(in \wedge \Uparrow \Rightarrow (\bigwedge_{1 \leq i \leq n} haut_i \Rightarrow \mathrm{L}_0(in, out, bas_i)))$,
– $\mathtt{G}(in \wedge \Downarrow \Rightarrow (\bigwedge_{1 \leq i \leq n} droite_i \Rightarrow \mathrm{L}_0(in, out, gauche_i)))$,
– $\mathtt{G}(in \wedge \Downarrow \Rightarrow (\bigwedge_{1 \leq i \leq n} haut_i \Rightarrow \mathrm{L}_1(in, out, bas_i)))$.

Soit REC la formule qui énonce que la couleur 1 est répétée infiniment souvent:

$$\mathtt{GF}(in \wedge \bigvee_{s \in \{gauche, droite, haut, bas\}} s_1).$$

Ainsi le jeu de dominos Dom peut couvrir $\mathbb{N}^2$ en répétant la couleur 1 infiniment souvent ssi

$$\text{PAVAGE} \wedge \text{SERPENT} \wedge \text{DIRECTION} \wedge \text{CONTRAINTES} \wedge \text{REC}$$

est satisfaisable dans LTL + $\{\mathrm{L}_0, \mathrm{L}_1\}$. CQFD

Par conséquent, SAT(LTL + $\{\mathrm{L}_0\}$) n'est pas récursivement énumérable. La preuve de la proposition 2.3.5 est inspirée de la preuve d'indécidabilité de PDL avec le langage hors-contexte $\{a_1^n \cdot a_2 \cdot a_1^n : n \geq 0\}$ [HKT00, chapitre 9].

En ce qui concerne le problème de model-checking, grâce à une réduction en temps exponentiel entre SAT(LTL(X) + $\{\mathrm{L}_0\}$) et MC$^\exists$(LTL(X) + $\{\mathrm{L}_0\}$) on obtient le corollaire suivant.

Corollaire 2.3.6. MC$^\exists$(LTL(X) + $\{\mathrm{L}_0\}$) est Σ_1^1-complet.

En fait on peut même montrer que le problème de model-checking pour le calcul propositionnel enrichi de l'unique opérateur temporel défini à partir de L_1 est Σ_1^1-difficile. En effet, on a

– $\mathtt{X}\phi = \mathrm{L}_1(\bot, \top, \phi)$,
– $\mathtt{F}\phi = \mathrm{L}_1(\top, \phi, \top)$,
– $\mathrm{L}_0 = a_1 \mathrm{L}_1$ et donc $\mathrm{L}_0(\phi_1, \phi_2, \phi_3) \equiv \phi_1 \wedge \mathtt{X}\mathrm{L}_1(\phi_1, \phi_2, \phi_3)$.

Comme L_1 peut être défini à l'aide d'un CQDD [BH99, FIS03], on obtient naturellement que LTL augmenté des opérateurs définis à partir des CQDD est indécidable. Ce résultat, finalement pas très surprenant est à rapprocher de celui dans [DFGvD06] énonçant qu'une classe de systèmes à compteurs (fonctionnels, plats et dont les boucles admettent des fermetures transitives définissables dans l'arithmétique de Presburger) a un problème de model-checking décidable pour des propriétés arborescentes avec des opérateurs linéaires définissables avec des CQDD.

Soit L_2 le langage $\{a_1^k a_2 a_3^k a_4 : k \geq 0\}$. Ce langage est un "visibly pushdown language" (noté VPL dans la suite) [AM04], c'est-à-dire un langage reconnu par un automate à pile dont l'alphabet est divisé en trois parties disjointes: soit la lecture d'une lettre permet seulement d'empiler, soit elle permet seulement de dépiler soit elle laisse la pile intacte. Les langages acceptés par de tels automates

sont fermés par union, intersection et complémentation et admettent d'autres propriétés désirables [AM04]. Alors que PDL généralisé aux termes de programmes définissant des VLP est décidable [LS06], nous avons le résultat suivant.

Proposition 2.3.7. Le problème de model-checking pour LTL + L_2 est indécidable.

En effet, l'opérateur L_2 peut facilement exprimer L_1 avec $\mathrm{L}_1(\phi_1, \phi_2, \phi_3) \equiv \mathrm{L}_2(\phi_1, \phi_2, \phi_1, \phi_3)$.

2.4 Model-checking symbolique

Le problème de l'explosion du nombre d'états demeure un obstacle important à l'efficacité des algorithmes de vérification puisque le système à vérifier est souvent représenté comme un produit de systèmes [Rab00, LS00]. Ainsi le système produit est de taille exponentielle en la somme des tailles de chacun des composants. Pour tenter d'éviter ce problème, on peut observer que la complexité algorithmique du problème de la vérification dépend de deux mesures: la taille du modèle et la taille de la spécification (souvent une formule d'une logique temporelle). En pratique, on constate que la taille du modèle peut être grande tandis que la taille de la spécification peut être relativement petite puisqu'il s'agit souvent de vérifier une propriété prédéfinie (atteignabilité, vivacité, équité, etc). La complexité paramétrée de Downey et Fellows [DF99] offre un cadre séduisant pour analyser la complexité des problèmes de vérification lorsque l'on désire raffiner l'analyse du coût algorithmique en distinguant différents paramètres. La restriction consiste à fixer la valeur de certains paramètres et d'analyser alors la complexité du fragment ainsi obtenu. Avec une telle approche, on peut espérer expliquer pourquoi des problèmes algorithmiquement coûteux dans le pire des cas sont en pratique faciles.

Nous explorons le coût algorithmique du problème de l'explosion du nombre d'états à la lumière de la complexité paramétrée. Nous commençons par rappeler quelques notions élémentaires de cette théorie puisqu'elle n'est pas aussi répandue que la théorie standard.

2.4.1 Quelques notions de complexité paramétrée

Nous suivons les définitions présentées dans [DF99] concernant la complexité paramétrée. Un langage (problème) paramétré P est un ensemble de paires $\langle x, k\rangle$ pour $x \in \Sigma^*$ et $k \in \mathbb{N}$ où Σ est un alphabet fini. Dans $\langle x, k\rangle$ le paramètre est k. Un problème paramétré P est facile au sens paramétré ("fixed-parameter tractable") ssi il existe une fonction calculable $f : \mathbb{N} \mapsto \mathbb{N}$ et une constante $N \in \mathbb{N}$ tels que la question "$\langle x, k\rangle \in P$?" puisse être résolue en temps de calcul $f(k) \times |x|^N$, voir par exemple [DF99, chapitre 2]. La classe de tels problèmes est dénotée par FPT. Ce qui nous intéresse dans ce travail est de déterminer si des problèmes de model-checking où les structures sont représentées comme des produits de k structures

sont faciles au sens paramétré lorsque k est un paramètre. Ce serait alors une façon d'atténuer les effets du problème de l'explosion du nombre d'états.

Continuons avec quelques définitions utiles dans la suite. On dit qu'il existe une fp-réduction entre un problème paramétré P et un problème paramétré P' ssi il existe des fonctions calculables $f_1 : k \mapsto k'$, $f_2 : k \mapsto k''$, $f_3 : \langle x, k\rangle \mapsto x'$ et une constante $N \in \mathbb{N}$ tels que $\langle x, k\rangle \mapsto x'$ est calculable en temps $k''|x|^N$ et $\langle x, k\rangle \in P$ ssi $\langle x', k'\rangle \in P'$. Il est à noter que k' dépend seulement de k et pas de l'entrée x. P et P' sont dits fp-équivalents ssi il existe une fp-réduction de P vers P' et réciproquement.

Dans [DF99], Downey et Fellows ont introduit la hiérarchie AW de classes de problèmes paramétrès dont voici quelques degrés ci-dessous.

$$\begin{aligned}\mathrm{FPT} \subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \cdots \subseteq \mathrm{W}[\mathrm{SAT}] \subseteq \\ \subseteq \mathrm{AW}[1] \subseteq \mathrm{AW}[\mathrm{SAT}] \subseteq \mathrm{AW}[\mathrm{P}] \subseteq \mathrm{XP}.\end{aligned}$$

En fait, ces classes ont été introduites dans divers articles et la monographie [DF99] constitue un premier pas pour proposer une vision unifiée sur le sujet. On sait par exemple que FPT est différent de XP. Cette dernière classe contient exactement les problèmes paramétrés qui peuvent être résolus en temps $\mathcal{O}(|x|^{f(k)})$ où f est une fonction calculable. C'est l'analogue de la classe EXPTIME dans le cadre standard [Pap94]. De plus, la classe W[1] est considérée comme l'analogue paramétrée de la classe NP et par conséquent un problème W[1]-difficile n'est probablement pas facile même au sens paramétré. Non seulement la complexité paramétrée raffine le cadre standard de complexité algorithmique mais il n'existe pas de correspondence univoque entre les classes standard de complexité et les classes de complexité paramétrée. Par exemple, il existe des problèmes NP-complets dont la version paramétrée est dans FPT alors qu'il existe aussi des problèmes NP-complets dont la version paramétrée est W[1]-difficile, voir des exemples dans [DF99].

Comme dans le cadre classique, certains problèmes fondamentaux en complexité paramétrée utilisent des machines de Turing. Dans la suite nous considérons des machines déterministes (DTM), des machines non-déterministes (NDTM) ou des machines alternantes (ATM). Par défaut, ces machines ont un unique ruban. Le problème SHORT DTM COMPUTATION est le suivant:

entrée: un machine de Turing déterministe M et un entier positif k codé en unaire,

paramètre: k,

question: est-ce que M en commençant avec la chaîne vide peut atteindre un état acceptant en moins de k pas de calcul?

On peut aussi définir des problèmes analogues en contraignant l'espace mémoire utilisé. Le problème COMPACT ATM COMPUTATION est le suivant:

entrée: un machine de Turing alternante M et un entier positif k codé en unaire,

paramètre: k,

question: est-ce que M en commençant avec la chaîne vide peut atteindre un état acceptant en utilisant moins de k cellules mémoire sur le ruban?

On peut facilement montré que SHORT DTM COMPUTATION est FPT alors que SHORT NDTM COMPUTATION est W[1]-complet [DF99]. Les problèmes impliquant des contraintes sur la mémoire peuvent atteindre des classes plus élevées dans la hiérarchie. Par exemple, il a été établi dans [CF03] que COMPACT DTM COMPUTATION est AW[SAT]-difficile. Même si on ignore la complexité de tous les problèmes paramétrés sur les machines de Turing on connait des problèmes de ce type qui soient complets pour les classes W[1], W[2] et W[P] [Ces03]. Par exemple, SHORT NDTM COMPUTATION avec des machines non déterministes admettant plusieurs rubans est W[2]-complet [Ces03].

Il aussi intéressant de noter que dans la monographie [DF99] des problèmes paramétrés avec des machines alternantes ne sont pas considérés. Nous avons pu montrer les résultats suivants qui ont un intérêt non seulement pour la théorie de la complexité paramétrée mais aussi pour les problèmes paramétrés de model-checking comme on le verra dans la suite.

Théorème 2.4.1. [DLS06] SHORT ATM COMPUTATION est AW[1]-complet.

La preuve du théorème 2.4.1 consiste à montrer que SHORT ATM COMPUTATION est fp-équivalent au problème PARAMETERIZED-QBFSAT$_t$ montré AW[1]-complet dans [DF99, chapitre 14].

Théorème 2.4.2. [DLS06] COMPACT ATM COMPUTATION est XP-complet.

La preuve du théorème 2.4.2 consiste à montrer que le problème PEBBLE GAME est fp-équivalent à COMPACT ATM COMPUTATION alors que PEBBLE GAME a été prouvé XP-complet dans [DF99, théorème 15.5].

2.4.2 Problèmes paramétrés d'accessibilité

Un système de transition $\mathcal{A}$ est un tuple $\langle \Sigma, Q, \rightarrow \rangle$ tel que Σ est un alphabet (pas nécessairement fini), Q est un ensemble d'états (pas nécessairement fini) et $\rightarrow$ est une relation de transition $\rightarrow \subseteq Q \times \Sigma \times Q$. $\mathcal{A}$ est dit fini lorsque Σ et Q sont finis. Dans cette section, une représentation succincte d'un système de transition est une séquence de systèmes $\mathcal{A}_1, \ldots, \mathcal{A}_k$ qui représente le système $\langle \Sigma, Q, \rightarrow \rangle$ avec

- $Q \stackrel{\text{def}}{=} \prod_{i=1}^{k} Q_i$ où $\mathcal{A}_i = \langle Q_i, \Sigma_i, \rightarrow_i \rangle$ pour chaque i,
- $\Sigma \stackrel{\text{def}}{=} \bigcup_{i=1}^{k} \Sigma_i$,
- $\rightarrow \subseteq Q \times \Sigma \times Q$.

La définition exacte de la relation de transition $\rightarrow$ dépend du mode de synchronisation. Ici, nous allons distinguer la synchronisation forte de la synchronisation binaire et dans la suite par défaut on utilise la synchronisation forte. Dans la synchronisation forte, tous les composants bougent en même temps: $\langle s_1, \ldots, s_k \rangle \xrightarrow{a}_{\text{str}} \langle t_1, \ldots, t_k \rangle$ ssi pour $i \in \{1, \ldots, k\}$, $s_i \xrightarrow{a}_i t_i$. Par contraste, dans la synchronisation binaire, seuls deux composants se synchronisent tandis que les autres ne bougent pas: $\langle s_1, \ldots, s_k \rangle \xrightarrow{a}_{\text{bin}} \langle t_1, \ldots, t_k \rangle$ ssi il existe $i \neq j$ tels que $s_i \xrightarrow{a}_i t_i$, $s_j \xrightarrow{a}_j t_j$ et pour $l \notin \{i, j\}$, $s_l = t_l$.

Nous présentons ci-dessous les quatre problèmes d'accessibilité avec le système représenté succinctement.

– Accessibilité Exacte (EXACT-REACH):

  entrée: k systèmes de transition finis $\mathcal{A}_1, \cdots, \mathcal{A}_k$, deux configurations $\bar{s}$ et $\bar{t}$ de $\mathcal{A}_1 \times \cdots \times \mathcal{A}_k$.

  question: A-t-on $\bar{s} \xrightarrow{*} \bar{t}$?

– Accessibilité Locale (LOCAL-REACH):

  entrée: k systèmes de transition finis $\mathcal{A}_1, \cdots, \mathcal{A}_k$, k ensembles d'états F_1, ..., F_k avec $F_i \subseteq Q_i$ et une configuration $\bar{s}$ de $\mathcal{A}_1 \times \cdots \times \mathcal{A}_k$,

  question: A-t-on $\bar{s} \xrightarrow{*} \bar{t}$ pour une configuration $\bar{t} \in \bar{F}$ avec $\bar{F} = F_1 \times \cdots \times F_k$?

– Accessibilité Répétée (REP-REACH):

  entrée: comme pour LOCAL-REACH,

  question: A-t-on $\bar{s} \xrightarrow{*} \bar{t} \xrightarrow{+} \bar{t}$ pour une configuration $\bar{t} \in \bar{F}$?

– Accessibilité Equitable (FAIR-REACH):

  entrée: k systèmes de transition finis $\mathcal{A}_1, \cdots, \mathcal{A}_k$, des ensembles d'états $(F_i^j)_{i=1,\ldots,k}^{j=1,\ldots,p}$ avec $F_i^j \subseteq Q_i$ pour i, j, et une configuration $\bar{s}$ de $\mathcal{A}_1 \times \cdots \times \mathcal{A}_k$.

  question: A-t-on $\bar{s} \xrightarrow{*} \bar{t_1} \xrightarrow{*} \bar{t_2} \ldots \xrightarrow{*} \bar{t_p} \xrightarrow{+} \bar{t_1}$ pour des configurations $\bar{t_1}, \ldots, \bar{t_p} \in \bar{F^1}, \ldots, \bar{F^p}$ avec $\bar{F^j} = F_1^j \times \cdots \times F_k^j$ pour chaque j?

La contrainte d'accessibilité dans le problème REP-REACH fait référence à la condition d'acceptation dans les automates de Büchi tandis que celle dans le problème FAIR-REACH impose une condition d'équité. Pour chaque problème de la forme $\star$-REACH on note k-$\star$-REACH le problème paramétré correspondant où le nombre k de systèmes est le paramètre du problème. Les quatre problèmes ci-dessus (non paramétrés) sont connus pour être équivalents à réduction logarithmique en espace près puisqu'ils sont PSPACE-complets. Une analyse un peu plus fine permet d'établir que les quatre problèmes paramétrés de la forme k-$\star$-REACH sont aussi équivalents au sens des réductions paramétrées.

Théorème 2.4.3. [DLS02] Les quatre problèmes paramétrés d'accessibilité sont fp-équivalents.

Dans la suite, on notera k-$\star$-REACH n'importe lequel des quatre problèmes paramétrés. On peut en fait caractériser la complexité de k-$\star$-REACH.

Théorème 2.4.4. [DLS02, théorème 5.1] k-$\star$-REACH est fp-équivalent à COMPACT NDTM COMPUTATION.

Le théorème 2.4.4 implique que chaque problème k-$\star$-REACH est AW[SAT]-difficile ce qui place déjà assez haut ce problème dans la hiérarchie AW de Downey et Fellows. Par conséquent, il est improbable que k-$\star$-REACH puisse se résoudre en temps $\mathcal{O}(f(k) \times (\Sigma_i |\mathcal{A}_i|)^N)$ pour un degré fixe N et une fonction calculable $f(\cdot)$.

On peut aussi montrer que les deux résultats précédents sont très robustes si on considère de légères variantes de ces problèmes. On note k-$\star$-REACH$_{bin}$ les variantes respectives de k-$\star$-REACH avec synchronisation binaire au lieu de synchronisation forte. De même, on note $k, |\Sigma|$-$\star$-REACH$_{bin}$ les variantes de k-$\star$-REACH$_{bin}$ où les paramètres sont k et $|\Sigma|$ et k-$\star$-REACH$_{|\Sigma|=2}$ le sous-problème de k-$\star$-REACH avec un alphabet binaire.

La robustesse du théorème 2.4.4 est illustrée par le résultat suivant.

Théorème 2.4.5. [DLS02] Les problèmes suivants sont fp-équivalents à COMPACT NDTM COMPUTATION:

- k-$\star$-REACH$_{bin}$,
- $k, |\Sigma|$-$\star$-REACH$_{bin}$,
- $k, |\Sigma|$-$\star$-REACH,
- k-$\star$-REACH$_{|\Sigma|=2}$.

On peut aller plus loin en considérant des systèmes déterministes. On note k-$\star$-REACH$_{|\Sigma|=2,det}$ [resp. $k, |\Sigma|$-$*$-REACH$_{det}$] la restriction de k-$\star$-REACH$_{|\Sigma|=2}$ [resp. $k, |\Sigma|$-$*$-REACH] aux systèmes de transition déterministes.

Théorème 2.4.6. [DLS02] Les problèmes suivants sont fp-équivalents à COMPACT NDTM COMPUTATION:

- k-$\star$-REACH$_{det}$,
- $k, |\Sigma|$-$\star$-REACH$_{det}$,
- k-$\star$-REACH$_{|\Sigma|=2}, det$.

Comme effet de bord nous obtenons que les problèmes FAI-II et FAI-III de [DF99, page 470] sont fp-équivalents à COMPACT NDTM COMPUTATION et donc ces problèmes sont AW[SAT]-difficiles. Cela raffine la meilleure borne de complexité inférieure connue à savoir que ces problèmes étaient W[t]-difficiles pour tous les $t \geq 0$ [DF99].

2.4.3 Problèmes paramétrés de model-checking

Dans cette section, nous allons nous intéresser à des problèmes paramétrés de model-checking pour logiques temporelles lorsque le système de transition est représenté symboliquement. Dans la suite un modèle de Kripke est compris comme un système de transition enrichi d'une valuation construit sur un ensemble infini dénombrable PROP de variables propositionnelles. Il s'agit d'une généralisation de la notion utilisée dans la section 2.1 dans laquelle les transitions sont étiquetées. C'est d'ailleurs aussi celle utilisée dans le chapitre 5.

Plus précisément, un modèle de Kripke $\mathcal{M}$ est une structure de la forme $\langle \Sigma, Q, \rightarrow, V \rangle$ avec $V : Q \rightarrow \mathcal{P}(\mathrm{PROP})$. $\mathcal{M}$ est dit fini lorsque $\langle \Sigma, Q, \rightarrow \rangle$ est un système de transition fini et chaque ensemble de la portée de V est fini. On peut aussi définir le produit de modèles de Kripke construit sur le produit de systèmes de transition. Ainsi si $\mathcal{M}_1 = \langle \Sigma_1, Q_1, \rightarrow_1, V_1 \rangle, \cdots, \mathcal{M}_k = \langle \Sigma_k, Q_k, \rightarrow_k, V_k \rangle$ alors $\mathcal{M} = \langle \Sigma, Q, \rightarrow, V \rangle = \mathcal{M}_1 \times \cdots \times \mathcal{M}_k$ avec $\langle \Sigma, Q, \rightarrow \rangle = \langle \Sigma_1, Q_1, \rightarrow_1 \rangle \times$

$\cdots \times \langle \Sigma_k, Q_k, \rightarrow_k \rangle$ et V est construit à partir de $V_1, \ldots, V_k$. Par exemple, on suppose dans la suite que $V(\langle s_1, \ldots, s_k \rangle) = \bigcup_i V_i(s_i)$. Nous allons à présent définir le problème de model-checking pour une logique temporelle $\mathcal{L}$ (par exemple parmi LTL, CTL, CTL$\star$, le μ-calcul modal, etc.) lorsque la structure de Kripke est représentée symboliquement comme un produit de structures. En fait par souci de simplification, on supposera qu'une structure de Kripke produit est construite sur un alphabet singleton, c'est-à-dire $\langle s_1, \ldots, s_k \rangle \rightarrow \langle t_1, \ldots, t_k \rangle$ dans $\mathcal{M}$ ssi il existe une lettre $a \in \Sigma_1 \cap \cdots \cap \Sigma_k$ tel que pour $i \in \{1, \ldots, k\}$, $s_i \xrightarrow{a} t_i$. En calculant le produit, on renomme donc implicitement les étiquettes.
Le problème paramétré de model-checking pour la logique $\mathcal{L}$, noté PMC($\mathcal{L}$) est défini de la façon suivante:

entrée: k structures de Kripke finies $\mathcal{M}_1, \ldots, \mathcal{M}_k$, une formule ϕ de $\mathcal{L}$ et une configuration $\bar{s}$ de $\mathcal{M}_1 \times \cdots \times \mathcal{M}_k$,
paramètre: $k + |\phi|$,
question: A-t-on $\mathcal{M}_1 \times \cdots \times \mathcal{M}_k, \bar{s} \models \phi$?

Nos résultats sur les problèmes paramétrés d'accessibilité et la possibilité de transformer une formule de LTL en un automate de Büchi équivalent (voir par exemple [VW94]) permettent d'établir ce résultat important concernant LTL.

Théorème 2.4.7. [DLS02, théorème 6.1] PMC(LTL) est fp-équivalent à COMPACT NDTM COMPUTATION.

Il est possible de se réjouir que la complexité paramétrée de PMC(LTL) admette une caractérisation élégante. Cependant, en pratique, la AW[SAT]-dureté du problème interdit tout espoir pour contourner le problème de l'explosion du nombre d'états. De plus, même un fragment ridiculement inexpressif de LTL produit un problème paramétré de model-checking W[1]-difficile. Soit LTL$_{poor}$ le fragment de LTL réduit à une unique variable propositionnelle avec uniquement les opérateurs $\vee$ et X.

Théorème 2.4.8. [DLS02, théorème 6.2] PMC(LTL$_{poor}$) est W[1]-complet.

Passons à présent à quelques logiques temporelles du temps arborescent. Considérons d'abord la logique modale de Hennessy-Milner [HM85b] avec variables propositionnelles dont la syntaxe est rappelée ci-dessous:

$$\phi ::= \mathrm{p} \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \Box\phi \mid \Diamond\phi,$$

où $\mathrm{p} \in \mathrm{PROP}$. C'est juste la logique modale K (voir chapitre 5). Un modèle de HML est une structure de Kripke avec un alphabet unaire et $\mathcal{M}, x \models \Box\phi$ est vérifié lorsque $\mathcal{M}, x' \models \phi$ pour tous les états successeurs x' de x. HML ne permet d'exprimer des questions d'accessibilité car l'interprétation des opérateurs $\Box$ et $\Diamond$ n'impliquent que les successeurs immédiats.

Théorème 2.4.9. [DLS02] PMC(HML) est AW[1]-complet.

La preuve dans [DLS02] établit que PMC(HML) est fp-équivalent à SHORT ATM COMPUTATION qui est aussi montré AW[1]-complet dans [DLS02]. Lorsqu'on ajoute à HML des opérateurs de point fixe, on obtient le μ-calcul modal [AN01] et dans ce cas, la complexité du problème paramétré du model-checking est très élévée.

Théorème 2.4.10. [DLS02] PMC(μ) est XP-complet.

Pour montrer que PMC(μ) est dans XP, on utilise qu'une instance de PMC(μ) peut être résolue en temps $\mathcal{O}\big((|\phi|.n^k)^{|\phi|}\big)$ avec $n = \Sigma_i|\mathcal{M}_i|$ [KVW00, théorème 6.4]. La XP-dureté est obtenue par réduction de COMPACT ATM COMPUTATION qui a aussi été montré XP-complet dans [DLS02]. On peut remarquer que déjà dans [Rab97, Rab00] la version non paramétrée de PMC(μ) a été montrée EXPTIME-complète alors que le model-checking pour le μ-calcul modal est connu pour être dans UP$\cap$ co-UP [Jur98].

D'autres problèmes paramétrés avec des modèles symboliques pour des questions relatives à des équivalences comportementales comme la bisimulation forte ont été étudiés dans [DLS02] raffinant les résultats de EXPTIME-complétude pour la version non paramétrée de ces problèmes [JM96, LS00] (et quelquefois en utilisant certaines des preuves). La figure 2.4 contient un résumé des résultats de [DLS02] incluant certains problèmes non traités explicitement dans ce mémoire.

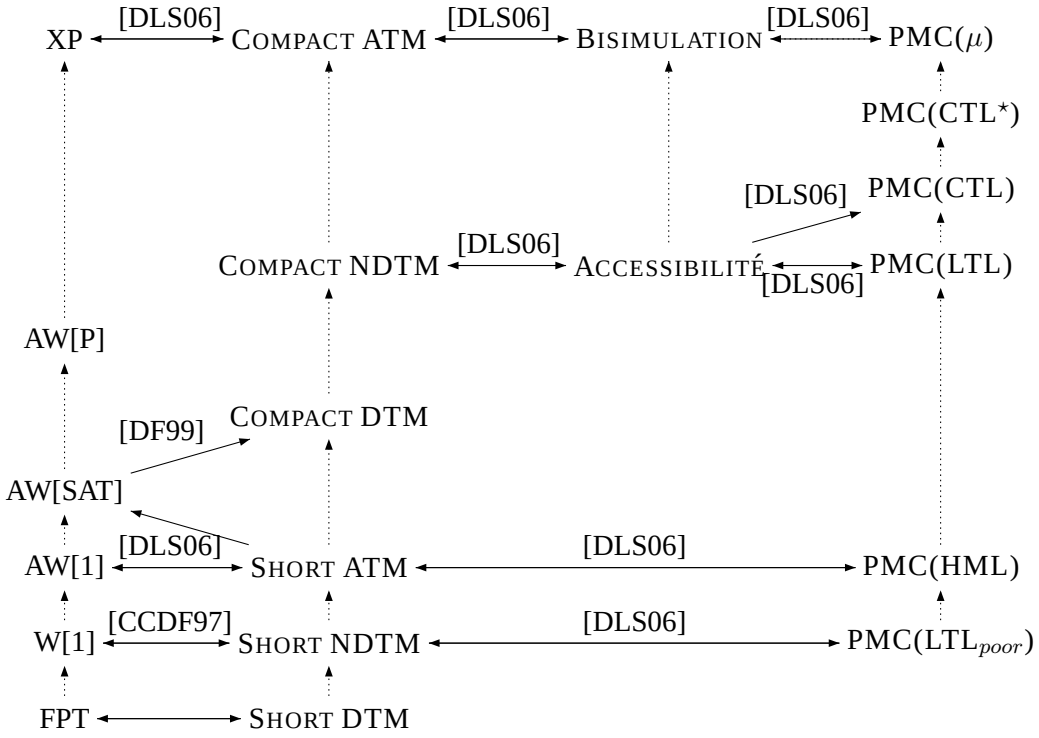


FIG. 2.4 – *Complexité des problèmes de model-checking paramétrés*

Alors que NPSPACE est à égale à PSPACE, on ignore si COMPACT DTM

COMPUTATION est strictement inclus dans COMPACT NDTM COMPUTATION.

Problème ouvert 3. Y-a-t-il une fp-réduction de COMPACT NDTM COMPUTATION vers COMPACT DTM COMPUTATION? ◯

En ce qui concerne les problèmes liés aux logiques temporelles, on sait seulement que PMC(CTL) et PMC(CTL$\star$) sont AW[SAT]-difficiles.

Problème ouvert 4. Quelle est la complexité de PMC(CTL) et PMC(CTL*)? En particulier, peut-on réduire PMC(CTL$\star$) à COMPACT NDTM COMPUTATION? ◯

2.5 Extension de LTL avec des mots transfinis

Modéliser l'interaction entre un ordinateur et un système physique nécessite la possibilité de travailler avec des échelles de temps très différentes en particulier si l'on cherche à contrôler un tel système. Certains systèmes physiques dont les comportements sont régis par des équations différentielles admettent des comportements Zeno pour lesquels un nombre infini d'actions a lieu en un temps borné. Ces comportements sont naturellement exclus des systèmes informatiques même si l'analyse de systèmes admettant des comportements Zeno peut avoir du sens [BP97, HLMR02], voir aussi un usage de séquences de longueur supérieure à ω pour la vérification dans [GW94].

Dans le travail présenté ci-dessous, nous nous sommes intéressés à la conception de langages de spécification pour exprimer des comportements Zeno dans le but de contrôler des systèmes physiques. Cette motivation nous a conduit à introduire dans un premier temps une famille de logiques temporelles étendant LTL (section 2.5.1) et à étudier les problèmes de model-checking et satisfaisabilité (section 2.5.2). Finalement, la section 2.5.3 présente un problème de contrôle qui a motivé l'introduction de ces logiques.

2.5.1 Une famille d'extensions de LTL

Nous rappelons brièvement qu'un ordinal est une classe d'équivalence d'ordres linéaires bien ordonnés, voir [Ros82] pour les définitions complètes.

Soient α un ordinal dénombrable et fermé par addition (pour $\beta, \beta' < \alpha$, on a $\beta + \beta' < \alpha$) et PROP un ensemble de variables propositionnelles infini dénombrable. La logique $\mathrm{LTL}(\alpha)$ a pour modèle les séquences $\sigma : \alpha \to \mathcal{P}(\mathrm{PROP})$ où comme d'habitude α est compris comme l'ensemble $\{\beta : \beta < \alpha\}$. Les formules de $\mathrm{LTL}(\alpha)$ sont définies par la grammaire suivante:

$$\phi ::= \mathrm{p} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \mathtt{X}^\beta\phi \mid \phi_1 \mathtt{U}^{\beta'} \phi_2,$$

avec $\mathrm{p} \in \mathrm{PROP}$, $\beta < \alpha$ et $\beta' \leq \alpha$. La relation de satisfaction est définie inductivement ci-dessus où σ est un modèle de $\mathrm{LTL}(\alpha)$ et $\beta < \alpha$:

– $\sigma, \beta \models \mathrm{p}$ ssi $\mathrm{p} \in \mathcal{M}(\beta)$,

– $\sigma, \beta \models \neg\phi$ ssi $\sigma, \beta \not\models \phi$,
– $\sigma, \beta \models \phi_1 \wedge \phi_2$ ssi $\sigma, \beta \models \phi_1$ et $\sigma, \beta \models \phi_2$,
– $\sigma, \beta \models \mathtt{X}^{\beta'}\phi$ ssi $\sigma, \beta + \beta' \models \phi$,
– $\sigma, \beta \models \phi_1 \mathtt{U}^{\beta'} \phi_2$ ssi il existe $\gamma < \beta'$ tel que $\sigma, \beta + \gamma \models \phi_2$ et pour chaque $\gamma' < \gamma$, on a $\sigma, \beta + \gamma' \models \phi_1$.

La fermeture par addition garantit que $\beta + \beta'$ et $\beta + \gamma$ sont bien strictement inférieurs à α. L'ensemble des modèles de ϕ, noté $\mathrm{Mod}(\phi)$, est $\{\sigma : \alpha \to \mathcal{P}(\mathrm{PROP}) : \sigma, 0 \models \phi\}$.

L'opérateur $\mathtt{X}^\beta$ constitue une généralisation naturelle de l'opérateur $\mathtt{X}$ de LTL qui permet d'effectuer un saut d'une longueur β. De même, l'opérateur $\mathtt{U}^\beta$ généralise l'opérateur $\mathtt{U}$ de LTL. Dans la suite on notera aussi $\mathtt{F}^\beta\phi$ pour $\top\mathtt{U}^\beta\phi$ et $\mathtt{G}^\beta\phi$ pour $\neg\mathtt{F}^\beta\neg\phi$. On peut facilement montrer que $\mathrm{LTL}(1)$ est équivalent au calcul propositionnel tandis que $\mathrm{LTL}(\omega)$ a la même expressivité que LTL mais pas la même concision si les entiers ont une représentation binaire.

Dans la suite on utilise la forme normale de Cantor pour représenter les ordinaux inférieurs à ω^ω et par défaut les entiers utilisés pour une telle représentation sont codés en binaire.

Voici quelques exemples de propriétés nécessitant des états limites qui peuvent facilement s'exprimer dans la logique $\mathrm{LTL}(\omega^k)$ ($k \geq 2$):

1. "p est vérifiée dans toutes les positions limites inférieures à ω^k":

$$\mathtt{G}^{\omega^k}(\mathtt{X}^\omega \mathrm{p} \wedge \cdots \wedge \mathtt{X}^{\omega^{k-1}} \mathrm{p}).$$

2. Pour $1 \leq k' \leq k-2$, "si p est vérifiée infiniment souvent aux positions de la forme $\omega^{k'} \times n$ pour $n \geq 1$, alors q est vraie à la position $\omega^{k'+1}$":

$$(\mathtt{G}^{\omega^{k'+1}} \mathtt{F}^{\omega^{k'+1}} \mathtt{X}^{\omega^{k'}} \mathrm{p}) \Rightarrow (\mathtt{X}^{\omega^{k'+1}} q).$$

Proposition 2.5.1. [DN07] Soit α un ordinal tel que $0 \leq \alpha \leq \omega$. Le problème de la satisfaisabilité pour $\mathrm{LTL}(\omega^\alpha)$ est décidable.

La preuve de [DN07] est par réduction vers la théorie monadique du second ordre de $\langle \omega^\omega, < \rangle$ qui est montrée décidable dans [BS73, théorème 4.12]. Une autre réduction vers la théorie du premier ordre a été proposée dans [Cac06]. Cela permet d'obtenir en effet la décidabilité mais avec une borne de complexité non-élémentaire [Mey73]. Dans la suite nous allons caractériser exactement la complexité du problème de satisfaisabilité pour $\mathrm{LTL}(\omega^k)$ avec $k < \omega$ en adoptant l'approche de [VW94]. Il est important de rappeler que l'approche introduite par Büchi dans [Büc62, Büc64, Büc65, BS73] a aussi consisté à résoudre un problème logique en le réduisant à un problème sur des automates idoines et qu'historiquement ce fut la première de ce type.

Problème ouvert 5. Est-ce que pour tous les ordinaux dénombrables et fermé par addition α, la logique $\mathrm{LTL}(\alpha)$ est décidable? ◯

Dans [Roh97], une logique interprétée sur des α-séquences et munie des opérateurs "Next" et "Until" (sans décoration) a été introduite. Il a été montré que

le problème de satisfaisabilité qui admet en entrée une formule (qui est aussi une formule de LTL) et un ordinal dénombrable α est dans EXPTIME [Roh97]. Outre la théorie monadique du second ordre de $\langle \alpha, < \rangle$ avec α dénombrable [BS73], on peut aussi citer la logique temporelle introduite dans [BLZ96] où des sauts de temps sont modélisés par la notion de limite dans les ordinaux.

Avant de définir le problème de model-checking pour $\mathrm{LTL}(\alpha)$, nous rappelons la notion d'automate transfini qui généralise celle des automates de Muller.

Définition 2.5.1. Un automate transfini $\mathcal{A}$ est un tuple $\langle Q, \Sigma, \delta, E, I, F \rangle$ tel que

- Q est un ensemble fini d'états,
- Σ est un alphabet fini,
- $\delta \subseteq Q \times \Sigma \times Q$ est la relation de transition,
- $E \subseteq \mathcal{P}(Q) \times Q$ est la relation de transition limite,
- $I \subseteq Q$ est l'ensemble des états initiaux et $F \subseteq Q$ est l'ensemble des états finaux.

∇

On écrira $q \xrightarrow{a} q'$ lorsque $\langle q, a, q' \rangle \in \delta$, $q \rightarrow q'$ lorsque $q \xrightarrow{a} q'$ pour une lettre $a \in \Sigma$. Un chemin de longueur $\alpha + 1$ est une application $r : \alpha + 1 \rightarrow Q$ telle que

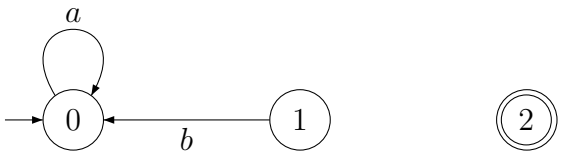
- pour $\beta \in \alpha$, $r(\beta) \rightarrow r(\beta + 1)$,
- pour chaque ordinal limite $\beta \leq \alpha$, il existe une transition limite $P \rightarrow r(\beta) \in E$ telle que $P = inf(\beta, r)$ avec

$$inf(\beta, r) \stackrel{\text{def}}{=} \{q \in Q : \text{pour chaque } \gamma \in \beta, \text{ il existe } \gamma' \text{ tel que } \gamma < \gamma' < \beta \text{ et } r(\gamma') = q\}.$$

Un calcul de longueur $\alpha + 1$ est un chemin de longueur $\alpha + 1$ tel que $r(0) \in I$. Si de plus, $r(\alpha) \in F$ alors le calcul r est dit acceptant. L'ensemble des séquences reconnues par l'automate $\mathcal{A}$, dénoté par $\mathrm{L}(\mathcal{A})$, est l'ensemble des α-séquences $\sigma : \alpha \rightarrow \Sigma$ pour lesquelles il y a un calcul acceptant de longueur $\alpha + 1$ vérifiant pour chaque $\beta \in \alpha$ que $r(\beta) \xrightarrow{\sigma(\beta)} r(\beta + 1)$.

Les automates transfinis de la définition 2.5.1 sont ceux définis dans [HW91]. Ce sont aussi exactement les B'-automates de [Bed98, page 35], une variante de automates de Wojciechowski sans lettre sur les transitions limites [Woj84]. L'équivalence entre ces divers formalismes est montré dans [Bed98, section 2.5]. On retrouve une définition analogue dans [Car01, définition 17] et une généralisation très élégante est faite dans [BC01] pour reconnaître des séquences sur des ordres linéaires dispersés.

A titre d'exemple, l'automate $\mathcal{A}$ ci-dessous avec pour transitions limites $\{0\} \rightarrow 1$ et $\{0, 1\} \rightarrow 2$ reconnaît seulement des ω^2-séquences.



Plus précisément, $\mathrm{L}(\mathcal{A}) = (a^\omega \cdot b)^\omega$.

Nous pouvons à présent proposer une version existentielle du problème de model-checking pour $\mathrm{LTL}(\alpha)$:

entrée: un automate transfini $\mathcal{A}$ dont l'alphabet est un sous-ensemble de $\mathcal{P}(\mathrm{PROP})$ et ϕ est une formule de $\mathrm{LTL}(\alpha)$,

question: est-ce qu'il existe une α-séquence σ reconnue par $\mathcal{A}$ telle que $\sigma, 0 \models \phi$?

2.5.2 Analyse de la complexité avec automates succincts

La proposition 2.1.1 admet un analogue dans le cas transfini. On peut en effet construire un automate transfini qui reconnaisse les modèles d'une formule. Si ce n'était que cela, ce fait serait une conséquence de [Büc64], cependant nous pouvons obtenir de cette traduction une borne de complexité élémentaire.

Proposition 2.5.2. [DN07] Soit ϕ une formule de $\mathrm{LTL}(\omega^k)$ avec $k < \omega$. Il existe un automate transfini $\mathcal{A}_\phi = \langle Q, \Sigma, \delta, E, I, F\rangle$ tel que

1. $\mathrm{L}(\mathcal{A}_\phi) = \mathrm{Mod}(\phi)$,
2. $|Q|$ est doublement exponentiel en $|\phi|$,
3. $|\delta \cup E|$ est triplement exponentiel en $|\phi|$.

Comme le problème de vacuité pour les automates transfinis est dans PTIME d'après [Car02, proposition 6], on obtient une première borne supérieure 3EXPTIME pour le problème de satisfaisabilité pour $\mathrm{LTL}(\omega^k)$ avec $k < \omega$. Il serait peut-être possible de raffiner légérement ce résultat si le problème de vacuité était connu pour ne pas être PTIME-difficile. Cependant le problème suivant est ouvert.

Problème ouvert 6. Est-ce que le problème de vacuité pour les automates transfinis est PTIME-difficile? ◯

Nous allons à présent entamer notre analyse de complexité en énonçant le résultat ci-dessous.

Lemme 2.5.3. [DN07]

(I) Le problème de satisfaisabilité pour $\mathrm{LTL}(\omega)$ est EXPSPACE-complet.

(II) Pour tous les ordinaux α, le problème de satisfaisabilité pour $\mathrm{LTL}(\omega^\alpha)$ est EXPSPACE-difficile.

L'énoncé (II) du lemme 2.5.3 est une conséquence de l'énoncé (I). La EXPSPACE-dureté dans l'énoncé (I) est obtenu en adaptant légèrement la preuve de [Har83, théorème 4.7] établissant la PSPACE-dureté de LTL. La borne supérieure EXPSPACE est obtenue par traduction exponentielle vers LTL. Cependant, si on suppose que les entiers sont codés en unaire dans $\mathrm{LTL}(\omega)$, la satisfaisabilité est alors seulement PSPACE-complète comme pour LTL.

Nous allons à présent compléter notre analyse de complexité et pour ce faire nous proposons d'introduire une nouvelle classe d'automates. Il s'agit d'une sous-classe d'automates transfinis représentés succinctement et reconnaissant des ω^k-séquences. Des automates similaires ont été définis dans [Cho78, HW91, Bed98] sauf pour les propriétés de concision qui vont suivre.

Définition 2.5.2. Un automate transfini $\mathcal{A} = \langle Q, \Sigma, \delta, E, I, F\rangle$ est dit de niveau $k \geq 1$ s'il existe une application $l : Q \to \{0, \ldots, k\}$ telle que

- pour $q \in F$, $l(q) = k$,
- $q \xrightarrow{a} q' \in \delta$ alors $l(q') = 0$ et $l(q) < k$,
- si $P \rightarrowtail q \in E$ alors $max\{l(q') : q' \in P\} + 1 = l(q)$.

∇

On notera $\langle Q, \Sigma, \delta, E, I, F, l\rangle$ un automate transfini avec fonction de niveau l. Chaque ensemble d'états ayant le même niveau correspond à une couche dans les automates de Choueka [Cho78]. L'automate de la proposition 2.5.2 est bien un automate de niveau k lorsque ϕ est une formule de $\mathrm{LTL}(\omega^k)$. Cependant sa taille est trop importante. C'est pourquoi nous introduisons une sous-classe d'automates transfinis qui peuvent représenter succinctement un nombre exponentiel de transitions limites comme les automates de Büchi généralisés peuvent être compris comme des automates de Muller représentés succinctement.

Définition 2.5.3. [DN07] Soient $p(\cdot)$ un polynôme et $k \geq 1$. Un automate transfini $p(\cdot)$-succinct de niveau k est une structure $\mathcal{A} = \langle Q, \Sigma, \delta, E, I, F, l\rangle$ définie comme un automate de niveau k sauf que la relation E est précisément un ensemble de tuples $\langle P_0, P_1, \ldots, P_n, q\rangle$ avec $n \geq 0$, $q \in Q$ et $P_0, \ldots, P_n \subseteq Q$ tels que

- si $\langle P_0, P_1, \ldots, P_n, q\rangle \in E$ alors
  1. $1 \leq l(q) \leq k$,
  2. chaque état de P_0 est de niveau $l(q) - 1$,
  3. chaque état de $P_1 \cup \cdots \cup P_n$ est de niveau inférieur ou égal à $l(q) - 1$,
  4. $n \leq p(|Q|)$,
- pour chaque état q de niveau strictement positif, il existe au plus une transition dans E de la forme $\langle P_0, P_1, \ldots, P_n, q\rangle$.

∇

En fait, chaque tuple $\langle P_0, P_1, \ldots, P_n, q\rangle$ code succinctement l'ensemble des transitions limites

$$trans(\langle P_0, P_1, \ldots, P_n, q\rangle) \stackrel{\text{def}}{=}$$

$$\{P \rightarrowtail q : \ P \subseteq Q, \ \forall\, i \ P_i \cap P \neq \emptyset \text{ et } \forall q' \in P, \ l(q') < l(q)\}.$$

Le langage reconnu par un automate transfini $p(\cdot)$-succinct de niveau k est celui reconnu par l'automate transfini de niveau k équivalent obtenu en considérant l'ensemble des transitions limites issu de l'ensemble E. La concision de tels automates se résume dans le fait que la taille de E est en $\mathcal{O}(|Q|^2 \times p(|Q|))$. En général,

un automate transfini de niveau k ordinaire peut avoir un nombre de transitions limites exponentiel en $|Q|$.

Proposition 2.5.4. [DN07] Soit ϕ une formule de $\mathrm{LTL}(\omega^k)$ avec $k < \omega$ et $p_0(\mathsf{x}) = \mathsf{x}$ (identité). Il existe un automate transfini $p_0(\cdot)$-succinct de niveau k $\mathcal{A}_\phi$ tel que

1. $\mathrm{L}(\mathcal{A}_\phi) = \mathrm{Mod}(\phi)$,
2. la taille de $\mathcal{A}_\phi$ est exponentielle en $|\phi|$ si les entiers dans ϕ sont codés en unaire. Lorsque les entiers sont codés en binaire, la taille de $\mathcal{A}_\phi$ devient doublement exponentielle en $|\phi|$.

L'automate de la proposition 2.5.4 est obtenu en observant que l'automate de la proposition 2.5.2 peut-être représenté succinctement. Il se trouve qu'à k fixé, le problème de vacuité est aussi dans NLOGSPACE comme pour les automates de Büchi.

Théorème 2.5.5. [DN07] Pour $k \geq 0$,

(I) le problème de vacuité pour les automates de niveau k est NLOGSPACE-complet,

(II) pour chaque polynôme $p(\cdot)$, le problème de vacuité pour les automates transfinis $p(\cdot)$-succincts de niveau k est NLOGSPACE-complet.

Par conséquent nous retrouvons un résultat classique lorsque $k = 1$.

Corollaire 2.5.6. Le problème de vacuité pour les automates de Muller est NLOGSPACE-complet.

Nous pouvons à présent conclure concernant la complexité des problèmes de model-checking et satisfaisabilité.

Théorème 2.5.7. [DN07] Pour $k \geq 1$, les problèmes de model-checking et satisfaisabilité pour $\mathrm{LTL}(\omega^k)$ sont EXPSPACE-complets quand les entiers sont codés en binaires et PSPACE-complets si les entiers sont codés en unaire.

Problème ouvert 7. Quelle est la complexité des problèmes de satisfaisabilité et model-checking pour $\mathrm{LTL}(\omega^\omega)$? ◯

Une autre façon de montrer le théorème 2.5.7 suggérée par A. Rabinovich [Rab06] consiste à montrer que LTL avec les opérateurs stricts "Since" et "Until" sur les ω^ω-séquences est dans PSPACE. En effet, il est alors possible de définir de façon concise une formule φ_i qui exprime que la position courante est un multiple de ω^i pour $i \in \omega$. Nos opérateurs $\mathtt{U}^{\omega^i}$ et $\mathtt{X}^{\omega^i}$ pour $i \geq 1$ sont alors définissables:

$$\psi \mathtt{U}^{\omega^i} \psi' \equiv \psi' \vee ((\neg\varphi_i \wedge \psi)\mathtt{U}(\neg\varphi_i \wedge \psi'))$$

$$\mathtt{X}^{\omega^i}\psi \equiv ((\neg\varphi_i)\mathtt{U}(\varphi_i \wedge \psi)).$$

Pour obtenir une réduction qui soit logarithmique en espace, il faut cependant utiliser une technique de renommage qui s'applique bien ici.

Problème ouvert 8. Est-ce que pour tous les ordinaux dénombrables α, la logique LTL avec les opérateurs stricts "Since" et "Until" interprétée sur les α-séquences est dans PSPACE? ◯

Au moment d'imprimer ce document, une réponse positive peut être donnée à ce problème et fera l'objet d'une publication prochaine.

2.5.3 Un problème de contrôle

Ce qui a motivé l'introduction des logiques $\mathrm{LTL}(\alpha)$ dans [DN07] est un problème de contrôle dont par ailleurs une première solution a déjà été proposée dans [Cac06]. Avant d'en donner une définition formelle, ce qui est l'objet de cette section, en voici une description succincte. Etant donné un système physique $\mathcal{S}$ modélisé par un automate transfini reconnaissant des ω^k-séquences et une propriété ϕ de $\mathrm{LTL}(\omega^k)$, existe-t-il un contrôleur $\mathcal{C}$ reconnaissant des ω-séquences tel que le système $\mathcal{S} \times \mathcal{C}$ vérifie ϕ? L'opérateur de synchronisation $\times$ prend en compte les différentes échelles de temps de $\mathcal{S}$ et $\mathcal{C}$, et les vecteurs de synchronisation dépendent de l'ensemble des actions observables du système.

Afin de synchroniser le système avec un contrôleur travaillant sur des ω-séquences, nous devons transformer un automate transfini de niveau 1 $\mathcal{A}$ en un automate transfini de niveau $k \geq 2$ $\mathit{lift}_k(\mathcal{A})$ tel que pour tous les mots $\sigma \in \Sigma^{\omega^k}$, $\sigma \in \mathrm{L}(\mathit{lift}_k(\mathcal{A}))$ ssi l'ω-séquence $\sigma' \in \Sigma^\omega$, définie par $\sigma'(i) = \sigma(\omega^{k-1} \times i)$, appartient à $\mathrm{L}(\mathcal{A})$. On peut facilement contruire un tel automate $\mathit{lift}_k(\mathcal{A})$ à partir de $\mathcal{A}$ dans le cas général. Par exemple, soit $\mathcal{A}$ l'automate transfini de niveau 1 décrit à gauche de la figure 2.5 avec pour unique transition limite $\{q_0, q_1, q_2\} \to q_3$. L'automate $\mathit{lift}_2(\mathcal{A})$ avec les transitions limites $\{\langle 0, q_0\rangle\} \to \langle 1, q_0\rangle$, $\{\langle 0, q_1\rangle\} \to \langle 1, q_1\rangle$, $\{\langle 0, q_2\rangle\} \to \langle 1, q_2\rangle$, et $\{\langle 0, q_0\rangle, \langle 1, q_0\rangle, \langle 0, q_1\rangle, \langle 1, q_1\rangle, \langle 0, q_2\rangle, \langle 1, q_2\rangle\} \to q_3$ est décrit à droite de la figure 2.5.

Un système physique est défini comme un tuple $\langle \mathcal{A}, \mathit{Act_C}, \mathit{Act_O}, \mathit{Act}\rangle$ tel que:

- $\mathcal{A}$ est un automate transfini de niveau k sur l'alphabet $\mathcal{P}(\mathit{Act})$ où *Act* est un ensemble fini d'actions,
- $\mathit{Act_O} \subseteq \mathit{Act}$ est l'ensemble des actions observables,
- $\mathit{Act_C} \subseteq \mathit{Act_O}$ est l'ensemble des actions contrôlables. L'ensemble des actions incontrôlables est noté par $\mathit{Act_{nC}}$.

Une spécification du système $\mathcal{S}$ est une formule ψ de $\mathrm{LTL}(\omega^k)$. Un contrôleur $\mathcal{C}$ pour $\langle \mathcal{S}, \psi\rangle$ est un système dont les exécutions complètes sont des ω-séquences (par exemple un automate transfini de niveau 1) vérifiant les propriétés suivantes.

- Seules les actions observables sont présentes dans le contrôleur. La synchronisation entre $\mathcal{S}$ et $\mathcal{C}$ va garantir que les actions inobservables ne vont pas modifier l'état du contrôleur:

  (obs) l'alphabet de $\mathcal{C}$ est $\mathcal{P}(\mathit{Act_O})$ et pour chaque état q de $\mathcal{C}$, il existe une transition $q \xrightarrow{\emptyset} q$.

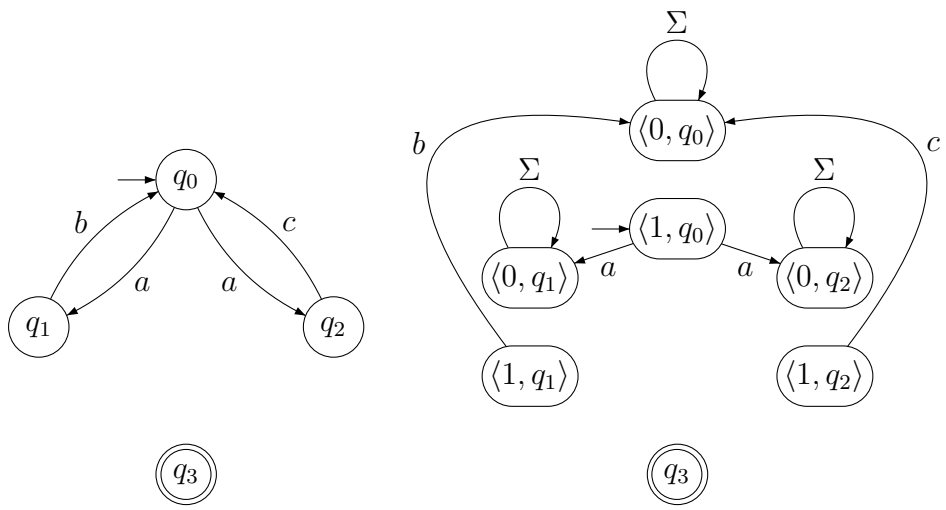


FIG. 2.5 – *Un automate et son lifting*

– De tous les états de $\mathcal{C}$, les actions incontrôlables peuvent toujours être exécutées:

  (unc) $\forall q \cdot \forall a \subseteq Act_O \setminus Act_C$, il y a une transition $q \xrightarrow{b} q'$ dans $\mathcal{C}$ telle que $b \cap Act_{NC} = a$.

– Finalement, le système $\mathcal{S}$ contrôlé par $\mathcal{C}$ satisfait ψ. Comme $\mathcal{S}$ et $\mathcal{C}$ ne reconnaissent pas nécessairement des mots de même longueur, le système contrôlé est en fait $lift_k(\mathcal{C}) \times_Y \mathcal{S}$ pour un ensemble Y de vecteurs de synchronisation. $\mathcal{S}$ et $\mathcal{C}$ se synchronisent sur les actions observables

  (syn) $Y = \{\langle X, X', X''\rangle \in Act \times Act_O \times Act : X \cap Act_O = X',\ X = X''\}$.

  Cela revient dont à vérifier que l'automate produit ci-dessous reconnaît un langage non-vide $(\mathcal{S} \times_Y lift_k(\mathcal{C})) \times \mathcal{A}_{\neg\psi}$. L'opérateur de synchronisation $\times_Y$ est défini dans [DN07] et correspond à la notion naturelle (encore faut-il faire attention aux transitions limites).

Ainsi, le problème de contrôle pour $\mathrm{LTL}(\omega^k)$ est le suivant:

entrée: un système $\mathcal{S}$ (avec un automate transfini de niveau k) et une formule ϕ de $\mathrm{LTL}(\omega^k)$ construite sur les variables propositionnelles dans *Act*.

question: existe-t-il un automate $\mathcal{C}$ de niveau 1 vérifiant (obs), (unc) et tel que pour tous les mots de longueur ω^k acceptés par $\mathcal{S} \times_Y lift_k(\mathcal{C})$ vérifient ϕ avec Y satisfaisant (syn).

Ce problème est déjà une simplification puisque on pourrait seulement supposer que le contrôleur $\mathcal{C}$ se synchronise avec $\mathcal{S}$ aux positions $0 < \alpha_1 < \alpha_2 < \ldots$ du système avec $lim_{i\to\omega}\alpha_i = \omega^k$.

Théorème 2.5.8. [Cac06] Le problème de contrôle pour $\mathrm{LTL}(\omega^k)$ est décidable.

L'exemple du contrôle d'une balle qui rebondit est résolu dans [Cac06] (voir aussi une variante dans [DN07]).

Problème ouvert 9. Pour $k \geq 1$, quelle est la complexité du problème de contrôle pour $\mathrm{LTL}(\omega^k)$? ◯

Chapitre 3

Vérification de propriétés qualitatives et quantitatives

Ce chapitre est en grande partie une traduction de l'article [Dem06a] retraçant principalement des résultats présentés dans [DG05, DL06a, DLN07, Dem06b, DD07].

Comme nous l'avons rappelé dans le chapitre 2, la logique du temps linéaire LTL avec des opérateurs de passé est équivalente à la logique du premier ordre sur $\langle \mathbb{N}, < \rangle$ [Kam68] et les problèmes de model-checking et satisfaisabilité sont PSPACE-complets [SC85]. Une variable propositionnelle p représente une propriété de l'état courant du système. Par exemple, p peut être vraie lorsque la valeur de la variable $\mathtt{x}$ est supérieure à la valeur de la variable $\mathtt{y}$ après un pas de calcul. Une solution plus fine consiste donc à autoriser dans le langage logique la possibilité d'exprimer directement des contraintes entre variables du programme abandonnant donc l'abstraction usuelle avec les variables propositionnelles. Lorsque les variables sont typées, elles peuvent être interprétées dans des domaines concrets spécifiques comme les entiers, les réels, les chaînes de caractères etc. Ces domaines concrets sont appelés des systèmes de contraintes dans la suite. Ainsi, une proposition de la forme "$\mathtt{x}$ est supérieure à la prochaine valeur de $\mathtt{y}$" peut se coder dans une syntaxe enrichie par "$\mathtt{x} < \mathtt{X}\mathtt{y}$" où $\mathtt{X}\mathtt{y}$ fait référence à la prochaine valeur de $\mathtt{y}$ (souvent aussi notée $\mathtt{y}'$). Avec une telle extension les modèles linéaires sont des séquences d'états structurés qui contiennent une interprétation des variables. C'est pourquoi ma motivation initiale dans ce travail a été nourrie par le souhait de concevoir des logiques temporelles définies sur des systèmes de contraintes permettant de raffiner l'ensemble des formules atomiques et de comparer des valeurs de variables à des positions successives de l'exécution des programmes. Des exemples représentatifs du type de logiques temporelles que j'ai souhaité étudier sont les logiques introduites dans [AH94, CC00]. L'objectif dans ce travail consiste à développer des méthodes génériques, principalement à base d'automates, qui permettent de décider des problèmes de model-checking et de satisfaisabilité pour les logiques du temps linéaire dont les formules atomiques sont construites sur des systèmes de contraintes. Il s'agit en particulier de déterminer quels domaines concrets vérifient les meilleures propriétés pour la vérification au-

tomatique. On regarde ces deux types de problèmes car l'approche par automate ne fait pas beaucoup de différences entre eux.

3.1 LTL sur des systèmes de contraintes

3.1.1 Comment raffiner LTL

Soit $\mathrm{VAR} = \{\mathsf{x}_0, \mathsf{x}_1, \ldots\}$ un ensemble infini dénombrable de variables. Un système de contraintes est une paire $\mathcal{D} = \langle D, (R_\alpha)_{\alpha \in I} \rangle$ où D est un ensemble spécifique dans lequel les variables sont interprétées et $(R_\alpha)_{\alpha \in I}$ est une famille dénombrable de relations sur D. Il s'agit donc d'une structure relationnelle où l'ensemble I peut être compris comme un alphabet éventuellement structuré. Une contrainte atomique du système $\mathcal{D}$ est une expression de la forme $\mathsf{R}(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ où R est interprétée comme une relation de $\mathcal{D}$ et n est l'arité de cette relation. On supposera implicite l'arité des relations et la correspondance entre une relation et son symbole de relation associé. Une valuation de $\mathcal{D}$ est une application $v : \mathrm{VAR} \to D$. Ainsi, une contrainte est satisfaite par la valuation v, noté $v \models_{\mathcal{D}} \mathsf{R}(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ ssi $\langle v(\mathsf{x}_1), \ldots, v(\mathsf{x}_n) \rangle \in R$ où R est la relation dans $\mathcal{D}$ associée au symbole de relation R.

Le problème de consistance pour $\mathcal{D}$ consiste à vérifier s'il existe une valuation satisfaisant un ensemble fini de contraintes atomiques de $\mathcal{D}$. De même, le problème de consistance maximale pour $\mathcal{D}$ consiste à vérifier si un ensemble de contraintes est maximalement consistant par rapport à un ensemble de variables et à un ensemble de symboles de relation. Par exemple, le problème de consistance maximale pour le système $\langle \mathbb{R}, <, = \rangle$ est NLOGSPACE-complet car cela revient principalement à détecter des cycles dans un graphe fini.

Le problème de l'implication pour le système $\mathcal{D}$ consiste à tester si chaque valuation vérifiant un ensemble fini X de contraintes atomiques satisfait aussi une contrainte c (noté $X \models_{\mathcal{D}} c$). Par exemple, avec $\mathcal{D} = \langle \mathbb{R}, < \rangle$, nous avons $\{\mathsf{x} < \mathsf{y}, \mathsf{y} < \mathsf{z}\} \models_{\mathcal{D}} \mathsf{x} < \mathsf{z}$ car la relation "$<$" est transitive.

La logique $\mathrm{CLTL}(\mathcal{D})$ est définie comme une extension de LTL où les variables propositionnelles sont raffinées en contraintes atomiques de $\mathcal{D}$ construites sur des termes. Un terme est une variable x_j préfixée par un nombre fini i du symbole "next" $\mathtt{X}$, noté $\mathtt{X}^i\mathsf{x}_j$. Le codage de "$\mathtt{X}^i\mathsf{x}_j$" nécessite un espace mémoire de taille $\mathcal{O}(i + log\ j)$. Le terme $\mathtt{X}^i\mathsf{x}_j$ est naturellement interprété comme la valeur de la variable x_j au i^{eme} état suivant. Le symbole $\mathtt{X}$ est donc surchargé puisque $\mathtt{X}$ est aussi un opérateur temporel mais cela ne devrait pas entraîner de confusion dans la suite. Les formules de $\mathrm{CLTL}(\mathcal{D})$ sont définies par la grammaire suivante:

$$\phi ::= \mathsf{R}(\mathtt{X}^{l_1}\mathsf{x}_{j_1}, \ldots, \mathtt{X}^{l_n}\mathsf{x}_{j_n}) \mid \phi \wedge \phi \mid \neg\phi \mid \mathtt{X}\phi \mid \phi \mathtt{U} \phi.$$

Les opérateurs $\mathtt{X}$ et $\mathtt{U}$ sont interprétés comme dans LTL et ce langage sera dans la suite quelquefois enrichi d'opérateurs de passé ou d'opérateurs définissables dans la logique monadique du second ordre, voir par exemple [GK03]. Les formules de la forme $\mathsf{R}(\mathtt{X}^{l_1}\mathsf{x}_{j_1}, \ldots, \mathtt{X}^{l_n}\mathsf{x}_{j_n})$ sont appelées des contraintes atomiques

temporelles. Une formule $\mathsf{R}(\mathtt{X}^{l_1}\mathsf{x}_{j_1}, \ldots, \mathtt{X}^{l_n}\mathsf{x}_{j_n})$ est dite locale lorsque $l_1, \ldots, l_n \leq 1$. Etant donnée une formule ϕ de $\mathrm{CLTL}(\mathcal{D})$, la $\mathtt{X}$-hauteur de ϕ, notée $|\phi|_{\mathtt{X}}$ est le nombre maximal i pour lequel il existe un terme de la forme $\mathtt{X}^i\mathsf{x}$ apparaissant dans ϕ. Intuitivement, cette hauteur correspond à la largeur d'une fenêtre qui glisserait sur un modèle pour tenir compte des contraintes atomiques temporelles de ϕ.

Les modèles de $\mathrm{CLTL}(\mathcal{D})$ sont des ω-séquences de valuations de $\mathcal{D}$ de la forme $\sigma : \mathbb{N} \to (\mathrm{VAR} \to D)$ et la relation de satisfaction est définie comme pour LTL sauf pour les formules atomiques:

- $\sigma, i \models \mathsf{R}(\mathtt{X}^{l_1}\mathsf{x}_{j_1}, \ldots, \mathtt{X}^{l_n}\mathsf{x}_{j_n})$ ssi $\langle \sigma(i+l_1)(\mathsf{x}_{j_1}), \ldots, \sigma(i+l_n)(\mathsf{x}_{j_n}) \rangle \in R$,
- $\sigma, i \models \phi \wedge \phi'$ ssi $\sigma, i \models \phi$ et $\sigma, i \models \phi'$,
- $\sigma, i \models \neg\phi$ ssi $\sigma, i \not\models \phi$,
- $\sigma, i \models \mathtt{X}\phi$ ssi $\sigma, i+1 \models \phi$,
- $\sigma, i \models \phi \mathtt{U} \phi'$ ssi il existe $j \geq i$ tel que $\sigma, j \models \phi'$ et pour $i \leq l < j$, nous avons $\sigma, l \models \phi$.

Comme d'habitude, une formule $\phi \in \mathrm{CLTL}(\mathcal{D})$ est dite satisfaisable quand il existe un modèle σ tel que $\sigma, 0 \models \phi$. Le problème de satisfaisabilité sera noté SAT($\mathrm{CLTL}(\mathcal{D})$).

On note $\mathrm{CLTL}^l_k(\mathcal{D})$ la restriction de $\mathrm{CLTL}(\mathcal{D})$ aux formules avec au plus k variables et dont la $\mathtt{X}$-hauteur est bornée par l. La logique LTL [SC85] peut être vue comme la logique $\mathrm{CLTL}(\{\top, \bot\}, True)$ où $True = \{\top\}$.

Lemme 3.1.1. Soit $\mathcal{D}$ un système de contraintes avec égalité. Il existe un réduction logarithmique en espace de SAT($\mathrm{CLTL}(\mathcal{D})$) vers SAT($\mathrm{CLTL}^1_\omega(\mathcal{D})$).

La preuve du lemma 3.1.1 renomme les termes et requiert un nombre infini dénombrable de variables de $\mathrm{CLTL}^1_\omega(\mathcal{D})$.

Le problème de model-checking pour $\mathrm{CLTL}(\mathcal{D})$ consiste à vérifier des propriétés exprimées dans $\mathrm{CLTL}(\mathcal{D})$ sur une classe d'automates à contraintes [Rev02]. Un $\mathcal{D}$-automate $\mathcal{A}$ à k variables est une structure $\langle Q, \delta, I, F \rangle$ telle que

- Q est un ensemble fini d'états,
- $I \subseteq Q$ est l'ensemble des états initiaux,
- $F \subseteq Q$ est l'ensemble des états finaux,
- δ est la relation de transition, sous-ensemble de $Q \times \Sigma \times Q$ où Σ est un sous-ensemble fini de $1SC_k$ qui dénote l'ensemble des combinaisons Booléennes des contraintes locales construites sur l'ensemble des variables $\{\mathsf{x}_1, \ldots, \mathsf{x}_k\}$.

On utilise la notation $q \xrightarrow{c} q'$ comme une abréviation pour $\langle q, c, q' \rangle \in \delta$. Lorsque le système de contraintes est un fragment de l'arithmétique de Presburger, les $\mathcal{D}$-automates constituent une forme particulière d'automates à compteurs. La figure 3.1 contient une représentation graphique d'un $\langle \mathbb{N}, +1, = 2 \rangle$-automate à 1 variable où "$= 2$" est un prédicat unaire interprété par $\{2\}$, ce qui correspond à un test à la valeur deux.

Le langage accepté par un $\mathcal{D}$-automate est noté $\mathrm{L}(\mathcal{A})$. Un modèle de $\mathrm{CLTL}(\mathcal{D})$ σ est une réalisation d'une ω-séquence de formules $\phi_0\phi_1 \ldots$ ssi pour $i \geq 0$, nous avons $\sigma, i \models \phi_i$. On note aussi $\mathrm{L}^{\mathcal{D}}(\mathcal{A})$ la classe des modèles de $\mathrm{CLTL}(\mathcal{D})$ qui sont la réalisation d'un mot de $\mathrm{L}(\mathcal{A})$.

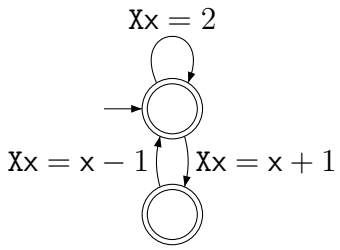


FIG. 3.1 – *Un* $\langle \mathbb{N}, +1, =2 \rangle$*-automate* $\mathcal{A}_0$ *à 1 variable*

Le problème de model-checking pour $\mathrm{CLTL}(\mathcal{D})$, noté MC($\mathrm{CLTL}(\mathcal{D})$) est défini ainsi:

entrée: un $\mathcal{D}$-automate $\mathcal{A}$ et une formule ϕ de $\mathrm{CLTL}(\mathcal{D})$,

question: est-ce qu'il existe un modèle de $\mathrm{L}^{\mathcal{D}}(\mathcal{A})$, tel que $\sigma, 0 \models \phi$ (noté $\mathcal{A} \models_{\exists} \phi$).

Cela revient en effet à vérifier si un calcul de $\mathcal{A}$ satisfait la formule ϕ. Par exemple, $\mathcal{A}_0 \models_{\exists} (\mathtt{x} = 0) \wedge \mathtt{GF}(\mathtt{x} = 0)$ avec $\mathcal{A}_0$ défini dans la figure 3.1. En effet, $((\mathtt{Xx} = \mathtt{x}+1) \cdot (\mathtt{Xx} = \mathtt{x}-1))^{\omega} \in \mathrm{L}(\mathcal{A}_0)$ et $(0 \cdot 1)^{\omega}$ est une réalisation de cette ω-séquence.

Une version universelle de ce problème peut être définie de façon analogue (on utilise alors la notation $\mathcal{A} \models_{\forall} \phi$). Pour la restriction du problème de model-checking à $\mathrm{CLTL}^{l}_{k}(\mathcal{D})$, on suppose que la formule ϕ appartient à $\mathrm{CLTL}^{l}_{k}(\mathcal{D})$ et $\mathcal{A}$ est un $\mathcal{D}$-automate à k variables.

Pour tous les systèmes à contraintes non triviaux considérés dans ce travail, le problème de satisfaisabilité pour $\mathrm{CLTL}(\mathcal{D})$ sera équivalent à réduction logarithmique en espace près à la version existentielle du problème de model-checking. De même, comme les $\mathcal{D}$-automates sont des modèles opérationnels qui autorisent plus de comportements que des machines déterministes, il existe un $\mathcal{D}$-automate $\mathcal{A}_{\top}$ tel que $\mathrm{L}^{\mathcal{D}}(\mathcal{A}_{\top})$ est exactement la classe de tous les modèles de $\mathrm{CLTL}(\mathcal{D})$. Par conséquent, il existe une réduction en espace logarithmique entre SAT($\mathrm{CLTL}(\mathcal{D})$) et MC($\mathrm{CLTL}(\mathcal{D})$).

Pour conclure cette section, voici quelques repères pour comparer les logiques $\mathrm{CLTL}(\mathcal{D})$ avec des formalismes existants. L'analyse de problèmes d'accessibilité dans les automates à compteurs est incontournable dès que l'on s'intéresse à la vérification de systèmes avec un nombre infini de configurations, voir par exemple [CJ98, ISD+00, FL02, DPK03] et cela motive partiellement l'introduction des versions de LTL construites sur des systèmes de contraintes. Des logiques temporelles avec des contraintes de Presburger ont été étudiées dans [Čer94a, BEH95, CC00, BDR03]. Certaines admettent des fragments décidables vraiment expressifs. Cependant, toutes ne considèrent pas le même type de modèle qu'ici. Par exemple, dans [BEH95] les modèles sont ceux de LTL mais les contraintes de Presburger expriment des relations entre des nombres d'occurrence d'événements. Des logiques spatio-temporelles, voir par exemple [WZ00, GKWZ03, GKK+03], sont aussi des exemples de versions de LTL avec systèmes de contraintes pour lesquelles le système est une structure faisant référence à des relations spatiales. Pour les logiques terminologiques, l'introduction des domaines concrets remonte

à [BH91] et depuis, de nombreux formalismes conçus pour la représentation des connaissances ont été intensivement étudiés, voir par exemple [Lut03, Lut04]. Ainsi, la logique $\mathrm{CLTL}(\mathcal{D})$ peut aussi être comprise comme un cas particulier de logique terminologique construite sur le domaine concret $\mathcal{D}$ pour laquelle seuls les modèles isomorphes à $\langle \mathbb{N}, < \rangle$ sont retenus. Cependant, l'introduction de LTL avec contraintes de Presburger est motivée par le besoin de vérifier des systèmes à compteurs tandis que pour les logiques terminologiques l'introduction des domaines concrets se justifie par le besoin d'exprimer des concepts faisant référence à des qualités concrètes.

3.1.2 Fragments de LTL avec contraintes de Presburger

L'arithmétique de Presburger (PA) est la théorie du premier ordre de la structure $\langle \mathbb{Z}, + \rangle$ [Pre29]. Cette théorie est décidable en temps triplement exponentiel [FR74] et de nombreux fragments admettent une complexité beaucoup plus modeste (voir des exemples plus loin). Parmi les propriétés remarquables de (PA), on peut noter ici que (PA) définit exactement les ensembles semilinéaires [GS66].

Etant donnés une formule de Presburger $A(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ contenant au plus les variables libres de $\vec{\mathsf{x}} = \langle \mathsf{x}_1, \ldots, \mathsf{x}_n \rangle$, et un tuple $\vec{a} = \langle a_1, \ldots, a_n \rangle \in \mathbb{Z}^n$, la satisfaction de $A(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ par rapport à l'interprétation $\vec{a}$ est notée $\vec{a} \models A(\vec{\mathsf{x}})$. On note aussi $\mathrm{sol}(A(\vec{\mathsf{x}}))$ l'ensemble des éléments $\vec{a}$ de $\mathbb{Z}^n$ vérifiant la formule $A(\vec{\mathsf{x}})$. Dans la suite, tout fragment FPA de (PA) compris comme un sous-ensemble de (PA) définit implicitement un système de contraintes

$$\mathcal{D}_{\mathrm{FPA}} = \langle \mathbb{Z}, (\mathrm{sol}(A(\vec{\mathsf{x}})))_{A(\vec{\mathsf{x}}) \in \mathrm{FPA}} \rangle.$$

Ainsi, le système $\mathcal{D}_{\mathrm{FPA}}$ peut contenir un nombre infini de relations et FPA est compris comme un alphabet structuré. On écrira aussi $\mathrm{CLTL}(\mathrm{FPA})$ au lieu de $\mathrm{CLTL}(\mathcal{D}_{\mathrm{FPA}})$. Par exemple, les contraintes de la forme $\mathsf{x} \geq d$ avec $d \in \mathbb{Z}$ induisent le système de contraintes

$$\langle \mathbb{Z}, (\{n \in \mathbb{Z} : n \geq d\})_{d \in \mathbb{Z}} \rangle,$$

muni d'un ensemble infini dénombrable de relations.

Dans le reste de cette section, nous allons présenter quelques fragments de (PA) utiles pour la suite. Commençons par définir des langages du premier ordre avec contraintes de périodicité. Le langage de constraintes IPC est défini par la grammaire ci-dessous:

$$A ::= \mathsf{x} \equiv_k \mathsf{y} + c \mid \mathsf{x} \equiv_k c \mid A \wedge A \mid \neg A,$$

avec $k, c \in \mathbb{N}$ et $\mathsf{x}, \mathsf{y} \in \mathrm{VAR}$. Soit X un sous-ensemble de $\{\exists, [], <, \mathtt{Eq}\}$, nous définissons une extension IPC^X de IPC en ajoutant les clauses suivantes à la définition de IPC:

- si $\exists \in X$, alors la clause $\exists\, \mathsf{x}\, A$ est ajoutée (quantification existentielle),
- si $[] \in X$, alors la clause $\mathsf{x} \equiv_k \mathsf{y} + [c_1, c_2]$ avec $c_1, c_2 \in \mathbb{N}$ est ajoutée,
- si $\mathtt{Eq} \in X$, alors la clause $\mathsf{x} = \mathsf{y}$ avec $\mathsf{x}, \mathsf{y} \in \mathrm{VAR}$ est ajoutée,

– si $<\in X$, alors les clauses $\mathsf{x} < d \mid \mathsf{x} > d \mid \mathsf{x} = d$ avec $\mathsf{x} \in \mathrm{VAR}$ et $d \in \mathbb{Z}$ sont ajoutées.

La présence de "Eq" dans X permet d'exprimer des contraintes d'équalité entre variables alors que la présence de "$<$" autorise des contraintes entre une variable et une constante. A titre d'exemple, nous donnons la sémantique des formules de la forme $\mathsf{x} \equiv_k \mathsf{y} + [c_1, c_2]$ (les autres cas utilisent l'interprétation usuelle): $v \models \mathsf{x} \equiv_k \mathsf{y} + [c_1, c_2] \overset{\text{def}}{\Leftrightarrow}$ il existe $c \in [c_1, c_2]$ et $l \in \mathbb{Z}$ tel que $(v(\mathsf{x}) - v(\mathsf{y})) = l \times k + c$. Ainsi $\mathsf{x} \equiv_k \mathsf{y} + [c_1, c_2]$ est juste plus concis que $\bigvee_{c \in [c_1, c_2]} \mathsf{x} \equiv_k \mathsf{y} + c$ lorsque les constantes sont codées avec une représentation binaire.

Dans la suite, on note IPC^+ [resp. IPC^{++}] le langage $\mathrm{IPC}^{\{\exists,[],<\}}$ [resp. le langage $\mathrm{IPC}^{\{\exists,[],<,\mathtt{Eq}\}}$]. Il est bon de rappeler ici que IPC^{++} est une extension du langage de contraintes de périodicité introduit dans [TC98] mais avec la présence de la négation comme dans [BBFS96]. Ce qui est appelé "IPC" dans [TC98] est précisément défini par

$$A ::= \mathsf{x} \equiv_k \mathsf{y} + c \mid \mathsf{x} \equiv_k c \mid A \wedge A \mid \exists \mathsf{x}\, A.$$

Comme (PA), le langage IPC^{++} satisfait la propriété d'élimination des quantificateurs mais la complexité du problème de consistance est bien moindre.

Théorème 3.1.2. [Dem06b]

(I) Le problème de consistance pour IPC^{++} est PSPACE-complet.

(II) Etant donnée une constrainte A de IPC^{++}, on peut calculer une formule équivalente A' sans quantificateur en espace polynomial en $|A|$ et $|A'|$ est en $\mathcal{O}(2^{|A|})$.

Soit DL le fragment de (PA) constitué de contraintes de différence:

$$A ::= \mathsf{x} \sim \mathsf{y} + d \mid \mathsf{x} \sim d \mid A \wedge A \mid \neg A$$

où $\mathsf{x}, \mathsf{y} \in \mathrm{VAR}$, $\sim \in \{<, =\}$ et $d \in \mathbb{Z}$. Nous utilisons les notations $\mathsf{x} \leq \mathsf{y}$, $\mathsf{x} \geq \mathsf{y}$ et $\mathsf{x} > \mathsf{y}$ comme des abréviations pour respectivement $\mathsf{x} < \mathsf{y} \vee \mathsf{x} = \mathsf{y}$, $\neg(\mathsf{x} < \mathsf{y})$ et $\neg(\mathsf{x} \leq \mathsf{y})$. Soit $v : \mathrm{VAR} \to \mathbb{Z}$ une valuation, la relation de satisfaction $v \models A$ est définie de la façon attendue. Il est clair que DL est un fragment propre de (PA) sans quantification. En effet, des contraintes de périodicité de la forme $\mathsf{x} \equiv_k c$ ou des comparaisons de la forme $\mathsf{x} + \mathsf{y} + \mathsf{z} < 5$ ne font pas partie de DL.

Le dernier fragment de (PA) avec lequel nous allons travailler est (PA) sans quantification, noté QFP. Ce fragment principalement utilisé dans la section 3.4.6 est défini par la grammaire suivante:

$$A ::= \sum_{i \in J} a_i \mathsf{x}_i = d \mid \sum_{i \in J} a_i \mathsf{x}_i < d \mid \sum_{i \in J} a_i \mathsf{x}_i \equiv_k c \mid \neg A \mid A \wedge A$$

avec $a_i, d \in \mathbb{Z}$, $k, c \in \mathbb{N}$ et J est un ensemble fini d'indices. Le langage QFP est aussi expressif que (PA) mais moins concis (le problème de consistance est dans NP, une conséquence de [Pap81]).

3.2 Contraintes quantitatives et indécidabilité

Dans cette section, nous présentons des versions de LTL construites principalement sur des fragments de (PA) qui sont indécidables même si des restrictions syntaxiques drastiques seront quelquefois considérées.

3.2.1 Machines de Minsky

L'indécidabilité de LTL avec contraintes de Presburger peut être principalement établie en réduisant le problème de l'arrêt ou le problème de récurrence pour les machines de Minsky non déterministes. C'est pourquoi, nous nous permettons ici de rappeler brièvement ces problèmes. Une machine de Minsky non déterministe M se compose de deux compteurs C_1 et C_2, et d'une séquence de $n \geq 1$ instructions. Chacune des instructions peut incrémenter, décrémenter un des compteurs ou bien atteindre une instruction si un des compteurs est à zéro. Lorsqu'une incrémentation ou une décrementation a eu lieu, la machine continue en choisissant de façon non déterministe parmi deux instructions, ce choix n'étant possible que pour les machines non déterministes. L'instruction l a une des formes suivantes:

l: $\mathsf{C}_i := \mathsf{C}_i + 1$; aller à l' ou aller à l''

l: si $\mathsf{C}_i = 0$ alors aller à l' sinon $\mathsf{C}_i := \mathsf{C}_i - 1$; aller à l' ou aller à l''

Les configurations de M sont des triplets $\langle l, c_1, c_2\rangle$ où $l \in \{1, \ldots, n\}$ et $c_1, c_2 \geq 0$. Chaque valeur c_i correspond à la valeur courante du compteur C_i. Un calcul de M est une ω-séquence de configurations respectant les instructions de M et dont la configuration initiale est $\langle 1, 0, 0\rangle$. Un calcul est récurrent s'il contient un nombre infini de configurations avec pour instruction courante 1. Le problème de récurrence consiste à déterminer si une machine de Minsky non déterministe admet un calcul récurrent. Ce problème est hautement indécidable, Σ_1^1-difficile d'après [AH94, section 4.1]. De même, le problème qui consiste à déterminer si à partir de la configuration initiale $\langle 1, 0, 0\rangle$ une configuration avec instruction courante 1 est atteignable en au moins un pas de calcul, est indécidable [Min67] – problème de l'arrêt.

3.2.2 Systèmes avec un mécanisme de comptage

La Σ_1^1-dureté du problème de satisfaisabilité pour $\mathrm{CLTL}_\omega^1(\mathbb{N}, =, +1)$ peut être facilement montrée en réduisant le problème de récurrence pour les machines de Minsky non déterministes. Plus généralement, il est possible de définir trois propriétés d'un système de contraintes pour qu'il admette une mécanisme de comptage qui induise l'indécidabilité comme dans $\mathrm{CLTL}_\omega^1(\mathbb{N}, =, +1)$.

Définition 3.2.1. Un système de contraintes $\mathcal{D}$ admet un mécanisme de comptage implicite si les conditions suivantes sont vérifiées:

1. $\mathcal{D}$ contient l'égalité,

2. $\mathcal{D}$ possède une relation binaire R telle que
   (a) $R = \{\langle a, b\rangle \in D^2 : f(a) = b\}$ pour une injection $f : D \to D$,
   (b) $\langle D, R\rangle$ est un DAG.

∇

Ainsi, lorsque $\mathcal{D}$ a un mécanisme de comptage implicite pour chaque $a \in D$,

$$a \xrightarrow{R} f^1(a) \xrightarrow{R} \dots f^i(a) \xrightarrow{R} \dots$$

est isomorphe à $\langle \mathbb{N}, <\rangle$. Par exemple, pour $D \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$ et pour $i \in D \setminus \{0\}$, le système $\langle D, =, =_{+i}\rangle$ a un mécanisme de comptage implicite où $n =_{+i} n'$ ssi $n = n' + i$. De même, le système $\langle D \setminus \{0\}, =, =_{\times i}\rangle$ a un mécanisme de comptage implicite où $n =_{\times i} n'$ ssi $n = n' \times i$ avec $i \neq -1$.

Théorème 3.2.1. [DD07] Le problème de satisfaisabilité pour $\mathrm{CLTL}^1_\omega(\mathcal{D})$ est Σ^1_1-difficile pour chaque système $\mathcal{D}$ admettant un mécanisme de comptage implicite.

La preuve est par réduction du problème de récurrence pour les machines de Minsky non déterministes (voir la section 3.2.1) sur le modèle de la preuve de [CC00, théorème 3]. En effet, les calculs de telles machines peuvent être codés comme les modèles d'une formule de $\mathrm{CLTL}^1_\omega(\mathcal{D})$ sur la base du passage du model-checking vers la satisfaisabilité pour LTL [SC85]. C'est par exemple facile d'exprimer qu'une variable contenant la valeur de l'instruction courante peut prendre la valeur 1 infiniment souvent. C'est ainsi que $\mathrm{CLTL}^1_3(\mathrm{DL})$ est montrée indécidable dans [CC00]. Par exemple, l'instruction

– l: $\mathsf{C}_1 := \mathsf{C}_1 + 1$; aller à l' ou aller à l''

peut se coder par une formule temporelle de la forme

$$\mathtt{G}(\mathsf{x}_l = \mathsf{y}_l \Rightarrow (\mathsf{R}(\mathsf{z}_1, \mathtt{X}\mathsf{z}_1) \wedge \mathtt{X}(\mathsf{x}_{l'} = \mathsf{y}_{l'} \vee \mathsf{x}_{l''} = \mathsf{y}_{l''}))),$$

où z_1 est une variable attachée au compteur C_1 et R est le symbole de prédicat associé à la relation R de la définition 3.2.1.

Nous rappelons au passage que la Σ^1_1-dureté implique que la logique n'est pas récursivement axiomatisable. Nous obtenons les corollaires suivants.

Corollaire 3.2.2. Soit $D \in \{\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{Q}_+\mathbb{R}, \mathbb{R}_+\}$.

(I) Pour $i \in D \setminus \{0\}$, SAT($\mathrm{CLTL}^1_\omega(D, =, =_{+i})$) et MC($\mathrm{CLTL}^1_\omega(D, =, =_{+i})$) sont Σ^1_1-difficiles.

(II) Pour $i \in D \setminus \{0, 1, -1\}$, SAT($\mathrm{CLTL}^1_\omega(D, =, =_{\times i})$) et MC($\mathrm{CLTL}^1_\omega(D, =, =_{\times i}$)) sont Σ^1_1-difficiles.

3.2.3 Restreindre les ressources syntaxiques

Dans la section 3.2.2, nous avons vu que le problème de satisfaisabilité pour $\mathrm{CLTL}^1_\omega(\mathbb{N}, =, +1)$ est hautement indécidable. La façon la plus naturelle de coder les calculs des machines de Minsky consiste à faire usage d'un nombre non borné de variables. Dans cette section, nous allons aller un peu plus loin en montrant certaines restrictions syntaxiques préservent l'indécidabilité. Par exemple, le lemme 3.2.3 énonce comment réduire la X-hauteur des formules et le lemme 3.2.4 comment réduire le nombre de variables.

Lemme 3.2.3. [DG06] Pour $k, l, k', l' \in \mathbb{N} \setminus \{0\}$ et pour tous les systèmes $\mathcal{D}$, il existe une réduction de SAT($\mathrm{CLTL}^l_k(\mathcal{D})$) vers SAT($\mathrm{CLTL}^{l'}_{k'}(\mathcal{D})$) (exponentielle en temps) avec $k \times l = k' \times l'$ et $k' = k \times m$ pour un $m \geq 2$.

L'idée de la preuve consiste à représenter m états consécutifs d'un modèle de $\mathrm{CLTL}^l_k(\mathcal{D})$ avec $k' = km$ en un seul état d'un modèle de $\mathrm{CLTL}^{l'}_{k'}(\mathcal{D})$. Par exemple, si $k' = 2k$, alors le modèle de $\mathrm{CLTL}^l_k(\mathcal{D})$ ci-dessous

$$\begin{pmatrix} 1 \\ 2 \\ \dots \\ k \end{pmatrix} \begin{pmatrix} k+1 \\ k+2 \\ \dots \\ 2k \end{pmatrix} \begin{pmatrix} 2k+1 \\ 2k+2 \\ \dots \\ 3k \end{pmatrix} \dots$$

se code par le modèle de $\mathrm{CLTL}^{l'}_{k'}(\mathcal{D})$ ci-dessous

$$\begin{pmatrix} 1 \\ 2 \\ \dots \\ 2k \end{pmatrix} \begin{pmatrix} 2k+1 \\ 2k+2 \\ \dots \\ 4k \end{pmatrix} \dots$$

Le lemme 3.2.4 indique comment réduire le nombre de variables dans les formules.

Lemme 3.2.4. [DG06] Pour $k, l, k', l' \in \mathbb{N} \setminus \{0\}$ et pour les systèmes $\mathcal{D}$ avec égalité et au moins trois éléments, il existe une réduction logaritmique en espace de SAT($\mathrm{CLTL}^l_k(\mathcal{D})$) vers SAT($\mathrm{CLTL}^{l'}_{k'}(\mathcal{D})$) où $3k \times l \leq k' \times l'$ et $k = k' \times m$ pour un $m \geq 2$.

L'idée de la preuve consiste à coder un état d'un modèle de $\mathrm{CLTL}^l_k(\mathcal{D})$ vers $3m$ états d'un modèle de $\mathrm{CLTL}^{l'}_{k'}(\mathcal{D})$. Seul un état sur trois contient une information pertinente sur les valeurs des variables. Les autres états intermédiaires sont utilisés pour déterminer quand une séquence de $3m$ états d'un modèle de $\mathrm{CLTL}^{l'}_{k'}(\mathcal{D})$ correspond à un état d'un modèle de $\mathrm{CLTL}^l_k(\mathcal{D})$. Par exemple, si $k' = 1$ et $k = 2$, alors le modèle de $\mathrm{CLTL}^l_k(\mathcal{D})$ ci-dessous

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} \begin{pmatrix} 3 \\ 4 \end{pmatrix} \dots$$

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

$$\overbrace{1 = b \neq b \neq 2 \neq b \neq b \neq}^{\text{position 0}}\overbrace{3 = b \neq b \neq 4 \neq b \neq b \ldots}^{\text{position 1}}$$

où b dénote une valeur arbitraire satisfaisant les bonnes relations avec ses voisins (chaque occurrence de b peut correspondre à une valeur différente). Afin de garantir que de telles valeurs existent, nous avons supposé que le domaine a au moins trois éléments disincts. Le début du codage d'un état d'un modèle de $\mathrm{CLTL}^{l}_{k}(\mathcal{D})$ se caractérise par la présence de deux valeurs consécutives de la variable identiques.

Un fragment plat de $\mathrm{CLTL}^{1}_{\omega}(\mathrm{DL})$ est montré décidable dans [CC00] avec une procédure de décision qui a au moins la complexité de (PA). Ce fragment a été défini en restreignant l'usage de l'opérateur `U`: en particulier les arguments gauches ne peuvent contenir d'opérateurs temporels. L'indécidabilité de la logique $\mathcal{L}_p$ de [CC00] est montrée en réduisant le problème de l'arrêt des machines de Minsky. De même, l'indécidabilité de $\mathrm{CLTL}^{\omega}_{2}(\mathrm{DL})$ est montrée dans [DD07] avec des contraintes de la forme $\mathsf{x} = \mathsf{y}$, $\mathsf{x} = \mathsf{y} + 1$. Il est possible de raffiner davantage ces résultats.

Théorème 3.2.5. [DG06] Le problème de la satisfaisabilité pour $\mathrm{CLTL}^{2}_{1}(\mathrm{DL})$ est Σ^1_1-complet.

Une conséquence du théorème 3.2.5 et du lemme 3.2.3 est que nous pouvons améliorer la Σ^1_1-dureté de $\mathrm{CLTL}^{1}_{3}(\mathrm{DL})$ établie dans [CC00].

Corollaire 3.2.6. Le problème de satisfaisabilité pour $\mathrm{CLTL}^{1}_{2}(\mathrm{DL})$ est Σ^1_1-complet.

Par conséquent la logique $\mathcal{L}_p$ de [CC00] restreinte à deux variables est aussi hautement indécidable. De plus, le problème de satisfaisabilité peut être facilement réduit au model-checking car $\phi \in \mathrm{CLTL}(\mathrm{DL})$ est satisfaisable ssi $\mathcal{A}_\top \models \phi$ où $\mathcal{A}_\top$ est l'automate à un état $\langle \{q\}, \delta, \{q\}, \{q\} \rangle$ avec pour unique transition $q \xrightarrow{\top} q$. Nous obtenons ainsi le corollaire suivant:

Corollaire 3.2.7. [DG06] MC($\mathrm{CLTL}^{1}_{2}(\mathrm{DL})$) et MC($\mathrm{CLTL}^{2}_{1}(\mathrm{DL})$) sont Σ^1_1-complets.

3.2.4 Satisfaisabilité pour CLTL^{1}_{1}(QFP)

L'extension de $\mathrm{CLTL}(\mathrm{DL})$ avec des contraintes de la forme $a\mathsf{x} + b\mathsf{y} = 0$ avec $a, b \in \mathbb{Z}$ conduit à une logique indécidable même si on se restreint à une seule variable et à une `X`-hauteur 1.

Théorème 3.2.8. [DG06] Le problème de satisfaisabilité pour $\mathrm{CLTL}^{1}_{1}(\mathrm{QFP})$ est indécidable.

En effet, les valeurs des deux compteurs $\langle c_1, c_2 \rangle$ dans la configuration d'une machine de Minsky peuvent être codées par la valeur $2^{c_1}3^{c_2}$ d'une variable. Les tests à zéro, incrémentations et décrémentations se codent respectivement avec les contraintes $\mathsf{x} \equiv_2 0$, $\mathsf{x} \equiv_3 0$, $\mathtt{X}\mathsf{x} = 2\mathsf{x}$ (incrémentation du premier compteur) etc. La valeur de l'instruction courante l peut par exemple se coder en répétant l fois la configuration. Par conséquent, le problème du model-checking est indécidable même pour le fragment avec une unique variable et restreint aux formules de X-hauteur au plus 1. Il s'agit d'une conséquence d'un résultat de [Min67, section 14.2] pour les machines à un compteur avec division et multiplication par des constantes.

3.3 Modèles symboliques

Dans cette section, nous allons expliquer les grandes lignes de l'approche qui nous a permis d'établir la décidabilité du model-checking et de la satisfaisabilité pour $\mathrm{CLTL}(\mathcal{D})$ avec divers systèmes $\mathcal{D}$. Par défaut, $\mathcal{D}$ est un système de contraintes $\mathcal{D} = \langle D, (R_\alpha)_{\alpha \in I} \rangle$ avec I éventuellement infini.

3.3.1 ω-régularité et abstraction

Commençons par une analogie. Etant donnée une formule ϕ de LTL construite sur les variables propositionnelles de $\{\mathrm{p}_1, \ldots, \mathrm{p}_s\}$, les modèles de ϕ sont des ω-séquences $\sigma : \mathbb{N} \to \Sigma$ avec $\Sigma = \mathcal{P}(\{\mathrm{p}_1, \ldots, \mathrm{p}_s\})$. Pour être plus précis, on peut se restreindre à ces modèles pour déterminer la satisfaisabilité de ϕ. L'approche par automates introduite dans [VW94] démontre qu'il est possible de construire efficacement un automate de Büchi $\mathcal{A}_\phi$ tel que $\mathrm{L}(\mathcal{A}_\phi)$ est exactement l'ensemble des modèles de ϕ (voir la proposition 2.1.1). Cette méthode permet d'ailleurs de résoudre les problèmes de satisfaisabilité et model-checking pour LTL en espace mémoire polynomial. En effet, le problème de vacuité pour les automates de Büchi est NLOGSPACE-complet.

Etant donnée une formule ϕ de $\mathrm{CLTL}(\mathcal{D})$ construite sur les variables $\mathsf{x}_1, \ldots, \mathsf{x}_k$ les modèles de ϕ peuvent être vus comme des séquences de la forme $\mathbb{N} \to D^k$. L'ensemble D^k n'est pas nécessairement fini et donc les modèles de $\mathrm{CLTL}(\mathcal{D})$ ne sont pas des ω-séquences sur un alphabet fini. Dans le but de réutiliser des résultats sur les automates de Büchi une stratégie consiste à abstraire les modèles de ϕ comme des séquences de la forme $\mathbb{N} \to \{0,1\}^m$ où $m \geq 1$ est un entier à définir qui dépend de ϕ et du système $\mathcal{D}$. Chaque élément de $\{0,1\}^m$ est compris comme un ensemble fini de propriétés locales (par exemple "la prochaine valeur de y est égale à la valeur courante de x"). Etant donnés une formule ϕ et un modèle $\sigma : \mathbb{N} \to D^k$, on peut définir un modèle symbolique $abs(\phi, \sigma) : \mathbb{N} \to \{0,1\}^m$. Afin de développer une méthode pour $\mathrm{CLTL}(\mathcal{D})$ similaire à celle pour LTL, le mieux que nous puissions espérer est que l'ensemble des modèles symboliques dérivés d'une formule ϕ de $\mathrm{CLTL}(\mathcal{D})$ soit ω-régulier et qu'un automate de Büchi pour ce langage puisse être construit efficacement.

Le reste de cette section présente les principales étapes qui permettent de passer des modèles concrets (séquences de valuations) aux modèles symboliques.

3.3.2 Mesure syntaxique

Pour vérifier si une formule ϕ de $\mathrm{CLTL}(\mathcal{D})$ est satisfaisable, nous avons besoin de déterminer quelles sont les ressources syntaxiques de ϕ pertinentes. Par exemple, pour une formule de LTL il s'agit simplement des variables propositionnelles apparaissant dans la formule. Comme nous avons affaire à des systèmes de contraintes, nous devons étendre cette notion.

Définition 3.3.1. Une mesure syntaxique μ est un triplet $\langle k, l, X\rangle$ tel que $k \in \mathbb{N} \setminus \{0\}$ (le nombre de variables de VAR), $l \in \mathbb{N}$ (la $\mathtt{X}$-hauteur) et X est un ensemble fini de symboles de relation de $\mathcal{D}$. ∇

Par exemple, pour une formule ϕ de la logique $\mathrm{CLTL}(\mathbb{N}, <, =)$, une mesure $\langle k, l, X\rangle$ pour ϕ peut être définie telle que k est le nombre de variables distinctes dans ϕ (sans perte de généralité on supposera qu'il s'agit de $\mathsf{x}_1, \ldots, \mathsf{x}_k$), $l = |\phi|_{\mathtt{X}}$, et $X = \{<, =\}$. Plus généralement, lorsque I est fini, on supposera que les symboles de relation dans une mesure sont exactement ceux issus de $\mathcal{D}$.

3.3.3 Etat symbolique

L'ensemble des formules atomiques temporelles de $\mathrm{CLTL}(\mathcal{D})$ définies par une mesure $\mu = \langle k, l, X\rangle$, noté CONS^{μ}, est défini comme l'ensemble ci-dessous

$$\{\mathsf{R}(t_1, \ldots, t_n) : \mathsf{R} \in X\}$$

où chaque terme t_i est de la forme $\mathtt{X}^{l'}\mathsf{x}_{k'}$ avec $0 \leq l' \leq l$ et $1 \leq k' \leq k$. Le cardinal de CONS^{μ} est au plus exponentiel en $k + l + |X|$. Si l'arité des relations de $\mathcal{D}$ est bornée comme dans $\langle \mathbb{R}, <, =\rangle$, alors le cardinal de CONS^{μ} est polynomial en $k + l + |X|$. On note FOR^{μ} l'ensemble des formules de $\mathrm{CLTL}(\mathcal{D})$ construites sur les formules atomiques temporelles de CONS^{μ}. Nous définissons un état symbolique comme un ensemble fini de propriétés locales satisfaites à une position donnée d'un modèle. Il s'agit d'une abstraction d'une valuation qui peut a priori avoir une portée avec des valeurs non bornées.

Définition 3.3.2. Pour une mesure μ, un état symbolique est un sous-ensemble de CONS^{μ}. ∇

Remarquons qu'avec la définition ci-dessus, un état symbolique n'est pas nécessairement [resp. maximalement] consistant. L'ensemble des états symboliques est noté SYMB^{μ}. Un modèle symbolique ρ pour une mesure μ est une séquence $\rho : \mathbb{N} \to \mathrm{SYMB}^{\mu}$. C'est bien de la forme $\mathbb{N} \to \{0, 1\}^m$ pour un m donné, voir la section 3.3.1. Nous sommes à présent en position de définir la relation de satisfaction symbolique $\models_{\mu}$ (pour une mesure μ donnée). Les formules sont dans FOR^{μ} et les modèles symboliques sont des ω-séquences sur l'alphabet SYMB^{μ}. Les

opérateurs Booléens et temporels sont définis de façon homomorphique comme pour la relation classique $\models$. Seul le cas atomique nécessite un traitement particulier qui est en fait similaire à ce qui se passe pour LTL: pour toute formule $\phi \in \mathrm{CONS}^\mu$, $\rho, i \models_\mu \phi \stackrel{\text{def}}{\Leftrightarrow} \phi \in \rho(i)$.

3.3.4 Abstraction

Pour un modèle σ de $\mathrm{CLTL}(\mathcal{D})$ et une mesure μ, on note ρ^μ_σ le modèle symbolique tel que pour $i \geq 0$,

$$\rho^\mu_\sigma(i) \stackrel{\text{def}}{=} \{\phi \in \mathrm{CONS}^\mu : \sigma, i \models \phi\}.$$

Lemme 3.3.1. Soient μ une mesure, σ un modèle, ϕ une formule de FOR^μ et $i \in \mathbb{N}$. Si $\sigma, i \models \phi$, alors $\rho^\mu_\sigma, i \models_\mu \phi$.

Définition 3.3.3. Une abstraction pour $\mathrm{CLTL}(\mathcal{D})$ est une fonction calculable f de l'ensemble des formules vers l'ensemble des mesures. ∇

Une abstraction est complète lorsque pour chaque formule ϕ, pour tous les modèles σ et $i \in \mathbb{N}$, on a $\sigma, i \models \phi$ ssi $\rho^{f(\phi)}_\sigma, i \models_{f(\phi)} \phi$.

Théorème 3.3.2. Soit f une abstraction complète. Alors ϕ est $\mathrm{CLTL}(\mathcal{D})$ satisfaisable ssi il existe un modèle symbolique ρ tel que
(I) $\rho, 0 \models_{f(\phi)} \phi$,
(II) il existe un modèle σ de $\mathrm{CLTL}(\mathcal{D})$ tel que $\rho = \rho^{f(\phi)}_\sigma$.

Dans la preuve, si $\sigma, 0 \models \phi$, alors par le lemme 3.3.1, $\rho^{f(\phi)}_\sigma, 0 \models_{f(\phi)} \phi$. Evidemment, $\rho^{f(\phi)}_\sigma$ a un modèle concret. Réciproquement, si ρ satisfait les conditions (I) et (II) alors comme l'abstraction est complète, nous avons $\sigma, 0 \models \phi$.

Pour montrer que le problème de satisfaisabilité pour $\mathrm{CLTL}(\mathcal{D})$ est dans PSPACE, on pourra montrer que

- $\mathrm{CLTL}(\mathcal{D})$ possède une abstraction complète dont les éléments consistants de $\mathrm{SYMB}^{f(\phi)}$ peuvent être codés en espace polynomial et vérifier si $X \subseteq \mathrm{CONS}^{f(\phi)}$ est consistant peut se faire en espace polynomial,
- les modèles symboliques satisfaisant la condition (I) du théorème 3.3.2 peuvent être reconnus par un automate de Büchi calculable en espace polynomial,
- les modèles symboliques satisfaisant la condition (II) du théorème 3.3.2 peuvent être reconnus par un automate de Büchi calculable en espace polynomial.

Afin de garantir la borne PSPACE, les automates de Büchi peuvent être remplacés par n'importe quelle classe de modèles opérationnels dont le problème de vacuité soit dans NLOGSPACE, voir par exemple ce qui est fait dans [DG06] avec des automates à un compteur. Des exemples de borne PSPACE établie avec cette approche peuvent être trouvés dans [DD07, DG05, Gas05, Dem06b, DG06]. Cependant, alors que la classe des modèles symboliques vérifiant la condition (I) est ω-régulière, la classe des modèles symboliques verifiant la condition (II) peut ne pas l'être, voir par exemple [DD07].

3.3.5 Automates de Büchi pour la satisfaction symbolique

Nous définissons ci-dessous un automate de Büchi acceptant les modèles symboliques construits sur la mesure $f(\phi)$ qui vérifient symboliquement ϕ (condition (I) du théorème 3.3.2). Il s'agit d'étendre légèrement la traduction standard de LTL vers les automates de Büchi [VW94].

On note $cl(\phi)$ la fermeture de ϕ et un atome de ϕ est un sous-ensemble maximalement consistant de $cl(\phi)$. Comme d'habitude, $\psi_1 \mathtt{U} \psi_2 \in cl(\phi)$ implique $\mathtt{X}(\psi_1 \mathtt{U} \psi_2) \in cl(\phi)$. De plus, $\psi_1 \mathtt{U} \psi_2$ appartient à un atome X ssi $\phi_2 \in X$ ou bien ϕ_1, $\mathtt{X}(\psi_1 \mathtt{U} \psi_2) \in X$. Soit $\mathcal{A}_\phi^{f(\phi)}$ l'automate de Büchi généralisé $\langle Q, \delta, I, \mathcal{F} \rangle$ sur l'alphabet $\mathrm{SYMB}^{f(\phi)}$ tel que

- Q est l'ensemble des atomes de ϕ et $I = \{X \in Q : \phi \in X\}$,
- $X \xrightarrow{Z} Y$ ssi
  - pour toutes les formules atomiques temporelles A de X, $Z \models_{\mathcal{D}} A$ (une instance du problème d'implication),
  - pour $\mathtt{X}\psi \in cl(\phi)$, $\mathtt{X}\psi \in X$ ssi $\psi \in Y$,
- Soit $\{\psi_1 \mathtt{U} \phi_1, \ldots, \psi_n \mathtt{U} \phi_n\}$ l'ensemble des formules "Until" de $cl(\phi)$. On pose $\mathcal{F} = \{F_1, \ldots, F_n\}$ avec $F_i = \{X \in Q : \psi_i \mathtt{U} \phi_i \notin X \text{ ou } \phi_i \in X\}$ pour $i \in \{1, \ldots, n\}$.

On peut maintenant facilement établir le résultat suivant:

Lemme 3.3.3. $\mathrm{L}(\mathcal{A}_\phi^{f(\phi)}) = \{\rho : \mathbb{N} \to \mathrm{SYMB}^{f(\phi)} \mid \rho, 0 \models_{f(\phi)} \phi\}$.

3.4 Résultats de décidabilité

Dans cette section, différents fragments décidables de LTL avec contraintes de Presburger vont être principalement présentés.

3.4.1 Propriété de complétion

Un problème général concernant la classe de logiques $\mathrm{CLTL}(\mathcal{D})$ consiste à identifier les conditions suffisantes sur le système de contraintes $\mathcal{D}$ pour que les problèmes de model-checking et satisfaisabilité pour $\mathrm{CLTL}(\mathcal{D})$ admettent des procédures de décision efficaces, si possible en espace polynomial comme pour LTL. Très souvent cela revient à s'assurer que $\mathcal{D}$ admette une abstraction complète et si pour chaque mesure μ, l'ensemble $\{\rho_\sigma^\mu : \sigma \text{ est un modele de } \mathrm{CLTL}(\mathcal{D})\}$ des modèles symboliques (par rapport à la mesure μ) est ω-régulier ou non. Dans cette section, nous présentons une classe de systèmes de contraintes pour lesquels l'ensemble ci-dessus est ω-régulier et il existe un moyen simple de le définir avec un automate de Büchi.

Un modèle symbolique ρ par rapport à la mesure μ est consistant pas à pas ssi pour $i \geq 0$,

- $\rho(i)$ est maximalement consistant par rapport à μ,

– pour toute contrainte atomique temporelle $\mathsf{R}(\mathtt{X}^{l_1}\mathsf{x}_{j_1}, \ldots, \mathtt{X}^{l_n}\mathsf{x}_{j_n}) \in \mathrm{CONS}^\mu$ avec $l_1, \ldots, l_n \geq 1$,
$\mathsf{R}(\mathtt{X}^{l_1}\mathsf{x}_{j_1}, \ldots, \mathtt{X}^{l_n}\mathsf{x}_{j_n}) \in \rho(i)$ ssi $\mathsf{R}(\mathtt{X}^{l_1-1}\mathsf{x}_{j_1}, \ldots, \mathtt{X}^{l_n-1}\mathsf{x}_{j_n}) \in \rho(i+1)$.

Pour le système $\langle \mathbb{R}, <, = \rangle$ et la mesure $\langle 1, 2, \{<, =\} \rangle$, le modèle symbolique ci-dessous est consistant pas à pas:

$$\rho_0 = \{\mathsf{x} > \mathtt{X}\mathsf{x}, \mathtt{X}\mathsf{x} > \mathtt{XX}\mathsf{x}, \mathsf{x} > \mathtt{XX}\mathsf{x}, \mathsf{x} = \mathsf{x}, \mathtt{X}\mathsf{x} = \mathtt{X}\mathsf{x}, \mathtt{XX}\mathsf{x} = \mathtt{XX}\mathsf{x}\}^\omega.$$

La consistance pas à pas constitue une condition nécessaire pour qu'un modèle symbolique ρ ait un modèle σ de $\mathrm{CLTL}(\mathcal{D})$ tel que $\rho^\mu_\sigma = \rho$. Nous définissons ci-dessous une classe de systèmes de contraintes pour lesquels cette condition est suffisante.

Un système $\mathcal{D}$ a la propriété de complétion [DD07] ssi pour chaque mesure μ de la forme $\langle k, 0, X \rangle$, pour chaque ensemble maximalement consistant $Y \subseteq \mathrm{CONS}^\mu$ et pour $1 \leq k' < k$, si

– $Y' = \{\phi \in Y : \phi \in \mathrm{CONS}^{\langle k', 0, X \rangle}\}$ (restriction de Y aux contraintes sur les variables dans $\{\mathsf{x}_1, \ldots, \mathsf{x}_{k'}\}$) et,
– $v : \{\mathsf{x}_1, \ldots, \mathsf{x}_{k'}\} \to D$ telle que $Y' = \{\phi \in \mathrm{CONS}^{\langle k', 0, X \rangle} : v \models \phi\}$,

alors il existe une valuation $v' : \{\mathsf{x}_1, \ldots, \mathsf{x}_k\} \to D$ telle que v' restreinte à l'ensemble de variables $\{\mathsf{x}_1, \ldots, \mathsf{x}_{k'}\}$ est v et $Y = \{\phi \in \mathrm{CONS}^\mu : v' \models \phi\}$. Des propriétés similaires ont été aussi introduites dans [Dec92, BC02, LM05] sous le terme de "consistance globale".

Lorsque $\mathcal{D}$ satisfait la propriété de complétion, l'ensemble des modèles symboliques obtenus par abstraction admet une caractérisation simple.

Lemme 3.4.1. Soient $\mathcal{D}$ un système de contraintes vérifiant la propriété de complétion et μ une mesure. Un modèle symbolique ρ par rapport à μ a un modèle concret ssi ρ est consistant pas à pas.

A titre d'exemple, $\langle \mathbb{R}, <, = \rangle$, $\langle \mathbb{R}_+, <, = \rangle$, $\langle \mathbb{Q}, <, = \rangle$, $\langle \mathbb{Q}_+, <, = \rangle$ ainsi que $\langle D, = \rangle$ pour un ensemble non vide D satisfont la propriété de complétion. Le modèle symbolique ρ_0 ci-dessus a donc un modèle concret. Par contre, ρ_0 n'a pas de modèle concret avec le système de contraintes $\langle \mathbb{N}, <, = \rangle$ car $\mathbb{N}$ est bien fondé.

Soient μ une mesure et ϕ une formule de FOR^μ. On note $\mathrm{L}^\mu(\phi)$ l'ensemble des modèles symboliques ρ par rapport à μ tels que $\rho, 0 \models_\mu \phi$ et ρ a un modèle concret. On peut remarquer que ϕ est satisfaisable ssi $\mathrm{L}^\mu(\phi)$ est non vide.

Lemme 3.4.2. Soit $\mathcal{D}$ un système de contraintes ayant une abstraction complète f et satisfaisant la propriété de complétion. Pour chaque formule ϕ de $\mathrm{CLTL}(\mathcal{D})$, $\mathrm{L}^{f(\phi)}(\phi)$ est ω-régulier.

L'automate de Büchi acceptant le langage $\mathrm{L}^{f(\phi)}(\phi)$ peut être défini comme l'intersection de $\mathcal{A}^{f(\phi)}_\phi$ et de l'automate de Büchi reconnaissant les modèles symboliques consistants pas à pas.

En s'appuyant sur les propriétés ci-dessus, nous pouvons établir le résultat de complexité suivant.

Théorème 3.4.3. [BC02, DD02] Soit $\mathcal{D}$ un système de contraintes muni d'un nombre fini de relations. Lorsque $\mathcal{D}$ satisfait la propriété de complétion et les problèmes d'implication et de consistance maximale sont dans PSPACE, alors les problèmes de model-checking et satisfaisabilité sont dans PSPACE.

La preuve dans [BC02] utilise des arguments analogues à ceux de [SC85] pour montrer que LTL est dans PSPACE alors que la preuve dans [DD02] tire profit de l'approche par automates de [VW94].

Corollaire 3.4.4. Les problèmes de model-checking et satisfaisabilité pour les logiques $\mathrm{CLTL}(\mathbb{R}, <, =)$, $\mathrm{CLTL}(\mathbb{Q}, <, =)$ et $\mathrm{CLTL}(D, =)$ avec D ayant au moins deux éléments sont PSPACE-complets.

L'ajout dans $\mathrm{CLTL}(\mathbb{Q}, <, =)$ de contraintes de la forme $\mathsf{x} \sim c$ avec $c \in \mathbb{Q}$ et $\sim \in \{<, =\}$ conserve la borne supérieure PSPACE pour ce qui est de la satisfaisabilité. En effet, les constantes dans une formule (codées avec une représentation binaire) peuvent être simulées par des variables dont la valeur reste constante le long du modèle. Nous avons seulement besoin de spécifier comment les variables se comparent entre elles.

3.4.2 Un cas particulier: systèmes de contraintes finis

Lorsque $\mathcal{D}$ est fini, nous pouvons aussi avoir des résultats de complexité assez simplement. La finitude de $\mathcal{D}$ signifie que $\mathcal{D}$ est de la forme $\langle D, R_1, \ldots, R_N \rangle$ où D est un ensemble fini $\{d_1, \ldots, d_M\}$. Il n'est pas surprenant que dans ce cas, le problème de satisfaisabilité pour $\mathrm{CLTL}(\mathcal{D})$ soit dans PSPACE car on peut définir une réduction logarithmique en espace entre $\mathrm{CLTL}(\mathcal{D})$ et LTL.

Théorème 3.4.5. Soit $\mathcal{D}$ un système de contraintes fini. Le problème de satisfaisabilité pour $\mathrm{CLTL}(\mathcal{D})$ est dans PSPACE.

Une preuve consiste par exemple à introduire un système auxiliaire

$$\mathcal{D}' = \langle D, P_1, \ldots, P_M \rangle$$

tel que $P_i = \{d_i\}$ pour chaque i. Une réduction logarithmique en espace entre $\mathrm{CLTL}(\mathcal{D})$ et $\mathrm{CLTL}(\mathcal{D}')$ peut alors être construite et le problème de satisfaisabilité pour $\mathrm{CLTL}(\mathcal{D}')$ peut être montré dans PSPACE car $\mathcal{D}'$ admet une abstraction complète simple et vérifie la propriété de complétion, voir par exemple [Dem06b]. La réduction est en fait homomorphique pour les opérateurs temporels et Booléens et $\mathsf{R}(t_1, \ldots, t_n)$ se traduit en

$$\bigvee_{\langle d_{i_1}, \ldots, d_{i_n} \rangle \in R} P_{i_1}(t_1) \wedge \cdots \wedge P_{i_n}(t_n).$$

Ce pas atomique permet bien d'avoir une réduction logarithmique en espace car les arités des relations R_i et le cardinal de D sont des paramètres de la logique. La PSPACE-dureté de $\mathrm{CLTL}(\mathcal{D})$ est garantie en réduisant LTL dès que $\mathcal{D}$ est non trivial. Cela signifie qu'il existe une relation R dans $\mathcal{D}$ d'arité $n \geq 1$ telle que $R \neq \emptyset$ ou $R \neq D^n$.

3.4.3 Contraintes de périodicité

Le langage IPC^{++} est un fragment assez expressif de (PA). Par exemple, le problème de consistance est PSPACE-complet alors que pour le système $\langle \mathbb{N}, <, = \rangle$ le problème est seulement NLOGSPACE-complet. Les formules de $\mathrm{CLTL}(\mathrm{IPC}^{++})$ peuvent en effet représenter les notions de calendriers ("calendars") et tranches ("slices") de [NS92]. Un calendrier C est défini comme une partition $X_1, X_2, \ldots$ de $\mathbb{N}$ telle que (on omet ici le cas où la partition est finie)

(ordre) pour i, $x \in X_i$ et $y \in X_{i+1}$, nous avons $x < y$,

(suite) pour i, il existe $x \in X_i$ et $y \in X_{i+1}$ tels que $y = x + 1$.

Un calendrier $C = X_1, X_2, \ldots$ peut donc être représenté dans $\mathrm{CLTL}(\mathrm{IPC}^{++})$ par l'inteprétation d'une variable x dans un modèle σ de $\mathrm{CLTL}(\mathrm{IPC}^{++})$ de la forme $\sigma : \mathbb{N} \times \mathrm{VAR} \to \mathbb{Z}$ tel que des positions consécutives dans σ ayant la même valeur pour x appartiennent à la même classe:

$$\underbrace{\sigma(0,\mathsf{x}) = \sigma(1,\mathsf{x}) = \ldots = \sigma(i_1,\mathsf{x})}_{X_1=\{0,\ldots,i_1\}} \neq \underbrace{\sigma(i_1+1,\mathsf{x}) = \ldots = \sigma(i_2,\mathsf{x})}_{X_2=\{i_1+1,\ldots,i_2\}} \neq \ldots$$

Dans la plupart des cas, $\{\sigma(i,\mathsf{x}) : i \in \mathbb{N}\}$ est fini (minutes, heures, jours de la semaine, mois). Cela signifie que tels calendriers peuvent être codés de façon alternative comme des positions successives ayant la même valeur modulo un entier. Les granularités temporelles [Wij00] qui sont une variante des calendriers sont aussi définies comme des classes d'équivalence sur $\mathbb{N}$ et comme des ω-séquences sur un alphabet à trois lettres, voir les détails dans [Wij00].

Théorème 3.4.6. [Dem06b] Les problèmes de model-checking et satisfaisabilité pour $\mathrm{CLTL}(\mathrm{IPC}^{++})$ sont PSPACE-complets.

Un argument majeur pour la preuve du théorème 3.4.6 est que IPC^{++} admet une abstraction complète et il est possible de coder en espace polynomial les états symboliques maximalement consistants. Comme application du théorème 3.4.6, on peut caractériser la complexité du problème de l'équivalence pour les automates étendus à un mot ("extended single-string automata", abrégé en ESSA dans la suite) définis dans [LM01, section 5], voir aussi d'autres automates similaires dans [BMP04, Pup06]. Ce problème est central pour vérifier si deux granularités temporelles sont équivalentes [Wij00] lorsque les granularités sont codées par tels automates qui sont des automates de Büchi reconnaissant exactement une ω-séquence. Les gardes des transitions construites sur des contraintes de périodicité assurent la concision de ces automates à contraintes. De même, dans [CFP02], les auteurs expriment le besoin de concevoir une extension de LTL avec des

contraintes temporelles quantitatives telles que des contraintes de périodicité. La logique $\mathrm{CLTL}(\mathrm{IPC}^{++})$ étendue avec les opérateurs de passé, qui est la logique considérée dans [Dem06b], fournit une telle extension avec des problèmes qui demeurent PSPACE-complets.

Soit IPC' le fragment de $\mathrm{IPC}^{\{\exists\}}$ qui contient les combinaisons Booléennes de contraintes atomiques de la forme $\mathsf{x} \equiv_k c$ ou bien $\exists \mathsf{z}\ (\mathsf{x} \equiv_k \mathsf{z} \ \wedge\ \mathsf{y} \equiv_{k'} \mathsf{z})$. Une fonction de mise à jour g pour la variable x_i est une expression de la forme soit $\mathsf{x}_i := \mathsf{x}_i + c$ ou $\mathsf{x}_i := c$ avec $c \in \mathbb{Z}$. On note $\mathrm{UP}_{\mathsf{x}_1,\ldots,\mathsf{x}_n}$ l'ensemble des fonctions de mise à jour qui utilise des variables parmi $\{\mathsf{x}_1, \ldots, \mathsf{x}_n\}$. Un automate étendu à un mot $\mathcal{A}$ (ESSA) sur l'ensemble de variables $\{\mathsf{x}_1, \ldots, \mathsf{x}_n\}$ [LM01] est une structure de la forme $\langle Q, q_0, \overline{v_0}, \Sigma, \delta\rangle$ avec

- Q est un ensemble fini d'états et $q_0 \in Q$ (état initial),
- $\vec{v_0} \in \mathbb{Z}^n$ (valeur initiale des variables)
- Σ est un alphabet fini,
- $\delta \subseteq Q \times \Sigma \times Q \times (\{\top\} \cup \mathrm{IPC}') \times \mathcal{P}(\mathrm{UP}_{\mathsf{x}_1,\ldots,\mathsf{x}_n})$ et pour chaque $q \in Q$, il y a exactement deux u tels que $\langle q, u\rangle \in \delta$, disons u_1 et u_2, et dans ce cas u_1 est de la forme $\langle a_1, q_1, A, X_1\rangle$, u_2 est de la forme $\langle a_2, q_2, \neg A, X_2\rangle$ où A est une contrainte de IPC^* construite sur $\{\mathsf{x}_1, \ldots, \mathsf{x}_n\}$ et dans X_1 et X_2 il y a exactement une fonction de mise à jour pour chaque x_i.

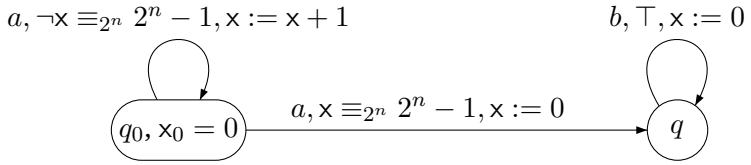
Les éléments de δ sont aussi notés $q \xrightarrow{a,A,X} a'$ (A est la garde et X est la fonction de mise à jour globale).

Une configuration est un élément de $\langle q, \vec{v}\rangle \in Q \times \mathbb{Z}^n$. La relation de transition $\xrightarrow{a}$ pour $a \in \Sigma$ est définie ainsi: $\langle q, \vec{v}\rangle \xrightarrow{a} \langle q', \vec{v'}\rangle$ ssi il existe une transition $q \xrightarrow{a,A,X} q' \in \delta$ telle que $[\mathsf{x}_1 \leftarrow v_1, \ldots, \mathsf{x}_n \leftarrow v_n] \models A$ (dans IPC^{++}) et pour chaque $g \in X$,

- si g est égale à $\mathsf{x}_i := \mathsf{x}_i + c$ alors $v'_i = v_i + c$,
- si g est égale à $\mathsf{x}_i := c$ alors $v'_i = c$.

On peut facilement vérifier qu'il existe exactement une unique ω-séquence $w = a_1 a_2 \ldots \in \Sigma^\omega$ telle que $\langle q_0, \vec{v_0}\rangle \xrightarrow{a_1} \langle q_1, \vec{v_1}\rangle \xrightarrow{a_2} \ldots$. L'unique ω-séquence générée par $\mathcal{A}$ est notée $w_\mathcal{A}$. Le problème de l'équivalence de ESSA détermine si $w_\mathcal{A} = w_{\mathcal{A}'}$ étant donnés deux ESSA $\mathcal{A}$ et $\mathcal{A}'$. Ce problème introduit dans [LM01] est central pour vérifier l'équivalence de granularités temporelles lorsque les granulatrités sont codées comme des automates.

Par exemple, l'ω-séquence associée au ESSA ci-dessous est $a^{2^n} \cdot b^\omega$ avec pour valeur initiale 0:



Théorème 3.4.7. [Dem06b] Le problème de l'équivalence pour les ESSA est PSPACE-complet.

Pour obtenir la borne supérieure PSPACE, étant donnés deux ESSA $\mathcal{A}$ et $\mathcal{A}'$, nous construisons un $\mathrm{IPC}^{\{\exists\}}$-automate $\mathcal{B}$ en espace logarithmique tel que $\mathcal{B} \models_\exists \top$ ssi $w_\mathcal{A} = w_{\mathcal{A}'}$. $\mathcal{B}$ est en fait un produit de $\mathcal{A}$ et $\mathcal{A}'$. La borne inférieure PSPACE est obtenue en réduisant QBF. Ainsi, une conséquence de la preuve du théorème 3.4.7 est que $w_\mathcal{A} = w_{\mathcal{A}'}$ peut être vérifiée en temps

$$\mathcal{O}(2^{2\times maxsize^2 \times n} \times |Q| \times |Q'|),$$

où n est le nombre de variables utilisées dans $\mathcal{A}, \mathcal{A}'$ et $maxsize$ est la taille du plus grand entier k apparaissant dans des gardes de $\mathcal{A}, \mathcal{A}'$ avec $\equiv_k$. Ainsi, le plus grand entier apparaissant dans $\mathcal{A}, \mathcal{A}'$ a une valeur en $\mathcal{O}(2^{maxsize})$. En fait, notre preuve établit aussi que la PSPACE-dureté est conservée si le plus grand entier apparaissant dans des gardes avec $\equiv_k$ est 2 or si les entiers sont codés avec une représentation unaire.

Problème ouvert 10. Est-ce que le problème de l'équivalence de ESSA lorsque chaque automate est réduit à une variable est PSPACE-difficile? ◯

Lorsque chaque automate peut contenir jusqu'à deux variables, le problème de l'équivalence est PSPACE-difficile [Dem06b]. Ce problème semble posséder des similitudes avec la caractérisation du problème de vacuité pour les automates temporisés à deux horloges [LMS04] et avec le problème ouvert 13.

Un autre problème plus simple apparaissant avec les granularités temporelles consiste à trouver la n^{ieme} occurrence d'un symbole dans une chaîne [LMP03, section 4]. Le problème de l'occurrence pour les ESSA a la définition suivante:

entrée: un ESSA $\mathcal{A}$, $a \in \Sigma$ et $n, m \in \mathbb{N}$ (en binaire).

question: est-ce que la n^{ieme} occurrence de a dans $w_\mathcal{A}$ est à une position inférieure à m?

La preuve du théorème 3.4.7 peut être facilement adaptée pour montrer le résultat suivant.

Théorème 3.4.8. [Dem06b] Le problème de l'occurrence pour les ESSA est PSPACE-complet.

3.4.4 Décidabilité de CLTL$(\mathbb{Z}, <, =)$

Même si les problèmes de consistance pour $\langle \mathbb{Z}, <, = \rangle$ et $\langle \mathbb{R}, <, = \rangle$ sont identiques, et ces deux systèmes admettent des abstractions complètes, la résolution du problème de satisfaisabilité pour $\mathrm{CLTL}(\mathbb{Z}, <, =)$ en PSPACE nécessite une technique bien plus complexe. En effet, non seulement $\langle \mathbb{Z}, <, = \rangle$ ne vérifie pas la propriété de complétion mais l'ω-régularité de $\mathrm{L}^\mu(\phi)$ n'est pas garantie ce qui peut invalider l'usage d'automates de Büchi.

Lemme 3.4.9. [DD07] Il existe une mesure μ et une formule ϕ dans FOR^μ tels que $\mathrm{L}^\mu(\phi)$ n'est pas ω-régulier.

Le problème est qu'il existe une mesure μ pour laquelle la classe des modèles symboliques qui sont consistants pas à pas et qui admettent un modèle concret n'est pas nécessairement ω-régulière. Pour contourner cette difficulté, nous avons introduit dans [DD07] une surapproximation de cette classe, qui est ω-régulière et nous avons montré que tout modèle symbolique ultimement périodique consistant pas à pas qui vérifie cette nouvelle condition admet un modèle concret. Cela nous a permis de montrer le résultat suivant.

Théorème 3.4.10. [DD07] Les problèmes de satisfaisabilité et model-checking pour $\mathrm{CLTL}(\mathbb{N}, <, =)$ et $\mathrm{CLTL}(\mathbb{Z}, <, =)$ sont PSPACE-complets.

Soit Σ un alphabet fini et $\subset$ une relation binaire sur les mots de Σ^* comme la relation sous-mot, préfixe, suffixe, etc... Dans la suite on suppose juste que $\subset$ est la relation sous-mot stricte. La logique $\mathrm{CLTL}(\Sigma, \subset, =)$ est précisément la logique $\mathrm{CLTL}(\mathbb{N}, <, =)$ lorsque Σ est un singleton. Dans le cas général, on ignore complétement si les problèmes SAT($\mathrm{CLTL}(\Sigma, \subset, =)$) et MC($\mathrm{CLTL}(\Sigma, \subset, =)$) sont décidables.

Problème ouvert 11. Est-ce que les problèmes SAT($\mathrm{CLTL}(\{0,1\}^*, \subset, =)$) et MC($\mathrm{CLTL}(\{0,1\}^*, \subset, =)$) où $\subset$ est la relation sous-mot stricte sont décidables? ◯

Les chaînes de caractères formant un type de données omniprésent dans les langages de programmation, les problèmes relatifs à $\mathrm{CLTL}(\Sigma, \subset, =)$ méritent donc une étude approfondie. On ne sait même pas si la cardinalité de l'alphabet revête une importance à l'opposé de ce qui se passe avec les machines de Turing.

Problème ouvert 12. Existe-t-il toujours une réduction logarithmique en espace entre le problème de satisfaisabilité pour $\mathrm{CLTL}(\Sigma, \subset, =)$ avec Σ quelconque et le problème de satisfaisabilité pour $\mathrm{CLTL}(\{0,1\}^*, \subset, =)$? ◯

Le théorème 3.4.10 a été étendu dans [DG05]. Les contraintes du langage IPC^*, notées A, sont définies par la grammaire suivante:

$$A ::= A' \mid \mathsf{x} < \mathsf{y} \mid A \wedge A \mid \neg A$$

$$A' ::= \mathsf{x} \equiv_k [c_1, c_2] \mid \mathsf{x} \equiv_k \mathsf{y} + [c_1, c_2] \mid \mathsf{x} = \mathsf{y} \mid \mathsf{y} < d \mid \mathsf{x} = d \mid A' \wedge A' \mid \neg A' \mid \exists \mathsf{x}\, A'$$

avec $\mathsf{x}, \mathsf{y} \in \mathrm{VAR}$, $k \in \mathbb{N} \setminus \{0\}$, $c_1, c_2 \in \mathbb{N}$ et $d \in \mathbb{Z}$. Le langage IPC^* étend à la fois $\langle \mathbb{Z}, <, = \rangle$ et IPC^{++}. Cependant, les contraintes de IPC^* n'autorisent pas d'avoir des occurrences de $\mathsf{x} < \mathsf{y}$ dans la portée de $\exists$ ce qui permettrait alors de coder l'incrémentation.

Jusqu'à maintenant, le langage IPC^* est la classe optimale de contraintes qualitatives sur $\mathbb{Z}$ pour laquelle $\mathrm{CLTL}(\mathrm{IPC}^*)$ est décidable en espace polynomial. Par contrainte qualitative, nous entendons par exemple une contrainte qui est interprétée comme une relation binaire non déterministe comme $\mathsf{x} < \mathsf{y}$ et $\mathsf{x} \equiv_{2^k}$

$\mathsf{y} + 5$ (la relation entre x et y est un peu lâche). Les automates à contraintes avec des contraintes qualitatives sur $\mathbb{Z}$ paraissent un modèle opérationnel séduisant car ils peuvent être interprétés comme des abstractions d'automates à compteurs où les opérations d'incrémentation et de décrémentation sont abstraites par des opérations modulo une puissance de deux. De nombreux langages de programmation utilisent des opérateurs arithmétiques modulo 2^k, voir dans [MOS05] des motivations analogues, typiquement k est 32 ou 64. Ainsi, $\mathsf{x} = \mathsf{y} + 1$ peut être abstrait par $\mathsf{x} \equiv_{2^k} \mathsf{y} + 1 \ \wedge \ \mathsf{y} < \mathsf{x}$ ce qui est exprimable dans IPC^*.

En étendant adéquatement les techniques de preuves de [DD07, Dem06b], nous pouvons caractériser la complexité de $\mathrm{CLTL}(\mathrm{IPC}^*)$.

Théorème 3.4.11. [DG05] Les problèmes de model-checking et satisfaisabilité pour $\mathrm{CLTL}(\mathrm{IPC}^*)$ sont PSPACE-complets.

De façon assez étonnante, le résultat ci-dessus nous permet de caractériser la complexité du model-checking des automates relationnels intégraux introduits dans [Čer94a].

Corollaire 3.4.12. Le problème de model-checking pour les automates relationnels intégraux restreint au fragment LTL de CCTL* introduit dans [Čer94a] est dans PSPACE.

On sait cependant que le problème de model-checking pour les automates relationnels intégraux restreint au fragment CTL de CCTL* est indécidable [Čer94a].

3.4.5 Le fragment $\mathrm{CLTL}_1^1(\mathrm{DL})$

Nous avons montré précédemment que MC($\mathrm{CLTL}_2^1(\mathrm{DL})$) et MC($\mathrm{CLTL}_1^2(\mathrm{DL})$) sont indécidables. C'est seulement en se restreignant à une variable et aux formules de $\mathtt{X}$-hauteur au plus une, qu'il est possible de regagner la borne supérieure PSPACE.

Théorème 3.4.13. [DG07] SAT($\mathrm{CLTL}_1^1(\mathrm{DL})$) et MC($\mathrm{CLTL}_1^1(\mathrm{DL})$) sont PSPACE-complets.

La preuve s'appuie sur le fait que DL admet une abstraction complète. De plus, la classe des modèles symboliques (par rapport à $\mu = \langle 1, 1, X \rangle$) qui admettent un modèle concret peut être reconnue par un automate à un compteur pour lequel

- le compteur est interprété dans $\mathbb{Z}$,
- il y a des tests à zéro et des tests de signe,
- les mots acceptés sont des ω-séquences et la condition d'acceptance est une condition de Büchi,
- les mises à jour du compteur sont parmi 0,-1,1.

Les automates de Büchi standard forment une sous-classe de ces automates à un compteur. De plus, pour obtenir la borne PSPACE nous avons montré que le problème de vacuité pour cette classe d'automates à un compteur est NLOGSPACE-complet [DG06].

Si on étend le modèle en autorisant des mises à jour dans $d \in \mathbb{Z}$ avec d codé avec une représentation binaire, le problème de vacuité est NP-difficile et dans PSPACE.

Problème ouvert 13. Est-ce que le problème de vacuité pour la classe étendue d'automates à un compteur est dans NP? ◯

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

Nous avons vu dans la section 3.2.4 que le problème de satisfaisabilité pour $\mathrm{CLTL}_1^1(\mathrm{QFP})$ est hautement indécidable. Par conséquent, il en est de même pour le model-checking de $\mathrm{CLTL}_1^1(\mathrm{QFP})$. Cependant, il existe une restriction pertinente de ce problème qui est décidable. En effet, considérons comme modèles opérationnels les automates à un compteur où le compteur est interprété dans $\mathbb{Z}$, il y a des tests à zéro et de signe, les mots acceptés sont des ω-séquences, et les mises à jour du compteur sont dans $\mathbb{Z}$.

Théorème 3.4.14. [DG06] Le problème du model-checking pour $\mathrm{CLTL}_1^\omega(\mathrm{QFP})$ sur les automates à un compteur avec mise à jour dans $\mathbb{Z}$ est PSPACE-complet.

3.4.7 Autres extensions décidables

Une des caractéristiques intéressantes du théorème 3.3.2 repose sur les traitements séparés entre la satisfaction symbolique et l'existence de modèles concrets pour les modèles symboliques. Ainsi, soit $\mathcal{D}$ un système de contraintes tel que le problème de satisfaisabilité pour $\mathrm{CLTL}(\mathcal{D})$ a été montré dans PSPACE dans une section précédente. Soit LTL^+ une extension de LTL dont on sait traduire les formules en automates de Büchi en espace polynomial. Cela inclut par exemple les extensions suivantes:

- avec des opérateurs de passé comme "previous" et "since" [LP00],
- plus généralement avec un nombre fini d'opérateurs temporels définissables dans la logique monadique du second-ordre (MSO) [GK03],
- avec des opérateurs à base d'automates [Wol83],
- avec un opérateur de point fixe, voir par exemple [Var88].

Notre technique de preuve nous assure la conséquence suivante.

Théorème 3.4.15. Les problèmes de model-checking et satisfaisabilité pour la logique $\mathrm{CLTL}^+(\mathcal{D})$ sont dans PSPACE.

Les extensions de $\mathrm{CLTL}(\mathcal{D})$ par ajout d'opérateurs temporels définissables dans MSO (mais pas nécessairement en quantité finie) sont décidables si $\mathrm{CLTL}(\mathcal{D})$ peut être montrée décidable avec les techniques ci-dessus. Il suffit alors d'adapter la définition de $\mathcal{A}_\phi^{f(\phi)}$ pour passer de LTL à LTL^+.

3.4.8 Un bref état de l'art de formalismes voisins

Dans cette section, nous allons brièvement mettre en relation les problèmes de model-checking et satisfaisabilité des logiques $\mathrm{CLTL}(\mathcal{D})$ avec des formalismes introduits par ailleurs. Une présentation plus complète est faite dans [Dem06a].

Vérification d'automates à compteurs. Comme nous l'avons déjà remarqué, l'analyse des questions d'accessibilité pour les automates à compteurs est omniprésente dès qu'il s'agit de vérifier des systèmes avec un nombre infini de configurations, voir exemple les modèles opérationnels de [ISD+00, CJ98, FL02, DPK03, FS00]. On peut remarquer que même si la décidabilité est assurée seulement au prix de restrictions quelquefois drastiques sur les systèmes, il existe une classe de systèmes à compteurs qui permet de capturer de très nombreuses études de cas. Par exemple, les systèmes applatissables [LS05] admettent un dépliage plat du graphe de contrôle qui préserve l'ensemble des configurations accessibles. La contrainte de platitude implique que chaque état du graphe de contrôle appartient à au plus un cycle. Des propriétés autres que l'accessibilité sont prises en compte dans [DFGvD06].

Si on s'intéresse à des propriétés plus riches exprimables dans un cadre logique, des logiques temporelles avec contraintes de Presburger ont été développées dans [Čer94a, BEH95, BGP97, CC00, BDR03], quelques-unes ayant des fragments décidables assez expressifs. Cependant, l'indécidabilité du problème d'accessibilité peut être établie pour des classes d'automates à compteurs de prime abord inoffensifs, voir par exemple [Cor02, Pot04]. De même, plusieurs résultats de décidabilité sont obtenus par réduction vers l'arithmétique de Presburger [FO97, FL02, BDR03, DFGvD06].

De par leur restriction, les automates à un compteur ont des problèmes moins complexes, voir par exemple les résultats sur les équivalences comportementales dans [Kuč00, JKMS04]. La classe des automates à un compteur est évidemment équivalente aux automates à pile avec un alphabet de pile réduit à un singleton et donc de nombreux résultats obtenus pour cette classe d'automates s'appliquent aussi aux automates à un compteur. Par exemple, le problème de model-checking sur les automates à un compteur avec le μ-calcul modal est dans EXPTIME [Wal01] et ce résultat a été raffiné dans [Ser04, section 7.2] où il est montré que ce problème est dans PSPACE (voir aussi [Ser06]). En général les formules atomiques de ces logiques sont des états de contrôle et pas exactement des contraintes sur la valeur du compteur, une différence notable avec le formalisme impliqué dans le théorème 3.4.14. De même dans [BEM97], le problème du model-checking d'automates à pile avec le μ-calcul linéaire est montré dans EXPTIME.

Même s'il est vrai que les résultats sur les automates à un compteur n'ont pas la prétention de donner lieu à de nombreuses applications (l'obtention d'une taxonomie est la vertu principale semble-t-il), ces automates ont été utilisés par exemple pour la vérification de protocoles cryptographiques [LLT05] ou pour la validation de flux XML (chaînes représentant des documents XML) en codant les DTDs récursifs avec des automates à un compteur [CR04].

Logiques temporelles du premier ordre Les logiques de la forme $\mathrm{CLTL}(\mathcal{D})$ et $\mathrm{CLTL}^{\downarrow}(\mathcal{D})$ (voir la section 3.5) peuvent être vues comme des fragments de LTL du premier ordre où le domaine d'interprétation des variables est fixe ainsi que l'interprétation des symboles de relation. De plus, les variables flexibles de $\mathrm{CLTL}(\mathcal{D})$ correspondent à des symboles de prédicat unaires interprétés par des singletons (les valeurs des variables). Cependant, $\mathrm{CLTL}(\mathcal{D})$ n'a pas de quantification sur les éléments du domaine (sauf dans la section 3.5) et en ce sens cela correspond à un fragment très restreint de LTL du premier ordre. On rappelle de plus que LTL du premier ordre est hautement indécidable [Aba89] même dans le cas où les domaines non interprétés sont finis [Tra63]. De même, LTL du premier ordre sur des structures temporelles finis est hautement indécidable [CMP99].

Une variante de LTL du premier ordre a été introduite dans [DSV04] pour vérifier des applications Web. L'interaction entre les opérateurs temporels et les quantifications du premier ordre est restreinte car aucune quantification ne peut apparaître dans la portée d'opérateurs temporels, ce qui assure de bonnes propriétés algorithmiques.

3.5 Un mécanisme de registres

Les contraintes atomiques temporelles de $\mathrm{CLTL}(\mathcal{D})$ permettent de comparer des valeurs de variables pour des positions de distance bornée comme dans $\mathsf{x} < \mathtt{X}^2\mathsf{y}$. Les langages temporels précédents n'ont pas la possibilité d'exprimer une propriété de la forme: "il existe $i \geq 0$ tel que $\mathsf{x} < \mathtt{X}^i\mathsf{y}$ est vérifié" ce qui peut s'écrire "$\bigvee_i \mathsf{x} < \mathtt{X}^i\mathsf{y}$" avec une disjonction généralisée. De même, une propriété comme "toutes les valeurs futures de x sont différentes de la valeur courante de x", qui pourrait s'écrire $\bigwedge_{i>0} \neg(\mathsf{x} = \mathtt{X}^i\mathsf{x})$, ne peut pas être exprimée avec les langages vus précédemment. Dans cette section, nous allons présenter des extensions de $\mathrm{CLTL}(\mathcal{D})$ qui peuvent exprimer de telles propriétés en ajoutant l'opérateur "freeze". L'usage de cet opérateur offre la possibilité de stocker une valeur de D (typiquement la valeur d'une variable) et de la comparer avec la valeur d'une variable mais à une position de distance non bornée avec la position où la première valeur a été stockée. L'extension de $\mathrm{CLTL}(\mathcal{D})$ avec l'opérateur "freeze" $\downarrow$ est notée $\mathrm{CLTL}^{\downarrow}(\mathcal{D})$.

3.5.1 Définition

Afin de définir $\mathrm{CLTL}^{\downarrow}(\mathcal{D})$, l'ensemble des variables VAR est divisé entre deux ensembles infinis disjoints: VAR_r est l'ensemble des variables rigides et VAR_f est l'ensemble des variables flexibles. L'ensemble des variables de $\mathrm{CLTL}(\mathcal{D})$ (sans opérateur "freeze") est constitué uniquement de variables flexibles. La clause $\downarrow_{\mathsf{y}=\mathtt{X}^j\mathsf{x}} \phi$ avec $\mathsf{y} \in \mathrm{VAR}_r$ et $\mathsf{x} \in \mathrm{VAR}_f$ est ajoutée à la définition des formules de $\mathrm{CLTL}(\mathcal{D})$. Les formules atomiques de $\mathrm{CLTL}^{\downarrow}(\mathcal{D})$ sont des expressions de la forme $\mathsf{R}(t_1, \ldots, t_n)$ où chaque t_i est soit une variable rigide soit un terme de la forme $\mathtt{X}^i\mathsf{x}$ avec $\mathsf{x} \in \mathrm{VAR}_f$. Un modèle σ de $\mathrm{CLTL}^{\downarrow}(\mathcal{D})$ est une séquence infinie de valuations $\sigma : \mathbb{N} \times \mathrm{VAR}_f \to D$ (seules les variables flexibles sont interprétées

dans le modèle) et la relation de satisfaction est indicée par un environnement $e : \mathrm{VAR}_r \to D$ (elle admet donc un argument supplémentaire). La définition de $\models_e$ est la suivante pour le nouveau type de formules:

$$\sigma, i \models_e \downarrow_{\mathtt{y}=\mathtt{X}^j\mathtt{x}} \phi \overset{\text{def}}{\Leftrightarrow} \sigma, i \models_{e'} \phi \text{ où } e' \text{ est la variante de } e \text{ en modifiant éventuellement la valeur de y: } e'(\mathsf{y}) = \sigma(i+j)(\mathsf{x}).$$

La satisfaction des formules atomiques utilise à la fois le modèle σ et l'environnement e selon que les variables sont dans VAR_f ou dans VAR_r. Sans perte de généralité, on peut aussi supposer que dans toutes les formules de $\mathrm{CLTL}^{\downarrow}(\mathcal{D})$, les variables libres de ϕ sont nécessairement flexibles. Les problèmes de model-checking et satisfaisabilité sont définis comme pour $\mathrm{CLTL}(\mathcal{D})$ mais avec un langage un peu plus riche.

Ce mécanisme de liaison est en fait présent dans de nombreux formalismes. Nous en donnons un bref aperçu ci-dessous mais des comparaisons plus approfondies peuvent être trouvées dans [DLN07, DL06a, Seg06].

- Les logiques temps-réel comme la version discrète de TPTL [AH94] (voir aussi [Hen90]) utilisent un tel mécanisme. TPTL peut être définie comme un fragment de la logique $\mathrm{CLTL}^{\downarrow}(\mathcal{D})$ avec
  - $D = \mathbb{N}$ et la seule variable flexible est t inteprétée comme la valeur courante du temps,
  - les relations de $\mathcal{D}$ sont définies à partir des prédicats suivants:

    $$(\mathsf{x} \leq c)_{c\in\mathbb{Z}}, (\mathsf{x} \leq \mathsf{y} + c)_{c\in\mathbb{Z}}, (\mathsf{x} \equiv_d c)_{c,d\in\mathbb{N}}, (\mathsf{x} \equiv_d \mathsf{y} + c)_{c,d\in\mathbb{N}}$$

  - on se restreint aux formules de la forme

    $$\overbrace{\mathtt{G}(\mathtt{t} \leq \mathtt{Xt})}^{\text{monotonie}} \wedge \overbrace{\mathtt{GF}(\mathtt{t} < \mathtt{Xt})}^{\text{progression}} \wedge \phi$$

    où tout usage de l'opérateur "freeze" est de la forme $\downarrow_{\mathsf{x}=\mathtt{t}}$ et il n'y pas d'autre occurrence de t.

  L'abandon de la condition de monotonie du temps entraîne l'indécidabilité de la logique [AH94, théorème 5].
- Des extensions de logiques temporelles standard contiennent aussi des mécanismes qui permettent de stocker en mémoire un état du modèle. Dans les exemples ci-dessous, les extensions sont aussi expressives que la logique temporelle standard correspondante. Dans [Lar94] une extension de LTL (avec passé) contient l'opérateur "Now" tel que la formule $\mathtt{Now}\,\phi$ est vérifiée lorsque ϕ est vraie dans la structure où l'origine est la position courante en oubliant le passé. Cet opérateur que W. Smith utilisait en 1984 admet une formulation équivalente qui consiste à introduire un registre stockant la position de l'origine et l'effet de l'opérateur "Now" revient à mettre dans le registre la position courante. Cette logique est aussi expressive que LTL mais plus concise [LMS02]. Un autre exemple est fourni par la logique

arborescente avec mémoire [KV06] qui utilise une version alternative du quantificateur existentiel de chemin. La formule $\mathtt{E}\ \phi$ est vérifiée à l'état x s'il existe un chemin π de la racine passant par x tel que ϕ est vérifiée à la racine le long de π et la proposition atomique spéciale $present$ (qui peut apparaître dans ϕ) prend la valeur x. Ce mécanisme stocke l'état x et teste plus tard cette valeur avec la valeur d'un état courant. Cette logique demeure aussi expressive que CTL*.

– Dans [Fit02] un mécanisme de "λ-abstraction" a été introduit pour résoudre le fameux problème d'interprétation des constantes dans les logiques modales du premier ordre. Alors que divers résultats de décidabilité et d'indécidabilité sont établis dans [Fit02] pour des classes de modèles pas nécessairement linéaires, ce mécanisme est essentiellement celui de l'opérateur "freeze" même si le lien entre les différents travaux [Hen90, Fit02, LP05] n'a été effectué que tardivement. De façon indépendante, des résultats d'indécidabilité dans le cas de logiques temporelles du temps linéaire ont été présentés dans [LP05, DLN05].

– Dans [BPT03, Bou02], des langages de données sont introduits comme des ensembles de mots de données dans $(\Sigma \times D)^*$ où Σ est un alphabet fini et D est un domaine infini. Cela généralise la notion de langage de mots temporisés et des automates reconnaissant des mots de données sont étudiés dans [BPT03, Bou02]. Ce type d'automates n'est pas sans rapport avec les automates à registres reconnaissant des mots sur des alphabets infinis [NSV04].

  Une logique du premier ordre interprétée sur des mots de données est introduite dans [BMS$^+$06] avec des motivations relatives aux langages de requêtes pour données semi-structurées (voir aussi [BDM$^+$06] pour le cas arborescent). Un résultat majeur de [BMS$^+$06] établit que la logique FO$^2(\sim, <, +1)$ sur les mots de données finis est décidable en réduisant le problème de satisfaisabilité à un problème d'accessibilité dans les réseaux de Petri. Cette logique n'utilise que deux variables individuelles (qui peuvent être recyclées), $<$ et $+1$ sont les relations usuelles sur les positions et la relation $\sim$ spécifie que deux positions ont la même donnée. D'autres problèmes, en particulier l'extension aux mots de données infinis, sont montrés décidables dans [BMS$^+$06].

– Les logiques hybrides, voir par exemple [Gor96, ABM99, FdRS03], contiennent un mécanisme de liaison analogue à l'opérateur "freeze": $\downarrow_{\mathsf{x}} \phi(\mathsf{x})$ est vérifiée ssi $\phi(\mathsf{x})$ est vérifiée lorsque la variable propositionnelle x est interprétée comme un singleton contenant l'état courant. Des exemples de logiques hybrides avec cet opérateur pour l'interrogation de données semi-structurées peuvent être trouvés dans [BCT04, Thi04, BC06]. Sur des structures arbitraires, il s'agit d'un mécanisme très puissant qui conduit à des logiques indécidables (voir par exemple [Mar02]). La complexité de fragments dans le cas de modèles linéaires avec un nombre borné de variables rigides est étudiée dans [SW07], voir aussi [FdRS03].

La formule ci-dessous de $\mathrm{CLTL}^{\downarrow}(\mathbb{N},=)$ exprime que toutes le valeurs de la variable x sont distinctes:

$$\varphi_0 = \mathtt{G} \downarrow_{\mathtt{y}=\mathtt{x}} \mathtt{XG}(\mathtt{x} \neq \mathtt{y}).$$

L'opérateur "freeze" est en fait extrêmement puissant car une formule ϕ satisfaisable dans $\mathrm{CLTL}(\mathbb{N},=)$ a toujours un modèle avec un nombre fini de valeurs distinctes [DLN07]. Par contre, les modèles de φ_0 ont toujours un nombre infini de valeurs.

3.5.2 Fragments indécidables

De façon surprenante, l'ajout de l'opérateur "freeze" en présence du système de contraintes élémentaire $\langle \mathbb{N},= \rangle$ conduit à l'indécidabilité même si finalement ce système est assez rudimentaire. On peut tout de même rappeler ici que $\langle \mathbb{N},= \rangle$ vérifie la propriété de complétion, $\mathrm{CLTL}(\mathbb{N},=)$ admet une abstraction complète et MC($\mathrm{CLTL}(\mathbb{N},=)$) est PSPACE-complet.

Théorème 3.5.1. [DLN07] SAT($\mathrm{CLTL}^{\downarrow}(\mathbb{N},=)$) restreint à deux variables rigides et à une variable rigide est Σ_1^1-complet.

L'indécidabilité avec trois variables rigides a été montrée indépendamment dans [LP05]. L'indécidabilité est conservée si on considère des modèles finis. En présence d'une seule variable rigide, l'indécidabilité est encore vraie mais dans une moindre mesure.

Théorème 3.5.2. [DL06a] Le problème de la satisfaisabilité pour $\mathrm{CLTL}^{\downarrow}(\mathbb{N},=)$ restreint à une variable flexible et à une variable rigide est Π_1^0-complet.

La preuve est par réduction du problème de vacuité pour les automates à compteurs avec erreur d'incrémentation dans le cas infini. Cette nouvelle classe d'automates introduite dans [DL06a] est la contrepartie avec gain sur les compteurs des automates à compteurs avec perte (une classe particulière d'automates communicants à perte [Sch02]). Dans ces automates, un compteur peut augmenter à tout moment.

3.5.3 Décidabilité sans être récursif primitif

Les résultats d'indécidabilité présentés dans la section 3.5.2 ne laissent que peu d'espoir sur l'usage de l'opérateur "freeze" dans des formalismes admettant des procédures de décision efficaces. En effet, les résultats d'indécidabilité sont établis pour des systèmes de contraintes très simples. Par contre, si on s'intéresse aux modèles finis et si on se restreint à une variable rigide, on obtient la décidabilité avec une complexité conséquente.

Théorème 3.5.3. [DL06a] Le problème de satisfaisabilité pour $\mathrm{CLTL}^{\downarrow}(\mathbb{N},=)$ restreint à une variable rigide, une variable flexible et pour les modèles finis, est décidable mais non récursif primitif.

La preuve de décidabilité se divise en deux étapes.

1. Chaque formule de ce dernier fragment peut être réduit en un automate alternant à un registre équivalent [DL06a].
2. Le problème de vacuité pour cette classe d'automates reconnaisant des mots de données finis peut être réduit au problème de vacuité des automates à compteurs avec erreur d'incrémentation dans le cas fini [DL06a].

Le fait que le problème est non récursif primitif est montrée en deux étapes.

1. Le problème de vacuité des automates à compteurs avec erreur d'incrémentation dans le cas fini est non primitif récursif en adaptant la preuve de [Sch02].
2. Ce problème peut être réduit en espace logarithmique à la satisfaisabilité de $\mathrm{CLTL}^{\downarrow}(\mathbb{N}, =)$ réduit à une seule variable rigide.

On peut remarquer ici que nous avons introduit une classe d'automates à registres alternants dans [DL06a] qui nous ont permis de caractériser la complexité de logiques temporelles avec l'opérateur "freeze". C'est l'objet de la prochaine section.

3.5.4 Problème de vacuité pour les automates à registres

Avant de définir les automates à registres, nous avons besoin de spécifier les structures qu'ils reconnaissent. En effet, ce ne sont pas exactement les modèles de $\mathrm{CLTL}(\mathbb{N}, =)$.

Un mot de données sur un alphabet fini Σ est une chaîne non vide dans $\Sigma^{<\omega}$ ou Σ^{ω} munie d'une relation d'équivalence $\sim^{\sigma}$ sur les positions. Dans le cas infini, un mot de données auquel on aurait effacé les lettres de Σ est donc équivalent à un modèle de $\mathrm{CLTL}^{\downarrow}(\mathbb{N}, =)$ restreint à une unique variable flexible. On peut d'ailleurs facilement simuler l'existence d'un alphabet fini dans les modèles de $\mathrm{CLTL}^{\downarrow}(\mathbb{N}, =)$ à l'aide de variables flexibles supplémentaires. Ce type de modèles est apparu dans [Dav04, BMS^{+}06], voir aussi [Bou02, BPT03].

On note $|\sigma|$ la longueur de σ, $\sigma(i)$ la i^{eme} lettre de σ et $\Sigma^{<\omega}(\sim)$ [resp. $\Sigma^{\omega}(\sim)$] l'ensemble des mots de données finis [resp. infinis]. Une valuation de registres v est une fonction partielle finie de $\mathbb{N}\setminus\{0\}$ vers les indices de σ. En effet, les valeurs indéfinies des registres vont être les valeurs initiales des automates.

Soient Q un ensemble fini et $n \in \mathbb{N}$. L'ensemble $\mathrm{TR}(Q, n)$ des formules de transition construites avec Q et n est défini ci-dessous avec $r \in \{1, \ldots, n\}$ et $q \in Q$:

$$\begin{aligned}\varphi \;::=\; & \top \mid \bot \mid \uparrow_r \sim \mid \uparrow_r \not\sim \mid \texttt{beg} \mid \texttt{nbeg} \mid \texttt{end} \mid \texttt{nend} \mid \\ & \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \downarrow_r \varphi \mid \langle q, 1\rangle \mid \langle q, -1\rangle\end{aligned}$$

Une formule de transition est dite locale lorsqu'elle ne contient pas de sous-formule de la forme $\langle q, o\rangle$. Pour toute formule de transition locale φ, on note $\widetilde{\varphi}$ sa formule duale obtenue en remplaçant chaque formule atomique par sa négation et en interchangeant $\wedge$ et $\vee$. Par exemple, `beg` and `nbeg` sont duales, de même que

$\uparrow_r\sim$ et $\uparrow_r\not\sim$. Ainsi la formule locale `beg` [resp. `end`] teste si la tête de lecture est en début [resp. fin] de mot.

Un automate à registres $\mathcal{A}$ (alternant et bi-directionnel) est une structure de la forme $\langle \Sigma, Q, q_I, n, \delta, \rho\rangle$ telle que

- Σ est un alphabet fini,
- Q est un ensemble fini d'états,
- $q_I \in Q$ est l'état initial,
- $n \in \mathbb{N}$ est un nombre de registres (codé en unaire),
- $\delta : Q \times \Sigma \to \mathrm{TR}(Q, n)$ est la fonction de transition,
- $\rho : Q \to \mathbb{N}$ est une fonction de rang telle que si $\langle q', o\rangle$ est une sous-formule de $\delta(q, a)$, alors $\rho(q') \leq \rho(q)$.

Supposons que $\sigma \in \Sigma^{\leq\omega}(\sim)$. Afin de définir les calculs de $\mathcal{A}$ sur σ, nous définissons d'abord une configuration de $\mathcal{A}$ pour σ comme un triplet $\langle i, q, v\rangle$ où i est un indice de σ, q est un état de $\mathcal{A}$ et v est une valuation de registres pour σ.

Nous interprétons la fonction de transition δ à l'aide d'une relation de satisfaction $S \models^{\sigma,i}_v \varphi$ où S est un ensemble fini de configurations pour σ et φ est une formule de transition. Les cas des constantes Booléennes, des opérateurs Booléens et des clauses duales sont omis:

- $S \models^{\sigma,i}_v \uparrow_r\sim$ ssi $v(r) \sim^\sigma i$ et $v(r)$ est défini,
- $S \models^{\sigma,i}_v$ `beg` ssi $i = 0$,
- $S \models^{\sigma,i}_v \downarrow_r \varphi$ ssi $S \models^{\sigma,i}_{v[r\mapsto i]} \varphi$,
- $S \models^{\sigma,i}_v$ `end` ssi $i = |\sigma| - 1$,
- $S \models^{\sigma,i}_v \langle q, o\rangle$ ssi $\langle i + o, q, v\rangle \in S$.

Un calcul de $\mathcal{A}$ de longueur $0 < \kappa \leq \omega$ sur le mot de données σ est un graphe orienté acyclique G tel que G est constitué des éléments suivants:

- pour $0 \leq j < \kappa$, un ensemble fini $G(j)$ de configurations de $\mathcal{A}$ pour σ,
- pour j vérifiant $j + 1 < \kappa$, une relation binaire $\to_j$ entre $G(j)$ et $G(j + 1)$,

telle que:

(i) $G(0) = \{\langle 0, q_I, \emptyset\rangle\}$ où $\emptyset$ est la valuation de registre partout indéfinie,

(ii) pour chaque j avec $G(j) \neq \emptyset$, on a $j+1 < \kappa$, $G(j+1) = \bigcup_{\langle i,q,v\rangle \in G(j)} S^j_{\langle i,q,v\rangle}$ où chaque ensemble $S^j_{\langle i,q,v\rangle}$ est un ensemble minimal satisfaisant $S^j_{\langle i,q,v\rangle} \models^{\sigma,i}_v \delta(q, \sigma(i))$,

(iii) $\langle i, q, v\rangle \to_j \langle i', q', v'\rangle$ ssi $\langle i', q', v'\rangle \in S^j_{\langle i,q,v\rangle}$.

Pour tout chemin π d'un calcul, le rang ne peut pas croître strictement. Par conséquent, si π est infini, alors le rang va se stabiliser autour d'une valeur, notée $\rho(\pi)$. Un mot de données est accepté ssi il admet un calcul tel que pour chaque chemin infini π, $\rho(\pi)$ est pair.

Nos automates ont des similarités fortes avec ceux définis dans [BPT03, Bou02, LW05, OW05]. De plus, il diffère sur quelques points techniques avec les automates considérés dans [KF94, SI00, NSV04]. Ils ont été conçus dans [DL06a] pour répondre aux questions relatives aux logiques temporelles.

Un automate est uni-directionnel ssi il ne contient pas de sous-formules de transition de la forme $\langle q, -1\rangle$. Une formule de transition est nondéterministe ssi

chaque sous-formule de transition qui est une conjonction de formules non locales est de la forme $(\varphi \vee \varphi') \wedge (\widetilde{\varphi} \vee \varphi'')$ avec φ locale. Une formule de transition est universelle ssi chaque sous-formule de transition qui est une disjonction de formules non locales est de la forme $(\varphi \wedge \varphi') \vee (\widetilde{\varphi} \wedge \varphi'')$ avec φ locale. Une formule de transition est déterministe ssi elle est à la fois nondéterministe et universelle. Un automate à registres est nondéterministe [resp. universel, déterministe] ssi chaque formule de transition est nondéterministe [resp. universelle, déterministe]

Les classes des automates à registres sont notées $dC\text{RA}(\sim)$ avec $d \in \{1, 2\}$ et $C \in \{\text{A}, \text{N}, \text{U}, \text{D}\}$ spécifiant les restrictions de direction ou de contrôle. On note aussi $dC\text{RA}_n(\sim)$ la sous-classe avec n registres.

Dans la figure 3.2, nous résumons les résultats de complexité concernant le problème de vacuité pour diverses classes d'automates à registres. Ces résultats ont été établis dans [DL06b]. La figure de gauche traite du cas fini (acceptation de mots de données finis) tandis que la figure de droite traite du cas infini. L'échelle centrale de complexité indique dans quelle classe de complexité appartiennent les divers problèmes de vacuité définis à partir de restriction. Pour les problèmes correspondant à $\mathsf{D} \setminus \mathsf{RP}$, le problème de vacuité est décidable mais pas récursif primitif. Les autres problèmes sont complets par rapport à la classe associée sur l'échelle. Une flêche entre deux classes indiquent simplement que la classe inférieure est une sous-classe syntaxique de la classe supérieure. Finalement, n est un entier positif.

3.5.5 Autres cas de décidabilité

Le théorème 3.4.5 a sa contrepartie en présence de l'opérateur "freeze" au prix d'un coût algorithmique plus élevé.

Théorème 3.5.4. [DLN07] Soit $\mathcal{D}$ un système de contraintes fini. Le problème de satisfaisabilité pour $\text{CLTL}^{\downarrow}(\mathcal{D})$ est décidable en espace exponentiel.

La preuve consiste à réduire en temps exponentiel $\text{CLTL}^{\downarrow}(\mathcal{D})$ à $\text{CLTL}(\mathcal{D}')$ où $\mathcal{D}'$ est un système fini auxiliaire et à utiliser alors le théorème 3.4.5. On peut aussi se demander si ce coût exponentiel en espace est optimal. Après tout, dans le cas fini, le problème de satisfaisabilité pour $\text{CLTL}(\mathcal{D})$ est dans PSPACE. En fait, l'optimalité est atteinte.

Théorème 3.5.5. [DLN07] Soit $\mathcal{D}$ un système de contraintes muni de l'égalité et ayant au moins deux éléments. Le problème de satisfaisabilité pour $\text{CLTL}^{\downarrow}(\mathcal{D})$ est EXPSPACE-difficile.

Le problème de satisfaisabilité pour $\text{CLTL}^{\downarrow}(\mathcal{D})$ restreint aux formules plates peut être aussi montré décidable. Dans le reste de cette section, $\mathcal{D}$ n'est pas nécessairement fini. Des fragments plats de LTL ou de variantes ont été étudiés dans [Dam99, CC00, ID01] (voir aussi le chapitre 2) et la définition de platitude tient compte de la polarité des sous-formules gouvernées par l'opérateur `U`.

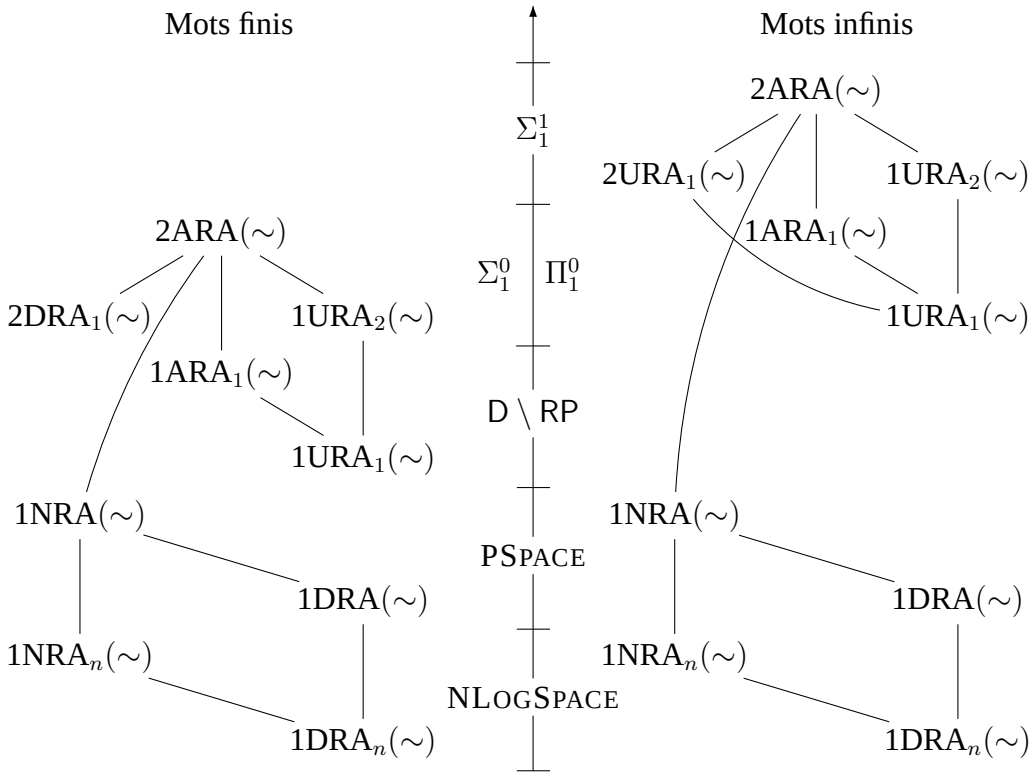


FIG. 3.2 – *Complexité du problème de vacuité pour diverses classes d'automates à registres*

Le fragment plat de $\mathrm{CLTL}^{\downarrow}(\mathcal{D})$ est formé des formules dont pour chaque occurrence positive [resp. négative] d'une sous-formule de la forme $\phi_1 \mathtt{U} \phi_2$, l'opérateur $\downarrow$ n'apparaît pas dans ϕ_1 [resp. pas dans ϕ_2]. Ce concept de platitude contraint l'interaction entre les opérateurs temporels et l'opérateur "freeze", une approche aussi suivie pour des formalismes voisins [BH96, CC00, Mar02, tCF05]. Pour saisir rapidement en quoi les formules plates sont plus faciles à analyser, on peut remarquer que dans une formule plate comme $\downarrow_{\mathtt{y}=\mathtt{x}} \mathtt{F}\phi$, seule la valeur $\mathtt{x}$ a besoin d'être stockée. Par contre, dans une formule non plate comme $\mathtt{G} \downarrow_{\mathtt{y}=\mathtt{x}} \phi$, il est nécessaire de stocker autant de valeurs de $\mathtt{x}$ qu'il y a de positions ultérieures.

Nous supposons que les variables flexibles de $\mathrm{CLTL}^{\downarrow}(\mathcal{D})$ sont $\{\mathtt{x}_0, \mathtt{x}_1, \ldots\}$ et les variables rigides sont $\{\mathtt{y}_0, \mathtt{y}_1, \ldots\}$. Pour simplifier la présentation, nous supposons que l'ensemble des variables flexibles de $\mathrm{CLTL}(\mathcal{D})$ est composé de deux ensembles disjoints: $\{\mathtt{x}_0, \mathtt{x}_1, \ldots\}$ et $\{\mathtt{y}_0^{\text{new}}, \mathtt{y}_1^{\text{new}}, \ldots\}$. Nous définissons une fonction de traduction t du fragment plat de $\mathrm{CLTL}^{\downarrow}(\mathcal{D})$ vers $\mathrm{CLTL}(\mathcal{D})$: t remplace chaque variable $\mathtt{y}_j$ par $\mathtt{y}_j^{\text{new}}$ dans les formules atomiques, est homomorphique pour les opérateurs Booléens et temporels et

$$t(\downarrow_{\mathtt{y}=\mathtt{X}^n\mathtt{x}} \psi) \stackrel{\text{def}}{=} \mathtt{y}^{\text{new}} = \mathtt{X}^n\mathtt{x} \wedge \mathtt{G}(\mathtt{y}^{\text{new}} = \mathtt{X}\mathtt{y}^{\text{new}}) \wedge t(\psi)$$

On peut montrer que $t(\phi)$ se calcule en espace logarithmique en la taille de ϕ.

Lemme 3.5.6. Soit $\mathcal{D}$ un système de contraintes muni de l'égalité. Pour chaque formule plate ϕ de $\mathrm{CLTL}^{\downarrow}(\mathcal{D})$, ϕ est satisfaisable ssi $t(\phi)$ est satisfaisable.

Nous obtenons donc les corollaires suivants.

Théorème 3.5.7. [DLN07] Les fragments plats de $\mathrm{CLTL}^{\downarrow}(\mathbb{Z}, <, =)$, $\mathrm{CLTL}^{\downarrow}(\mathbb{N}, <, =)$, $\mathrm{CLTL}^{\downarrow}(\mathbb{R}, <, =)$, et $\mathrm{CLTL}^{\downarrow}(\mathcal{D})$ avec $\mathcal{D}$ fini sont PSPACE-complets.

En dépit de la réduction de la partie plate de $\mathrm{CLTL}^{\downarrow}(\mathbb{N}, =)$ vers $\mathrm{CLTL}(\mathbb{N}, =)$ préservant la satisfaisabilité, certaines formules plates de $\mathrm{CLTL}^{\downarrow}(\mathbb{N}, =)$ n'ont pas d'équivalent dans $\mathrm{CLTL}(\mathbb{N}, =)$, voir les détails dans [DLN07].

Par ailleurs, on a pu aussi montrer le résultat suivant.

Théorème 3.5.8. [Dem06b] Le problème de satisfaisabilité pour $\mathrm{CLTL}^{\downarrow}(\mathrm{IPC}^{+})$ est EXPSPACE-complet.

Par contre, $\mathrm{CLTL}^{\downarrow}(\mathrm{IPC}^{++})$ est indécidable ce qui est une conséquence du théorème 3.5.1 et le problème de satisfaisabilité pour $\mathrm{CLTL}(\mathrm{IPC}^{+})$ est seulement PSPACE-complet.

Un mécanisme de quantification alternatif est tout simplement la quantification existentielle. Pour définir l'extension $\mathrm{CLTL}^{\exists}(\mathcal{D})$, les catégories syntaxiques sont identiques à celles de $\mathrm{CLTL}^{\downarrow}(\mathcal{D})$ sauf que nous ajoutons la clause $\exists\, \mathtt{y}\ \phi$ avec $\mathtt{y} \in \mathrm{VAR}_r$ (il n'y a plus d'opérateur "freeze"). Les logiques $\mathrm{CLTL}^{\exists}(\mathcal{D})$ et $\mathrm{CLTL}^{\downarrow}(\mathcal{D})$ possèdent les mêmes modèles et la relation de satisfaction est indicée par un environnement $e : \mathrm{VAR}_r \to D$. La relation $\models_e$ a la définition suivante:

$\sigma, i \models_e \exists\, \mathtt{y}\ \phi \stackrel{\text{def}}{\Leftrightarrow}$ il existe $a \in D$ tel que $\sigma, i \models_{e'} \phi$ où e' diffère avec e avec éventuellement la valeur pour $\mathtt{y}$: $e'(\mathtt{y}) = a$.

	LTL/PLTL	LTL/PLTL + $\downarrow$	LTL/PLTL + $\exists$
$\{\mathsf{x} < \mathsf{y}, \mathsf{x} = \mathsf{y}\}$	PSPACE [DD07]	Σ_1^1 [Dem04, section 7] (avec passé) [DLN07, LP05] (sans passé)	Σ_1^1
$\{\mathsf{x} - \mathsf{y} = c,\ \mathsf{x} = c\}$	Σ_1^1 [CC00]	Σ_1^1	Σ_1^1
$\mathrm{IPC} + \{\mathsf{x} < \mathsf{y}, \mathsf{x} = \mathsf{y}\}$	PSPACE [DG05]	Σ_1^1	Σ_1^1
IPC^{+}	PSPACE [Dem06b]	EXPSPACE [Dem06b]	EXPSPACE [Dem06b]
IPC^{++}	PSPACE [Dem06b]	Σ_1^1 [DLN07, LP05]	Σ_1^1

FIG. 3.3 – *Complexité de LTL avec contraintes de périodicité*

Lorsque $\mathcal{D}$ est muni de l'égalité, $\mathrm{CLTL}^{\downarrow}(\mathcal{D})$ peut être vu comme un fragment syntaxique de $\mathrm{CLTL}^{\exists}(\mathcal{D})$ mais ce n'est pas toujours le cas comme avec IPC^{+}. En effet, la formule $\downarrow_{\mathsf{y}=\mathsf{x}} \phi$ est alors équivalente à $\exists\, \mathsf{y}\ \mathsf{x} = \mathsf{y} \wedge \phi$.

Théorème 3.5.9. [Dem06b] Le problème de satisfaisabilité pour $\mathrm{CLTL}^{\exists}(\mathrm{IPC}^{+})$ est EXPSPACE-complet.

Un tableau récapitulatif concernant LTL avec contraintes de périodicité est présenté dans la figure 3.3. PLTL est l'extension de LTL avec opérateurs de passé $\mathtt{X}^{-1}$ ("previous") et $\mathtt{U}^{-1}$ ("since"). Chaque problème est complet pour la classe associée. A l'intersection d'une ligne L et d'une colonne $\mathcal{L}/\mathcal{L}'$, nous donnons la complexité de SAT($\mathcal{L}(\mathrm{L})$) et SAT($\mathcal{L}'(\mathrm{L})$), respectivement. Comme il n'y aura pas de différence entre LTL et PLTL, une unique information sera présentée.

D'autres fragments décidables de $\mathrm{CLTL}^{\downarrow}(\mathbb{N}, =)$ sont établis dans [Laz06, DL06a, BMS^{+}06, DDG07]. Par exemple, le fragment simple de $\mathrm{CLTL}^{\downarrow}(\mathbb{N}, =)$ avec des opérateurs de passé est restreint aux formules avec une variable flexible x, une variable rigide y telles que

- les opérateurs temporels sont parmi $\mathtt{X}$, $\mathtt{X}^{-1}$, $\mathtt{XXF}$ and $\mathtt{X}^{-1}\mathtt{X}^{-1}\mathtt{F}^{-1}$,
- chaque occurrence de ces opérateurs est immédiatement précédé de $\downarrow_{\mathsf{y}=\mathsf{x}}$,
- les seuls termes dans les formules atomiques sont x et y.

Ce fragment simple est en fait équivalent à FO$^2(\sim, <, +1)$ [BMS^{+}06, DL06a] et par conséquent est décidable en utilisant les résultats de [BMS^{+}06].

Théorème 3.5.10. [DL06a] Le problème de satisfaisabilité pour le fragment simple de $\mathrm{CLTL}^{\downarrow}(\mathbb{N}, =)$ avec opérateurs de passé est décidable.

La complexité du problème de l'accessibilité dans les réseaux de Petri est aussi difficile que la question suivante [BMS$^+$06, DL06a].

Problème ouvert 14. Quelle est la complexité du problème de satisfaisabilité pour le fragment simple de $\mathrm{CLTL}^{\downarrow}(\mathbb{N},=)$ avec opérateurs de passé? ◯

Pour finir cette section, nous présentons une dernière extension de $\mathrm{CLTL}(\mathcal{D})$, qui s'avère en général moins expressive que $\mathrm{CLTL}^{\downarrow}(\mathcal{D})$. En effet, la seule utilisation de l'opérateur "freeze" est faite pour spécifier qu'une valeur va se répéter dans le futur. Pour définir l'extension $\mathrm{CLTL}^{\diamond}(\mathcal{D})$ de $\mathrm{CLTL}(\mathcal{D})$, les catégories syntaxiques sont identiques à celles de $\mathrm{CLTL}(\mathcal{D})$ sauf que nous ajoutons les formules atomiques de la forme $\mathsf{x} = \diamond\mathsf{y}$. La relation de satisfaction est simplement étendue par $\sigma, i \models \mathsf{x} = \diamond\mathsf{y}$ ssi il existe $j > 0$ tel que $\sigma(i)(\mathsf{x}) = \sigma(i+j)(\mathsf{y})$. Ainsi $\mathsf{x} = \diamond\mathsf{y}$ est sémantiquement équivalente à une disjonction infinie $\bigvee_{j>0} \mathsf{x} = \mathtt{X}^j\mathsf{y}$ et ce type de contraintes est apparue dans [WZ00, section 7]. "$\mathsf{x} = \diamond\mathsf{y}$" est aussi équivalent à "$\downarrow_{\mathsf{z}=\mathsf{x}} \mathtt{XF}(\mathsf{x} = \mathsf{z})$".

Théorème 3.5.11. [DDG07] Le problème de satisfaisabilité pour $\mathrm{CLTL}^{\diamond}(\mathbb{N},=)$ est décidable.

La décidabilité est préservée en présence d'opérateurs de passé ou si l'on se restreint à des modèles finis [DDG07], ce qui permet de capturer de nombreuses propriétés de [LP05, section 3]. Comme la décidabilité est obtenue par réduction vers un problème qui consiste à vérifier des propriétés d'équité dans les réseaux de Petri [Jan90], la complexité n'est pas connue.

Problème ouvert 15. Quelle est la complexité de SAT($\mathrm{CLTL}^{\diamond}(\mathbb{N},=)$)? ◯

La fragment de SAT($\mathrm{CLTL}^{\diamond}(\mathbb{N},=)$) restreint à une variable est par ailleurs PSPACE-complet [DDG07]. Le théorème 3.5.11 peut aussi être décliné avec un problème de model-checking décidable mais la question en suspens qui nous semble la plus intéressante est pour quels systèmes $\mathcal{D}$ dont on a montré que $\mathrm{CLTL}(\mathcal{D})$ est décidable a-t-on que $\mathrm{CLTL}^{\diamond}(\mathcal{D})$ est aussi décidable? En particulier, la question suivante est ouverte.

Problème ouvert 16. Est-ce que SAT($\mathrm{CLTL}^{\diamond}(\mathbb{R},<,=)$) est décidable où on autorise des formules de la forme $\mathsf{x} < \diamond\mathsf{y}$? ◯

Une autre façon d'étendre le résultat de décidabilité pour SAT($\mathrm{CLTL}^{\diamond}(\mathbb{N},=)$) consiste à considérer une restriction analogue à celle dans [Mar02, tCF05].

Problème ouvert 17. Est-ce que SAT($\mathrm{CLTL}^{\downarrow}(\mathbb{N},=)$) restreint aux formules dont pour toutes les sous-formules de la forme $\downarrow_{\mathsf{y}=\mathtt{X}^i\mathsf{x}} \varphi$, la sous-formule φ ne contient ni d'occurrence négative de $\mathtt{U}$ ni d'occurrence positive dont le premier argument n'est pas propositionnel, est décidable? ◯

Chapitre 4

Contraintes de régularité

Ce chapitre est construit sur la base de [ADdR01, ADdR03, Dem05, DL06d, DL06c].

Les contraintes de régularité dans les systèmes de transition ou plus généralement dans les graphes peuvent s'exprimer dans divers formalismes comme la logique monadique du second ordre [Cau03], la logique dynamique propositionnel [Pra79, FL79] ou encore dans le μ-calcul modal. Dans ce chapitre, nous nous intéressons à plusieurs formalismes logiques qui permettent d'exprimer des contraintes de régularité sur des structures obtenues de diverses façons (documents XML, la toile, politiques d'action).

Une contrainte de régularité exprime simplement qu'une séquence appartient à un langage régulier donné. Dans la logique PDL, on peut directement exprimer l'existence d'un chemin étiqueté par un mot fini d'une langage régulier. Les contraintes de chemin pour les données semi-structurées utilisent cette forme de régularité [AV99]. En présence de structures arborescentes dont les noeuds fils sont ordonnés, comme dans la représentation de documents XML ("eXtended Markup Language"), une contrainte de régularité exprime une contrainte sur l'ordre de noeuds fils [SSMH04, ZL06]. Finalement, dans ce chapitre, nous considérons les contraintes de chemins qui correspondent à des contraintes sur les séquences d'actions autorisées pour des formalismes qui spécifient des politiques au sens des logiques déontiques [vdM96, PW04].

4.1 Contraintes de chemin et données semi-structurées

4.1.1 Classes de contraintes de chemin

Les données semi-structurées regroupent communément des données moins structurées que celles que l'on trouve dans les bases de données relationnelles. Par exemple, le schéma n'est pas fixé et la structure du document est supposée aller de soi. C'est un terme générique qui regroupe aussi bien les documents XML que les pages Web. Les données semi-structurées sont souvent représentées comme

```
<bibliography name="HDR-D">
<book>
<title> Modal Logic </title>
<author> Blackburn </author>
<author> de Rijke </author>
<author> Venema </author>
<publisher> Cambridge University Press </publisher>
<year> 2001 </year>
</book>
<book language = "french">
<title> 1984 </title>
<author> Orwell </author>
<publisher> Gallimard </publisher>
<year> 1950 </year>
</book>
</bibliography>
```

FIG. 4.1 – *Un document XML*

des graphes dont les noeuds et les arêtes sont étiquetés. Il existe en fait deux familles distinctes de représentation. Les pages Web peuvent être représentées comme des graphes dont les noeuds correspondent aux URLs et les étiquettes aux liens hypertextes [ABS00]. On pourra par ailleurs supposer qu'il y a une ou plusieurs racines à partir desquelles toutes les pages sont accessibles. Cette vue, probablement assez simpliste, a été enrichie et des modèles plus riches ont été étudiés dans [dA01]. Ainsi, dans la figure 4.1 un document XML est présenté tandis que la figure 4.2 contient une présentation graphique possible.

De cette section, étant donné un ensemble d'étiquettes Σ (fini ou infini), une Σ-structure est un tuple $G = \langle S, rt, (R_a)_{a\in\Sigma}\rangle$ où

- S est un ensemble non vide de noeuds,
- rt est un noeud distingué de S (la racine),
- $(R_a)_{a\in\Sigma}$ est une famille de relations binaires sur S.

Une Σ-structure est un objet un peu plus riche que les systèmes de transition de la section 2.4.2. En effet, la racine est un élément distingué d'une telle structure. On suppose implicitement que tous les noeuds de S sont accessibles à partir de la racine. Ainsi l'énoncé des problèmes relatifs à ce type de modèles mettra en avant que seuls les noeuds de S accessibles de rt sont pertinents. Cette hypothèse prend sa source dans le souhait de ne considérer que les pages accessibles d'une page courante (la racine). Comme d'habitude, G est déterministe lorsque pour $a \in \Sigma$ et pour $x \in S$, il y a au plus un noeud $y \in S$ tel que $x \xrightarrow{a} y$ est dans G, c'est-à-dire que $\langle x, y\rangle \in R_a$.

Une expression de chemin p est définie avec la grammaire suivante:

$$p ::= a \mid \sharp \mid \epsilon \mid p;p \mid p^* \mid p \cup p,$$

où $a \in \Sigma$. ϵ est interprétée comme la chaîne vide et $\sharp$ est le symbole joker puisqu'il

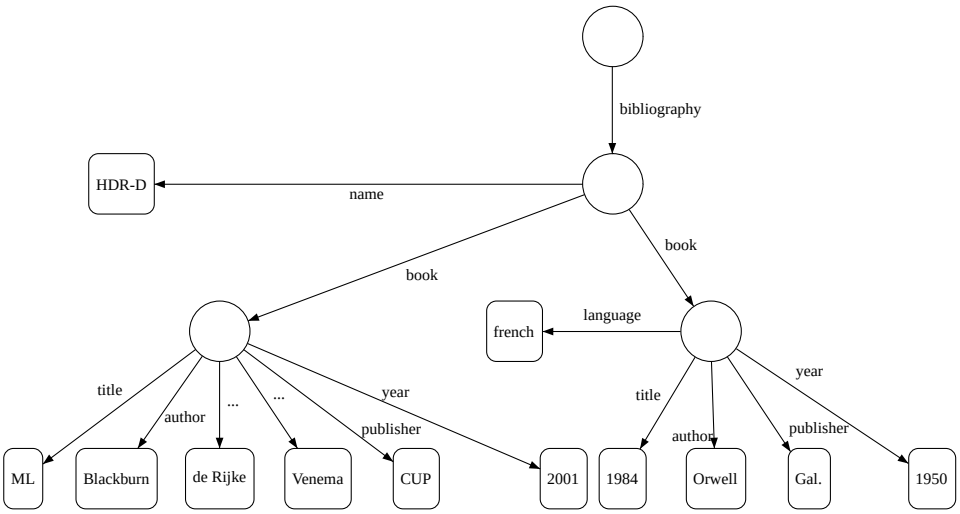


FIG. 4.2 – *Représentation graphique du document de la figure 4.1*

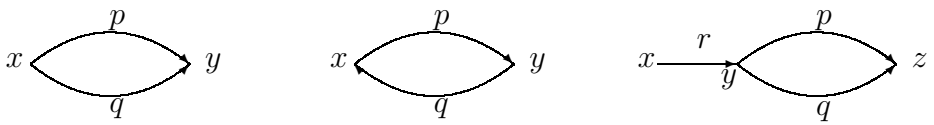
est interprété comme l'union de toutes les relations. Une expression de chemin est donc une expression régulière. Un mot est une expression de chemin n'utilisant que les lettres de Σ, ";" ou ϵ.

Etant donnée une Σ-structure G, chaque expression de chemin p est interprétée comme un sous-ensemble $[\![p]\!]$ de $S \times S$ avec la définition suivante:

$$
\begin{array}{ll}
[\![a]\!] & = R_a \\
[\![p^*]\!] & = \text{la fermeture réflexive et transitive de } [\![p]\!] \\
[\![\sharp]\!] & = \bigcup_{a \in \Sigma} R_a \\
[\![\epsilon]\!] & = \{\langle x, x\rangle : x \in S\} \\
[\![p; p']\!] & = [\![p]\!] \circ [\![p']\!] \text{ (composée)} \\
[\![p \cup p']\!] & = [\![p]\!] \cup [\![p']\!] \text{ (union)}
\end{array}
$$

En l'absence d'information sur le format des données, l'évaluation de requêtes avec des expressions régulières sur des Σ-structures peut être très inefficace. Ainsi, l'optimisation de requête nécessite la connaissance de certaines propriétés sur les données, des illustrations peuvent être trouvées dans [BFW98a]. Une contrainte de chemin du type "tout noeud accessible de la racine avec un chemin étiqueté par un mot de $\mathrm{L}(p)$ est accessible de la racine avec un chemin étiqueté par un mot de $\mathrm{L}(p')$" peut être une telle connaissance. Cette contrainte exprime simplement l'inclusion suivante:

$$\{x : \langle rt, x\rangle \in [\![p]\!]\} \subseteq \{x : \langle rt, x\rangle \in [\![p']\!]\}.$$



(a) Contrainte en avant (b) Contrainte en arrière (c) Contrainte en sucette

FIG. 4.3 – *Contraintes*

A cet effet, des contraintes d'inclusion de chemins ont été introduites dan [AV99].

Définition 4.1.1. Soient p, q des expressions de chemin. Une contrainte de chemin en avant (aussi appelée "contrainte d'inclusion") est une expression de la forme $p \subseteq_f q$. La Σ-structure G vérifie la contrainte $p \subseteq_f p' \stackrel{\text{def}}{\Leftrightarrow} \{x : \langle rt, x\rangle \in [\![p]\!]\} \subseteq \{x : \langle rt, x\rangle \in [\![q]\!]\}$. ∇

Les contraintes de chemin en arrière ont été aussi introduites dans [BFW98b].

Définition 4.1.2. Soient p, q des expressions de chemins. Une contrainte de chemin en arrière est une expression de la forme $p \subseteq_b q$. La Σ-structure G vérifie la contrainte $p \subseteq_b p' \stackrel{\text{def}}{\Leftrightarrow} \{x : \langle rt, x\rangle \in [\![p]\!]\} \subseteq \{x : \langle x, rt\rangle \in [\![q]\!]\}$. ∇

Si on admet que Σ contient des étiquettes en avant et des étiquettes en arrière, alors on peut aussi exprimer dans ce cadre plus général, une contrainte en arrière par une contrainte en avant avec la version étendue des étiquettes. Dans la suite, une contrainte de chemin $p \subseteq p'$ sans autre précision est entendue comme une contrainte en avant ou comme une contrainte en arrière.

Définition 4.1.3. Soient p, q, r des expressions de chemin. Une contrainte de chemin en sucette est une expression de la forme $r \rightsquigarrow p \subseteq q$. La Σ-structure G vérifie la contrainte $r \rightsquigarrow p \subseteq q \stackrel{\text{def}}{\Leftrightarrow}$ pour tous les noeuds $x \in [\![r]\!](rt)$, $\langle S, x, (R_a)_{a\in\Sigma}\rangle \models p \subseteq q$. ∇

Une illustration des divers types de contraintes est présentée dans la figure 4.3. On note $G \models c$ lorsque la structure G vérifie la contrainte c d'un des trois types. Maintenant que nous avons défini les contraintes de chemin, nous allons introduire le problème d'évaluation de requêtes et le problème d'implication.

Le problème d'évaluation de requêtes pour la classe C de contraintes de chemins est défini de la façon suivante:

entrée: une Σ-structure finie G et une contrainte c de C,
question: A-t-on $G \models c$?

Un problème plus difficile est celui de l'implication pour la classe C.

entrée: des constraintes $c_1, \ldots, c_{n+1}$ de C ($n \geq 0$),
question: est-ce que pour toutes les Σ-structures G, si $G \models c_1$ et $\cdots$ et $G \models c_n$ alors $G \models c_{n+1}$? (noté $c_1, \ldots, c_n \rightarrow c_{n+1}$).

Ces problèmes admettent des variantes, par exemple selon que l'on considère des sous-classes de Σ-structures. Pour certaines classes de contraintes et problèmes, le problème de l'implication a une complexité faible, ce qui est rappelé ci-dessous.

Théorème 4.1.1.

(I) Le problème de l'implication pour la classe des contraintes en avant restreint aux mots est dans PTIME [AV97].

(II) Le problème de l'implication pour la classe des contraintes en sucette restreint aux contraintes $r \rightsquigarrow p \subseteq q$ avec r,p et q des mots, et aux Σ-structures déterministes est décidable en espace linéaire [BFW98a, proposition 4.1].

(I) Le problème de l'implication pour la classe des contraintes en avant est dans EXPSPACE [AV97].

Par contre, l'indécidablité n'est pas très loin.

Théorème 4.1.2.

(I) Le problème de l'implication pour la classe des contraintes en sucette restreint aux expressions de chemins faites de mots est indécidable [BFW00, théorème 3.1].

(II) Le problème de l'implication pour la classe des contraintes en sucette restreint aux Σ-structures déterministes est indécidable même si Σ ne contient que deux étiquettes [BFW98a, théorème 6.1].

4.1.2 Logique des chemins $\mathrm{PDL}^{\mathrm{path}}$

L'approche défendue dans ce travail consiste à exprimer les problèmes de contraintes de chemin dans des logiques modales proches de la logique dynamique propositionnelle PDL [Pra79, FL79]. Après tout, les termes de programme dans PDL sans l'opérateur de test sont des expressions de chemin privées de ϵ et de $\sharp$. L'usage de logiques modales pour l'analyse de problèmes relatifs aux données semi-structurées n'est cependant pas nouveau. Dès 1997, dans [Ale97] la subsomption de schémas est résolue dans une logique modale hybride. Le terme "hybride" fait référence à la présence de nominaux et à celle de l'opérateur "freeze". Plus abstraitement, les logiques hybrides sont des formalismes logiques qui sont à mi-chemin entre des logiques modales et la logique classique, d'où leur caractère hybride [Hen90, Bla00b]. Ce même problème est codé dans une logique terminologique dans [CGL98]. Dans la même veine, l'équivalence de DTDs ("Document Type Definitions") pour des documents XML est résolue dans une logique terminologique à la PDL avec la possibilité d'avoir des contraintes sur le nombre de fils et un opérateur testant si une relation est bien fondée [CGL99]. Cette approche a aussi été suivie dans [BCT04, Thi04] où des problèmes sur des DTDs avec typage de références sont traduits vers une logique modale hybride, voir aussi [BC06]. L'usage de logique modale hybride est aussi préconisé dans [FdR06].

Nous étendons légèrement la notion d'expression de chemin pour tenir compte des transitions inverses. L'ensemble des expressions de chemin étendues admet l'opérateur $^{-1}$ tel que $[\![p^{-1}]\!]$ est la relation inverse de $[\![p]\!]$.

Les formules de la logique $\mathrm{PDL}^{\mathrm{path}}$ sont définies par la grammaire ci-dessous:

$$\top \mid \bot \mid \texttt{root} \mid \neg\phi \mid \phi \wedge \phi \mid [p]\phi \mid \langle p\rangle\phi$$

où p est une expression de chemin étendue. La formule atomique `root` est un nominal, c'est-à-dire une variable propositionnelle vraie dans un unique état (la racine).

Les modèles de $\mathrm{PDL}^{\mathrm{path}}$ sont des Σ-structures et la relation de satisfaction $\models$ est définie inductivement (on omet les clauses pour les opérateurs Booléens):

– $G, x \models \texttt{root}$ ssi $x = rt$,
– $G, x \models [p]\phi$ ssi pour $y \in [\![p]\!](x)$, on a $G, y \models \phi$,
– $G, x \models \langle p\rangle\phi$ ssi il existe $y \in [\![p]\!](x)$ tel que $G, y \models \phi$.

Une formule ϕ est satisfaite dans le modèle G ssi $G, rt \models \phi$. Une formule est valide ssi elle est satisfaite dans tous les modèles. Ainsi, la notion de satisfaction est liée à la racine.

La logique $\mathrm{PDL}^{\mathrm{path}}$ est donc une variante de la logique dynamique propositionnelle PDL [Pra79, FL79] avec une unique variable propositionnelle interprétée par la racine. Cependant, il ne faut pas négliger la présence de l'expression de chemin $\sharp$ qui n'existe pas dans PDL. Quand Σ est infini, $\sharp$ est bien une union infinie ce qui contribue aussi à la spécifité de $\mathrm{PDL}^{\mathrm{path}}$. Cependant $\mathrm{PDL}^{\mathrm{path}}$ sans $\sharp$ peut être considérée comme un fragment de CPDL avec nominaux [dG95] ou comme un fragment du μ-calcul hybride [SV01]. CPDL est compris ici comme PDL avec l'opérateur inverse de programmes. De plus, comme la logique modale HML [HM85b], $\mathrm{PDL}^{\mathrm{path}}$ n'a pas de variables propositionnelles à part le nominal `root`.

Le problème de satisfaisabilité [resp. validité] consiste à déterminer si une formule est satisfaisable [resp. valide]. Le problème de model-checking pour $\mathrm{PDL}^{\mathrm{path}}$ consiste à vérifier si une formule est satisfaite dans une Σ-structure finie.

Nous concluons cette section em mettant en lumière ce qui rend notre approche par logique attrayante pour analyser les problèmes relatifs aux constraintes de chemin. A toute constrainte en avant $c = p \subseteq_f q$ [resp. en arrière $c = p \subseteq_b q$], on note φ_c la formule $[p]\langle q^{-1}\rangle\texttt{root}$ [resp. $[p]\langle q\rangle\texttt{root}$].

Lemme 4.1.3.

(I) Soient G une Σ-structure et une constrainte de chemin $c = p \subseteq q$. Alors $G \models c$ iff φ_c est satisfaite dans G.

(II) Soient $c_1, \ldots, c_{n+1}$ des contraintes de chemin. Alors $c_1, \ldots, c_n \rightarrow c_{n+1}$ ssi $\varphi_{c_1} \wedge \cdots \wedge \varphi_{c_n} \Rightarrow \varphi_{c_{n+1}}$ est valide.

La preuve du lemme 4.1.3 est en fait assez immédiate et il n'est pas possible d'obtenir une extension avec les contraintes de chemin en sucette. Comme on le verra dans la suite, $\mathrm{PDL}^{\mathrm{path}}$ est décidable alors le problème de l'implication pour les contraintes en sucette est indécidable.

Lemme 4.1.4. Soit C la classe des Σ-structures ou celle des Σ-structures déterministes.

(I) Le problème d'évaluation de requêtes pour les contraintes de chemin se réduit en espace logarithmique au problème de model-checking pour $\mathrm{PDL}^{\mathrm{path}}$.

(II) Le problème d'implication pour les contraintes en avant restreint aux Σ-structures dans C se réduit en espace logarithmique au problème de validité de $\mathrm{PDL}^{\mathrm{path}}$ restreint aux Σ-structures de C.

(III) Le problème d'implication pour les contraintes en arrière restreint aux Σ-structures dans C se réduit en espace logarithmique au problème de validité de $\mathrm{PDL}^{\mathrm{path}}$ restreint aux Σ-structures de C et sans l'usage de l'opérateur $^{-1}$.

4.1.3 Complexité des problèmes logiques

Comme $\mathrm{PDL}^{\mathrm{path}}$ est assez proche de PDL, il n'est pas surprenant que ces logiques aient un problème de model-checking de complexité analogue.

Théorème 4.1.5. [ADdR03, théorème 9] Soient $G = \langle S, rt, (R_a)_{a\in\Sigma}\rangle$ une Σ-structure, $x \in S$ et ϕ une formule de $\mathrm{PDL}^{\mathrm{path}}$. Déterminer si $G, x \models \phi$ peut être calculer en temps $\mathcal{O}(|G| \times |\phi|)$.

La preuve du théorème 4.1.5 est par réduction en temps linéaire vers le problème de model-checking pour PDL. Ce problème peut être aussi résolu en temps $\mathcal{O}(|G|\times |\phi|)$ en utilisant que le problème de model-checking pour le fragment du μ-calcul modal sans alternation est de complexité temporelle bilinéaire [AC88, CS91].

Corollaire 4.1.6. [ADdR03] Le problème de model-checking pour $\mathrm{PDL}^{\mathrm{path}}$ est PTIME-complet.

La borne inférieure de complexité est une conséquence de la PTIME-dureté du problème de model-checking pour CTL restreint à $\forall\mathtt{X}$ et $\exists\mathtt{X}$ qui peut facilement se coder dans $\mathrm{PDL}^{\mathrm{path}}$. Avec au moins deux étiquettes dans Σ, on peut même se restreindre aux Σ-structures déterministes tout en préservant la PTIME-dureté.

Théorème 4.1.7. [ADdR03, théorème 9] Les problèmes de satisfaisabilité et validité pour $\mathrm{PDL}^{\mathrm{path}}$ sont dans EXPTIME.

La preuve est par réduction vers la logique CPDL avec nominaux dont la borne supérieure EXPTIME est établie dans [dG95, théorème 49]. Si Σ est fini, disons $\Sigma = \{a_1, \ldots, a_n\}$, alors il suffit de remplacer $\sharp$ par $a_1 \cup \cdots \cup a_n$. Si Σ est infini, une formule de $\mathrm{PDL}^{\mathrm{path}}$ avec les étiquettes $\{a_1, \ldots, a_n\}$ admet une traduction où $\sharp$ est remplacé par $a_1 \cup \cdots \cup a_n \cup a_{n+1}$. Dans le second cas, la réduction n'est pas correcte avec des Σ-structures déterministes.

Problème ouvert 18. Est-ce que le problème de satisfaisabilité pour $\mathrm{PDL}^{\mathrm{path}}$ restreint aux Σ-structures déterministes lorsque Σ est infini est décidable? ◯

Théorème 4.1.8. [ADdR03, théorème 12] Le problème de satisfaisabilité pour $\mathrm{PDL}^{\mathrm{path}}$ est EXPTIME-difficile dès que Σ possède au moins une étiquette.

PDL est connue pour être EXPTIME-difficile [FL79, Spa93a] mais $\mathrm{PDL}^{\mathrm{path}}$ ne contient pas de variables propositionnelles et seulement un unique nominal. La preuve est par réduction du problème de saitsfaisabilité globale pour la logique modale K qui est EXPTIME-difficile [CL94, Hem96] et cela en adaptant la technique du noeud espion de [BS95]. En effet, dans cette technique on construit un noeud qui peut accéder à presque tous les autres noeuds (le noeud espion), ce qui permet de simuler une quantification sur tous les noeuds en accédant d'abord à ce noeud espion, un concept bien mis en pratique en Océania. Une axiomatisation complète de $\mathrm{PDL}^{\mathrm{path}}$ est aussi présentée dans [AS06].

La logique temporelle minimale K_t [RU71] avec une modalité future $\mathtt{F}$ et une modalité passé $\mathtt{F}^{-1}$ interprétée sur des graphes quelconques a un problème de satisfaisabilité PSPACE-complet [Lad77, Spa93b]. Un corollaire de la preuve du théorème 4.1.8 est le suivant.

Corollaire 4.1.9. [ADdR03] K_t sans variable propositionnelle mais augmentée d'un unique nominal a un problème de satisfaisabilité EXPTIME-difficile.

Ce corollaire raffine un résultat analogue de [ABM99] pour lequel des variables propositionnelles sont présentes. Par ailleurs, cela complète les résultats pour des logiques régulières avec nominaux présentés dans le chapitre 5.

4.1.4 Complexité des problèmes de chemins

Les résultats de la section 4.1.3 ont pour conséquence que déterminer si $G \models p \subseteq q$ peut se calculer en temps $\mathcal{O}(|G| \times (|p| + |q|))$. On peut faire un peu mieux en utilisant le lemme ci-dessous.

Lemme 4.1.10. Le problème ci-dessous est dans NLOGSPACE.

entrée: une Σ-structure finie G, une expression de chemin p et deux noeuds x, y de G,

question: A-t-on $\langle x, y\rangle \in [\![p]\!]$?

Le lemme 4.1.10 améliore légèrement le résultat dans [MW95] qui établit une borne supérieure PTIME. Cependant, ce léger progrès nous permet d'établir le théorème suivant.

Théorème 4.1.11. [ADdR03] Les problèmes d'évaluation de requêtes pour les classes de contraintes en avant, de contraintes en arrière ou de contraintes de chemins sont NLOGSPACE-complets aussi bien dans le cas où les Σ-structures sont déterministes que dans le cas où elles sont nondéterministes.

Considérons à présent le problème d'implication.

Théorème 4.1.12. Le problème d'implication pour les contraintes en avant est dans EXPTIME et PSPACE-difficile dès que Σ a au moins deux étiquettes.

La borne supérieure en EXPTIME est une conséquence du lemme 4.1.3(II) et du théorème 4.1.7 tandis que la borne inférieure PSPACE est par réduction du problème d'inclusion de langages pour les expressions régulières [HRS76, théorème 2.12(c)]. La borne EXPTIME améliore la borne EXPSPACE de [AV97].

On peut de même obtenir un résultat analogue avec les contraintes en arrière.

Théorème 4.1.13. Le problème d'implication pour les contraintes en arrière est dans EXPTIME et PSPACE-difficile dès que Σ a au moins trois étiquettes.

Problème ouvert 19. Est-ce que le problème d'implication pour les contraintes en avant [resp. en arrière] est dans PSPACE? ◯

Un résultat plus récent mérite ici d'être rappeler.

Théorème 4.1.14. [ACD$^+$04, Deb05] Lorsque les structures sont multi-racines, le problème de l'implication restreint aux instances de la forme $c_1, \ldots, c_{n+1}$ telles que pour $i \in \{1, \ldots, n\}$, c_i est de la forme $p_i \subseteq_f q_i$ et q_i est un mot, est PSPACE-complet.

Ce résultat qui permet presque de répondre au problème ouvert 19 utilise des résultats puissants sur la théorie du premier ordre de la réécriture préfixe [DT90]. Notons cependant que dans [ACD$^+$04, Deb05], il est supposé que les structures sont multi-racines, c'est-à-dire qu'au lieu de considérer une unique racine, la structure admet éventuellement plus d'une racine. Dans [Deb05], de nombreux autres problèmes sont analysés comme le problème de la borne finie ou celui de l'équivalent fini.

Pour conclure cette section, intéressons-nous aux Σ-structures déterministes. Le problème d'implication pour les contraintes de chemin de la forme $p \rightsquigarrow p' \subseteq_f q$ restreint aux Σ-structures déterministes est indécidable [BFW98a]. Des restrictions décidables sont introduites dans ce même papier. Comme corollaire des résultats précédents, nous pouvons établir le résultat suivant mettant en avant un nouveau cas de décidabilité.

Lemme 4.1.15. [ADdR03] Le problème d'implication pour les contraintes en arrière restreint aux Σ-structures déterministes lorsque Σ est fini est dans EXPTIME.

Si Σ est infini la question est ouverte.

Problème ouvert 20. Est-ce que le problème d'implication pour les contraintes en arrière restreint aux Σ-structures déterministes est décidable lorsque Σ est infini? ◯

On n'en sait pas tellement plus avec les contraintes en avant.

Problème ouvert 21. Est-ce que le problème d'implication pour les contraintes en avant restreint aux Σ-structures déterministes est décidable? ◯

4.2 Contraintes de Presburger et de régularité

L'interrogation de documents XML a donné naissance à différents formalismes logiques ou à base d'automates qui permettent d'exprimer des contraintes de régularité ou des contraintes de Presburger [ZL06, SSMH04, BT05, OTTR05]. Ici, les documents XML sont représentés comme des arbres avec ordre sur les fils et sans arité maximale sur le nombre de fils. Une contrainte de Presburger exprime une relation entre des nombres de noeuds fils satisfaisant certaines propriétés alors qu'une contrainte de régularité exprime une contrainte sur la lecture des noeuds fils de gauche à droite. Par exemple, une logique avec un opérateur de point fixe, des contraintes de régularité et de Presburger a été introduite dans [SSMH04] et prouvée décidable en espace exponentiel. Un peu à la même période, la logique SL ("Sheaves Logic") a été montrée décidable dans [ZL03] avec une complexité non-élémentaire. Comme cela a été vu dans la section 4.1, la conception de logiques modales pour les données semi-structurées que ce soit avec des modèles sous forme d'arbres [Mar03b, SSMH04, ABD$^+$05] ou avec des modèles sous forme de graphes plus généraux [CGL99, ADdR03, BCT04] est une approche séduisante qui permet de réutiliser des techniques bien comprises.

Ce travail a pour but d'introduire une logique modale qui peut exprimer des contraintes de Presburger plus riches que celles dans les logiques modales graduées (voir par exemple [BC85, Tob00, PS04]) ou dans les logiques terminologiques (voir par exemple [HB91, HST00, CG05]) tout en admettant des contraintes de régularité comme dans [Wol83, ZL03, SSMH04]. On recherche cependant une complexité en espace polynomial afin de raffiner les résultats de décidabilité et de complexité de [Tob00, SSMH04, ZL06]. Une telle logique serait d'ailleurs bien plus expressive que la logique modale minimale K qui est déjà PSPACE-complète [Lad77]. Une complexité en espace polynomial exclut la présence d'opérateurs de point fixe dans toute leur généralité car le problème de satisfaisabilité pour le μ-calcul modal est EXPTIME-complet. De même, les contraintes de Presburger ne pourront pas être celles du premier ordre car alors le problème de satisfaisabilité pour ce fragment de l'arithmétique est déjà 2EXPTIME-difficile [FR74].

4.2.1 Une logique modale étendue: EXML

Etant donnés un ensemble infini dénombrable de variables propositionnelles $\mathrm{PROP} = \{\mathrm{p}_1, \mathrm{p}_2, \ldots\}$ et un ensemble infini dénombrable de symboles de relation $\Sigma = \{\mathsf{R}_1, \mathsf{R}_2, \ldots\}$, les formules de EXML sont définies par la grammaire ci-dessous:

$$\phi ::= \mathrm{p} \mid \neg\phi \mid \phi \wedge \phi \mid t \sim b \mid t \equiv_k c \mid \mathcal{A}(\mathsf{R}, \phi_1, \ldots, \phi_n)$$

$$t ::= a \times \sharp^{\mathsf{R}}\phi \mid t + a \times \sharp^{\mathsf{R}}\phi,$$

avec

– $\mathrm{p} \in \mathrm{PROP}$, $\mathsf{R} \in \Sigma$,

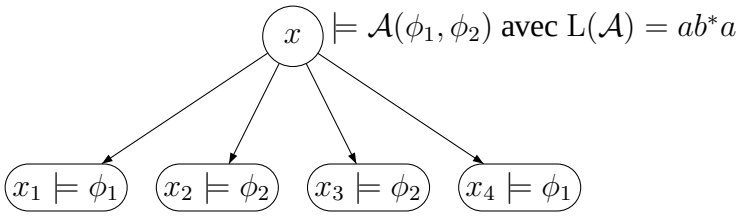


FIG. 4.4 – *Une contrainte de régularité*

- $b, k, c \in \mathbb{N}$, $a \in \mathbb{Z}$,
- $\sim \in \{<, >, =\}$,
- $\mathcal{A}$ est un automate non-déterministe avec un alphabet de n lettres linéairement ordonnées.

Dans cette section, le degré modal de ϕ, noté $\mathrm{md}(\phi)$, est le nombre maximal d'emboîtements du symbole $\sharp$ dans ϕ. Un terme de la forme $a_1 \times \sharp^{\mathsf{R}_1}\phi_1 + \ldots + a_m \times \sharp^{\mathsf{R}_m}\phi_m$ est abrégé en $\Sigma_i a_i \sharp^{\mathsf{R}_i}\phi_i$. Tous les entiers dans la suite sont codés avec une représentation binaire.

Un modèle de EXML est une structure $\mathcal{M} = \langle T, (R_{\mathsf{R}})_{\mathsf{R}\in\Sigma}, (<_x^{\mathsf{R}})_{x\in T}, V\rangle$ où

- T est un ensemble de noeuds éventuellement infini,
- $(R_{\mathsf{R}})_{\mathsf{R}\in\Sigma}$ est une famille de relations binaires sur T telle que pour $\mathsf{R} \in \Sigma$ et $x \in T$, l'ensemble $\{x' \in T : \langle x, x'\rangle \in R_{\mathsf{R}}\}$ est fini (branchement fini),
- chaque relation $<_x^{\mathsf{R}}$ est un ordre total sur les R_{R}-successeurs de x,
- $V : T \to \mathcal{P}(\mathrm{PROP})$ est un étiquetage des noeuds.

On écrit $R_{\mathsf{R}}(x) = x_1 < \ldots < x_\alpha$ lorsque

$$R_{\mathsf{R}}(x) \stackrel{\text{def}}{=} \{x' \in T : \langle x, x'\rangle \in R_{\mathsf{R}}\} = \{x_1, \ldots, x_\alpha\},$$

et $x_1 <_x^{\mathsf{R}} \ldots <_x^{\mathsf{R}} x_\alpha$. De même, étant donnée une relation binaire $R \subseteq T \times T$ à branchement fini, on note $R^\sharp(x)$ le cardinal de l'ensemble $\{x' \in T : \langle x, x'\rangle \in R\}$. La relation de satisfaction $\models$ est définie ainsi:

- $\mathcal{M}, x \models \mathrm{p}$ ssi $\mathrm{p} \in V(x)$,
- $\mathcal{M}, x \models \neg\phi$ ssi $\mathcal{M}, x \not\models \phi$,
- $\mathcal{M}, x \models \phi_1 \wedge \phi_2$ ssi $\mathcal{M}, x \models \phi_1$ et $\mathcal{M}, x \models \phi_2$,
- $\mathcal{M}, x \models \Sigma_i a_i \sharp^{\mathsf{R}_i}\phi_i \sim b$ ssi $\Sigma_i a_i R^\sharp_{\mathsf{R}_i,\phi_i}(x) \sim b$ avec $R_{\mathsf{R}_i,\phi_i} = \{\langle x', x''\rangle \in T \times T : \langle x', x''\rangle \in R_{\mathsf{R}_i}, \text{ et } \mathcal{M}, x'' \models \phi_i\}$,
- $\mathcal{M}, x \models \Sigma_i a_i \sharp^{\mathsf{R}_i}\phi_i \equiv_k c$ ssi il existe $n \in \mathbb{N}$ tel que $\Sigma_i a_i R^\sharp_{\mathsf{R}_i,\phi_i}(x) = nk + c$,
- $\mathcal{M}, x \models \mathcal{A}(\mathsf{R}, \phi_1, \ldots, \phi_n)$ ssi il existe $a_{i_1} \cdots a_{i_\alpha} \in \mathrm{L}(\mathcal{A})$ tel que $R_{\mathsf{R}}(x) = x_1 < \ldots < x_\alpha$ et pour $j \in \{1, \ldots, \alpha\}$, $\mathcal{M}, x_j \models \phi_{i_j}$.

La présence des opérateurs Booléens et la propriété d'élimination des quantificateurs dans l'arithmétique de Presburger assurent que toute l'expressivité de l'arithmétique de Presburger est présente dans EXML à défaut de sa concision.

La figure 4.4 illustre la sémantique des formules à base d'automates. L'usage des automates dans EXML est identique à celui dans ETL [Wol83] pour définir des opérateurs temporels à base d'automates (voir aussi la section 2.3). L'opérateur

modal $\Diamond$ (voir par exemple [BdRV01]) est défini par $\Diamond\phi \approx \sharp^{\mathsf{R}}\phi \geq 1$ et donc $\Box\phi \approx \sharp^{\mathsf{R}}\neg\phi = 0$. Une formule $\Diamond_{\geq n}\phi$ dans la logique modale graduée GML se définit par $\Diamond_{\geq n}\phi \approx \sharp^{\mathsf{R}}\phi \geq n$. Cependant, EXML est beaucoup plus expressive que GML. En effet, EXML peut exprimer facilement qu'il y a exactement deux fois plus de noeuds fils vérifiant p que de noeuds fils vérifiant q, ce qui peut s'écrire $\sharp^{\mathsf{R}}\mathrm{p} - 2\sharp^{\mathsf{R}}\mathrm{q} = 0$. De même, comme dans [PS04], on peut aussi exprimer que plus de la moitié des noeuds fils vérifie la proposition p: $2\sharp\mathrm{p} - \sharp\top > 0$.

Comme d'habitude, une formule est dite satisfaisable ssi il existe un modèle $\mathcal{M} = \langle T, (R_{\mathsf{R}})_{\mathsf{R}\in\Sigma}, (<^{\mathsf{R}}_x)_{x\in T}, V\rangle$ et $x \in T$ tels que $\mathcal{M}, x \models \phi$.

On note RML [resp. PML] le fragment de EXML sans contraintes de Presburger [resp. sans contraintes de régularité]. Le fragment PML est déjà plus expressive que les formalismes définis dans [Tob00, PS04, KRM05]. De même, pour $k \geq 0$, on note EXML_k le fragment de EXML dont la somme des cardinalités des alphabets des contraintes régulières est bornée par k. Il s'agit d'une variante de la notion introduite dans [DL06c] pour laquelle nous pourrons finalement borner la taille des solutions d'un système d'équations.

En utilisant la technique de dépliage des graphes et celle qui permet de passer d'un étiquetage des transitions vers un étiquetage des noeuds, on peut montrer le résultat suivant.

Lemme 4.2.1. [ADdR03] Il existe une réduction logarithmique en espace entre le problème de satisfaisabilité pour EXML et sa restriction aux modèles avec un unique symbole de relation interprété comme un arbre fini.

Dans la suite, on s'intéresse aux modèles satisfaisant cette restriction. Nous allons déterminer plusieurs fragments de EXML qui sont dans PSPACE.

4.2.2 Des formules aux systèmes d'équations

Nous définissons ci-dessous une notion de fermeture à la Fischer-Ladner [FL79]. Grossièrement, la fermeture $\mathrm{cl}(X)$ d'un ensemble X contient toutes les formules utiles pour déterminer si X est satisfaisable.

Définition 4.2.1. Soit X un ensemble fini. $\mathrm{cl}(X)$ est le plus petit ensemble de formules tel que

- $X \subseteq \mathrm{cl}(X)$,
- $\mathrm{cl}(X)$ est fermé par sous-formule,
- si $\psi \in \mathrm{cl}(X)$, alors $\neg\psi \in \mathrm{cl}(X)$ (on identifie $\neg\neg\psi$ avec ψ),
- si $t \sim b \in \mathrm{cl}(X)$, alors $t \sim' b \in \mathrm{cl}(X)$ pour chaque $\sim' \in \{<, >, =\}$,
- soit K le ppcm de toutes les constantes k apparaissant dans des sous-formules de la forme $t \equiv_k c$. Si $t \equiv_k c \in \mathrm{cl}(X)$, alors $t \equiv_K c' \in \mathrm{cl}(X)$ pour tous les $c' \in \{0, \ldots, K-1\}$.

∇

Un ensemble X est dit fermé ssi $\mathrm{cl}(X) = X$. A présent, nous raffinons le concept de fermeture en ajoutant un nouveau paramètre n: la distance du noeud

racine au noeud courant où les formules sont évaluées. Chaque ensemble $\mathrm{cl}(n, \phi)$ sera un sous-ensemble de $\mathrm{cl}(\{\phi\})$.

Définition 4.2.2. Soit ϕ une formule de EXML. Pour $n \in \mathbb{N}$, $\mathrm{cl}(n, \phi)$ est le plus petit ensemble tel que:

- $\mathrm{cl}(0, \phi) = \mathrm{cl}(\{\phi\})$,
- pour $n \in \mathbb{N}$, $\mathrm{cl}(n, \phi)$ est fermé,
- pour $n \in \mathbb{N}$ et $\sharp\psi$ apparaissant dans une formule de $\mathrm{cl}(n, \phi)$, $\psi \in \mathrm{cl}(n + 1, \phi)$,
- pour $n \in \mathbb{N}$ et $\mathcal{A}(\phi_1, \ldots, \phi_m) \in \mathrm{cl}(n, \phi)$, $\{\phi_1, \ldots, \phi_m\} \subseteq \mathrm{cl}(n + 1, \phi)$.

∇

Sans perte de généralité, on suppose dans la suite que pour chaque n tel que $\mathrm{cl}(n, \phi)$ est non-vide, le ppcm de toutes les constantes apparaissant dans $\mathrm{cl}(n, \phi)$ de la forme $t \equiv_k c$ est égale au ppcm de tous les k apparaissant dans ϕ. De plus, on supposera que $\equiv_K$ n'apparaît pas dans ϕ.

Nous introduisons ci-dessous une notion de consistance locale qui est finalement très classique quand on manipule ce type d'ensembles.

Définition 4.2.3. Un ensemble $X \subseteq \mathrm{cl}(n, \phi)$ est dit n-localement consistant ssi les conditions suivantes sont vérifiées:

- si $\neg\psi \in \mathrm{cl}(n, \phi)$, alors $\neg\psi \in X$ ssi $\psi \notin X$,
- si $\psi_1 \wedge \psi_2 \in \mathrm{cl}(n, \phi)$, alors $\psi_1 \wedge \psi_2 \in X$ ssi $\psi_1, \psi_2 \in X$,
- si $t \sim b \in \mathrm{cl}(n, X)$ alors il existe un unique $\sim' \in \{<, >, =\}$ tel que $t \sim' b \in X$,
- si $t \equiv_k c \in \mathrm{cl}(n, X)$, alors il existe un unique $c' \in \{0, \ldots, K - 1\}$ tel que $t \equiv_K c' \in X$,
- si $t \equiv_k c \in \mathrm{cl}(n, X)$, alors $\neg t \equiv_k c \in X$ ssi il existe $c' \in \{0, \ldots, K - 1\}$ tel que $t \equiv_K c' \in X$ et non $c' \equiv_k c$,
- si $t \sim b \in \mathrm{cl}(n, X)$ alors $\neg t \sim b \in X$ ssi il existe $\sim' \in \{<, >, =\} \setminus \{\sim\}$ tel que $t \sim' b \in X$.

∇

Pour une formule donnée, les ensembles localement consistants peuvent se coder en espace polynomial.

Lemme 4.2.2. Soient ϕ une formule et $n \in \mathbb{N}$.

(I) Chaque ensemble n-localement consistant est de cardinal au plus $2 \times |\phi|$ et peut être codé avec un espace mémoire de taille polynomiale en $|\phi|$.

(II) $\mathrm{cl}(|\phi|, \phi) = \emptyset$.

Un système d'équations $\mathcal{S}$ construit sur l'ensemble de variables $\{\mathsf{x}_1, \ldots, \mathsf{x}_n\}$ est une formule de Presburger construite sur $\{\mathsf{x}_1, \ldots, \mathsf{x}_n\}$ qui est une combinaison Booléenne de constraintes atomiques de la forme $\Sigma_j a_j \times \mathsf{x}_{i_j} = b$ avec $a_j \in \mathbb{Z}$ et $b \in \mathbb{N}$. Une solution positive de $\mathcal{S}$ est un élément de $\overline{\mathsf{x}} \in \mathbb{N}^n$ tel que $\overline{\mathsf{x}} \models \mathcal{S}$

est vérifié dans l'arithmétique de Presburger. Une conséquence de [Pap81] est le résultat suivant.

Lemme 4.2.3. Soit $\mathcal{S}$ un système d'équations construit sur $\{\mathsf{x}_1, \ldots, \mathsf{x}_n\}$. $\mathcal{S}$ a une solution positive ssi $\mathcal{S}$ a une solution positive dont tous les coefficients sont bornés par $(n + 2 \times m) \times (2 \times m + (a + 1))^{4m+1}$ où a est la valeur absolue maximale des constantes de $\mathcal{S}$ et m est le nombre de contraintes atomiques dans $\mathcal{S}$.

Si $\mathcal{S}$ admet une solution positive, alors on note M le coefficient $(n+2\times m)\times (2 \times m + (a + 1))^{4m+1}$.

Lemme 4.2.4. [DL06c] Soit ϕ une formule de EXML, X un ensemble n-localement consistant pour $n \leq |\phi|$ et N le nombre d'ensembles $n+1$-localement consistants.

(I) Il existe un système d'équations $\mathcal{S}$ sur les variables $\mathsf{x}_1, \ldots, \mathsf{x}_N$ tel que X est satisfaisable ssi $\mathcal{S}$ a un solution positive.

(II) Si ϕ appartient à EXML$_k$ ou à RML et $\mathcal{S}$ a une solution positive, alors M est exponentiel en $|\phi|$.

(III) Si ϕ appartient à EXML et $\mathcal{S}$ a une solution positive, alors M est doublement exponentiel en $|\phi|$.

La preuve de ce lemme est basée sur [SSMH04] où par exemple des contraintes de régularité sont transformées en système d'équations en utilisant les images de Parikh des langages réguliers. Cependant, contrairement à ce que nous avons cru dans [DL06d], je ne pense pas que [SSMH04] permet de conclure que M soit toujours simplement exponentiel en $|\phi|$ pour toute formule de EXML. Si c'était quand même le cas, la complexité de EXML serait alors complétement caractérisée.

4.2.3 Un algorithme à la Ladner

La fonction SAT définie ci-dessous va nous permettre de tester la satisfaisabilité d'une formule de EXML. Elle admet trois arguments: une formule ϕ dont on cherche à déterminer la satifaisabilité, un entier d et un sous-ensemble de $\mathrm{cl}(d, \phi)$. La figure 4.5 contient cet algorithme qui est un algorithme à la Ladner [Lad77] (voir d'autres exemples dans [Spa93b, Dem03]). En effet, c'est un algorithme direct qui n'utilise pas d'automates, pas de calculs de séquents ou de formalismes voisins. SAT est un algorithme non-déterministe et le graphe des appels récursifs induit un modèle arborescent pour la formule passée en argument.

Dans un premier temps, on peut caractériser la complexité de cet algorithme.

Lemme 4.2.5. Pour tous les ensembles 0-localement consistants X et pour tous les calculs de $\mathrm{SAT}(\phi, X, 0)$,

- la profondeur de la pile des appels récursifs est bornée par $|\phi|$,
- chaque appel nécessite un espace mémoire polynomial en
  - l'espace mémoire utile pour coder les ensembles 0-localement consistants,

`function` $\mathrm{SAT}(X, \phi, d)$

(consistance) Si X n'est pas d-localement consistant alors `abort`;

(cas de base) Si X ne contient que des variables propositionnelles alors retourner `true`;

(témoins)

(initialisation-compteurs) Pour chaque formule $\psi \in \mathrm{cl}(d+1, \phi)$ qui n'est pas de la forme $t \equiv_K c$, $C_\psi := 0$;

(initialisation-états) Pour chaque formule $\mathcal{A}(\psi_1, \ldots, \psi_\alpha) \in X$, choisir un état initial q_0 de $\mathcal{A}$ et $q_{\mathcal{A}(\psi_1,\ldots,\psi_\alpha)} := q_0$;

(initialisation-états-complément) Pour chaque formule $\neg\mathcal{A}(\psi_1, \ldots, \psi_\alpha) \in X$, $q_{\neg\mathcal{A}(\psi_1,\ldots,\psi_\alpha)} := I$ où I est l'ensemble des états initiaux de $\mathcal{A}$ et $b_{\neg\mathcal{A}(\psi_1,\ldots,\psi_\alpha)} := 0$;

(devine-nb-fils) Choisir nb dans $\{0, \ldots, nb(d+1) \times M\}$;

(devine-fils-de-gauche-a-droite) Pour $i = 1$ à nb faire

1. Choisir $x \in \{1, \ldots, nb(n+1)\}$;
2. $\mathrm{SAT}(Y_x, \phi, d+1)$;
3. Pour chaque formule $\psi \in \mathrm{cl}(d+1, \phi)$ qui n'est pas une contrainte de periodicité, si $\psi \in Y_x$ alors $C_\psi := C_\psi + 1$;
4. Pour chaque contrainte de régularité $\mathcal{A}(\psi_1, \ldots, \psi_\alpha) \in X$,
   (a) Choisir une transition $q_{\mathcal{A}(\psi_1,\ldots,\psi_\alpha)} \xrightarrow{a_i} q'$ de $\mathcal{A}$ avec $\Sigma_\mathcal{A} = a_1, \ldots, a_\alpha$;
   (b) Si $\psi_i \in Y_x$ alors $q_{\mathcal{A}(\psi_1,\ldots,\psi_\alpha)} := q'$ sinon `abort`;
5. Pour $\neg\mathcal{A}(\psi_1, \ldots, \psi_\alpha) \in X$, si $\neg\psi_1, \ldots, \neg\psi_\alpha \in X$ ou bien $b_{\neg\mathcal{A}(\psi_1,\ldots,\psi_\alpha)} = 1$ alors $b_{\neg\mathcal{A}(\psi_1,\ldots,\psi_\alpha)} := 1$ sinon
   (a) $q_{\neg\mathcal{A}(\psi_1,\ldots,\psi_\alpha)} := \{q : \exists\, q' \in q_{\neg\mathcal{A}(\psi_1,\ldots,\psi_\alpha)}, q' \xrightarrow{a_i} q, \psi_i \in Y_x\}$;

(vérification-finale)

1. Pour $\Sigma_i a_i \sharp \psi_i \sim b \in X$, si $\Sigma_i a_i \times C_{\psi_i} \sim b$ n'est pas vérifié alors `abort`,
2. Pour $\Sigma_i a_i \sharp \psi_i \equiv_k c \in X$, si $\Sigma_i a_i \times C_{\psi_i} \equiv_k c$ n'est pas vérifié alors `abort`,
3. Pour $\mathcal{A}(\psi_1, \ldots, \psi_\alpha) \in X$, si $q_{\mathcal{A}(\psi_1,\ldots,\psi_\alpha)}$ n'est pas un état final de $\mathcal{A}$, alors `abort`;
4. Pour $\neg\mathcal{A}(\psi_1, \ldots, \psi_\alpha) \in X$, si $q_{\neg\mathcal{A}(\psi_1,\ldots,\psi_\alpha)}$ contient un état final de $\mathcal{A}$ et $b_{\neg\mathcal{A}(\psi_1,\ldots,\psi_\alpha)} = 0$, alors `abort`;

(retourner-vrai) Retourner `true`.

FIG. 4.5 – *Algorithme SAT*

– le logarithme de M (taille des plus petites solutions des systèmes de contraintes).

Par conséquent, si ϕ appartient à EXML$_k$, PML ou RML seul un espace polynomial est nécessaire pour chaque calcul de $\mathrm{SAT}(\phi, X, 0)$ (cf. le lemme 4.2.4). La correction de l'algorithme, énoncée dans le lemme 4.2.6, est fastidieuse à montrer mais ne nécessite pas d'astuces particulières.

Lemme 4.2.6. ϕ est satisfaisable ssi il existe $X \subseteq \mathrm{cl}(0, \phi)$ tel que $\mathrm{SAT}(X, \phi, 0)$ a un calcul acceptant.

On peut donc obtenir les résultats de complexité suivants.

Théorème 4.2.7. [DL06c]

(I) Pour chaque $k \geq 1$, le problème de satisfaisabilité pour EXML$_k$ est PSPACE-complet.

(II) Les problèmes de satisfaisabilité pour RML et PML sont PSPACE-complets.

(III Le problème de satisfaisabilité pour EXML est dans EXPSPACE.

Nous avons annoncé dans [DL06d] que le problème de satisfaisabilité pour EXML est dans PSPACE, ce que nous avons corrigé dans [DL06c]. Il n'est pas exclus que ce résultat soit correct cependant notre argument basé sur la preuve du théorème 6 de [SSMH04] n'est pas valide à moins que la compréhension de cette preuve nous échappe.

Problème ouvert 22. Est-ce que le problème de satisfaisabilité pour EXML est dans PSPACE? ◯

Les logiques modales graduées sont évidemment les ancêtres de EXML où les contraintes de Presburger ont leur plus simple expression dans les formules $\Diamond_{\geq n}\phi$, voir par exemple [Fin72, BC85, Cer90, vdH92, vdHdR95]. Des extensions plus élaborées ont été considérées dans des logiques épistémiques [vdHM91] et des logiques terminologiques [HB91, CG05] sans que les contraintes de Presburger aient été enrichies. C'est seulement tardivement dans [Tob00] que la logique modale graduée minimale a été montrée dans PSPACE tandis que des résultats de décidabilité avaient été obtenus de façon systématique dans [Cer94b]. La PSPACE-complétude de chaque fragment EXML$_k$ (incluant PML) étend donc le résultat principal de [Tob00]. Il existe encore d'autres extensions graduées de logiques déjà bien expressives par ailleurs mais l'indécidabilité est souvent causée par la possibilité de coder une grille infinie [BP04]. Cependant, la EXPTIME-complétude de μ-calcul gradué [KSV02] demeure à ce jour un tour de force. De plus, il y a eu plusieurs tentatives pour coder de façon concise des logiques avec comptage vers des logiques sans comptage explicite (voir par exemple [OSH96, MP97, Kaz04]) mais aucune d'entre elles n'a permis à ma connaissance d'obtenir des bornes de complexité dans PSPACE. Finalement, EXML peut être vue comme la logique introduite dans [SSMH04] sans les opérateurs de point-fixe et c'est d'ailleurs ce travail qui a initialement attiré mon attention. Une des logiques de [ZL06] a aussi un rapport avec la logique SL ("Sheaves Logic") et c'est justement l'objet de la prochaine section.

4.2.4 Complexité élémentaire de “Sheaves Logic”

Nous rappelons ici la définition de la logique SL (“Sheaves Logic”) introduite dans [ZL03, ZL06]. Nous adoptons des notations qui permettront ultérieurement de faire facilement le lien avec EXML. De plus, contrairement à [ZL03, ZL06], nous autorisons les opérateurs Booléens dans les éléments (notés E) comme c'est déjà le cas pour les documents (notés D). Les éléments et les documents sont inductivement définis avec la grammaire suivante:

- $E := \alpha[D] \mid \delta \mid \neg E \mid E \wedge E \mid \texttt{true}$,
- $D := \mathcal{A}(E_1, \ldots, E_p) \mid \exists \mathsf{x}_1, \ldots, \mathsf{x}_p : \phi(\mathsf{x}_1, \ldots, \mathsf{x}_p) : \mathsf{x}_1 E_1 \& \cdots \& \mathsf{x}_p E_p \mid \texttt{true} \mid \neg D \mid D \wedge D'$,

où

- α appartient à un ensemble infini dénombrable d'étiquettes TAGS,
- δ appartient à un ensemble infini dénombrable de types de données DATATYPES (disjoint de TAGS),
- $\mathcal{A}$ est un automate non-déterministe sur un alphabet à p lettres linéairement ordonnées,
- $\phi(\mathsf{x}_1, \ldots, \mathsf{x}_p)$ est une combinaison Booléenne de formules de Presburger construites sur les variables $\mathsf{x}_1, \ldots, \mathsf{x}_p$ et de la forme $t \sim b$ ou $t \equiv_k c$ avec $t = \Sigma a_i \mathsf{x}_i$.

Pour $k \geq 0$, nous notons SL_k le fragment de SL défini sur le modèle de EXML_k.

Un modèle de SL est une structure de la forme $\mathcal{M} = \langle T, R, (<_x)_{x \in T}, V \rangle$ où

- T est un ensemble fini d'états,
- $\langle T, R \rangle$ est un arbre et chaque relation $<_x$ est un ordre total sur $R(x)$,
- $V : T \to \mathrm{TAGS} \cup \mathrm{DATATYPES}$ est une fonction d'étiquetage telle que pour $x \in T$, si x est une feuille de $\langle T, R \rangle$ alors $V(x) \in \mathrm{DATATYPES}$ sinon $V(x) \in \mathrm{TAGS}$.

La relation de satisfaction $\models$ admet la définition suivante:

- $\mathcal{M}, x \models \delta$ ssi $\delta = V(x)$,
- $\mathcal{M}, x \models \alpha[D_1 \wedge D_2]$ ssi $\mathcal{M}, x \models \alpha[D_1]$ et $\mathcal{M}, x \models \alpha[D_2]$,
- $\mathcal{M}, x \models \alpha[\neg D]$ ssi $\alpha = V(x)$ et $\mathcal{M}, x \not\models \alpha[D]$,
- $\mathcal{M}, x \models \alpha[\texttt{true}]$ ssi $\alpha = V(x)$,
- $\mathcal{M}, x \models \alpha[\exists \mathsf{x}_1, \ldots, \mathsf{x}_p : \phi(\mathsf{x}_1, \ldots, \mathsf{x}_p) : \mathsf{x}_1 E_1 \& \cdots \& \mathsf{x}_p E_p]$ ssi $\alpha = V(x)$, $R(x) = x_1 < \cdots < x_k$, et il existe $i_1, \ldots, i_k$ tels que pour $j \in \{1, \ldots, k\}$, $\mathcal{M}, x_j \models E_{i_j}$ et $[\mathsf{x}_1 \leftarrow n_1, \ldots, \mathsf{x}_p \leftarrow n_p] \models \phi(\mathsf{x}_1, \ldots, \mathsf{x}_p)$ avec $n_i = \mathrm{card}(\{s \in \{1, \ldots, k\} : i_s = i\})$,
- $\mathcal{M}, x \models \alpha[\mathcal{A}(E_1, \ldots, E_p)]$ ssi $\alpha = V(x)$, $R(x) = x_1 < \cdots < x_k$, et il existe $i_1, \ldots, i_k$ tels que pour $j \in \{1, \ldots, k\}$, $\mathcal{M}, x_j \models E_{i_j}$ et $a_{i_1} \cdots a_{i_k} \in \mathrm{L}(\mathcal{A})$ avec $\Sigma_{\mathcal{A}} = a_1, \ldots, a_p$.

Une différence cruciale avec la sémantique de EXML (voir aussi [SSMH04]) est que pour SL une contrainte de Presburger ne compte qu'une seule fois un fils.

Soit ϕ une formule de SL avec les étiquettes $\{\alpha_1, \ldots, \alpha_n\}$ et les types de données $\{\delta_1, \ldots, \delta_{n'}\}$. Nous allons définir une formule ϕ' de EXML construite au

moins sur les variables propositionnelles suivantes:

$$VP = \{\mathrm{p}_{\alpha_1}, \ldots, \mathrm{p}_{\alpha_n}, \mathrm{p}_{\alpha_{new}}\} \cup \{\mathrm{p}_{\delta_1}, \ldots, \mathrm{p}_{\delta_{n'}}, \mathrm{p}_{\delta_{new}}\}.$$

Dans la suite la conjonction $\bigwedge_{i=0}^{m} \overbrace{\Box \ldots \Box}^{i \text{ fois}} \varphi$. sera abrégée en $\forall^m \varphi$. La formule ϕ' est une conjonction $\phi'_{val} \wedge t(\phi)$ où $t(\phi)$ est définie récursivement sur la structure de ϕ et ϕ'_{val} énonce des contraintes sur l'interprétation des étiquettes et des types de données. Pour chaque document de la forme $D = \exists \mathsf{x}_1 \cdots \mathsf{x}_p : \phi(\mathsf{x}_1, \ldots, \mathsf{x}_p) : \mathsf{x}_1 E_1 \ \& \cdots \& \ \mathsf{x}_p E_p$ de ϕ, nous introduisons de nouvelles variables propositionnelles $\mathrm{p}^1_D, \ldots, \mathrm{p}^p_D$.

La formule ϕ'_{val} est la conjonction des formules suivantes:

- $\forall^{|\phi|} \bigvee_{\mathrm{p} \in VP} (\mathrm{p} \wedge \bigwedge_{\mathrm{q} \in VP \setminus \{\mathrm{p}\}} \neg \mathrm{q}) \ \wedge \overbrace{\forall^{|\phi|}(\Diamond \mathtt{true} \Rightarrow \bigvee_{\alpha \in \{\alpha_1, \ldots, \alpha_n, \alpha_{new}\}} \mathrm{p}_\alpha)}^{\text{noeuds internes}}$
- $\underbrace{\forall^{|\phi|}(\Box \mathtt{false} \Rightarrow \bigvee_{\delta \in \{\delta_1, \ldots, \delta_{n'}, \delta_{new}\}} \mathrm{p}_\delta)}_{\text{feuilles etiquetees par des types de donnees}}$
- $\forall^{|\phi|}(\bigwedge_{D \text{ is of the form } \exists \ldots E_p} (\bigwedge_{i \neq j \in \{1, \ldots, p\}} \neg(\mathrm{p}^i_D \wedge \mathrm{p}^j_D)) \wedge \mathrm{p}^i_D \Leftrightarrow t(E_i))$.

où t est la traduction de SL vers EXML définie ci-dessous. On peut remarquer que comme $\Leftrightarrow$ ne fait pas partie du langage orginal (c'est une abréviation), $\mathrm{p}^i_D \Leftrightarrow t(E_i)$ utilise deux copies de $t(E_i)$ mais cela ne cause pas d'explosion combinatoire (sinon on aurait utilisé la technique de renommage qui s'applique ici).

- t est homomorphique pour les opérateurs Booléens et $t(\mathtt{true}) = \mathtt{true}$,
- $t(\alpha_i[D]) = \mathrm{p}_{\alpha_i} \wedge t(D)$,
- $t(\delta_i) = \mathrm{p}_{\delta_i}$,
- $t(\mathcal{A}(E_1, \ldots, E_p)) = \mathcal{A}(t(E_1), \ldots, t(E_p))$,
- $t(\exists \mathsf{x}_1 \cdots \mathsf{x}_p : \phi(\mathsf{x}_1, \ldots, \mathsf{x}_p) : \mathsf{x}_1 E_1 \ \& \cdots \& \ \mathsf{x}_p E_p)$ est égale à la formule ci-dessous:

  $$\phi(\mathsf{x}_1, \ldots, \mathsf{x}_p)[\mathsf{x}_1 \leftarrow \sharp(\mathrm{p}^1_D), \ldots, \mathsf{x}_p \leftarrow \sharp(\mathrm{p}^p_D)] \wedge \neg\sharp(\neg \mathrm{p}^1_D \wedge \cdots \wedge \neg \mathrm{p}^p_D) > 0.$$

  où $\phi(\mathsf{x}_1, \ldots, \mathsf{x}_p)[\mathsf{x}_1 \leftarrow \sharp(\mathrm{p}^1_D), \ldots, \mathsf{x}_p \leftarrow \sharp(\mathrm{p}^p_D)]$ est obtenue à partir de $\phi(\mathsf{x}_1, \ldots, \mathsf{x}_p)$ en remplaçant chaque ocurrence de x_i par $\sharp(\mathrm{p}^i_D)$.

Lemme 4.2.8. t est une réduction logarithmique en espace telle que ϕ est satisfaisable ssi ϕ' est satisfaisable.

Comme t préserve le nombre de contraintes de régularité ($\Box$ et $\Diamond$ sont codés comme des contraintes de Presburger), la borne de complexité non-élémentaire pour SL établie dans [ZL06] peut être considérablement raffinée.

Proposition 4.2.9. [DL06c]

(I) Pour $k \geq 1$, le problème de satisfaisabilité pour SL_k est PSPACE-complet.

(II) Le problème de satisfaisabilité pour SL est dans EXPSPACE.

Problème ouvert 23. Est-ce que le problème de satisfaisabilité pour SL est dans PSPACE? ◯

4.2.5 Une extension indécidable

Dans le papier [ABD+05] une logique à la PDL, nommée $\mathsf{PDL}_{\mathtt{tree}}$, est introduite dont les modèles sont des arbres finis avec ordre sur les fils. Le langage logique peut faire référence aux relations suivantes: frère-gauche, frère-droit, parent, fils. D'autres relations peuvent être générées avec les opérateurs de programmes de PDL (itération, test, union et composition). Le problème de satisfaisabilité pour $\mathsf{PDL}_{\mathtt{tree}}$ est montré EXPTIME-complet dans [ABD+05].

Nous définissons ci-dessous un fragment de $\mathsf{PDL}_{\mathtt{tree}}$ auquel ont été ajoutées des contraintes de Presburger. Cette logique, notée simplement ici $\mathcal{L}$, aura des caractérisques à la fois de EXML et de $\mathsf{PDL}_{\mathtt{tree}}$.

Etant donnés une ensemble infini dénombrable de variables propositionnelles $\mathrm{PROP} = \{\mathrm{p}_1, \mathrm{p}_2, \ldots\}$ et l'ensemble de symboles de relation $\Sigma = \{\downarrow, \downarrow^*, \rightarrow, \rightarrow^*, \leftarrow, \leftarrow^*, \uparrow, \uparrow^*\}$ les formules de $\mathcal{L}$ admettent la définition suivante:

- $\phi ::= \mathrm{p} \mid \neg\phi \mid \phi \wedge \phi \mid t \sim b$
- $t ::= a \times \sharp^{\mathsf{R}}\phi \mid t + a \times \sharp^{\mathsf{R}}\phi$,

où $\mathrm{p} \in \mathrm{PROP}$, $\mathsf{R} \in \Sigma$, $b \in \mathbb{N}$, $a \in \mathbb{Z}$ et $\sim \in \{<, >, =\}$. Les programmes dans $\mathsf{PDL}_{\mathtt{tree}}$ sont beaucoup plus nombreux que ceux dans Σ grâce à la présence de l'itération, du test, de choix non-déterministe et de la composition. De même, les contraintes de Presburger de EXML sont plus riches que celles de $\mathcal{L}$. Un modèle pour $\mathcal{L}$ est une structure de la forme

$$\mathcal{M} = \langle T, R_\downarrow, R_{\downarrow^*}, R_\rightarrow, R_{\rightarrow^*}, R_\leftarrow, R_{\leftarrow^*}, R_\uparrow, R_{\uparrow^*}, V \rangle$$

avec

- $\langle T, R_\downarrow, R_\rightarrow \rangle$ est une arbre fini ordonné où $R_\downarrow$ et $R_\rightarrow$ sont respectivement les relations "fils" et "frère-droit",
- $V : T \rightarrow \mathcal{P}(\mathrm{PROP})$ est un étiquetage,
- pour chaque relation $\mathsf{R} \in \{\downarrow, \rightarrow, \leftarrow, \uparrow\}$, $R^*_{\mathsf{R}} = R_{\mathsf{R}^*}$ (R^*_{R} est la fermeture réflexive et transitive de R_{R}), $R_\rightarrow = R^{-1}_\leftarrow$ et $R_\uparrow = R^{-1}_\downarrow$,

La relation de satisfaction se définit facilement à partir de la définition pour EXML.

Proposition 4.2.10. [DL06d] Le problème de satisfaisabilité pour $\mathcal{L}$ est indécidable.

La preuve est par réduction du problème de l'arrêt pour les machines de Minsky. Cependant, un examen de la preuve (voir [DL06c]) dévoile que seuls les symboles de relation dans $\{\downarrow^*, \downarrow, \rightarrow^*, \leftarrow\}$ sont utilisés, De plus, il est nécessaire d'utiliser

des contraintes de Presburger manipulant plusieurs relations.

Problème ouvert 24. Est-ce que le fragment de $\mathcal{L}$ restreint aux formules telles que si $\Sigma_i a_i \sharp^{\mathsf{R}_i} \phi_i$ est une sous-formule alors $j \neq j'$ implique $\mathsf{R}_j \neq \mathsf{R}_{j'}$, est décidable? ◯

Cela nous incite aussi à regarder le problème suivant.

Problème ouvert 25. Est-ce que EXML augmentée avec la relation $\leftarrow$ est décidable? ◯

La proposition 4.2.10 laisse en fait un champ beaucoup plus large d'investigation à savoir comment étendre EXML tout en préservant la décidabilité, sans même se soucier à ce stade des questions de complexité.

4.3 Dynamique des politiques

Dans cette section, nous changeons de domaine d'application. Nous allons expliquer brièvement comment une logique qui permet de spécifier dynamiquement des politiques d'action peut être traduite vers la logique bien connue PDL. Il s'agit d'un travail ponctuel qui touche aux contraintes de régularité et à la notion novatrice de sabotage introduite dans [vB05]. En effet, dans un réseau ouvert, un acte élémentaire de sabotage consiste à supprimer une connexion ou un noeud. Une logique de sabotage, comme celles introduites dans [vB05, LR03], permet de spécifier des propriétés du réseau après un acte de sabotage même si on ignore où il a eu lieu.

4.3.1 Politiques d'action

Les logiques déontiques sont des formalismes qui contiennent des modalités qui expriment l'obligation, l'interdiction ou encore la permission [Gar79]. En effet, pouvoir comparer des comportements idéaux avec ceux observés prend tout son sens dès que l'on souhaite décrire des situations normales ou mettre en place des politiques de sécurité, voir par exemple [WM94]. Nombreuses de ces logiques, comme celles dans [Mey88, vdM96, Bro03, PW04] peuvent être vues comme des variantes de la logique PDL [HKT00].

Parmi ces logiques déontiques, la logique dynamique de permission DLP de R. van der Meyden [vdM96] est centrale et peut être définie comme une version de PDL sans opérateur "test" sur les programmes et avec la possibilité de distinguer les transitions autorisées (les transitions "vertes") des transitions interdites (les transitions "rouges"). Dans un modèle, l'ensemble des transitions vertes est appelé une politique et certaines modalités de DLP tiennent compte de la couleur des transitions. La possibilité de raisonner sur l'autorisation d'une séquence d'actions a motivée l'introduction de DLP [vdM96] avec le but d'améliorer la logique introduite dans [Mey88] qui distingue les états autorisés des états interdits (au lieu des transitions autorisées des transitions interdites dans DLP). Une axiomatisation

pour DLP est définie dans [vdM96] et une borne supérieure de complexité NEXPTIME pour la satisfaisabilité est aussi établie dans [vdM96] même si les questions de complexité ne sont pas explicitement discutées dans ce papier. Un peu plus tard, dans l'article [PW04], la logique DLP a été étendue de façon significative en ajoutant dans le langage logique la possibilité de mettre à jour les politiques, c'est-à-dire de pouvoir ajouter ou retrancher des transitions dans l'ensemble de transitions représentant une politique. Une axiomatisation de cette nouvelle logique appelée $\mathrm{DLP}_{\mathrm{dyn}}$ et une borne supérieure de complexité NEXPTIME sont établies pour cette extension dans [PW04] même si les preuves sont promises pour une version longue qui n'a pas encore vu le jour. Cette faculté d'ajouter et surtout de retrancher des transitions admet une forte analogie avec la logique modale de sabotage SML de J. van Benthem [vB05] pour laquelle une variante a été montrée indécidable dans [LR03] en autorisant la suppression de transitions au lieu d'états. Une différence notable mérite cependant d'être soulignée. Alors que dans SML il est possible de quantifier sur tous les modèles alternatifs obtenus par suppression d'un état ou d'une transition, dans $\mathrm{DLP}_{\mathrm{dyn}}$ les transitions ajoutées ou supprimées sont spécifiées dans le langage logique lui-même.

La motivation première de ce travail part du constat que de nombreuses logiques peuvent être traduites vers PDL (et donc vers le μ-calcul modal) même si ce n'était pas toujours évident a priori. On en trouve des exemples parmi les logiques épistémiques [FI87], les logiques déontiques [Bro03], les logiques terminologiques [Sch91, dGL94] ou encore les logiques pour l'information incomplète [DG00a]. Cette liste n'est bien-sûr pas exhaustive et nous allons montrer dans la suite que $\mathrm{DLP}_{\mathrm{dyn}}$ pourra être aussi traduite vers PDL ce qui nous permettra d'établir de nouveaux résultats de complexité. De plus, cela fournit implicitement des explications sur la bonne marche des techniques de PDL utilisées dans [vdM96, PW04].

4.3.2 Logique dynamique de permission $\mathrm{DLP}^{+}_{\mathrm{dyn}}$

Soit $\Pi_0 = \{a_1, a_2, \ldots\}$ un ensemble infini dénombrable d'actions. Les formules de $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ sont définies par la grammaire ci-dessous:

$$\begin{aligned}\phi ::= {} & \mathrm{p} \mid \neg\phi \mid \phi \wedge \phi \mid [\alpha]\phi \mid \langle\alpha\rangle\phi \mid \\ & \mathrm{perm}(\alpha)\phi \mid \mathrm{freeperm}(\alpha)\phi \mid \mathrm{grant}(\psi,\psi')\phi \mid \mathrm{revoke}(\psi,\psi')\phi\end{aligned}$$

où $\mathrm{p} \in \mathrm{PROP}$ et α est un terme d'action défini comme une expression régulière construite dont l'alphabet fini est inclus dans Π_0 et dans l'ensemble des formules (voir aussi la section 5.2.1). Les formules de $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ incluent celles PDL. On notera $\mathrm{L}(\alpha)$ le langage régulier associé à l'action α.

Un modèle de $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ est une structure de la forme $\mathcal{M} = \langle S, (R_a)_{a\in\Pi_0}, V, P\rangle$ où

- S est un ensemble non-vide d'états. Chaque état représente une configuration courante.
- $(R_a)_{a\in\Pi_0}$ est une famille de relations binaires de S. Chaque relation R_a décrit l'effet de l'action a,

– $V : S \to \mathcal{P}(\mathrm{PROP})$ spécifie quelles propriétés atomiques sont vérifiées pour chaque état.
– $P \subseteq S \times S$ représente une politique comprise comme l'ensemble des transitions autorisées.

La relation de satisfaction $\mathcal{M}, x \models \phi$ est défini inductivement comme suit où $x \xrightarrow{\phi} y$ est une abréviation pour $x = y$ et $\mathcal{M}, x \models \phi$. Un chemin de la forme $x_0 \xrightarrow{A_0} x_1 \xrightarrow{A_1} \cdots \xrightarrow{A_{n-1}} x_n$ dans $\mathcal{M}$ est dit P-vert ssi pour chaque $A_i \in \Pi_0$, on a $\langle x_i, x_{i+1}\rangle \in P$. Un chemin P-vert correspond à une séquence légale d'actions. Le chemin est dit P-rouge s'il n'est pas P-vert.

– $\mathcal{M}, x \models [\alpha]\phi$ ssi pour tous les chemins de la forme $x_0 \xrightarrow{A_0} x_1 \xrightarrow{A_1} \cdots \xrightarrow{A_{n-1}} x_n$ dans $\mathcal{M}$ avec $A_0 A_1 \cdots A_n \in \mathrm{L}(\alpha)$ et $x_0 = x$, on a $\mathcal{M}, x_n \models \phi$.
– $\mathcal{M}, x \models \langle\alpha\rangle\phi$ ssi il existe un chemin de la forme $x_0 \xrightarrow{A_0} x_1 \xrightarrow{A_1} \cdots \xrightarrow{A_{n-1}} x_n$ dans $\mathcal{M}$ avec $A_0 A_1 \cdots A_n \in \mathrm{L}(\alpha)$ et $x_0 = x$ tel que $\mathcal{M}, x_n \models \phi$.
– $\mathcal{M}, x \models \mathrm{perm}(\alpha)\phi$ ssi il existe un chemin P-vert de la forme $x_0 \xrightarrow{A_0} x_1 \xrightarrow{A_1} \cdots \xrightarrow{A_{n-1}} x_n$ dans $\mathcal{M}$ avec $A_0 A_1 \cdots A_n \in \mathrm{L}(\alpha)$ et $x_0 = x$ tel que $\mathcal{M}, x_n \models \phi$.
– $\mathcal{M}, x \models \mathrm{freeperm}(\alpha)\phi$ ssi il n'existe pas de chemin P-rouge de la forme $x_0 \xrightarrow{A_0} x_1 \xrightarrow{A_1} \cdots \xrightarrow{A_{n-1}} x_n$ dans $\mathcal{M}$ avec $A_0 A_1 \cdots A_n \in \mathrm{L}(\alpha)$, $x_0 = x$ et $\mathcal{M}, x_n \models \phi$.
– $\mathcal{M}, x \models \mathrm{grant}(\psi, \psi')\phi$ ssi $\langle S, (R_a)_{a\in\Pi_0}, V, P \cup P_{\mathcal{M}}^{\psi,\psi'}\rangle, x \models \phi$ avec

$$P_{\mathcal{M}}^{\psi,\psi'} = \{\langle y, y'\rangle : \mathcal{M}, y \models \psi \text{ and } \mathcal{M}, y' \models \psi'\}.$$

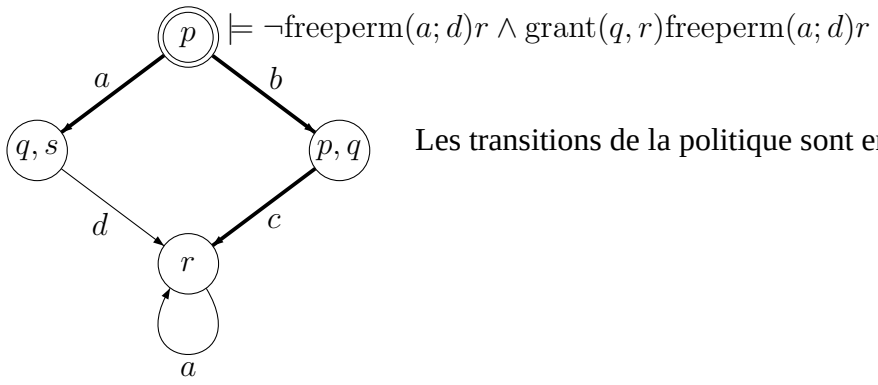
– $\mathcal{M}, x \models \mathrm{revoke}(\psi, \psi')\phi$ ssi $\langle S, (R_a)_{a\in\Pi_0}, v, P \setminus P_{\mathcal{M}}^{\psi,\psi'}\rangle, x \models \phi$.

Une formule ϕ dite satisfaisable ssi il existe un modèle de $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ et un état x tels que $\mathcal{M}, x \models \phi$. La formule $\mathrm{perm}(\alpha)\phi$ est une variante syntaxique de $\diamond(\alpha, \phi)$ dans [vdM96] et $\mathrm{freeperm}(\alpha)\phi$ correspond à $\pi(\alpha, \phi)$. Nous avons donc adapté la notation de [PW04] qui utilise aussi les opérateurs "grant" et "revoke" non présents dans [vdM96]. Plusieurs exemples de propriétés intéressantes exprimables dans DLP peuvent être trouvés dans [PW04].

La figure 4.3.2 illustre la sémantique de $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ dans un cas simple. Dans l'état doublement encerclé, la formule $\mathrm{freeperm}(a; d)r$ n'est pas satisfaite car l'unique chemin étiqueté par $a \cdot d$ commençant dans cet état est P-rouge. A l'opposé, la formule $\mathrm{grant}(q, r)\mathrm{freeperm}(a; d)r$ est satisfaite parce que l'effet de l'opérateur $\mathrm{grant}(q, r)$ est d'inclure $\{q, s\} \xrightarrow{d} \{r\}$ dans la nouvelle valeur de la politique.

La logique $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ a été introduite dans [Dem05]. Pour se repérer un peu, la logique PDL est équivalente au fragment de $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ restreint aux formules sans les quatre opérateurs faisant appel aux politiques. La logique $\mathrm{DLP}_{\mathrm{dyn}}$ introduite dans [PW04] est égale au fragment de $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ restreint aux formules

– sans la possibilité qu'une lettre d'une action α soit une formule (correspondant à l'absence de l'opérateur "test" de PDL),



Les transitions de la politique sont en gras.

FIG. 4.6 – *Sémantique de* $\mathrm{DLP}^{+}_{\mathrm{dyn}}$

– les arguments de $\mathrm{grant}(\psi, \psi')$ et $\mathrm{revoke}(\psi, \psi')$ sont nécessairement des formules du calcul propositionnel.

Le langage de la logique DLP introduite dans [vdM96] est celui de $\mathrm{DLP}_{\mathrm{dyn}}$ sans les opérateurs “grant” et “revoke”. Cependant, la définition de la sémantique des formules de DLP dans [vdM96] est légérement différente de celle utilisée ici pour $\mathrm{DLP}^{+}_{\mathrm{dyn}}$. Il a été établi dans [Dem05, lemme 2.1] que les deux sémantiques assurent le même ensemble de formules satisfaisables.

La profondeur de changement d'une formule ϕ, notée $cd(\phi)$, est l'imbrication maximale d'opérateur “grant” et “revoke” dans ϕ. Pour les formules de DLP, cette profondeur est nulle. En voici une définition formelle pour lever toute ambiguité.

– $cd(\mathrm{p}) = 0$, $cd(\neg\phi) = cd(\phi)$, $cd(\phi_1 \wedge \phi_2) = max(cd(\phi_1), cd(\phi_2))$,
– $cd([\alpha]\phi) = cd(\mathrm{perm}(\alpha)\phi) = cd(\mathrm{freeperm}(\alpha)\phi)) = cd(\alpha) + cd(\phi)$,
– $cd(\mathrm{revoke}(\psi_1, \psi_2)\phi) = 1 + max(cd(\psi_1), cd(\psi_2)) + cd(\phi)$,
– $cd(\mathrm{grant}(\psi_1, \psi_2)\phi) = cd(\mathrm{revoke}(\psi_1, \psi_2)\phi)$,
– $cd(\alpha; \beta) = cd(\alpha \cup \beta) = max(cd(\alpha), cd(\beta))$,
– $cd(a) = 0$, $cd(\alpha^*) = cd(\alpha)$.

4.3.3 Difficultés avec $\mathrm{DLP}^{+}_{\mathrm{dyn}}$

La logique $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ a de nombreuses caractéristiques qui rendent improbable sa décidabilité, lorsque ce n'est pas sa traduction vers PDL. Nous passons en revue ci-dessous quelques aspects gênants.

$\mathrm{DLP}^{+}_{\mathrm{dyn}}$ intersecte

La sémantique de l'opérateur “perm” déjà présent dans DLP peut être paraphrasée dans PDL en remplaçant $\mathrm{perm}(\alpha)$ par $\langle t^{\forall \mathrm{g}}(\alpha)\rangle$ où $t^{\forall \mathrm{g}}(\alpha)$ est obtenu en remplaçant dans α chaque action atomique a par $a \cap \alpha_P$. L'opérateur $\cap$ est l'intersection de relation et α_P est une action dont l'interprétation est la politique P. Cependant, PDL avec intersection est décidable en 2EXPTIME [Dan84], voir aussi [Lut05].

$\mathrm{DLP}^{+}_{\mathrm{dyn}}$ **complémente**

En poursuivant ce type de codage vers PDL, la présence de la complémentation semble aussi nécessaire. En effet, la sémantique de l'opérateur "freeperm" peut être paraphrasée dans PDL en remplaçant $\mathrm{freeperm}(\alpha)$ par $\neg\langle t^{\exists r}(\alpha)\rangle$. La formule $\langle t^{\exists r}(\alpha)\rangle\top$ exprime alors qu'il existe un chemin étiqueté par un mot de $\mathrm{L}(\alpha)$ qui contient une transition étiquetée par une action atomique qui n'appartient pas à P. La fonction $t^{\exists r}$ prenant en argument une action est définie inductivement de la façon suivante:

- $t^{\exists r}(\phi) = \bot$. En effet, la rougeur du chemin ne peut être due à une transition étiquetée par une formule.
- $t^{\exists r}(\alpha_1 \cup \alpha_2) = t^{\exists r}(\alpha_1) \cup t^{\exists r}(\alpha_2)$,
- $t^{\exists r}(\alpha_1;\alpha_2) = t^{\exists r}(\alpha_1);\alpha_2 \cup \alpha_1; t^{\exists r}(\alpha_2)$,
- $t^{\exists r}(\alpha^*) = \alpha^*; t^{\exists r}(\alpha); \alpha^*$,
- $t^{\exists r}(a) = a \cap -\alpha_P$.

$t^{\exists r}$ recherche donc la présence d'une transition rouge, c'est-à-dire le passage d'une transition qui ne soit pas dans P. Cependant, PDL avec complémentation est indécidable (voir par exemple [HKT00]) et PDL avec la complémentation juste devant les programmes atomiques est bien dans EXPTIME [LW04] mais ne peut capturer l'intersection. Une alternative consiste à coder "perm" et "freeperm" en décomposant chaque action atomique a par l'union d'actions atomiques $a^g \cup a^r$ (la partie verte et la partie rouge) telle que les conditions suivantes soient vérifiées:

(PROP1) R_{a^g} et R_{a^r} sont disjointes,

(PROP2) pour chaque paire $\langle x, y\rangle \in S \times S$, si $\langle x, y\rangle \in R_{a^g}$ alors $\langle x, y\rangle \notin R_{b^r}$ pour toutes les actions atomiques b et si $\langle x, y\rangle \in R_{a^r}$ alors $\langle x, y\rangle \notin R_{b^g}$ pour toutes les actions atomiques b.

Il se trouve que ces conditions ne sont pas définissables en logique modale. Cependant, la condition (PROP1) peut être imposée sans changer la classe des formules satisfaisables en utilisant des dépliages de modèles. La condition (PROP2) est une conséquence du fait qu'une politique est un ensemble de transitions sans information sur l'action entreprise.

$\mathrm{DLP}^{+}_{\mathrm{dyn}}$ **modifie**

Les points évoqués ci-dessus ne concernent en fait que la partie déjà présente dans DLP de $\mathrm{DLP}^{+}_{\mathrm{dyn}}$. Les opérateurs "grant" et "revoke" introduits dans [PW04] expriment la satisfaction dans un autre modèle, ce qui est analogue à l'aspect destructif de la logique modale de sabotage de J. van Benthem [vB05]. En effet, la logique modale de sabotage (SML) définie dans [vB05] admet l'ensemble suivant de formules:

$$\phi ::= \mathrm{p} \mid \neg\phi \mid \phi \wedge \phi \mid \diamond\phi \mid \langle -\rangle\phi.$$

Les modèles de SML sont les structures de Kripke usuelles de la forme $\mathcal{M} = \langle S, R, V\rangle$. La seule différence avec la sémantique habituelle est pour l'interprétation

des formules $\langle - \rangle \phi$: $\mathcal{M}, x \models \langle - \rangle \phi$ ssi il existe $x' \in S$ tel que $\mathcal{M}', x \models \phi$ avec $\mathcal{M}'$ la restriction de $\mathcal{M}$ à $S \setminus \{x'\}$. A ce jour, le problème suivant est encore ouvert.

Problème ouvert 26. Est-ce que le problème de satisfaisabilité pour SML est décidable? ◯

Il existe cependant une variante de SML, appelons-la SML′, pour laquelle le problème de satisfaisabilité a été montré indécidable dans [LR03]. Au lieu de supprimer des états comme dans SML, la logique SML′ a la possibilité de retrancher des transitions, une caractéristique commune avec $\mathrm{DLP}^{+}_{\mathrm{dyn}}$. Les formules de SML′ sont les suivantes:

$$\phi ::= \mathrm{p} \mid \neg\phi \mid \phi \wedge \phi \mid \diamond_a \phi \mid \langle - \rangle_a \phi,$$

où a appartient à un alphabet fini Σ. Les modèles de SML′ sont de la forme $\mathcal{M} = \langle S, (R_a)_{a \in \Sigma}, V \rangle$ et $\mathcal{M}, x \models \langle - \rangle_a \phi$ ssi il existe $\langle y, y' \rangle \in R_a$ tel que $\mathcal{M}', x \models \phi$ où $\mathcal{M}'$ est obtenu à partir de $\mathcal{M}$ en supprimant la transition $\langle y, y' \rangle$ de R_a. SML′ est indécidable dès que $|a| \geq 2$ [LR03].

Problème ouvert 27. Est-ce que SML′ est décidable lorsque Σ est un singleton? ◯

Ainsi SML′ et $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ ont toutes deux la possibilité de supprimer des arêtes et le pire pourrait être que $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ peut aussi en ajouter.

4.3.4 Traduction vers PDL

Avant de définir une réduction de $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ vers PDL qui préserve la satisfaisabilité, nous allons établir certaines propriétés qui vont justifier certains choix faits pour la traduction.

Le lemme 4.3.1 ci-dessous énonce que la présence de certaines paires dans une politique n'est pas utile dans l'évaluation d'une formule.

Lemme 4.3.1. Soient $\mathcal{M} = \langle S, (R_a)_{a \in \Pi_0}, V, P \rangle$ et $\mathcal{M}' = \langle S, (R_a)_{a \in \Pi_0}, V, P' \rangle$ deux modèles tels que $P' = P \cap (R_{a_1} \cup \cdots \cup R_{a_N})$, pour un ensemble fini d'actions atomiques $\{a_1, \ldots, a_N\}$. Pour toutes les formules construites sur les seules actions atomiques de $\{a_1, \ldots, a_N\}$ et pour $x \in S$, $\mathcal{M}, x \models \phi$ ssi $\mathcal{M}', x \models \phi$.

Dans la suite on suppose que $P \subseteq R_{a_1} \cup \cdots \cup R_{a_N}$ quand on s'intéresse à une formule construite sur les actions atomiques de $\{a_1, \ldots, a_N\}$.

Le lemme 4.3.2 ci-dessous montre que si α_P est une action dans PDL interprétée par $P \cap (R_{a_1} \cup \cdots \cup R_{a_N})$ alors on peut construire une action dans PDL pour l'ensemble $(P \cup P^{\psi,\psi'}_{\mathcal{M}}) \cap (R_{a_1} \cup \cdots \cup R_{a_N})$ quand ψ, ψ' sont construites sur les actions atomiques de $\{a_1, \ldots, a_N\}$. Il s'agit d'une propriété essentielle pour simuler l'usage des transitions vertes et rouges sans avoir recours aux opérateurs d'intersection et de complémentation sur les relations.

Lemme 4.3.2. Soient $\mathcal{M} = \langle S, (R_a)_{a \in \Pi_0}, V, P \rangle$ un modèle et $\{a_1, \ldots, a_N\}$ un ensemble d'actions atomiques tels qu'il existe des actions α_P et α_{-P} vérifiant
(I) $R_{\alpha_P} = P \cap (R_{a_1} \cup \cdots \cup R_{\alpha_N})$,

(II) $R_{\alpha_{-P}} = ((S \times S) \setminus P) \cap (R_{a_1} \cup \cdots \cup R_{a_N})$.
Alors, pour toutes les formules ψ, ψ' construites sur $\{a_1, \ldots, a_N\}$,

(III) $(P \cup P_{\mathcal{M}}^{\psi,\psi'}) \cap (R_{a_1} \cup \cdots \cup R_{a_N}) = R_\beta$ avec

$$\beta = \alpha_P \cup (\psi?; a_1; \psi'?) \cup \cdots \cup (\psi?; a_N; \psi'?)$$

(IV) $((S \times S) \setminus (P \cup P_{\mathcal{M}}^{\psi,\psi'})) \cap (R_{a_1} \cup \cdots \cup R_{a_N}) = R_{\beta'}$ avec

$$\beta' = (\neg\psi)?; \alpha_{-P} \cup \alpha_{-P}; (\neg\psi')?$$

Les politiques des points (III) et (IV) du lemme 4.3.2 sont typiquement obtenus avec les opérateurs “grant” et “revoke”.

Afin de représenter la valeur initiale d'une politique P par une action de PDL, chaque action atomique a est traduite en l'union $a^g \cup a^r$. Par exemple, si $a_1, \ldots, a_N$ sont les actions atomiques présentes dans une formule ϕ, nous allons imposons que les actions atomiques $a_1^g, a_1^r, \ldots, a_N^g, a_N^r$ soient interprétées par des relations disjointes deux à deux ce qui est possible grâce au lemme ci-dessous.

Lemme 4.3.3. Une formule ϕ de $\mathrm{DLP}^+_{\mathrm{dyn}}$ est satisfaisable ssi ϕ est satisfaisable dans un modèle dont toutes les relations pour les actions atomiques sont disjointes deux à deux.

La preuve est obtenue en dépliant les modèles. Ainsi, il est possible de garantir que (PROP1) et (PROP2) soient vérifiées. La valeur initiale de α_P est donc $a_1^g \cup \cdots \cup a_N^g$ et son complément $-\alpha_P$ relativement à $\bigcup_{1 \leq i \leq N}(a_i^g \cup a_i^r)$ est $a_1^r \cup \cdots \cup a_N^r$. En appliquant un nombre fini d'opérateurs “grant” ou “revoke”, le lemme 4.3.2 assure que les valeurs courantes de α_P et $-\alpha_P$ sont équivalentes à des unions d'actions de la forme

$$\psi_1?; \cdots; \psi_n?; a_i^\bullet; \psi_1'? \cdots; \psi_{n'}'?$$

avec $\bullet \in \{g, r\}$. $\psi_1, \cdots, \psi_n, \psi_1', \psi_{n'}'$ sont des formules de PDL et n n'est pas nécessairement égale à n'.

Récapitulons comment une action atomique a_i est traduite.

- Une occurrence de a_i dans la portée immédiate d'un opérateur $[\cdot]$ est traduite en $a_i^g \cup a_i^r$.
- Une occurrence de a_i dans la portée immédiate d'un opérateur $\mathrm{perm}(\cdot)$ est traduite en une union d'actions apparaissant dans α_P de la forme

  $$\psi_1?; \cdots; \psi_n?; a_i^\bullet; \psi_1'? \cdots; \psi_{n'}'?$$

  avec $\bullet \in \{g, r\}$.
- De façon analogue, une occurrence de a_i dans la portée immédiate d'un opérateur $\mathrm{freeperm}(\cdot)$ peut quelquefois se traduire en une union d'actions apparaissant dans $-\alpha_P$ de la forme ci-dessus.

Nous allons à présent définir la traduction d'une formule ϕ de $\mathrm{DLP}^{+}_{\mathrm{dyn}}$ construite sur les actions atomiques $a_1, \ldots, a_N$. La formule de PDL obtenue par traduction est construite sur les actions atomiques $a_1^g, a_1^r, \ldots, a_N^g, a_N^r$. Cette traduction est calculée à partir d'une fonction ternaire $T(\psi, \mathsf{G}, \mathsf{R})$ òu G et R sont des ensembles d'actions de la forme

$$\psi_1?; \cdots; \psi_n?; a; \psi_1'? \cdots; \psi_{n'}'?$$

avec $n, n' \geq 0$ et a est parmi $a_1^g, a_1^r, \ldots, a_N^g, a_N^r$. De plus, ψ sera une sous-formule de ϕ et $n, n' \leq |\phi|$. L'union des actions de G est interprétée comme la partie verte du modèle (sa politique) tandis que l'union des actions de R est interprétée comme sa partie rouge. Les ensembles G et R sont mis à jour à chaque fois que nous traversons un opérateur "grant" ou "revoke".

La traduction de ϕ est $T(\phi, \mathsf{G}_0, \mathsf{R}_0)$ avec

$$\mathsf{G}_0 = \{a_i^g : 1 \leq i \leq N\}, \quad \mathsf{R}_0 = \{a_i^r : 1 \leq i \leq N\}.$$

Durant la traduction, $(\bigcup_{\alpha \in \mathsf{G}} \alpha)$ [resp. $(\bigcup_{\alpha \in \mathsf{R}} \alpha)$] est interprétée comme la restriction de P [resp. $-P$] à $\bigcup_{1 \leq i \leq N}(R_{a_i^g} \cup R_{a_i^r})$. Un invariant de la traduction est donc que $\mathsf{G} \cup \mathsf{R}$ est toujours interprétée par $\bigcup_{1 \leq i \leq N}(R_{a_i^g} \cup R_{a_i^r})$. Observons que si α_P est égale à $(\bigcup_{\alpha \in \mathsf{G}} \alpha)$, alors $(a_i^g \cup a_i^r) \cap \alpha_P$ est équivalent à

$$\bigcup\{\psi_1?; \cdots; \psi_n?; a_i^{\bullet}; \psi_1'? \cdots; \psi_{n'}'? \in \mathsf{G} : \bullet \in \{g, r\}\}.$$

Une remarque analogue est vraie pour $(a_i^g \cup a_i^r) \cap -\alpha_P$. Voici à présent la définition complète de $T(\cdot)$:

- $T(\mathrm{p}, \mathsf{G}, \mathsf{R}) = \mathrm{p}$ et T est homomorphique pour les opérateurs Booléens.
- $T([\alpha]\psi, \mathsf{G}, \mathsf{R}) = [T(\alpha, \mathsf{G}, \mathsf{R})]T(\psi, \mathsf{G}, \mathsf{R})$ où l'extension de T sur les actions est définie suivant les clauses ci-dessous:
  - $T(a_i, \mathsf{G}, \mathsf{R}) = a_i^g \cup a_i^r$,
  - T est homomorphique pour les opérateurs ;, $\cup$ et *,
- $T(\mathrm{perm}(\alpha)\psi, \mathsf{G}, \mathsf{R}) = \langle t^{\forall g}(\alpha, \mathsf{G}, \mathsf{R})\rangle T(\psi, \mathsf{G}, \mathsf{R})$ avec $t^{\forall g}$ définie ainsi:
  - $t^{\forall g}$ est homomorphique pour les opérateurs ;, $\cup$ et *,
  - $t^{\forall g}(a_i, \mathsf{G}, \mathsf{R}) = \bigcup\{\psi_1?; \cdots; \psi_n?; a_i^{\bullet}; \psi_1'?; \cdots; \psi_{n'}'? \in \mathsf{G} : \bullet \in \{g, r\}\}$. Comme nous voulons que a et $a^g \cup a^r$ aient la même interprétation, pour chaque action $\alpha \in \mathsf{G}$ de la forme $\psi_1?; \cdots; \psi_n?; a_i^{\bullet}; \psi_1'?; \cdots; \psi_{n'}'?$, nous avons $R_\alpha \subseteq R_{a_i}$. Par ailleurs, pour chaque paire $\langle y, y'\rangle \in R_\alpha$, $\langle y, y'\rangle$ appartient à la politique courante. En effet, $\mathrm{perm}(\beta)\psi$ est vérifiée lorsqu'il existe une chemin P-vert étiqueté par un mot de $\mathrm{L}(\beta)$ qui conduit à un état vérifiant ψ.
- $T(\mathrm{freeperm}(\alpha)\psi, \mathsf{G}, \mathsf{R}) = \neg\langle t^{\exists r}(\alpha, \mathsf{G}, \mathsf{R})\rangle T(\psi, \mathsf{G}, \mathsf{R})$ où $t^{\exists r}$ est définie ainsi:
  - $t^{\exists r}(\psi, \mathsf{G}, \mathsf{R}) = \bot$,
  - $t^{\exists r}(\alpha_1 \cup \alpha_2, \mathsf{G}, \mathsf{R}) = t^{\exists r}(\alpha_1, \mathsf{G}, \mathsf{R}) \cup t^{\exists r}(\alpha_2, \mathsf{G}, \mathsf{R})$,
  - $t^{\exists r}(\alpha_1; \alpha_2, \mathsf{G}, \mathsf{R}) = (t^{\exists r}(\alpha_1, \mathsf{G}, \mathsf{R}); T(\alpha_2, \mathsf{G}, \mathsf{R})) \cup$ $(T(\alpha_1, \mathsf{G}, \mathsf{R}); t^{\exists r}(\alpha_2, \mathsf{G}, \mathsf{R}))$,

- $t^{\exists r}(\alpha^*, \mathsf{G}, \mathsf{R}) = T(\alpha^*, \mathsf{G}, \mathsf{R}); t^{\exists r}(\alpha, \mathsf{G}, \mathsf{R}); T(\alpha^*, \mathsf{G}, \mathsf{R})$,
- $t^{\exists r}(a_i, \mathsf{G}, \mathsf{R}) = \bigcup\{\psi_1?; \cdots; \psi_n?; a_i^\bullet; \psi'_1? \cdots; \psi'_{n'}? \in \mathsf{R} : \bullet \in \{g, r\}\}$. Pour chaque action $\alpha \in \mathsf{R}$ de la forme $\psi_1?; \cdots; \psi_n?; a_i^\bullet; \psi'_1?; \cdots; \psi'_{n'}?$, nous avons $R_\alpha \subseteq R_{a_i}$. De plus, pour chaque paire $\langle y, y' \rangle \in R_\alpha$, $\langle y, y' \rangle$ n'appartient pas à la politique courante. La fonction $t^{\exists r}$ est construite sur le même principe que la fonction homonymique précédemment introduite.

- $T(\mathrm{grant}(\psi_1, \psi_2)\ \psi, \mathsf{G}, \mathsf{R}) = T(\psi, \mathsf{G}', \mathsf{R}')$ avec
  - $\mathsf{G}' = \mathsf{G} \cup \{T(\psi_1, \mathsf{G}, \mathsf{R})?; a_i^g; T(\psi_2, \mathsf{G}, \mathsf{R})? : 1 \leq i \leq N\} \cup$ $\{T(\psi_1, \mathsf{G}, \mathsf{R})?; a_i^r; T(\psi_2, \mathsf{G}, \mathsf{R})? : 1 \leq i \leq N\}$ (voir le lemme 4.3.2(III)),
  - $\mathsf{R}' = \{(\neg T(\psi_1, \mathsf{G}, \mathsf{R}))?; \alpha : \alpha \in \mathsf{R}\} \cup \{\alpha; (\neg T(\psi_2, \mathsf{G}, \mathsf{R}))? : \alpha \in \mathsf{R}\}$ (voir le lemme 4.3.2(IV)).
- $T(\mathrm{revoke}(\psi_1, \psi_2)\psi, \mathsf{G}, \mathsf{R}) = T(\psi, \mathsf{G}', \mathsf{R}')$ avec
  - $\mathsf{G}' = \{\neg T(\psi_1, \mathsf{G}, \mathsf{R})?; \alpha : \alpha \in \mathsf{G}\} \cup \{\alpha; \neg T(\psi_2, \mathsf{G}, \mathsf{R})? : \alpha \in \mathsf{G}\}$ (voir le lemme 4.3.2(IV)).
  - $\mathsf{R}' = \mathsf{R} \cup \{T(\psi_1, \mathsf{G}, \mathsf{R})?; a_i^g; T(\psi_2, \mathsf{G}, \mathsf{R})? : 1 \leq i \leq N\} \cup$ $\{T(\psi_1, \mathsf{G}, \mathsf{R})?; a_i^r; T(\psi_2, \mathsf{G}, \mathsf{R})? : 1 \leq i \leq N\}$ (voir le lemme 4.3.2(III)).

Lemme 4.3.4. ϕ est satisfaisable dans $\mathrm{DLP}^+_{\mathrm{dyn}}$ ssi $T(\phi, \mathsf{G}_0, \mathsf{R}_0)$ est satisfaisable dans PDL.

Par conséquent, on peut déjà énoncer un résultat de décidabilité sachant que le problème de satisfaisabilité pour PDL est dans EXPTIME [Pra79] et la traduction exige un temps exponentiel.

Théorème 4.3.5. [Dem05] Le problème de satisfaisabilité pour $\mathrm{DLP}^+_{\mathrm{dyn}}$ est dans 2EXPTIME.

Une analyse un peu plus fine tenant compte de la profondeur de changement permet d'obtenir des résultats de EXPTIME-complétude.

Corollaire 4.3.6. Pour chaque $k \geq 0$ fixé, le problème de satisfaisabilité pour $\mathrm{DLP}^+_{\mathrm{dyn}}$ restreint aux formules ayant une profondeur de changement au plus k est EXPTIME-complet.

Lorsque $k = 0$, cette restriction syntaxique produit précisément la logique DLP [vdM96].

Corollaire 4.3.7. [Dem05] Le problème de satisfaisabilité pour DLP est EXPTIME-complet.

Ce résultat améliore la borne NEXPTIME établie dans [vdM96, PW04] et répond à une conjecture énoncée dans [vdM96, PW04] selon laquelle DLP est dans EXPTIME. La borne inférieure EXPTIME est héritée de celle de PDL [FL79].

Problème ouvert 28. Quelle est la complexité de $\mathrm{DLP}^+_{\mathrm{dyn}}$? Nous avons montré la borne supérieure 2EXPTIME et par héritage de PDL, la borne inférieure EXPTIME est aussi assurée. ◯

Selon toute vraisemblance, au moment d'imprimer ce document, ce problème a été résolu dans [Göl07].

Dans [vdM96], l'introduction de la logique DLP était motivée par le besoin de remplacer les états permis de [Mey88] par des actions permises. L'extension de DPL introduite dans [PW04] ajoute la possibilité de mettre à jour dynamiquement la politique à l'aide des opérateurs "grant" et "revoke". Alors qu'une politique dans [vdM96] est un sous-ensemble de $S \times S$, une politique dans [Mey88] est un sous-ensemble de S distinguant ainsi les bons des mauvais états. La prise en compte d'une politique dans [Mey88] consiste simplement à considérer une variable propositionnelle spéciale, disons vc ("violation constant"), et l'on voit bien que c'est beaucoup plus simple que ce qui s'est fait après dans [vdM96, PW04, Dem05]. Une version de la logique de [Mey88] avec mise à jour de la politique a été introduite aussi dans [Dem05] et montrée EXPTIME-complète.

Chapitre 5

Complexité des logiques modales grammaticales

Ce chapitre est construit sur la base de [Dem98, Dem00a, Dem00b, Dem01a, Dem03, DdN03, DdN05].

Il existe une multitude de logiques modales caractérisées par diverses classes de structures de Kripke, voir par exemple [BdRV01] et dont les domaines d'application dépendent des hypothèses qui sont faites sur les relations d'accessibilité (logiques temporelles, logiques terminologiques, logiques épistémiques etc.). Il y a donc un impérieux besoin pour définir des méthodes qui s'appliquent à de larges classes de logiques. La généralité pour les propriétés de complétude et de canonicité est présente dans [Sah75]. Pour ce qui est de la complexité, des résultats généraux peuvent être trouvés dans [Spa93a, Hem94]. De même, les rapports entre la logique classique et les logiques modales sont longuement étudiés dans [vB85].

Dans ce chapitre, nous nous intéressons aux questions de décidabilité et de complexité pour une large classe de logiques modales, à savoir celles dont les schémas d'axiomes peuvent être vus comme les règles d'un système semi-Thuéien fini hors-contexte. Les logiques modales définies à partir de règles de grammaire ont été introduites dans [dCP88]. L'approche orginale de ce travail consiste à rechercher à caractériser la complexité algorithmique d'une logique modale grammaticale en analysant la nature des langages engendrés par le système semi-Thuéien associé. Des traductions vers des extensions décidables de PDL ou vers le fragment gardé de la logique classique avec un nombre borné de variables individuelles mettront en évidence pourquoi de nombreuses logiques modales dans la littérature et appartenant aux cadres fixés dans [Var97, Grä99a] ont de bonnes propriétés algorithmiques.

5.1 Logiques modales grammaticales

5.1.1 Définitions

Soit PROP un ensemble infini dénombrable de variables propositionnelles et Σ un alphabet (éventuellement infini). Le langage multimodal ML(Σ) est défini

par la grammaire suivante:

$$\phi ::= \overbrace{\mathrm{p} \mid \neg\phi \mid \phi \wedge \phi' \mid \phi \vee \phi'}^{\text{calcul propositionnel}} \mid \overbrace{\langle a\rangle\phi \mid [a]\phi}^{\text{extension modale}}$$

où $a \in \Sigma$ et $\mathrm{p} \in \mathrm{PROP}$. L'opérateur unaire $[a]$ permet de quantifier universellement sur tous les états accessibles par la relation d'indice a. Pour $u \in \Sigma^*$, on utilise l'abréviation $[u]\phi$ avec $[\epsilon]\phi \stackrel{\text{def}}{=} \phi$ et $[a_1 \cdots a_n]\phi \stackrel{\text{def}}{=} [a_1] \cdots [a_n]\phi$.

Un modèle de Kripke est un triplet $\mathcal{M} = \langle S, (R_a)_{a\in\Sigma}, V\rangle$ tel que $\langle S, (R_a)_{a\in\Sigma}\rangle$ est un système de transition et $V : S \to \mathcal{P}(\mathrm{PROP})$ est un coloriage des états de S. La relation de satisfaction $\models$ ($x \in S$) est définie de la façon suivante:

- $\mathcal{M}, x \models \mathrm{p}$ ssi $\mathrm{p} \in \mathrm{PROP}$,
- $\mathcal{M}, x \models \phi_1 \wedge \phi_2$ ssi $\mathcal{M}, x \models \phi_1$ et $\mathcal{M}, x \models \phi_2$,
- $\mathcal{M}, x \models \neg\phi$ ssi $\mathcal{M}, x \models \phi$ n'est pas vérifié,
- $\mathcal{M}, x \models [a]\phi$ ssi pour tous les états $x' \in R_a(x)$, on a $\mathcal{M}, x' \models \phi$,
- $\mathcal{M}, x \models \langle a\rangle\phi$ ssi il existe $x' \in R_a(x)$ tel que $\mathcal{M}, x' \models \phi$.

Dans la suite, une logique modale $\mathcal{L} = \langle \Sigma, \mathcal{C}\rangle$ est une paire composée d'un alphabet et d'un ensemble de modèles de Kripke. Généralement, cet ensemble de modèles est lui-même caractérisé par un ensemble de systèmes de transition clos par isomorphismes. La genèse de ce type de sémantique est décrite dans [Cop02]. Une formule ϕ de ML(Σ) est dite satisfaisable pour la logique $\mathcal{L}$ s'il existe un modèle $\mathcal{M} = \langle S, (R_a)_{a\in\Sigma}, V\rangle$ dans $\mathcal{C}$ et un état $x \in S$ tels que $\mathcal{M}, x \models \phi$. Par dualité, une formule ϕ est valide ssi pour tous les modèles $\mathcal{M} = \langle S, (R_a)_{a\in\Sigma}, V\rangle$ de $\mathcal{C}$ et pour tous les états $x \in S$, $\mathcal{M}, x \models \phi$. De plus, une formule ϕ est dite globalement satisfaisable s'il existe un modèle $\mathcal{M} = \langle S, (R_a)_{a\in\Sigma}, V\rangle$ dans $\mathcal{C}$ tel que pour chaque état $x \in S$, nous ayons $\mathcal{M}, x \models \phi$.

A titre d'exemple, la logique modale S4 se définit comme la logique modale tel que Σ est un singleton et l'unique relation de transition des modèles de S4 est réflexive et transitive.

Théorème 5.1.1. [Lad77, Dem00a] Pour S4, les problèmes de satisfaisabilité, validité et de satisfaisabilité globale sont PSPACE-complets.

Dans la suite, nous n'apporterons que peu d'attention à une logique en particulier car nous recherchons les propriétés qui permettent les résultats les plus généraux possibles. Cependant, pour présenter une exception, il pourra nous arriver de mettre l'accent sur une logique.

Dans la suite, un alphabet avec passé est une paire $\langle \Sigma, \overline{\cdot}\rangle$ telle que

- $\Sigma = \Sigma^+ \uplus \Sigma^-$,
- $\overline{\cdot} : \Sigma \to \Sigma$ est une bijection dont l'image de $\overline{\cdot}$ restreint à Σ^+ est égale à Σ^- et pour chaque lettre a, l'égalité $\overline{\overline{a}} = a$ est vérifiée.

On note simplement Σ un alphabet avec passé en supposant implicite la bijection $\overline{\cdot}$ et la partition $\Sigma^+ \uplus \Sigma^-$. Par défaut Σ^+ dénote la partie positive de Σ (et non l'ensemble des mots finis non vide de Σ). Quand Σ est un alphabet avec passé, on demande à un modèle de Kripke de vérifier une propriété supplémentaire: pour chaque lettre $a \in \Sigma$, $R_{\overline{a}} = \{\langle y, x\rangle : \langle x, y\rangle \in R_a\}$.

Etant donné un modèle $\mathcal{M} = \langle S, (R_a)_{a\in\Sigma}, V\rangle$, on définit la relation R_u pour $u \in \Sigma^*$ de la façon suivante:

– $R_\epsilon = \{\langle x, x\rangle : x \in S\}$,

– $R_{a.v} = R_a \circ R_v$ (composée).

De même, dans le cas d'un alphabet avec passé, on étend la bijection $\overline{\cdot}$ aux éléments de Σ^*: $\overline{\epsilon} = \epsilon$ et $\overline{u \cdot a} = \overline{a} \cdot \overline{u}$ (attention à l'inversion!).

Dans la suite, nous allons définir des logiques modales à l'aide de systèmes semi-Thuéiens. Etant donné un alphabet Σ, un système semi-Thuéien $\mathcal{S}$ est juste un ensemble de règles de la forme $u \to v$ avec $u, v \in \Sigma^*$. Par rapport à la notion de grammaire formelle, il n'y a pas d'axiome ni de distinction entre symboles terminaux et symboles non-terminaux.

Chaque contrainte $R_{a_1} \circ \cdots \circ R_{a_n} \subseteq R_b$ dans les modèles va correspondre à la règle $b \to a_1 \cdots a_n$. Ainsi, les règles pour définir les modèles de S4 sont précisément $a \to aa$ et $a \to \epsilon$.

Un système de transition $\langle S, (R_a)_{a\in\Sigma}\rangle$ satisfait la règle $u \to v$ avec $u, v \in \Sigma^*$ ssi $R_v \subseteq R_u$. On peut noter l'inversion entre u et v car la règle $u \to v$ peut aussi être comprise comme le schéma d'axiomes $[u]\mathrm{p} \Rightarrow [v]\mathrm{p}$. Un modèle de Kripke $\mathcal{M} = \langle S, (R_a)_{a\in\Sigma}, V\rangle$ satisfait un système semi-Thuéien $\mathcal{S}$ ssi le système de transition sous-jacent de $\mathcal{M}$ satisfait toutes les règles de $\mathcal{S}$.

Définition 5.1.1. Une logique modale avec passé $\mathcal{L} = \langle \Sigma, \mathcal{C}\rangle$ est dite grammaticale ssi Σ est une alphabet fini avec passé et il existe un système semi-Thuéien $\mathcal{S}$ fini tel que $\mathcal{C}$ est l'ensemble des modèles satisfaisant $\mathcal{S}$. ∇

Une définition analogue pour les logiques modales sans passé admet que Σ soit quelconque (donc sans passé). Cependant, il pourra arriver qu'à une logique sans passé, on ajoute le passé pour la définir. Par exemple, la logique (mono)modale S5 caracterisée par les modèles dont l'unique relation est une relation d'équivalence est un fragment de la logique avec passé définie par le système ci-dessous:

$$\{a \to \epsilon, a \to aa, a \to \overline{a}, \overline{a} \to \epsilon, \overline{a} \to \overline{aa}\}.$$

D'ailleurs, dans tous les modèles de S5, les relations R_a et $R_{\overline{a}}$ sont égales.

Dans la suite, par défaut Σ est fini. Les logiques modales grammaticales ont été initialement définies à l'aide de grammaires formelles dans [dCP88] (voir aussi [Bal98, Dem01a]). Elles forment une sous-classe des logiques modales de Sahlqvist [Sah75] dont les contraintes sur les relations s'expriment dans le fragment Π_1 de la logique du premier ordre. La définition avec des systèmes semi-Thuéiens a été utilisée dans [CS94] car elle permet d'avoir une présentation plus simple des logiques.

Nous avons besoin de rappeler quelques définitions sur les systèmes semi-Thuéiens. Pour un système $\mathcal{S}$, la relation de dérivation $\Rightarrow_\mathcal{S}$ vérifie: $u \Rightarrow_\mathcal{S} v$ ssi il existe $u_1, u_2 \in \Sigma^*$ et $u' \to v'$ tels que $v = u_1 v' u_2$ et $u = u_1 u' u_2$. On note $\Rightarrow^*_\mathcal{S}$ la fermeture réflexive et transitive de la relation $\Rightarrow_\mathcal{S}$. Pour $u \in \Sigma^*$, on note $\mathrm{L}_\mathcal{S}(u)$ l'ensemble de mots dérivables de u à savoir l'ensemble $\{v : u \Rightarrow^*_\mathcal{S} v\}$.

logique	$\mathrm{L}_{\mathcal{S}}(a)$
K	$\{a\}$
KT	$\{a, \epsilon\}$
KB	$\{a, \overline{a}\}$
KTB	$\{a, \overline{a}, \epsilon\}$
K4	$\{a\} \cdot \{a\}^*$
KT4 = S4	$\{a\}^*$
KB4	$\{a, \overline{a}\} \cdot \{a, \overline{a}\}^*$
K5	$(\{\overline{a}\} \cdot \{a, \overline{a}\}^* \cdot \{a\}) \cup \{a\}$
KT5 = S5	$\{a, \overline{a}\}^*$
K45	$(\{\overline{a}\}^* \cdot \{a\})^*$

TAB. 5.1 – *Langages réguliers de logiques modales standard*

Un système $\mathcal{S}$ est dit hors-contexte ssi toutes les règles sont de la forme $a \rightarrow u$ avec $a \in \Sigma$ et $u \in \Sigma^*$. Un système hors-contexte est dit régulier [resp. VPL] ssi pour $a \in \Sigma$, $\mathrm{L}_{\mathcal{S}}(a)$ forme un langage régulier [resp. un VPL [AM04]]. Les langages dans VPL sont reconnus par des automates à pile dont l'alphabet est divisé en trois parties disjointes: soit la lecture d'une lettre permet seulement d'empiler, soit elle permet seulement de dépiler soit elle laisse la pile intacte. Les langages acceptés par de tels automates sont fermés par union, intersection et complémentation [AM04].

Etant donné un système $\mathcal{S}$ sur un alphabet avec passé, on note $\overline{\mathcal{S}}$ la fermeture de $\mathcal{S}$ définie par

$$\mathcal{S} \cup \{\overline{u} \rightarrow \overline{v} : u \rightarrow v \in \mathcal{S}\}.$$

$\mathcal{S}$ est dit clos ssi $\mathcal{S} = \overline{\mathcal{S}}$.

Définition 5.1.2. Une logique grammaticale avec passé $\mathcal{L} = \langle \Sigma, \mathcal{C} \rangle$ est dite hors-contexte [resp. régulière, VPL] ssi il existe un système fini clos $\mathcal{S}$ hors-contexte [resp. régulier, VPL] tel que $\mathcal{C}$ est l'ensemble des structures de Kripke satisfaisant $\mathcal{S}$. ∇

Dans la suite on supposera que les systèmes caractérisant des logiques grammaticales avec passé sont clos. On notera aussi SAT($\mathcal{S}$) le problème de satisfaisabilité de la logique modale grammaticale caractérisée par le système $\mathcal{S}$.

5.1.2 De l'omniprésence des logiques régulières avec passé

Les logiques modales régulières sont omniprésentes et étudiées dans divers contextes. On en trouve parmi les logiques modales standard (voir la table 5.1), les logiques terminologiques [SCM03] et les logiques temporelles avec passé [Bla90].

De nombreuses autres logiques sont soit des logiques modales régulières, soit des logiques qui peuvent facilement être réduites à de telles logiques: logiques terminologiques [HS99] (avec hiérarchie de rôles et rôles transitifs), logiques épistémiques comme $\mathrm{S5}_m$(DE) [FHMV95], logiques bimodales pour des logiques

modales intuitionnistes de la forme **IntK**$_\Box$, **IntK4**$_\Box$, **IntS4**$_\Box$ [WZ97, WZ98], fragments de logiques pour le contrôle d'accès de systèmes distribués [ABLP93, Mas97]. A toute logique modale régulière, on peut ajouter l'opérateur universel (associé à la relation $S \times S$) tout en préservant la régularité (voir aussi la section 5.3.3).

Ce qui fait aussi l'intérêt des résultats qui vont être présentés est leur généralité indépendamment des motivations qui ont gouverné l'introduction d'une logique. Soit $\mathcal{S}$ le système hors-contexte ci-dessous:

$$\overline{\{a \to bab, b \to \epsilon, b \to c, b \to bb\}}.$$

On peut facilement montrer que

- $\mathrm{L}_{\mathcal{S}}(a) = \{b, c\}^* a \{b, c\}^*$,
- $\mathrm{L}_{\mathcal{S}}(b) = \{b, c\}^*$ et $\mathrm{L}_{\mathcal{S}}(c) = \{c\}$.

Par conséquent la logique caractérisée par $\mathcal{S}$ est régulière et tous les résultats qui vont suivre s'appliqueront à cette logique.

5.1.3 Traduction relationnelle vers la logique classique

La traduction de logiques modales vers la logique classique du premier ordre avec le but explicite de mécaniser le raisonnement dans de telles logiques a été introduite dans [Mor76]. Ce papier fondateur (mais rarement cité) distingue deux types de traduction: la traduction sémantique (voir aussi [Fin75, vB76, Moo77]) et la traduction syntaxique qui consiste à réifer les formules (c'est-à-dire à les transformer en des termes du premier ordre) et ensuite à transformer les axiomes et les règles d'inférence d'un calcul en des formules de la logique classique. Même si les logiques modales sont décidables, ces traductions, en général, ne permettent pas d'avoir de procédure de décision. L'article [ONdRG01] présente diverses méthodes de traduction pour les logiques modales, en particulier certaines introduites dans [vB85, Her89, DMP95, DG00d].

Soit $\mathcal{L} = \langle \Sigma, \mathcal{C} \rangle$ une logique modale avec passé hors-contexte caractérisée par le système $\mathcal{S}$. Nous allons rappeler ci-dessous la traduction relationnelle de $\mathcal{L}$ vers la logique classique. En fait, toute logique modale dont $\mathcal{C}$ est définissable dans la logique classique peut être traduite de façon analogue mais nous n'avons pas besoin d'une telle généralité ici.

A une formule ϕ construite sur les variables propositionnelles $\{\mathrm{p}_1, \ldots, \mathrm{p}_n\}$, on associe une formule $\psi_{\mathcal{L}} \wedge t(\phi, \mathsf{x}_0)$ où $\psi_{\mathcal{L}}$ ne dépend que de la logique $\mathcal{L}$ et $t(\phi, \mathsf{x}_0)$ ne dépend que de la formule ϕ. Le vocabulaire utilisé dans la logique classique est composé de symboles de prédicat binaires R_a pour $a \in \Sigma$ et de symboles de prédicat unaires $\mathsf{P}_1, \ldots, \mathsf{P}_n$.

La formule $\psi_{\mathcal{L}}$ est une conjonction $\psi_\Sigma \wedge \psi_{\mathcal{S}}$ où ψ_Σ exprime une contrainte relative au fait que Σ est un alphabet avec passé et $\psi_{\mathcal{S}}$ exprime les contraintes d'inclusion pour chaque règle $u \to v \in \mathcal{S}$.

$$\psi_\Sigma \stackrel{\text{def}}{=} \bigwedge_{a \in \Sigma^+} \forall \mathsf{x}, \mathsf{y}\ \mathsf{R}_{\overline{a}}(\mathsf{x}, \mathsf{y}) \Leftrightarrow \mathsf{R}_a(\mathsf{y}, \mathsf{x}).$$

A chaque règle $r = a \to a_1 \cdots a_m$ de $\mathcal{S}$, on associe la formule

$$\psi_r \stackrel{\text{def}}{=} \forall \mathsf{x}_1, \ldots, \mathsf{x}_{m+1}\ (\mathsf{R}_{a_1}(\mathsf{x}_1, \mathsf{x}_2) \wedge \cdots \wedge \mathsf{R}_{a_m}(\mathsf{x}_m, \mathsf{x}_{m+1})) \Rightarrow \mathsf{R}_a(\mathsf{x}_1, \mathsf{x}_{m+1}).$$

La formule $\psi_{\mathcal{S}}$ est définie comme la conjonction des formules ψ_r pour chaque règle $r \in \mathcal{S}$. La fonction de traduction t admet la définition inductive suivante:

- $t(\mathrm{p}_i, \mathsf{x}) \stackrel{\text{def}}{=} \mathsf{P}_i(\mathsf{x})$,
- t est homomorphique pour les opérateurs Booléens,
- $t([a]\varphi, \mathsf{x}) = \forall \mathsf{y}\ \mathsf{R}_a(\mathsf{x}, \mathsf{y}) \Rightarrow t(\varphi, \mathsf{y})$ où y est une nouvelle variable,
- $t(\langle a \rangle \varphi, \mathsf{x}) = \exists \mathsf{y}\ \mathsf{R}_a(\mathsf{x}, \mathsf{y}) \wedge t(\varphi, \mathsf{y})$ où y est une nouvelle variable.

Théorème 5.1.2. [vB76] ϕ est satisfaisable ssi $\psi_{\mathcal{L}} \wedge t(\phi, x_0)$ est satisfaisable.

En fait, la traduction est fidèle sémantiquement car il existe une correspondance très forte entre les modèles respectifs de la formule modale et ceux de sa traduction. En recyclant les variables, on peut n'utiliser que deux variables dans t, voir par exemple [Gab81]. Ainsi, si M est la longueur maximale d'un règle de $\mathcal{S}$, $\psi_{\mathcal{L}} \wedge t(\phi, \mathsf{x}_0)$ peut ne contenir que $max(M, 2)$ variables. Par contre, dès que $M \geq 2$, la formule $\psi_{\mathcal{L}} \wedge t(\phi, \mathsf{x}_0)$ n'appartient pas au fragment gardé de la logique classique qui est connu pour être décidable [ANvB98] en temps doublement exponentiel [Grä99b].

5.2 Logiques grammaticales et variantes de PDL

L'objet de cette section est d'expliquer pourquoi les logiques grammaticales peuvent naturellement être traduites vers des variantes de PDL.

5.2.1 Logique dynamique avec langages hors-contexte

Soient $\Pi_0 = \{c_1, c_2, \ldots\} \uplus \{\overline{c_1}, \overline{c_2}, \ldots\}$ un ensemble infini dénombrable de programmes atomiques (ou actions) et PROP l'ensemble usuel de variables propositionelles. Les formules de la logique PDL(HC) sont définies inductivement de la façon suivante:

$$\phi ::= \mathrm{p} \mid \neg\phi \mid \phi \wedge \phi \mid [\mathcal{A}(\Sigma \uplus \{\phi_1, \ldots, \phi_m\})]\phi \mid \langle \mathcal{A}(\Sigma \uplus \{\phi_1, \ldots, \phi_m\}) \rangle \phi$$

où $\mathcal{A}$ est un automate à pile non déterministe (avec acceptation sur état final) dont l'alphabet d'entrée fini est $\Sigma \uplus \{\phi_1, \ldots, \phi_m\}$ et $\Sigma \subset \Pi_0$. Dans la suite, on écrira simplement $\mathcal{A}$ au lieu de $\mathcal{A}(\Sigma \uplus \{\phi_1, \ldots, \phi_m\})$. Les formules dans l'alphabet de $\mathcal{A}$ permettent d'avoir l'opérateur "test" de PDL.

Les modèles de PDL(HC) sont les structures de Kripke de la forme

$$\langle S, (R_c)_{c \in \Pi_0}, V \rangle$$

telles que chaque relation binaire $R_{\overline{c_i}}$ est égale à $R_{c_i}^{-1}$ (relation inverse de R_{c_i}). La relation de satisfaction $\mathcal{M}, x \models \phi$ est définie en introduisant une fonction

d'interprétation $[\![\cdot]\!]$ pour laquelle les formules sont interprétées comme des sous-ensembles de S et les automates à pile sont interprétés comme des relations binaires sur S. Ainsi, par définition $\mathcal{M}, x \models \phi$ ssi $x \in [\![\phi]\!]$. La fonction d'interprétation $[\![\cdot]\!]$ admet la définition inductive suivante:

$$\begin{array}{ll}
[\![\mathrm{p}]\!] & = \{x \in S : \mathrm{p} \in V(x)\} \\
[\![\phi_1 \wedge \phi_2]\!] & = [\![\phi_1]\!] \cap [\![\phi_2]\!] \\
[\![\neg\phi]\!] & = S \setminus [\![\phi]\!] \\
[\![\langle\mathcal{A}\rangle\phi]\!] & = \{x : \exists y\ \langle x, y\rangle \in [\![\mathcal{A}]\!] \text{ et } y \in [\![\phi]\!]\} \\
[\![[\mathcal{A}]\phi]\!] & = \{x : \text{pour chaque } \langle x, y\rangle \in [\![\mathcal{A}]\!], \text{on a } y \in [\![\phi]\!]\} \\
[\![\mathcal{A}]\!] & = \{\langle x, y\rangle : \langle x, y\rangle \in R_u, u \in \mathrm{L}(\mathcal{A})\} \\
 & \text{avec } R_u = R_{a_1} \circ \cdots \circ R_{a_n} \text{ si } u = a_1 \cdots a_n, \\
 & R_\epsilon = \{\langle x, x\rangle : x \in S\}, \\
 & \text{pour chaque formule } \phi,\ R_\phi = \{\langle x, x\rangle : x \in [\![\phi]\!]\}.
\end{array}$$

Les problèmes pour PDL(HC) de satisfaisabilité, validité ou de satisfaisabilité globale sont définis de la façon usuelle. On note PDL(AUT) le fragment de PDL(HC) réduit aux automates sans pile et PDL(REG) la variante de PDL(AUT) où les automates sont remplacés par des expressions régulières. Ainsi PDL(REG) n'est rien d'autre que la logique dynamique propositionnelle PDL [FL79] avec passé (aussi noté CPDL pour "converse PDL", voir les sections 4.3 et 4.1.2). On note aussi PDL(VPL) le fragment de PDL(HC) réduit aux automates à pile introduits dans [AM04].

Théorème 5.2.1.

(I) PDL(HC) est indécidable, voir par exemple [HKT00].

(II) PDL(AUT) et PDL(REG) sont EXPTIME-complets [FL79, Pra79, Pra81].

(III) PDL(VPL) restreint aux programmes atomiques $\{c_1, c_2, \ldots\}$, c'est-à-dire sans passé, est décidable [LS06].

L'indécidabilité de PDL(HC) peut être raffinée en ne considérant que l'unique langage hors-contexte $\{a^i b a^i : i \geq 0\}$, voir par exemple [HKT00, chapitre 9].

Problème ouvert 29. Est-ce que le problème de satisfaisabilité pour PDL(VPL) (avec passé) est décidable? Si oui, quelle en est la complexité? ◯

5.2.2 Traductions

Revenons à notre classe initiale de logiques modales. Soit $\mathcal{L} = \langle\Sigma, \mathcal{C}\rangle$ une logique modale grammaticale hors-contexte caractérisée par le système clos $\mathcal{S}$. Comme $\mathcal{S}$ est hors-contexte, pour chaque lettre $a \in \Sigma$, le langage $\mathrm{L}_\mathcal{S}(a)$ est un langage hors-contexte. On note $\mathcal{A}_a$ un automate à pile avec alphabet d'entrée Σ tel que $\mathrm{L}(\mathcal{A}_a) = \mathrm{L}_\mathcal{S}(a)$. $\mathcal{A}_a$ peut être construit en temps polynomial en la taille de $\mathcal{S}$. Si $\mathcal{S}$ peut être vu syntaxiquement comme une grammaire linéaire à gauche ou à droite, alors $\mathcal{A}_a$ peut être aussi construit en temps polynomial en la taille de $\mathcal{S}$ et $\mathcal{A}_a$ est un automate sans pile.

Nous allons définir une traduction de $\mathcal{L}$ vers PDL(HC) qui préserve la satisfaisabilité. En fait, cette traduction est très simple. On note $T(\phi)$ la formule de PDL(HC) construite à partir de la formule ϕ de $\mathcal{L}$ telle que chaque occurrence de $[a]$ [resp. $\langle a\rangle$] est remplacé par $[\mathcal{A}_a]$ [resp. $\langle\mathcal{A}_a\rangle$]. Sans perte de généralité, on suppose ici que Σ^+ (la partie positive de Σ) est un sous-ensemble de $\{c_1, c_2, \ldots\}$.

Lemme 5.2.2. ϕ est satisfaisable ssi $T(\phi)$ est satisfaisable.

Preuve(idée): Nous donnons ici les grandes lignes de la preuve qui s'appuie sur la preuve du cas plus simple avec $\mathcal{L}$ une logique régulière sans passé [Dem01a]. On note $C_{\mathcal{S}}$ une application qui transforme chaque modèle avec passé $\mathcal{M} = \langle S, (R_a)_{a\in\Sigma}, V\rangle$ en un modèle $C_{\mathcal{S}}(\mathcal{M}) = \langle S, (R'_a)_{a\in\Sigma}, V\rangle$ tel que pour chaque lettre a,

$$R'_a \stackrel{\text{def}}{=} \bigcup_{u\in \mathrm{L}_{\mathcal{S}}(a)} R_u.$$

Cet opérateur est précisément la substitution inverse h^{-1} de [Cau96, Cau03] quand la substitution étendue est définie par $a \mapsto \mathrm{L}_{\mathcal{S}}(a)$. On peut montrer les propriétés suivantes:

- pour chaque lettre a, $R_a \subseteq R'_a$,
- $C_{\mathcal{S}}(\mathcal{M}) = C_{\mathcal{S}}(C_{\mathcal{S}}(\mathcal{M}))$,
- $C_{\mathcal{S}}$ est monotone,
- $C_{\mathcal{S}}(\mathcal{M})$ est un modèle de $\mathcal{L}$.

Ainsi $\mathcal{C}$ couvre exactement la classe des modèles $\mathcal{M} = \langle S, (\llbracket\mathcal{A}_a\rrbracket)_{a\in\Sigma}, V\rangle$ obtenus à partir de tous les modèles $\langle S, (R_c)_{c\in\Pi_0}, V\rangle$ de PDL(HC) en extrayant les relations de la forme $\llbracket\mathcal{A}_a\rrbracket$ pour $a \in \Sigma$.

Lorsque ϕ est satisfaisable dans $\mathcal{M} = \langle S, (R_a)_{a\in\Sigma}, V\rangle$, la formule $T(\phi)$ est satisfaisable dans le modèle $\mathcal{M}' = \langle S, (R_c)_{c\in\Pi_0}, V\rangle$ où $\mathcal{M}$ et $\mathcal{M}'$ coïncident sur les relations indicées par $a \in \Sigma$. De même, si $T(\phi)$ est satisfaisable dans $\langle S, (R_c)_{c\in\Pi_0}, V\rangle$ alors ϕ est satisfaisable dans $\langle S, (\llbracket\mathcal{A}_a\rrbracket)_{a\in\Sigma}, V\rangle$ et on a la garantie qu'il s'agit d'un modèle de $\mathcal{L}$. CQFD

Ce résultat généralise les traductions entre certaines logiques modales et PDL, voir par exemple [FL79, Tuo90, Sch91].

Corollaire 5.2.3. Soit $\mathcal{L}$ une logique modale grammaticale régulière avec passé [resp. VPL sans passé]. Les problèmes de satisfaisabilité, validité et satisfaisabilité globale sont dans EXPTIME [resp. sont décidables].

Un énoncé analogue au corollaire 5.2.3 restreint aux logiques modales grammaticales régulières sans passé est établi dans [Dem01a]. L'ajout du passé ne pause pas de problème et ce sont les résultats de [LS06] qui nous permettent d'aller aussi plus loin.

Soit $\mathcal{L}$ une logique grammaticale caractérisée par le système

$$\mathcal{S} = \{a \to a^{k_1}, \ldots, a \to a^{k_n}, \overline{a} \to \overline{a}^{k_1}, \ldots, \overline{a} \to \overline{a}^{k_n}\}$$

avec $n \geq 1$ et $0 \leq k_1 < k_2 < \cdots < k_n$. Si $k_1 > 0$ alors

$$\mathrm{L}_{\mathcal{S}}(a) = \{a\} \cdot \{a^{k_1-1}, \ldots, a^{k_n-1}\}^*.$$

Par conséquent $\mathcal{L}$ est régulière et SAT($\mathcal{S}$) est dans EXPTIME. Nous capturons ainsi un nombre non négligeable de logiques modales standard. Si $n = 1$ et $k_1 = 1$ [resp. $n = 1$ et $k_1 = 0$] alors $\mathcal{L}$ est la logique K [resp. T], si $n = 1$ et $k_1 = 2$ [$n = 2$, $k_1 = 0$, $k_2 = 2$] alors $\mathcal{L}$ est la logique K4 [resp. S4].

La logique NIL introduite dans [Vak87] est caractérisée par un système $\mathcal{S}_{\mathrm{NIL}}$ dont l'ensemble de règles ci-dessous est construit sur un alphabet avec passé dont $\Sigma^+ = \{\mathrm{fin}, \mathrm{sim}\}$ (on omet la fermeture de quelques règles):

- $\mathrm{fin} \rightarrow \mathrm{fin} \cdot \mathrm{fin}$, $\mathrm{fin} \rightarrow \epsilon$,
- $\mathrm{sim} \rightarrow \overline{\mathrm{sim}}$, $\mathrm{sim} \rightarrow \epsilon$,
- $\mathrm{sim} \rightarrow \overline{\mathrm{fin}} \cdot \mathrm{sim} \cdot \mathrm{fin}$.

On peut facilement montrer que $\mathrm{L}_{\mathcal{S}}(\mathrm{sim}) = \{\overline{\mathrm{fin}}\}^* \cdot \{\mathrm{sim}, \overline{\mathrm{sim}}, \epsilon\} \cdot \{\mathrm{fin}\}^*$ ce qui implique que NIL est une logique régulière. Par conséquent SAT(NIL) est dans EXPTIME, ce qui améliore la borne NEXPTIME établie dans [Vak87]. En fait, en utilisant un algorithme à la Ladner assez complexe on peut faire mieux.

Théorème 5.2.4. [Dem00b] SAT(NIL) est PSPACE-complet.

5.2.3 Propriété d'uniformité

La preuve du corollaire 5.2.3 et certaines variantes permettent aussi de caractériser la complexité d'autres problèmes logiques, en particulier en tenant compte de l'uniformité de la preuve. On se restreint ici au cas où Σ est sans passé.

Un système semi-Thuéien $\mathcal{S}$ est dit linéaire [resp. linéaire à droite, linéaire à gauche] ssi il existe une façon de partitionner l'alphabet $\Sigma = \Sigma_{NT} \uplus \Sigma_T$ telle que $\mathcal{S}$ peut être vu syntaxiquement comme une grammaire linéaire [resp. linéaire à droite, linéaire à gauche]. Σ_T représente alors l'ensemble des symboles terminaux et Σ_{NT} l'ensemble des symboles non terminaux. L'ensemble des systèmes semi-Thuéiens finis hors-contexte [resp. linéaires, linéaires à droite, linéaires à gauche] est noté STHC [resp. STLIN, STLIND, STLING]. On peut décider si un système semi-Thuéien appartient à un de ces ensembles mais le problème qui consiste à déterminer si un système hors-contexte est régulier est indécidable, voir par exemple [MS97, page 31].

Soit C un ensemble de systèmes semi-Thuéiens parmi STHC, STLIN, STLIND, STLING. Le problème général de satisfaisabilité pour C, noté PGS(C), consiste à décider, étant donnés un système $\mathcal{S}$ de C et une formule ϕ de la logique modale grammaticale $\mathcal{L}$ caractérisée par $\mathcal{S}$, si ϕ est satisfaisable.

Théorème 5.2.5.

(I) PGS(HC) est indécidable [Bal98].

(II) PGS(STLIND) est décidable [Bal98].

(III) PGS(STLIND) et PGS(STLING) sont EXPTIME-complets [Dem01a].

La preuve de décidabilité de PGS(STLIND) dans [Bal98] utilise un algorithme non déterministe doublement exponentiel alors que notre borne supérieure pour PGS(STLIND) et PGS(STLING) est une conséquence du corollaire 5.2.3. La EXPTIME-dureté est obtenue en montrant que la logique caractérisée par le système $\{a \rightarrow ab\}$ a un problème de satisfaisabilité déjà EXPTIME-difficile. D'autres résultats de décidabilité/indécidabilité peuvent être trouvés dans [Dem01a].

5.3 Une autre traduction vers la logique classique

5.3.1 Le cas général

Soit $\mathcal{L} = \langle \Sigma, \mathcal{C} \rangle$ une logique modale grammaticale hors-contexte caractérisée par le système clos $\mathcal{S}$. Σ est un alphabet avec passé. Soit ϕ une formule de ML(Σ) sous forme normale négative (le symbole de négation n'apparaît que devant des variables propositionnelles). On peut facilement se convaincre que cette hypothèse n'enlève rien à la portée de la méthode. A chaque lettre $a \in \Sigma$, on associe un automate à pile $\mathcal{A}_a$ telle que $\mathrm{L}_{\mathcal{S}}(a) = \mathrm{L}(\mathcal{A}_a)$. Comme $\mathrm{L}_{\mathcal{S}}(a)$ est hors-contexte, un tel automate existe toujours.

On rappelle brièvement qu'un automate à pile non déterministe $\mathcal{A}$ est une structure $\langle Q, \Sigma, \Gamma, \Delta, Z, q_0, q_f \rangle$ où

- Q est l'ensemble fini des états, $q_0 \in Q$ est l'état initial et q_f est l'état accepteur,
- Σ est l'alphabet d'entrée, Γ est l'alphabet de pile et Z est le symbole initial de pile,
- Δ est la relation de transition définie comme une sous-ensemble de

$$Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \times Q \times \Gamma^*.$$

Nous n'allons pas rappeler complétement les notions de configuration, dérivation, exécution etc. mais une configuration est un élément de $Q \times \Sigma^* \times \Gamma^*$ et on utilisera le mode d'acceptation par état accepteur. Un élément de Δ sera aussi noté $q \xrightarrow{a,b,u} q'$.

Pour traduire une formule ϕ, le vocabulaire utilisé dans la logique classique est composé:

- de symboles de prédicat binaires R_a pour $a \in \Sigma$,
- de symboles de prédicat unaires $\mathsf{P}_1, \ldots, \mathsf{P}_n$ si ϕ est construite sur les variables propositionnelles $\{\mathrm{p}_1, \ldots, \mathrm{p}_n\}$,
- de symboles de prédicat binaires P_q^ψ pour chaque sous-formule $[a]\psi$ de ϕ et pour chaque état q de $\mathcal{A}_a$,
- de symboles de fonction unaires f_b^ψ pour chaque sous-formule $[a]\psi$ de ϕ et pour chaque lettre b de l'alphabet de pile de $\mathcal{A}_a$,
- d'une constante ϵ.

Un contenu de pile $b_1 \cdots b_k$ dans $\mathcal{A}_a$ pour la sous-formule $[a]\psi$ est représenté par le terme

$$\mathsf{f}_{b_1}^\psi(\mathsf{f}_{b_2}^\psi(\ldots \mathsf{f}_{b_k}^\psi(\epsilon)\ldots)).$$

Par convention, b_1 est en tête de pile.

La traduction de la formule ϕ, notée $T(\phi)$, utilise deux fonctions auxiliaires.

- La fonction de traduction $t(\psi, \mathsf{x})$ a pour premier argument une sous-formule de ϕ et pour second argument une variable parmi $\{\mathsf{x}_0, \mathsf{x}_1\}$. Nous allons en effet recycler les variables en utilisant la technique de [Gab81]. Cette fonction est une variante de la traduction relationnelle classique.
- Une fonction $t_{acc}([a]\psi)$ qui prend en argument une sous-formule de la forme $[a]\psi$, qui utilise $t(\cdot,\cdot)$ et qui simule les calculs acceptants de l'automate à pile $\mathcal{A}_a$.

Par définition, $T(\phi)$ est égale à la formule ci-dessous:

$$\Big(\bigwedge_{a\in\Sigma^+} \forall \mathsf{x}_0, \mathsf{x}_1\ \mathsf{R}_{\overline{a}}(\mathsf{x}_0, \mathsf{x}_1) \Leftrightarrow \mathsf{R}_a(\mathsf{x}_1, \mathsf{x}_0)\Big) \wedge t(\phi, \mathsf{x}_0) \wedge \bigwedge_{[a]\psi\in \mathrm{sub}(\phi)} t_{acc}([a]\psi).$$

La fonction de traduction t admet la définition suivante pour $i \in \{0, 1\}$:

- $t(\mathrm{p}_j, \mathsf{x}_i) \stackrel{\text{def}}{=} \mathsf{P}_j(\mathsf{x}_i)$,
- t est homomorphique pour les opérateurs Booléens,
- $t(\langle a\rangle\varphi, \mathsf{x}_i) = \exists \mathsf{x}_{1-i}\ \mathsf{R}_a(\mathsf{x}_i, \mathsf{x}_{i-1}) \wedge t(\varphi, \mathsf{x}_{1-i})$ (recyclage des variables en alternant x_i et x_{1-i}),
- $t([a]\varphi, \mathsf{x}_i) = \mathsf{P}^{\varphi}_{q_0}(\mathsf{x}_i, \mathsf{f}^{\varphi}_Z(\epsilon))$ où q_0 est l'état initial de $\mathcal{A}_a$ et Z est le symbole initial de pile.

Ainsi, la fonction t est presque définie comme la traduction relationnelle sauf pour les formules de la forme $[a]\varphi$. La seule variable libre dans $t(\varphi, \mathsf{x}_i)$ est x_i.

Chaque formule $t_{acc}([a]\psi)$ est la conjonction des formules suivantes:

- Pour chaque transition $q \xrightarrow{a,b,u} q'$ de $\mathcal{A}_a$ avec $a \in \Sigma$ et $u = b_1 \cdots b_k$:

$$\forall \mathsf{x}_0, \mathsf{x}_1, \mathsf{x}_2\ \mathsf{R}_a(\mathsf{x}_0, \mathsf{x}_1) \Rightarrow (\mathsf{P}^{\psi}_q(\mathsf{x}_0, \mathsf{f}^{\psi}_b(\mathsf{x}_2)) \Rightarrow \mathsf{P}^{\psi}_{q'}(\mathsf{x}_1, \mathsf{f}^{\psi}_{b_1}(\mathsf{f}^{\psi}_{b_2}(\ldots \mathsf{f}^{\psi}_{b_k}(\mathsf{x}_2)\ldots))))$$

- Pour chaque transition $q \xrightarrow{\epsilon,b,u} q'$ de $\mathcal{A}_a$:

$$\forall \mathsf{x}_0, \mathsf{x}_1\ (\mathsf{P}^{\psi}_q(\mathsf{x}_0, \mathsf{f}^{\psi}_b(\mathsf{x}_1)) \Rightarrow \mathsf{P}^{\psi}_{q'}(\mathsf{x}_0, \mathsf{f}^{\psi}_{b_1}(\mathsf{f}^{\psi}_{b_2}(\ldots \mathsf{f}^{\psi}_{b_k}(\mathsf{x}_1)\ldots))))$$

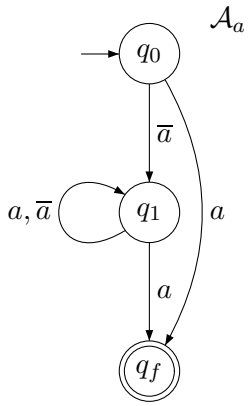
- Quand l'état accepteur est atteint, la traduction de ψ doit être vérifiée:

$$\forall \mathsf{x}_0, \mathsf{x}_1\ \mathsf{P}^{\psi}_{q_f}(\mathsf{x}_0, \mathsf{x}_1) \Rightarrow t(\psi, \mathsf{x}_0)$$

Exemple 5.3.1. Considérons la logique régulière K5 (voir la table 5.1). La figure 5.1 contient en partie gauche l'automate (sans pile) $\mathcal{A}_a$. On supposera que la pile contient constamment le symbole initial Z. A droite de la table se trouve la formule $t_{acc}([a]\mathrm{p})$.

∇

Lemme 5.3.1. Soit M la taille maximale des automates à piles $\mathcal{A}_a$ pour $a \in \Sigma$. Alors la taille de $T(\phi)$ est polynomiale en $|\phi| + M$.



$$\forall \mathsf{x}_0, \mathsf{x}_1, \mathsf{x}_2\ [\ \mathsf{R}_a(\mathsf{x}_0, \mathsf{x}_1) \Rightarrow (\mathsf{P}^{\mathrm{p}}_{q_0}(\mathsf{x}_0, \mathsf{f}_Z(\mathsf{x}_2)) \Rightarrow \mathsf{P}^{\mathrm{p}}_{q_f}(\mathsf{x}_1, \mathsf{f}_Z(\mathsf{x}_2)))]$$
$$\forall \mathsf{x}_0, \mathsf{x}_1, \mathsf{x}_2\ [\ \mathsf{R}_{\overline{a}}(\mathsf{x}_0, \mathsf{x}_1) \Rightarrow (\mathsf{P}^{\mathrm{p}}_{q_0}(\mathsf{x}_0, \mathsf{f}_Z(\mathsf{x}_2)) \Rightarrow \mathsf{P}^{\mathrm{p}}_{q_1}(\mathsf{x}_1, \mathsf{f}_Z(\mathsf{x}_2)))]$$

$$\forall \mathsf{x}_0, \mathsf{x}_1, \mathsf{x}_2\ [\ \mathsf{R}_a(\mathsf{x}_0, \mathsf{x}_1) \Rightarrow (\mathsf{P}^{\mathrm{p}}_{q_1}(\mathsf{x}_0, \mathsf{f}_Z(\mathsf{x}_2)) \Rightarrow \mathsf{P}^{\mathrm{p}}_{q_1}(\mathsf{x}_1, \mathsf{f}_Z(\mathsf{x}_2)))]$$
$$\forall \mathsf{x}_0, \mathsf{x}_1, \mathsf{x}_2\ [\ \mathsf{R}_{\overline{a}}(\mathsf{x}_0, \mathsf{x}_1) \Rightarrow (\mathsf{P}^{\mathrm{p}}_{q_1}(\mathsf{x}_0, \mathsf{f}_Z(\mathsf{x}_2)) \Rightarrow \mathsf{P}^{\mathrm{p}}_{q_1}(\mathsf{x}_1, \mathsf{f}_Z(\mathsf{x}_2)))]$$

$$\forall \mathsf{x}_0, \mathsf{x}_1, \mathsf{x}_2\ [\ \mathsf{R}_a(\mathsf{x}_0, \mathsf{x}_1) \Rightarrow (\mathsf{P}^{\mathrm{p}}_{q_1}(\mathsf{x}_0, \mathsf{f}_Z(\mathsf{x}_2)) \Rightarrow \mathsf{P}^{\mathrm{p}}_{q_f}(x_1, \mathsf{f}_Z(\mathsf{x}_2)))]$$

$$\forall \mathsf{x}_0, \mathsf{x}_1\ [\mathsf{P}^{\mathrm{p}}_{q_f}(\mathsf{x}_0, \mathsf{f}_Z(\mathsf{x}_1)) \Rightarrow \mathsf{P}(\mathsf{x}_0)]$$

FIG. 5.1 – $t_{acc}([a]\mathrm{p})$ *pour la logique K5*

Théorème 5.3.2. ϕ est satisfaisable ssi $T(\phi)$ est satisfaisable.

Avant de montrer le théorème qui est une légère variante de la preuve de [DdN05, théorème 3.3], nous allons établir une propriété intéressante. Un modèle pour la logique classique est une structure de la forme $\langle S, V\rangle$ où S est un ensemble non vide et V interprète les symboles de prédicat n-aires par des sous-ensembles de S^n et les symboles de fonction n-aires par des fonctions de profil $S^n \to S$. Le vocabulaire du modèle détermine sa forme exactement. Nous allons à présent définir une suite croissante de vocabulaires. Soit $[a_1]\psi_1, \ldots, [a_m]\psi_n$ une suite de sous-formules de ϕ qui contient toutes les sous-formules de la forme $[a]\psi$ et dont $i < j$ implique $[a_j]\psi_j$ n'est pas une sous-formule de $[a_i]\psi_i$. Le vocabulaire $\mathtt{voc}_0$ contient R_a pour $a \in \Sigma$, et P_i pour $i \in \{1, \ldots, n\}$. Ainsi un modèle du premier ordre sur le vocabulaire $\mathtt{voc}_0$ n'est rien d'autre qu'un modèle de Kripke sans la satisfaction de la contrainte de passé. Pour $i \in \{1, \ldots, m\}$, nous définissons le vocabulaire $\mathtt{voc}_i$ en ajoutant à $\mathtt{voc}_{i-1}$, les symboles $\mathsf{P}^{\psi_i}_q$ pour chaque état q de $\mathcal{A}_{a_i}$, les symboles $\mathsf{f}^{\psi_i}_b$ pour chaque lettre b de l'alphabet de pile de $\mathcal{A}_{a_i}$, et la constante ϵ.

Lemme 5.3.3. Soient $i \in \{0, \ldots, m-1\}$ et $\mathcal{M} = \langle S, V\rangle$ un modèle du premier ordre sur le vocabulaire $\mathtt{voc}_i$. Il existe une extension $\mathcal{M}' = \langle S', V'\rangle$ de $\mathcal{M}$ sur le vocabulaire $\mathtt{voc}_{i+1}$ ($S \subseteq S'$ et l'interprétation des symboles de $\mathtt{voc}_i$ reste inchangée) telle que pour $x \in S$, la propriété $(\star)$ est vérifiée:

($\star$) Si pour $b_1 \cdots b_l \in \mathrm{L}_{\mathcal{S}}(a_{i+1})$, et pour chaque séquence de S avec $\langle x, x_1\rangle \in V(\mathsf{R}_{b_1})$, …, $\langle x_{l-1}, x_l\rangle \in V(\mathsf{R}_{b_l})$, nous avons $\mathcal{M} \models_{[\mathsf{x}_j \mapsto x_l]} t(\psi_{i+1}, \mathsf{x}_j)$, alors $\mathcal{M}' \models_{[\mathsf{x}_j \mapsto x]} t_{acc}([a_{i+1}]\psi_{i+1}) \wedge \mathsf{P}^{\psi_{i+1}}_{q_0}(\mathsf{x}_j, \mathsf{f}^{\psi_{i+1}}_Z(\epsilon))$ où q_0 est l'état initial de $\mathcal{A}_{a_{i+1}}$.

Preuve: Nous notons $\mathrm{L}(\mathcal{A}^{q,u}_{a_{i+1}})$ l'ensemble des mots acceptés par $\mathcal{A}_{a_{i+1}}$ à partir de l'état q et avec u sur la pile. $\mathrm{L}(\mathcal{A}_{a_{i+1}})$ est précisément $\mathrm{L}(\mathcal{A}^{q_0,Z}_{a_{i+1}})$. Posons $S' = S \cup \Gamma^*$ où Γ est l'alphabet de pile de $\mathcal{A}_{a_{i+1}}$. Les symboles de fonction ont l'interprétation suivante:

- $V'(\epsilon) = \epsilon$,
- $V'(\mathsf{f}^{\psi_i}_b)(x) = bx$ pour $x \in \Gamma^*$, sinon $V'(\mathsf{f}^{\psi_i}_b)(x)$ prend une valeur arbitraire.

Le symbole de prédicat $\mathsf{P}^{\psi_{i+1}}_q$ a l'interprétation suivante pour q un état de contrôle de $\mathcal{A}_{a_{i+1}}$. $\langle x, u\rangle \in V'(\mathsf{P}^{\psi_{i+1}}_q)$ ssi

- $x \in S$, $u \in \Gamma^*$ et
- pour $b_1 \cdots b_l \in \mathrm{L}(\mathcal{A}^{q,u}_{a_{i+1}})$, pour chaque séquence de S avec $\langle x, x_1\rangle \in V(\mathsf{R}_{b_1})$, …, $\langle x_{l-1}, x_l\rangle \in V(\mathsf{R}_{b_l})$, nous avons $\mathcal{M} \models_{[\mathsf{x}_j \mapsto x_l]} t(\psi_{i+1}, \mathsf{x}_j)$.

Il n'est pas difficile de vérifier que $\mathcal{M}'$ satisfait $t_{acc}([a_{i+1}]\psi_{i+1}) \wedge \mathsf{P}_{q_0}^{\psi_i}(\mathsf{x}_j, \mathsf{f}_Z^{\psi_{i+1}}(\epsilon))$. CQFD

Preuve: (théorème 5.3.2) Supposons d'abord que ϕ soit satisfaisable. Soit $\mathcal{M} = \langle S, (R_a)_{a\in\Sigma}, V\rangle$ un modèle de $\mathcal{L}$ et $x_0 \in S$ tels que $\mathcal{M}, x_0 \models \phi$. On note $\mathcal{M}_0 = \langle S_0, V_0\rangle$ le modèle de la logique classique correspond à $\mathcal{M}$. Par ailleurs on note $\mathcal{M}_1 = \langle S_1, V_1\rangle, \ldots, \mathcal{M}_m = \langle S_m, V_m\rangle$ la séquence de modèles obtenue en appliquant le lemme 5.3.3. La structure $\mathcal{M}_l$ est notre modèle final. Remarquons que pour $a \in \Sigma$, $V_0(\mathsf{R}_a) = \cdots = V_l(\mathsf{R}_a)$ et pour $i \in \{1, \ldots, n\}$, $V_0(\mathsf{P}_i) = \cdots = V_m(\mathsf{P}_i)$. De plus, pour $j \in \{1, \ldots, m\}$ et pour tout état q de $\mathcal{A}_{a_j}$, $V_j(\mathsf{P}_q^{\psi_j}) = \cdots = V_m(\mathsf{P}_q^{\psi_j})$. Une propriété analogue peut être montrée pour les symboles de fonction unaires.

On montre alors par induction structurelle que pour toute sous-formule ψ de ϕ et pour $x \in S$, si $\mathcal{M}, x \models \psi$ alors $\mathcal{M}_l \models_{[\mathsf{x}_j\mapsto x]} t(\psi, \mathsf{x}_j)$. On établit ici la propriété pour les sous-formules gouvernées par des opérateurs modaux.

1. Soit $\psi = [a_k]\psi_k$ pour un $k \in \{1, \ldots, m\}$ et donc $t(\psi, \mathsf{x}_j) = \mathsf{P}_{q_0}^{\psi_k}(\mathsf{x}_j, \mathsf{f}_Z^{\psi_k}(\epsilon))$ pour l'état initial q_0 de $\mathcal{A}_{a_k}$. Supposons que $\mathcal{M}, x \models \psi$. Soient $b_1 \cdots b_l \in \mathrm{L}(\mathcal{A}_{a_k})$ et $\langle x, x_1\rangle \in V_m(\mathsf{R}_{b_1})$, …, $\langle x_{l-1}, x_l\rangle \in V_m(\mathsf{R}_{b_l})$ décrivant un chemin dans tous les modèles $\mathcal{M}_0$, …, $\mathcal{M}_m$. Comme $\mathcal{M}$ satisfait $\mathcal{S}$, $\langle x, x_l\rangle \in V_m(\mathsf{R}_{a_k})$. Par conséquent, $\mathcal{M}, x_l \models \psi_k$. Par hypothèse d'induction, on a $\mathcal{M}_m \models_{[\mathsf{x}_{1-j}\mapsto x_l]} t(\psi_k, \mathsf{x}_{1-j})$. En fait, on peut aussi vérifier que $\mathcal{M}_k \models_{[\mathsf{x}_{1-j}\mapsto x_l]} t(\psi_k, \mathsf{x}_{1-j})$. Par conséquent grâce au lemme 5.3.3, $\mathcal{M}_k \models_{[\mathsf{x}_j\mapsto x]} t_{acc}([a_k]\psi_k) \wedge \mathsf{P}_{q_0}^{\psi_k}(\mathsf{x}_j, \mathsf{f}_Z^{\psi_k}(\epsilon))$ où q_0 est l'état initial de $\mathcal{A}_{a_k}$. Comme $\mathcal{M}_m$ est une extension conservatrice de $\mathcal{M}_k$, $\mathcal{M}_m \models_{[\mathsf{x}_j\mapsto x]} \mathsf{P}_{q_0}^{\psi_k}(\mathsf{x}_j, \mathsf{f}_Z^{\psi_k}(\epsilon))$.
2. Soit $\psi = \langle a\rangle\psi'$. Supposons que $\mathcal{M}, x \models \psi$. Il existe $y \in S$ tel que $\langle x, y\rangle \in R_a$ et $\mathcal{M}, y \models \psi'$. Par hypothèse d'induction et comme $R_a = V_m(\mathsf{R}_a)$, $\mathcal{M}_m \models_{[\mathsf{x}_j\mapsto x]} \exists \mathsf{x}_{1-j}\, \mathsf{R}_a(\mathsf{x}_j, \mathsf{x}_{1-j}) \wedge t(\psi', \mathsf{x}_{1-j})$.

Supposons à présent que $T(\phi)$ soit satisfaisable. Il existe donc un modèle $\mathcal{M} = \langle S, V\rangle$ du premier ordre sur $\mathtt{voc}_m$ tel que $\mathcal{M} \models_{[\mathsf{x}_0\mapsto x_0]} T(\phi)$. Soit $\mathcal{M}' = \langle S, (R'_a)_{a\in\Sigma}, V'\rangle$ le modèle de Kripke obtenu à partir de $\mathcal{M}$ avec le vocabulaire $\mathtt{voc}_0$. Notons ensuite $\mathcal{M}'' = C_{\mathcal{S}}(\mathcal{M}') = \langle S, (R''_a)_{a\in\Sigma}, V'\rangle$ la fermeture du modèle $\mathcal{M}'$ (voir la preuve du lemme 5.2.2). Comme vu précédemment, $\mathcal{M}''$ est un modèle de Kripke avec passé satisfaisant $\mathcal{S}$. Montrons que pour toute sous-formule ψ de ϕ, si $\mathcal{M} \models_{[\mathsf{x}_j\mapsto x]} t(\psi, \mathsf{x}_j)$ alors $\mathcal{M}'', x \models \psi$.

1. Supposons que $\mathcal{M} \models_{[\mathsf{x}_j\mapsto x]} \exists \mathsf{x}_{1-j}\, \mathsf{R}_a(\mathsf{x}_j, \mathsf{x}_{1-j}) \wedge t(\psi', \mathsf{x}_{1-j})$. Il existe donc y tel que $\langle x, y\rangle \in V(\mathsf{R}_a)$ tel que $\mathcal{M} \models_{[\mathsf{x}_{1-j}\mapsto y]} t(\psi', \mathsf{x}_{1-j})$. Par hypothèse d'induction et comme l'opérateur $C_{\mathcal{S}}$ est conservateur, on obtient $\mathcal{M}'', x \models \langle a\rangle\psi'$.
2. Supposons à présent que $\mathcal{M} \models_{[\mathsf{x}_j\mapsto x]} \mathsf{P}_{q_0}^{\psi}(\mathsf{x}_j, \mathsf{f}_Z^{\psi}(\epsilon))$ où q_0 est l'état initial de $\mathcal{A}_a$. Comme $\mathcal{M} \models t_{acc}([a]\psi)$ on peut établir que pour tout $b_1 \cdots b_l \in \mathrm{L}_{\mathcal{S}}(a)$, et pour chaque séquence de S avec $\langle x, x_1\rangle \in V(\mathsf{R}_{b_1})$, …, $\langle x_{l-1}, x_l\rangle \in V(\mathsf{R}_{b_l})$, nous avons $\mathcal{M} \models_{[\mathsf{x}_j\mapsto x_l]} t(\psi, \mathsf{x}_j)$. Supposons que $\langle x, y\rangle \in R''_a$. Par définition de $C_{\mathcal{S}}$ il existe $u \in \mathrm{L}_{\mathcal{S}}(a)$ tel que $\langle x, y\rangle \in R'_u$. Par conséquent,

$\mathcal{M} \models_{[\mathsf{x}_j \mapsto y]} t(\psi, \mathsf{x}_j)$. Par hypothèse d'induction, $\mathcal{M}'', y \models \psi$. Ainsi $\mathcal{M}'', x \models [a]\psi$.

CQFD

Contrairement à la traduction relationnelle, cette traduction n'est pas sémantiquement exactement fidèle ne serait-ce par l'ajout de nouveaux symboles de prédicat pour coder le fonctionnement des automates à pile. Par contre, seules trois variables sont utilisées.

5.3.2 Traduction vers le fragment gardé

La traduction présentée dans la section 5.3.1 est assurément plus complexe que la traduction relationnelle même si elle s'appuie sur celle-ci. Cependant le codage des calculs acceptants des automates à pile est relativement explicite. Le mérite principal de cette traduction devient évident lorsque $\mathcal{L}$ est une logique régulière. En effet, dans ce cas les automates $\mathcal{A}_a$ n'utilisent pas la pile et par conséquent la constante ϵ et les symboles fonctionnels unaires deviennent inutiles. On peut alors supposer les symboles de prédicats P_q^ψ unaires. Par exemple, dans la définition de $t_{acc}([a]\psi)$, pour chaque transition $q \xrightarrow{a} q'$ de $\mathcal{A}_a$ avec $a \in \Sigma$ on considère plus simplement la formule

$$\forall \mathsf{x}_0, \mathsf{x}_1 \; \mathsf{R}_a(\mathsf{x}_0, \mathsf{x}_1) \Rightarrow (\mathsf{P}_q^\psi(\mathsf{x}_0) \Rightarrow \mathsf{P}_{q'}^\psi(\mathsf{x}_1))$$

En fait, on peut facilement montrer qu'alors $T(\phi)$ appartient au fragment gardé de la logique classique introduit dans [ANvB98], ce qui généralise les traductions ad hoc dans [dN99, dN01] pour les logiques S4 et K5. Dans ce fragment sans symbole fonctionnel, les quantifications sont de la forme

$$\forall \overline{\mathsf{x}} \, (\psi \Rightarrow \psi') \quad \exists \overline{\mathsf{x}} \psi \wedge \psi'$$

où $\overline{\mathsf{x}}$ est une séquence de variables, ψ est une formule atomique (la "garde") et toutes les variables libres de ψ' apparaissent dans ψ. On peut même faire mieux puisque $T(\phi)$ appartient alors à l'intersection du fragment gardé (noté GF dans la suite) et du fragment avec deux variables individuelles seulement (noté FO2) [Gab81, GKV97].

Le fragment gardé possède non seulement de nombreuses propriétés logiques intéressantes [ANvB98] mais le problème de satisfaisabilité pour GF restreint aux symboles de prédicats d'arité au plus $k \geq 2$ (fixé) est EXPTIME-complet [Grä99b]. Ainsi, sa complexité dans le pire des cas est comparable à celle du problème de satisfaisabilité globale pour K [Spa93a]. GF admet aussi diverses techniques de mécanisation, soit avec des calculs à base de la méthode de résolution [dN98, GdN99] soit avec des calculs par tableaux [Hla02]. Un démonstrateur pour FO2 est présenté dans [MMS99]. Ainsi, l'existence d'une traduction simple entre une logique modale et GF2 permet d'avoir une procédure de décision pour cette logique.

Des formalismes plus riches que GF2 mais décidables existent mais GF2 est le plus petit fragment qui nous permet de rester dans EXPTIME. En effet, dans [GW99] GF est étendu avec des opérateurs de point-fixe (noté μGF) et cette extension est montrée 2EXPTIME-complète dans [GW99] (GF est déjà 2EXPTIME-complet). Par contre, μGF restreint à deux variables, noté μGF2 est aussi dans EXPTIME [GW99]. On peut noter par ailleurs que μGF n'a pas la propriété du modèle fini contrairement à GF. Lorsqu'on ne souhaite pas avoir toute la puissance des opérateurs de point-fixe, on peut supposer que certains symboles de prédicat binaires vérifient des propriétés prédéfinies. Cependant GF2 avec des relations transitives prédéfinies est indécidable (voir la succession de papiers qui raffinent ce résultat: [Grä99b], [GMV99], [Kie05]). Une façon de regagner la décidabilité est d'imposer que les symboles de prédicat non unaires ne peuvent apparaître que dans la garde donnant naissance à divers fragments dits monadiques. Ainsi le fragment monadique de GF2 (noté MGF2) avec des relations transitives prédéfinies est montré décidable dans [GMV99] et en fait le problème de satisfaisabilité est 2EXPTIME-complet [ST04, Kie06]. L'article [GMV99] va plus loin: MGF2 avec des relations prédéfinies vérifiant une propriété de fermeture définissable dans MSO est décidable en effectuant une traduction vers la théorie monadique du second ordre des arbres k-aires. Comme les logiques modales régulières vérifient justement ce type de propriétés, on pourrait aussi les traduire vers ce formalisme mais cela nous fournirait une complexité non-élémentaire sans information supplémentaire, ce qui est moins intéressant que la borne EXPTIME de GF2.

Par conséquent, on obtient une autre preuve de la partie du corollaire 5.2.3 relative aux logiques régulières mais cette fois on peut utiliser toute la machinerie des démonstrateurs pour GF et FO2. En effet, on obtient une réduction en temps linéaire entre SAT($\mathcal{S}$) et GF2.

Corollaire 5.3.4. [DdN05] Soit $\mathcal{L}$ une logique modale grammaticale régulière avec passé. Les problèmes de satisfaisabilité, validité et satisfaisabilité globale sont dans EXPTIME.

Il est à noter que la propriété de transitivité n'appartient pas à GF mais la logique modale S4 caractérisée par les modèles avec relation réflexive et transitive peut être traduite dans GF2 avec notre approche, sans avoir à passer par μGF2 ou par MGF2 avec une relation transitive prédéfinie. Une traduction directe de la logique intuitionniste vers GF2 est présentée dans [DdN04].

On peut remarquer que la traduction des sous-formules de la forme $[a]\varphi$ consiste en partie à simuler l'automate $\mathcal{A}_a$ qui est directement lié aux contraintes de la relation R_a dans les modèles de la logique. C'est pourquoi, comme nous le faisions remarquer dans [DdN03, DdN05], cette traduction fournit aussi une explication sur le fait que de nombreuses logiques modales comme S4 ont des calculs analytiques relativement simples (voir par exemple [Fit83, Gor99, Mas00]). En effet, les règles de calcul pour les sous-formules de la forme $[a]\varphi$ (surtout les règles de propagation, voir [dCG02]) simule l'automate $\mathcal{A}_a$ comme le fait notre traduction. D'ailleurs, des calculs reprenant cette idée de simulation ont été définis explicite-

ment dans [GN05] (voir aussi [HS04]) pour les logiques grammaticales régulières sans passé.

A la différence de la traduction relationnelle classique, notre traduction mêle à la fois l'interprétation des symboles logiques et les contraintes sur les relations. La prise en compte de ces contraintes de cette façon est aussi présente dans [SH04, BH94, DG02b]. Cependant, la technique mise en place dans [SH04] est la plus proche de notre travail. Voici quelques points communs.

- Les deux traductions ont pour langage cible GF2 et permettent de traduire une classe assez importante de logiques modales.
- Les traductions pour les logiques K, T, K4 et S4 dans [SH04] et avec notre approche sont essentiellement les mêmes, une fois que des différences mineures sont écartées. Par exemple, le schéma de clause défini dans [SH04] pour l'axiome K4 $\Box \mathrm{p} \Rightarrow \Box\Box \mathrm{p}$ est le suivant:

  $$\forall\, \mathsf{x}\, Q_{\Box \mathrm{p}}(\mathsf{x}) \Rightarrow (\forall\, \mathsf{x}\, \mathsf{R}_a(\mathsf{x}, \mathsf{y}) \Rightarrow Q_{\Box \mathrm{p}}(\mathsf{y})).$$

  Ce schéma est obtenu à partir de $\Box \mathrm{p} \Rightarrow \Box\Box \mathrm{p}$ en effectuant une traduction partielle qui s'arrête avant les modalités les plus profondes et en renommant les sous-formules. Avec notre approche, la lettre a de l'opérateur modal $[a]$ est associée au langage régulier a^+ qui peut être accepté par l'automate $\mathcal{A}_0 = \langle \{q_0, q_1\}, q_0, q_1, \{q_0 \xrightarrow{a} q_1, q_1 \xrightarrow{a} q_1\} \rangle$. Dans notre traduction, un des conjoints de la formule $t_{acc}([a]\mathrm{p})$ est de la forme:

  $$\forall\, \mathsf{x}\, \mathsf{P}^{\mathrm{p}}_{q_1}(\mathsf{x}) \Rightarrow (\forall\, \mathsf{y}\, \mathsf{R}_a(\mathsf{x}, \mathsf{y}) \Rightarrow \mathsf{P}^{\mathrm{p}}_{q_1}(\mathsf{y})).$$

- Les deux méthodes nécessitent une connaissance préalable non-triviale sur la logique. Dans notre cas, nous devons savoir que la logique est régulière et donc connaître les automates $\mathcal{A}_a$.

5.3.3 Exemples

Outre le nombre important de logiques modales classiques qui peuvent être traduites vers GF2 avec notre méthode, nous présentons ci-dessous d'autres applications qui soulignent la portée de ce travail. Les logiques modales régulières contiennent les logiques modales intuitionnistes de la forme **IntK**$_\Box$ [WZ97]. En effet, soient $\mathcal{S}$ un système clos fini régulier sur l'alphabet Σ et $\Sigma' \subset \Sigma$ tels que pour $a \in \Sigma$, $\{a, \overline{a}\} \not\subseteq \Sigma'$. Le système

$$\mathcal{S} \cup \{b \rightarrow bab, \overline{b} \rightarrow \overline{b}\overline{a}\overline{b} : a \in \Sigma'\}$$

sur l'alphabet $\Sigma \uplus \{b, \overline{b}\}$ est aussi régulier. En utilisant [GMV99], la décidabilité de logiques modales intuitionnistes est aussi établie dans [AS03] de façon uniforme.

Dans le même ordre d'idée, à savoir les extensions que notre résultat permet de traiter, pour toute logique régulière caractérisée par le système $\mathcal{S}$ sur l'alphabet Σ, son extension avec un opérateur universel $[U]$ est aussi une logique régulière.

On rappelle que $[U]$ permet de quantifier sur tous les états du modèle. En utilisant [GP92], le système $\mathcal{S}'$ sur l'alphabet $\Sigma \uplus \{U, \overline{U}\}$ caractérisant une telle extension est simplement

$$\mathcal{S} \cup \{U \to a, \overline{U} \to \overline{a} : a \in \Sigma\} \cup \overline{\{U \to \overline{U}, U \to \epsilon, U \to UU\}}.$$

Si on ajoute des nominaux à une logique régulière on peut étendre la traduction vers GF2 avec égalité et constantes. Un nominal est une variable propositionnelle particulière vraie dans un unique état du modèle et des exemples de telles logiques se trouvent par exemple dans [Bla90] ou encore dans [Sat96, ABM00] avec des modèles dont les relations sont transitives. En effet, la fonction de traduction t est étendue de la façon suivante $t(\mathtt{i}, \mathtt{x}_j) \stackrel{\text{def}}{=} c_{\mathtt{i}} = \mathtt{x}_j$ où $\mathtt{i}$ est un nominal et $c_{\mathtt{i}}$ est une constante de la logique classique. L'extension de GF2 obtenue est aussi EXPTIME-complète [FtC05]. De plus, en utilisant [BM02, section 4], l'extension d'une logique régulière avec l'opérateur de Gregory [Gre01] peut être traduite vers une logique régulière avec nominaux.

Problème ouvert 30. Comment définir une traduction des logiques sans passé VPL vers GF sur le modèle de la traduction ci-dessus pour les logiques règulières? ◯

5.3.4 Limites de la traduction vers GF2

Soit Σ un alphabet avec passé. On note $\mathcal{S}^E_\Sigma$ le système infini suivant qui n'est pas hors-contexte:

$$\{u\overline{u}u \to u : u \in \Sigma^* \setminus \{\epsilon\}\}.$$

La correspondance entre l'inclusion de relations et la dérivation dans un système est spécifié dans le théorème 5.3.5.

Théorème 5.3.5. [DdN05] Soit Σ un alphabet avec passé et $\mathcal{S}$ un système hors-contexte construit sur Σ (pas nécessairement clos). Pour tous les $u, v \in \Sigma^*$, les propositions ci-dessous sont équivalentes:
(I) Pour tout modèle $\mathcal{M} = \langle S, (R_a)_{a\in\Sigma}, V\rangle$ satisfaisant $\mathcal{S}$, nous avons $R_v \subseteq R_u$.
(II) Il existe un mot $w \in \Sigma^*$ tel que $u \Rightarrow_{\overline{\mathcal{S}}} w$ et $w \Rightarrow_{\mathcal{S}^E_\Sigma} v$.
(III) $u \Rightarrow_{\overline{\mathcal{S}}\cup\mathcal{S}^E_\Sigma} v$.

Pour se convaincre de la nécessité de passer par l'extension $\mathcal{S}^E_\Sigma$, considérons le système clos $\mathcal{S} = \{a \to \overline{a}, b \to a^3, \overline{b} \to \overline{a}^3\}$. De façon évidente, $a \notin \mathrm{L}_{\mathcal{S}}(b)$. Cependant, tout modèle satisfaisant $\mathcal{S}$ vérifie $R_a \subseteq R_b$ car la relation R_a est alors symétrique. Par contre, si on ajoute la règle $a\overline{a}a \to a$ à $\mathcal{S}$, on peut en effet dériver a à partir de b.

Deux systèmes distincts peuvent caractériser la même logique, c'est-à-dire définir le même ensemble de modèles. Le théorème 5.3.5 donne une condition suffisante pour obtenir une telle situation.

Corollaire 5.3.6. Soient Σ un alphabet avec passé et $\mathcal{S}_1$, $\mathcal{S}_2$ deux systèmes hors-contexte clos finis. Si pour tous les $i \neq i'$ et règles $u \to v$ de $\mathcal{S}_i$, il existe $w \in \Sigma^*$ tel que $u \Rightarrow_{\mathcal{S}_{i'}} w$ et $w \Rightarrow_{\mathcal{S}^E_\Sigma} v$, alors $\mathcal{S}_1$ et $\mathcal{S}_2$ caractérisent la même logique.

Soit $\mathcal{S}_n$ le système semi-Thuéien hors-contexte: $\{a \to \overline{a}^n a, \overline{a} \to \overline{a}a^n\}$. Le système $\mathcal{S}_1$ définit en fait une logique régulière puisqu'il s'agit de K5. Cependant, on peut montrer que pour $n > 1$, $\mathrm{L}_{\mathcal{S}_n}(a)$ n'est pas régulier (voir le détail dans [DdN05]). En utilisant le corollaire 5.3.6, on peut établir que $\mathcal{S}_2$ et le système $\mathcal{S}_2'$ ci-dessous définissent la même logique:

$$\mathcal{S}_2' = \{a \to \overline{a}\overline{a}a, a \to \overline{a}aa, \overline{a} \to \overline{a}\overline{a}a\overline{a} \to \overline{a}aa\}.$$

Par ailleurs, $\mathcal{S}_2'$ définit une logique régulière [DdN05]. Par conséquent, la logique modale avec passé caractérisée par $\mathcal{S}_2$ admet un problème de satisfaisabilité dans EXPTIME. Il est aussi connu que le problème de satisfaisabilité pour la logique caractérisée par le système $\mathcal{S}_n$ mais restreint aux seules modalités avec a est décidable [Gab75, HS03] (sans passé).

Problème ouvert 31. Est-ce que toutes les logiques caractérisées par $\mathcal{S}_n$ sont régulières? Cette propriété est établie pour $n \in \{1, 2\}$. ◯

Il est erroné mais tentant de déduire des résultats des sections précédentes que la réduction vers le fragment gardé est réservée aux seules logiques modales grammaticales. Nous allons montrer dans le reste de cette section comment traduire la logique modale S4.2 [Seg71, Gor91] vers GF3 alors que S4.2 n'est pas grammaticale et les contraintes sur les modèles ne sont pas exprimables dans GF. Le problème de satisfaisabilité pour GF3 est EXPTIME-complet. On peut aussi trouver dans [DG00d] un exemple de traduction de la logique Grz du second-ordre vers GF2 en effectuant différentes traductions intermédiaires.

La logique modale S4.2 (sans passé) est une logique monomodale avec $\Sigma = \{a\}$ dont les modèles de la forme $\langle S, R_a, V\rangle$ vérifie les propriétés suivantes:

- R_a est réflexive et transitive (comme pour S4),
- R_a vérifie la propriété de Church-Rosser: si $\langle x, y\rangle \in R_a$ et $\langle x, z\rangle \in R_a$ alors il existe t tel que $\langle y, t\rangle \in R_a$ et $\langle z, t\rangle \in R_a$.

La décidabilité de S4.2 est établie dans [Seg71] alors qu'une borne supérieure de complexité PSPACE est montrée dans [dCG99].

Avant de définir la traduction de S4.2 vers GF3 qui s'inspire des idées présentes dans les sections précédentes, nous souhaitons énoncer une propriété qu'il est facile d'établir.

Lemme 5.3.7. Si ϕ est satisfaisable pour S4.2 alors il existe un modèle de S4.2 $\mathcal{M} = \langle S, R_a, V\rangle$ et $x_0 \in S$ tels que

1. $\mathcal{M}, x_0 \models \phi$ et $S = \{x \in S : \langle x_0, x\rangle \in R_a\}$,
2. pour tous les états $x, y \in S$ et sous-formules $[a]\psi$ de ϕ tels que $\mathcal{M}, x \models [a]\psi$, il existe $z \in S$ avec $\langle y, z\rangle \in R_a$ et $\mathcal{M}, z \models [a]\psi$.

Soit ϕ une formule sous forme normale négative. Sans perte de généralité, on suppose que $[a]\top$ est une sous-formule de ϕ. La traduction $T(\phi)$ est la conjonction des formules suivantes:

- $t(\phi, \mathsf{x}_0)$ où t est une variante de la fonction de traduction auxiliaire pour le cas des grammaires hors-contexte,

– $\bigwedge_{[a]\psi \in \mathrm{sub}(\phi)} per_{[a]\psi}$ où $per_{[a]\psi}$ code la persistence de la formule $[a]\psi$ (R_a est réflexive et transitive),
– $\bigwedge_{[a]\psi \in \mathrm{sub}(\phi)} cr_{[a]\psi}$ où $cr_{[a]\psi}$ code la propriété de Church-Rosser,
– $\mathsf{P}_{[a]\top}(\mathsf{x}_0)$.

La seule variable libre de $T(\phi)$ est x_0.

Le vocabulaire utilisé dans la logique classique est composé:

– du symbole de prédicat binaire R_a,
– de symboles de prédicat unaires $\mathsf{P}_1, \ldots, \mathsf{P}_n$ si ϕ est construite sur les variables propositionnelles $\{\mathrm{p}_1, \ldots, \mathrm{p}_n\}$,
– de symboles de prédicat unaires $\mathsf{P}_{[a]\psi}$ pour toute sous-formule de la forme $[a]\psi$ dans ϕ.

La fonction de traduction t admet la définition suivante pour $i \in \{0, 1\}$:

– $t(\mathrm{p}_j, \mathsf{x}_i) \stackrel{\text{def}}{=} \mathsf{P}_j(\mathsf{x}_i)$,
– t est homomorphique pour les opérateurs Booléens,
– $t(\langle a\rangle\varphi, \mathsf{x}_i) = \exists \mathsf{x}_{1-i}\, \mathsf{R}_a(\mathsf{x}_i, \mathsf{x}_{i-1}) \wedge t(\varphi, \mathsf{x}_{1-i})$,
– $t([a]\varphi, \mathsf{x}_i) = \mathsf{P}_{[a]\varphi}(\mathsf{x}_i)$.

Pour chaque sous-formule $[a]\psi$, la formule $per_{[a]\psi}$ est la suivante:

$$(\forall \mathsf{x}_0, \mathsf{x}_1\ \mathsf{R}_a(\mathsf{x}_0, \mathsf{x}_1) \Rightarrow (\mathsf{P}_{[a]\psi}(\mathsf{x}_0) \Rightarrow \mathsf{P}_{[a]\psi}(\mathsf{x}_1))) \wedge (\forall \mathsf{x}_0\ \mathsf{P}_{[a]\psi}(\mathsf{x}_0) \Rightarrow t(\psi, \mathsf{x}_0)).$$

Pour chaque sous-formule $[a]\psi$, la formule $cr_{[a]\psi}$ est la suivante:

$$\forall \mathsf{x}_0\ \mathsf{P}_{[a]\psi}(\mathsf{x}_0) \Rightarrow (\forall \mathsf{x}_1\ \mathsf{P}_{[a]\top}(\mathsf{x}_1) \Rightarrow (\exists \mathsf{x}_2\ \mathsf{R}_a(\mathsf{x}_1, \mathsf{x}_2) \wedge \mathsf{P}_{[a]\psi}(\mathsf{x}_2))).$$

Il est bon de remarquer que la formule $cr_{[a]\psi}$ énonce dans la logique classique la propriété 2. du lemme 5.3.7. On peut aussi noter que $T(\phi)$ est de taille linéaire en la taille de ϕ, que sa construction peut s'effectuer en espace logarithmique et que $T(\phi)$ est dans GF3. Afin que chaque formule $cr_{[a]\psi}$ soit bien dans GF nous avons fait appel à deux astuces puisque la propriété de Church-Rosser n'est ni exprimable dans GF ni grammaticale. D'abord, nous avons introduit le prédicat unaire $\mathsf{P}_{[a]\top}$ qui contient au moins tous les états accessibles avec l'interprétation de R_a. Cela évite l'usage d'une quantification de la forme

$$\forall \mathsf{x}_1\ \exists \mathsf{x}_2\ \mathsf{R}_a(\mathsf{x}_1, \mathsf{x}_2) \wedge \mathsf{P}_{[a]\psi}(\mathsf{x}_2)$$

qui n'est pas dans GF. Par ailleurs, dans la condition 2. du lemme 5.3.7, il n'est pas nécessaire de préciser que $\langle x, z\rangle \in R_a$ ce qui aurait requis une quantification de la forme

$$\forall \mathsf{x}_1\ \mathsf{P}_{[a]\top}(\mathsf{x}_1) \Rightarrow (\exists \mathsf{x}_2\ \mathsf{R}_a(\mathsf{x}_1, \mathsf{x}_2) \wedge (\mathsf{R}_a(\mathsf{x}_0, \mathsf{x}_2) \wedge \mathsf{P}_{[a]\psi}(\mathsf{x}_2))).$$

A nouveau, une telle quantification n'aurait pas été dans GF. Nos efforts pour une prise en compte de la propriété de Church-Rosser dans GF vont être récompensés dans la preuve du théorème suivant.

Théorème 5.3.8. ϕ est satisfaisable dans S4.2 ssi $T(\phi)$ est satisfaisable.

Preuve: Supposons d'abord que ϕ est satisfaisable. Il existe donc un modèle de $\mathcal{M} = \langle S, R_a, V\rangle$ vérifiant les propriétés 1. et 2. du lemme 5.3.7 et x_0 tels que $\mathcal{M}, x_0 \models \phi$. Construisons un modèle $\mathcal{M}' = \langle S', V'\rangle$ pour $T(\phi)$:

- $S' = S$, $V'(\mathsf{R}_a) = R_a$,
- pour $i \in \{1, \ldots, n\}$, $V'(\mathsf{P}_i) = V(\mathrm{p}_i)$,
- pour toute sous-formule $[a]\psi$, $V'(\mathsf{P}_{[a]\psi}) = \{x \in S : \mathcal{M}, x \models [a]\psi\}$.

Om peut facilement montrer par induction structurelle que pour toute sous-formule ψ de ϕ, si $\mathcal{M}, x \models \psi$ alors $\mathcal{M}' \models_{[\mathsf{x}_j \mapsto x]} t(\psi, \mathsf{x}_j)$. Par transitivité et réflexivité de R_a, il est facile d'établir que $\mathcal{M}' \models per_{[a]\psi}$ pour toute sous-formule $[a]\psi$. De même, les propriétés du lemme 5.3.7 assure que $\mathcal{M}' \models cr_{[a]\psi}$ pour toute sous-formule $[a]\psi$. Finalement, $\mathcal{M}' \models_{[\mathsf{x}_0 \mapsto x_0]} \mathsf{P}_{[a]\top}(\mathsf{x}_0)$ est trivialement vérifié.

Supposons à présent que $T(\phi)$ soit satisfaisable. Il existe donc un modèle $\mathcal{M} = \langle S, V\rangle$ et x_0 tels que $\mathcal{M} \models_{[\mathsf{x}_0 \mapsto x_0]} T(\phi)$.

Pour $x \in S$, on note $[a](x) = \{[a]\psi \in \mathrm{sub}(\phi) : x \in V(\mathsf{P}_{[a]\psi})\}$. Le modèle de Kripke $\mathcal{M}' = \langle S', R_a, V'\rangle$ est défini ainsi:

- $S' = S$,
- $\langle x, y\rangle \in R_a$ ssi $[a](x) \subseteq [a](y)$,
- $V'(\mathrm{p}_i) = V(\mathsf{P}_i)$.

La réflexivité et transitivité de R_a sont faciles à établir. Supposons que $\langle x, y\rangle \in R_a$ et $\langle x, z\rangle \in R_a$ et notons $[a](y) = \{[a_1]\psi_1, \ldots, [a_m]\psi_m\}$. Comme

$$\mathcal{M} \models cr_{[a_1]\psi_1} \wedge \cdots \wedge cr_{[a_m]\psi_m} \wedge per_{[a_1]\psi_1} \wedge \cdots \wedge per_{[a_m]\psi_m},$$

il existe $z_0, \ldots, z_m \in S$ tels que

$$[a](z) \cup \{[a_1]\psi_1, \ldots, [a_i]\psi_i\} \subseteq [a](z_i)$$

et $z_0 = z$. Par définition de R_a, $\langle z, z_m\rangle \in R_a$ et $\langle y, z_m\rangle \in R_a$. Par conséquent, R_a vérifie la propriété de Church-Rosser.

Nous montrons à présent que pour toute sous-formule ψ de ϕ, si $\mathcal{M} \models_{[\mathsf{x}_j \mapsto x]} t(\psi, \mathsf{x}_j)$ alors $\mathcal{M}', x \models \psi$. Nous traitons le seul cas vraiment intéressant pour $\psi = [a]\psi'$. Supposons que $\mathcal{M} \models_{[\mathsf{x}_j \mapsto x]} \mathsf{P}_{[a]\psi'}(\mathsf{x}_j)$. Soit $y \in R_a(x)$. Cela implique que $[a](x) \subseteq [a](y)$ par définition de R_a. Par conséquent, $\mathcal{M} \models_{[\mathsf{x}_j \mapsto y]} \mathsf{P}_{[a]\psi'}(\mathsf{x}_j)$. Comme $\mathcal{M} \models per_{[a]\psi'}$, alors $\mathcal{M} \models_{[\mathsf{x}_j \mapsto y]} t(\psi', \mathsf{x}_j)$. Par hypothèse d'induction, $\mathcal{M}', y \models \psi'$. CQFD

Corollaire 5.3.9. Le problème de satisfaisabilité pour S4.2 est dans EXPTIME.

On a vu donc que S4.2 qui n'est pas grammaticale et qui n'admet pas de prime abord une traduction vers les formalismes de [GMV99, GW99] plus simple que celle que nous proposons, admet une traduction assez simple vers GF3.

Problème ouvert 32. Comment définir une classe de logiques modales non grammaticales pour laquelle une traduction uniforme vers le fragment gardé existe? La logique S4.2 pourrait appartenir à une telle classe. ◯

5.4 Complexité algorithmique

Même si toutes les logiques modales grammaticales régulières ont un problème de satisfaisabilité dans EXPTIME, il est inexact que pour toutes ces logiques le problème de satisfaisabilité soit EXPTIME-difficile. Par exemple, pour la logique modale S5, le problème est seulement NP-complet [Lad77].

5.4.1 Logiques régulières et espace polynomial

Nous caractérisons ci-après des classes de logiques régulières dont le problème de satisfaisabilité est dans PSPACE ou PSPACE-difficile.

Théorème 5.4.1. [Dem01a] Soit $\mathcal{S}$ un système fini hors-contexte sans passé vérifiant une des conditions ci-dessous. Alors SAT($\mathcal{S}$) est PSPACE-difficile.

1. Il existe une lettre de Σ qui ne soit pas membre gauche d'une règle de $\mathcal{S}$.
2. $\mathcal{S}$ appartient à STLIND $\cup$ STLING.

La preuve du théorème 5.4.1 dans le cas 1. est par réduction du problème de satisfaisabilité pour la logique modale K qui est PSPACE-difficile [Lad77] tandis que dans le cas 2., on réduit le problème pour la logique modale T ou K. Nous avons identifié une sous-classe importante de systèmes de STLIND pour lesquels la logique sous-jacente a un problème de satisfaisabilité dans PSPACE. La preuve qui constitue une partie importante de l'article [Dem02] est par analyse de calculs de séquents.

Théorème 5.4.2. [Dem02] Soit $\mathcal{S}$ un système de STLIND sans passé avec $\Sigma = \Sigma_{NT} \uplus \Sigma_T$ tel que pour $a \in \Sigma_{NT}$, $\mathrm{L}_{\mathcal{S}}(a) \cap \Sigma_T^*$ est fini. SAT($\mathcal{S}$) est alors dans PSPACE.

Dans le théorème 5.4.2, il est à noter que l'ensemble $\mathrm{L}_{\mathcal{S}}(a)$ pour $a \in \Sigma_{NT}$ peut tout de même être infini. Un résultat similaire semble peu plausible avec STLING (à moins que PSPACE soit égale à EXPTIME) à cause de la EXPTIME-dureté de SAT($\mathcal{S}$) avec $\mathcal{S} = \{a \to ab\}$. Tester si $\mathrm{L}_{\mathcal{S}}(a) \cap \Sigma_T^*$ est fini peut se calculer en temps polynomial. Le système $\mathcal{S}$ ci-dessous vérifie les hypothèses du théorème 5.4.2:

$$\{a_1 \to a_3a_3a_1, a_1 \to a_2, a_1 \to a_4a_2, a_2 \to a_4a_4a_4a_2\},$$

avec $\Sigma_{NT} = \{a_1, a_2\}$ et $\Sigma_T = \{a_3, a_4\}$ et donc SAT($\mathcal{S}$) est dans PSPACE.

D'autres classes de logiques dans PSPACE et caractérisées par des systèmes linéaires à droite peuvent être trouvées dans [Dem02, théorème 8.1].

Problème ouvert 33. Y-a-t-il un fragment de GF dans PSPACE dans lequel on puisse traduire simplement les logiques S4, S4 avec passé, la logique de Grzegorczyk Grz ou la logique de Gödel G? ◯

5.4.2 Logiques régulières et temps exponentiel

Dans cette section, on s'intéresse à caractériser des logiques régulières pour lesquels le problème de satisfaisabilité est EXPTIME-difficile.

Théorème 5.4.3. [Dem01a, théorème 8.1] Soit $\mathcal{S}$ un système de STLING sans passé avec $\Sigma = \Sigma_{NT} \uplus \Sigma_T$. S'il existe $a \in \Sigma_{NT}$ avec $au^k \in \mathrm{L}_{\mathcal{S}}(a)$ où $|u|, k \geq 1$ et une lettre de u apparaît exactement une fois dans u alors SAT($\mathcal{S}$) est EXPTIME-difficile.

Le système $\{a \rightarrow ab\}$ vérifie trivialement cette propriété et par conséquent la logique bimodale avec la contrainte $R_a \circ R_b \subseteq R_a$ est EXPTIME-complète. La preuve du théorème 5.4.3 est par réduction du problème de satisfaisabilité globale pour la logique modale K [CL94, Hem96] (noté GSAT(K) dans la suite). Une autre classe de systèmes $\mathcal{S}$ dans STLING dont SAT($\mathcal{S}$) est EXPTIME-difficile est présentée dans le théorème 5.4.4 ci-dessous.

Théorème 5.4.4. [Dem01a, théorème 8.3] Soit $\mathcal{S}$ un système de STLING sans passé avec $\Sigma = \Sigma_{NT} \uplus \Sigma_T$. S'il existe $a \in \Sigma_{NT}$ avec $au \in \mathrm{L}_{\mathcal{S}}(a)$ tel que

- $|u| \geq 2$ et toute lettre de u apparaît au moins deux fois,
- $\mathrm{L}_{\mathcal{S}'}(a)$ ne contient aucun mot de la formule $u^k v$ où v est un préfixe strict de u et $\mathcal{S}' = \mathcal{S} \cup \{a \rightarrow \epsilon\}$,

alors SAT($\mathcal{S}$) est EXPTIME-difficile.

La preuve est aussi obtenue par réduction de GSAT(K). Une conséquence des théorèmes 5.4.3 et 5.4.4 est le corollaire suivant.

Corollaire 5.4.5. Soit $\mathcal{S}$ un système de STLING (sans passé) composé d'une unique règle de la forme $a \rightarrow au$ avec u un mot non vide sur Σ. Alors SAT($\mathcal{S}$) est EXPTIME-difficile.

Le théorème 5.4.3 admet un énoncé avec les grammaires linéaires à droite.

Théorème 5.4.6. [Dem01a, théorème 8.6] Soit $\mathcal{S}$ un système de STLIND sans passé avec $\Sigma = \Sigma_{NT} \uplus \Sigma_T$. S'il existe $a \in \Sigma_{NT}$ avec $u^k a \in \mathrm{L}_{\mathcal{S}}(a)$ et $u^{k'} v \in \mathrm{L}_{\mathcal{S}}(a)$ où

- $|u| \geq 1$ et $k > 0$,
- v est un préfixe strict de u et $k' \geq 0$,
- une lettre de u apparaît exactement une fois,

alors SAT($\mathcal{S}$) est EXPTIME-difficile.

La preuve du théorème 5.4.6 est aussi par réduction du problème de satisfaisabilité globale pour la logique modale K. Soit $\mathcal{L}_i$ la logique régulière (sans passé) caractérisée par le système $\mathcal{S}_i = \{a \rightarrow \epsilon, a \rightarrow b^i a\}$ pour $i \geq 1$. Une conséquence du théorème 5.4.6 est la EXPTIME-complétude de SAT($\mathcal{S}_i$). En fait, dans le cas où $\mathcal{S}$ est construit sur un alphabet à deux lettres et est linéaire à gauche ou à droite, on peut caractériser la complexité de la logique bimodale.

Théorème 5.4.7. [Dem02, théorème 7.5] Soit $\mathcal{S}$ un système dans STLIND $\cup$ STLING sans passé tel que $\Sigma = \{a, b\}$, $a \in \Sigma_{NT}$ et a est l'axiome.

(I) Si $a \rightarrow a \cdot b^i$ est une règle de $\mathcal{S}$ pour $i \geq 1$, alors SAT($\mathcal{S}$) est EXPTIME-complet.

(II) Si $\mathcal{S}$ appartient à STLIND et $\mathrm{L}_{\mathcal{S}}(a) \cap \Sigma_T^*$ est fini [resp. infini] alors SAT($\mathcal{S}$) est PSPACE-complet [resp. EXPTIME-complet].

Pour vérifier que tous les cas sont traités, on peut remarquer que si $\mathcal{S}$ ne contient pas de règle de la forme $a \rightarrow a \cdot b^i$, $\mathcal{S}$ est linéaire à droite.

Problème ouvert 34. Comment étendre le théorème 5.4.7 pour un alphabet Σ fini de taille quelconque? ◯

5.5 Logiques bimodales et autres variantes

Nous allons étudier dans cette section des logiques bimodales obtenues à partir de logiques monomodales en imposant une contrainte d'inclusion de la forme $R_a \subseteq R_b$ ce qui correspond à la règle $b \rightarrow a$. Ce type d'inclusion entre relations est très commun et on en trouve des exemples dans les logiques terminologiques [HS99], dans les logiques bimodales de prouvabilité [Vis95] ou dans des logiques avec approximation de l'opérateur de fermeture transitive [CH96] pour ne citer que quelques exemples. Certaines de ces logiques sont des logiques grammaticales régulières mais pas toutes.

Etant données deux logiques (mono)modales sans passé $\mathcal{L}_1 = \langle \{a\}, \mathcal{C}_1 \rangle$ et $\mathcal{L}_2 = \langle \{b\}, \mathcal{C}_2 \rangle$, on peut définir une nouvelle logique $\mathcal{L} = \mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2 = \langle \{a, b\}, \mathcal{C} \rangle$ telle que pour tout modèle $\mathcal{M} = \langle S, R_a, R_b, V \rangle$, $\mathcal{M} \in \mathcal{C}$ ssi $\langle S, R_a, V \rangle \in \mathcal{C}_1$, $\langle S, R_a, V \rangle \in \mathcal{C}_2$ et $R_a \subseteq R_b$.

Combiner de cette façon deux logiques peut accroître la complexité. En effet, la logique K $\oplus_{\subseteq}$ S5 est la logique modale K avec opérateur d'universalité [GP92]. SAT(K $\oplus_{\subseteq}$ S5) est EXPTIME-difficile [Spa93a] alors que SAT(K) est PSPACE-complet et SAT(S5) est NP-complet [Lad77]. Dans l'article [Dem00a], j'ai étudié de façon systématique la complexité de logiques bimodales obtenues à partir de logiques modales standard (K, T, B, S4 et S5). De nombreux résultats étaient déjà connus mais dans [Dem00a] j'ai aussi proposé des preuves plus uniformes. La table 5.2 récapitule les principaux résultats: la complétude de la classe est assurée pour tous les problèmes.

- Pour $\mathcal{L}_1 \in \{$S4,S5$\}$ et $\mathcal{L}_2 \in \{$T,B,S4,S5$\}$, un algorithme à la Ladner est présenté dans [Dem00a, section 3] établissant que le problème de satisfaisabilité pour $\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$ est dans PSPACE. La logique bimodale S4+5 dont le problème de satisfaisabilité est montré égal à celui pour S5 $\oplus_{\subseteq}$ S4 dans [Vak90] est donc PSPACE-complète.
- Pour $\mathcal{L}_1 \in \{$K,T,B$\}$ et $\mathcal{L}_2 \in \{$S4,S5$\}$, une réduction de GSAT($\mathcal{L}_1$) vers SAT($\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$) est établie assurant que SAT($\mathcal{L}_1 \oplus_{\subseteq} \mathcal{L}_2$) est EXPTIME-difficile. En effet, GSAT($\mathcal{L}_1$) a été montré EXPTIME-difficile dans [CL94]

$\oplus_{\subseteq}$	K	T	B	S4	S5
K	PSPACE	PSPACE	PSPACE	EXPTIME	EXPTIME
T	PSPACE	PSPACE	PSPACE	EXPTIME	EXPTIME
B	PSPACE	PSPACE	PSPACE	EXPTIME	EXPTIME
S4	PSPACE	PSPACE	PSPACE	PSPACE	PSPACE
S5	PSPACE	PSPACE	PSPACE	PSPACE	NP

TAB. 5.2 – *Complexité de logiques bimodales avec contrainte d'inclusion*

– Pour montrer que pour $\mathcal{L}_1, \mathcal{L}_2 \in \{\text{K,T,B}\}$, on peut utiliser un algorithme à la Ladner. Cependant dans [Dem00a], une traduction vers un fragment de la logique classique dans PSPACE a été effectuée. On peut aussi effectuer une traduction vers un fragment de PDL(REG) qui est dans PSPACE.

Pour établir que S5 $\oplus_{\subseteq}$ S5 est dans NP alors que S5 $\oplus$ S5 (sans inclusion de relations) est PSPACE-complet [HM85a], nous utilisons un résultat plus général, brièvement décrit ci-dessous.

Deux relations binaires R et R' sur l'ensemble S sont dites en accord local ssi pour $x \in S$, soit $R(x) \subseteq R'(x)$ soit $R'(x) \subseteq R(x)$. Cette condition a été introduite dans [Gar86] (voir plus de détails dans [DO02]). Nous allons introduire une classe de logiques modales avec un alphabet Σ (sans passé) telles que deux relations de n'importe lequel des modèles sont en accord local.

Une logique modale $\mathcal{L} = \langle \Sigma, \mathcal{C} \rangle$ est une logique avec accord local ssi il existe un ensemble non vide $\mathcal{O} = \{\sqsubseteq_1, \ldots, \sqsubseteq_N\}$ d'ordres linéaires sur Σ tel que pour tout modèle $\mathcal{M} = \langle S, (R_a)_{a\in\Sigma}, V \rangle$, $\mathcal{M} \in \mathcal{C}$ ssi

– pour $a \in \Sigma$, R_a est une relation d'équivalence,
– pour $x \in S$, il existe $\sqsubseteq \in \mathcal{O}$ tel que pour $a, a' \in \Sigma$, $a \sqsubseteq a'$ implique $R_a(x) \subseteq R_{a'}(x)$.

On peut remarquer qu'en effet deux relations de tout modèle sont en accord local.

Théorème 5.5.1. [Dem98] Soit $\mathcal{L}$ une logique avec accord local (Σ est fini sans passé). Une formule ϕ de $\mathcal{L}$ est satisfaisable ssi ϕ a un modèle dont le cardinal de l'ensemble des états est borné par $1 + |\Sigma| \times |\phi|^{|\Sigma|}$.

Le théorème 5.5.1 est prouvé en montrant que si $\mathcal{M}, x \models \phi$ avec $\mathcal{M} = \langle S, (R_a)_{a\in\Sigma}, V \rangle \in \mathcal{C}$, alors il existe $\mathcal{M}' = \langle S', (R'_a)_{a\in\Sigma}, V \rangle \in \mathcal{C}$ tel que $\mathcal{M}', x \models \phi$ avec $S' \subseteq S$, $|S'| \subseteq |\Sigma| \times |\phi|^{|\Sigma|}$ et V' est la restriction à S' de V. Ainsi, cette preuve généralise le lemme qui établit que toute formule satisfaisable de la logique S5 a un modèle de taille linéaire [Lad77].

Comme $|\Sigma|$ est une constante de $\mathcal{L}$ on obtient alors le résultat de complexité suivant.

Théorème 5.5.2. [Dem98] Pour toute logique avec accord local, le problème de satisfaisabilité est NP-complet.

La NP-dureté est bien-sûr héritée de celle du calcul propositionnel. Ainsi, soit $\mathcal{L}$ une logique avec accord local dont l'ensemble des ordres linéaires est un singleton. $\mathcal{L}$ a donc un problème de satisfaisabilité dans NP et par conséquent SAT(S5 $\oplus_{\subseteq}$ S5) est NP-complet.

De nombreux développements effectués dans ce chapitre peuvent être généralisés ou adaptés au cas où $\Sigma = \{a_1, a_2, \ldots\}$ est un ensemble infini dénombrable sans passé. Considérons cette hypothèse dans le reste de cette section pour énoncer deux résultats de PSPACE-complétude qui peuvent surprendre à la vue des résultats de NP-complétude précédents.

Soit $\mathcal{L}^{one}$ la logique modale caractérisée par la classe des modèles tel que chaque relation R_a est une relation d'équivalence et $R_{a_i} \subseteq R_{a_j}$ si $i \leq j$. La logique $\mathcal{L}^{one}$ peut être considérée comme la "limite" des logiques modales avec accord local (sur un alphabet fini) caractérisée par un unique ordre. Chacune de ces logiques est dans NP [Dem98]. Le résultat surprenant suivant a été établi pour une classe plus générale de logiques.

Théorème 5.5.3. [Dem03] SAT($\mathcal{L}^{one}$) est PSPACE-complet.

De même, soit $\mathcal{L}^{all}$ la logique modale caractérisée par la classe des modèles tel que chaque relation R_a est une relation d'équivalence et toutes les relations R_{a_i} et R_{a_j} sont en accord local. La logique $\mathcal{L}^{all}$ peut être considérée comme la "limite" des logiques modales avec accord local (sur un alphabet fini) caractérisée par tous les ordres linéaires sur l'alphabet. Chacune de ces logiques est dans NP [Dem98]. Un autre résultat étonnant est le suivant.

Théorème 5.5.4. [Dem03] SAT($\mathcal{L}^{all}$) est PSPACE-complet.

Il est à noter que ni la borne inférieure PSPACE ni la borne supérieure PSPACE ne sont immédiates. La borne inférieure est par réduction de QBF tandis que la borne supérieure est obtenue avec un algorithme à la Ladner. L'uniformité des preuves dans [Dem03] permet aussi d'établir la PSPACE-complétude de logiques introduites dans [Gar86, Nak93] et motivées par la modélisation du raisonnement en présence d'information incomplète.

Chapitre 6

Bilan et perspectives

Les travaux présentés dans les chapitres 2–5 déclinent l'approche logique pour l'analyse de systèmes informatiques lorsque les propriétés à vérifier ont un caractère qualitatif et quantitatif. Les systèmes de transition obtenus après une phase de modélisation peuvent être issus de données semi-structurées [ABS00], de systèmes complexes admettant une modélisation sous forme de systèmes à compteurs [EFM99] ou encore de programmes avec un nombre fini d'états [BBF+01]. Sans même compter les travaux non détaillés dans ce mémoire (voir par exemple la section 1.4), la diversité des formalismes étudiés ou introduits témoigne de mes divers intérêts. En bref, mes thèmes de recherche incluent

1. l'analyse du coût algorithmique des problèmes de model-checking et satisfaisabilité pour des logiques temporelles du temps linéaire (voir les chapitres 2 et 3),
2. l'approche par automates pour des extensions de LTL (voir le chapitre 3 et la section 2.5),
3. la résolution de contraintes de régularité sur des graphes (voir les chapitres 4 et 5),
4. la vérification de contraintes de Presburger dans des systèmes de transition (voir le chapitre 3 et la section 4.2).

On peut aussi rappeler que l'expressivité d'une logique pour la gestion dynamique de politique d'actions DLP est analysée dans la section 4.3 alors que la définition d'un problème de contrôle pour des systèmes physiques admettant des comportements Zeno est introduit dans la section 2.5.

En ce qui concerne les perspectives de recherche que j'envisage, il y a tout d'abord des directions de recherche évoquées dans le corps du document et qui me semblent intéressantes de poursuivre (voir aussi les problèmes ouverts évoqués dans le mémoire). Par exemple, dans le chapitre 3, je souhaite poursuivre les points suivants:

– étudier les logiques temporelles avec systèmes de contraintes à base de chaînes de caractères,
– définir les extensions arborescentes pour lesquelles le problème de model-checking demeure décidable (voir par exemple [BG06]),

– déterminer davantage de fragments de logiques temporelles avec l'opérateur "freeze" qui soient décidables.

De même, en rapport avec la section 4.2, la conception de formalismes logiques interprétés sur les arbres finis qui admettent des contraintes de Presburger et des contraintes de régularité tout en ayant de bonnes propriétés algorithmiques me paraît une question prometteuse. Par ailleurs, la vérification de systèmes à compteurs avec des propriétés arborescentes dans le but de vérifier par exemple des protocoles de communication [EFM99] ou des programmes avec variables de pointeur [BBH+06, BFLS06] demeure une direction que je souhaite poursuivre. En effet, même si ce mémoire ne traite que très peu des logiques temporelles du temps arborescent, dans l'article récent [DFGvD06] nous avons revisité certains résultats théoriques à l'origine de l'outil FAST [BFL04] en introduisant une extension de la logique temporelle CTL* dans laquelle les formules atomiques caractérisent des ensembles de configurations définissables dans l'arithmétique de Presburger. Nous avons ainsi conçu une sous-classe de systèmes à compteurs pour laquelle le model-checking est non seulement décidable mais aussi définissable dans l'arithmétique de Presburger, voir aussi [BDR03]. De façon plus générale, la question de l'extension des techniques du chapitre 3 au cas arborescent reste ouverte.

Comme nous l'avons vu, l'approche logique pour la vérification de systèmes informatiques est non seulement protéiforme mais elle peut aussi toucher des systèmes aussi divers que les protocoles cryptographiques [Gou02] ou les systèmes temps-réel [Lar05]. Parmi les axes de recherche sur lesquels je souhaite mettre l'accent ici figurent l'étude de formalismes logiques pour spécifier des propriétés sur les données semi-structurées avec application aux données circulant sur le Web et la vérification de systèmes logiciels avec gestion dynamique de la mémoire. Le premier point touche directement à l'approche suivie dans la section 4.2. Par exemple, on ne sait pas quelles interactions entre les contraintes de régularité et de Presburger conduisent à l'indécidabilité. Plus généralement, d'autres questions connexes se posent en particulier liées aux typages des données [BCT04] ou aux applications Web [DSV04] et ce sont des problèmes qui pourraient bénéficier d'une approche logique. Le second point, plus nouveau pour moi, consiste à développer des langages formels de spécification qui allient une composante temporelle liée à l'exécution du système et une composante propre à la gestion de la mémoire (voir par exemple [YRSW03]). Pour cette dernière composante, le souhait est de tirer profit des travaux sur la logique séparante [Rey02]. Même si l'approche logique n'est pas en soi nouvelle (voir par exemple PAL [JJKS97], TVLA [LAS00] ou la logique d'aliasing de [BIL04]), le défi principal ici consiste à concevoir un langage de spécification concernant l'allocation dynamique de la mémoire qui admette des techniques algorithmiques efficaces à base d'abstraction par exemple. De premiers résultats dans cette direction sont présentés dans [BDL07].

Bibliographie

[Aba89] M. Abadi. The power of temporal proofs. *Theoretical Computer Science*, 65:35–83, 1989.

[ABD+05] L. Afanasiev, P. Blackburn, I. Dimitriou, B. Gaiffe, E. Goris, M. Marx, and M. de Rijke. PDL for ordered trees. *Journal of Applied Non-Classsical Logics*, 15(2):115–135, 2005.

[ABLP93] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, 1993.

[ABM99] C. Areces, P. Blackburn, and M. Marx. A road-map on complexity for hybrid logics. In *CSL'99*, volume 1683 of *Lecture Notes in Computer Science*, pages 307–321. Springer, 1999.

[ABM00] C. Areces, P. Blackburn, and M. Marx. Complexity of hybrid temporal logics. *Logic Journal of the IGPL*, 8(5):653–679, 2000.

[ABS00] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web*. Morgan Kaufmann, 2000.

[AC88] A. Arnold and P. Crubille. A linear algorithm to solve fixed-point equations on transition systems. *Information Processing Letters*, 29(2):57–66, 1988.

[ACD+04] Y. André, A.-C. Caron, D. Debarbieux, Y. Roos, and S. Tison. Extraction and implication of path constraints. In *MFCS'04*, volume 3153 of *Lecture Notes in Computer Science*, pages 863–875. Springer, 2004.

[ADdR01] N. Alechina, S. Demri, and M. de Rijke. Paths constraints from a modal logic point of view. In *8th International Workshop on Knowledge Representation meets Databases*, 2001.

[ADdR03] N. Alechina, S. Demri, and M. de Rijke. A modal perspective on path constraints. *Journal of Logic and Computation*, 13(6):939–956, 2003.

[AH94] R. Alur and Th. Henzinger. A really temporal logic. *Journal of the Association for Computing Machinery*, 41(1):181–204, 1994.

[Ale97] N. Alechina. Semi-structured information: a modal logic approach. Technical Report CSR-97-08, School of Computer Science, The University of Birmingham, August 1997.

[AM04] R. Alur and P. Madhusudan. Visibly pushdown languages. In *STOC'04*, pages 202–211. ACM Press, 2004.

[AN01] A. Arnold and D. Niwinski. *Rudiments of μ-calculus*. Elsevier, 2001.

[ANvB98] H. Andréka, I. Németi, and J. van Benthem. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.

[AS03] N. Alechina and D. Shkatov. On decidability of intuitionistic modal logics. In *Third Workshop on Methods for Modalities, Nancy*, 2003.

[AS06] N. Alechina and D. Shkatov. Logics with an existential modality. In *AIML'06*, pages 31–48. College Publications, 2006.

[AV97] S. Abiteboul and V. Vianu. Regular path queries with constraints. In *PODS'97*, pages 122–133, 1997.

[AV99] S. Abiteboul and V. Vianu. Regular path queries with constraints. *Journal of Computer and System Sciences*, 58(3):428–452, 1999.

[Bal98] M. Baldoni. *Normal Multimodal Logics: Automated Deduction and Logic Programming*. PhD thesis, Università degli Studi di Torino, 1998.

[BBF+01] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and Ph. Schnoebelen. *Systems and Software Verification, Model-Checking Techniques and Tools*. Springer-Verlag, 2001.

[BBFS96] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati. Supporting periodic authorizations and temporal reasoning in database access control. In *22nd VLDB, Bombay, India*, pages 472–483, 1996.

[BBH+06] A. Bouajjani, M. Bozga, P. Habermehl, R. Iosif, and P. Moro nd T. Vojnar. Programs with lists are counter automata. In *CAV'06*, volume 4144 of *Lecture Notes in Computer Science*, pages 517–531. Springer, 2006.

[BC85] M. Fattorosi Barnaba and F. De Caro. Graded modalities. *Studia Logica*, 44(2):197–221, 1985.

[BC01] V. Bruyère and O. Carton. Automata on linear orderings. In *MFCS'01*, volume 2136 of *Lecture Notes in Computer Science*, pages 236–247. Springer-Verlag, 2001.

[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 Computer Science*, pages 162–176. Springer, 2002.

[BC06] N. Bidoit and D. Colazzo. Capturing well typed references in DTDs. In *BDA'06*, 2006.

[BCT04] N. Bidoit, S. Cerrito, and V. Thion. A first step towards modeling semistructured data in hybrid multimodal logic. *Journal of Applied Non-Classsical Logics*, 14(4):447–475, 2004.

[BD97] Ph. Balbiani and S. Demri. Prefixed tableaux systems for modal logics with enriched languages. In *IJCAI'97*, pages 190–195. Morgan Kaufmann, 1997.

[BDL07] R. Brochenin, S. Demri, and E. Lozes. Reasoning about sequences of memory states. In *LFCS'07*, volume 3634 of *Lecture Notes in Computer Science*, pages 100–114. Springer, 2007.

[BDM+06] M. Bojańczyk, C. David, A. Muscholl, Th. Schwentick, and L. Segoufin. Two-variable logic on data trees and XML reasoning. In *PODS'06*, pages 10–19. ACM, 2006.

[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.

[BdRV01] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[Bed98] N. Bedon. *Langages reconnaissables de mots indexés par des ordinaux*. PhD thesis, Université Marne-la-Vallée, 1998.

[BEH95] A. Bouajjani, R. Echahed, and P. Habermehl. On the verification problem of nonregular properties for nonregular processes. In *LICS'95*, pages 123–133, 1995.

[Bel82] N. Belnap. Display logic. *Journal of Philosophical Logic*, 11:375–417, 1982.

[BEM97] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model checking. In *CONCUR'97*, volume 1243 of *LNCS*, pages 135–150. Springer, 1997.

[BFL04] S. Bardin, A. Finkel, and J. Leroux. FASTer acceleration of counter automata in practice. In *TACAS'04*, volume 2988 of *Lecture Notes in Computer Science*, pages 576–590. Springer, March 2004.

[BFLS06] S. Bardin, A. Finkel, E. Lozes, and A. Sangnier. From pointer systems to counter systems using shape analysis. *Proceedings of the 5th International Workshop on Automated Verification of Infinite-State Systems (AVIS'06)*, 2006.

[BFW98a] P. Buneman, W. Fan, and S. Weinstein. Path constraints on deterministic graphs. Technical Report MS-CIS-98-33, LINCS, CIS, UPenn, 1998.

[BFW98b] P. Buneman, W. Fan, and S. Weinstein. Path constraints on semistructured and structured data. In *PODS'98*, pages 129–138, 1998.

[BFW00] P. Buneman, W. Fan, and S. Weinstein. Path constraints in semistructured databases. *Journal of Computer and System Sciences*, 61(2):146–193, 2000.

[BG06] L. Bozzelli and R. Gascon. Branching-time temporal logic extended with Presburger constraints. In *LPAR'06*, volume 4246 of *Lecture Notes in Computer Science*, pages 197–211. Springer, 2006.

[BGP97] T. Bultan, R. Gerber, and W. Pugh. Symbolic model checking of infinite state systems using Presburger arithmetic. In *CAV'97*, volume 1254 of *Lecture Notes in Computer Science*, pages 400–411. Springer, 1997.

[BH91] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *IJCAI'91*, pages 452–457, 1991.

[BH94] Ph. Balbiani and A. Herzig. A translation from the modal logic of provability into K4. *Journal of Applied Non-Classical Logics*, 4:73–77, 1994.

[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.

[BH99] A. Bouajjani and P. Habermehl. Symbolic reachability analysis of FIFO-channel systems with nonregular sets of configurations. *Theoretical Computer Science*, 221(1–2):211–250, 1999.

[BIL04] M. Bozga, R. Iosif, and Y. Lakhnech. On logics of aliasing. In *SAS'04*, volume 3148 of *Lecture Notes in Computer Science*, pages 344–360. Springer, 2004.

[Bla90] P. Blackburn. *Nominal Tense Logic and Other Sorted Intensional Frameworks*. PhD thesis, University of Edinburgh, 1990.

[Bla93] P. Blackburn. Nominal tense logic. *Notre Dame Journal of Formal Logic*, 34(1):56–83, 1993.

[Bla00a] P. Blackburn. Internalizing labelled deduction. *Journal of Logic and Computation*, 10(1):137–168, 2000.

[Bla00b] P. Blackburn. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of the IGPL*, 8(3):339–365, 2000.

[BLZ96] M. Baaz, A. Leitsch, and R. Zach. Completeness of a first-order temporal logic with time-gaps. *Theoretical Computer Science*, 160(1–2):241–270, 1996.

[BM02] P. Blackburn and M. Marx. Remarks in Gregory's "actually" operator. *Journal of Philosophical Logic*, 31(1):281–288, 2002.

[BMP04] D. Bresolin, A. Montanari, and G. Puppis. Time granularities and ultimately periodic automata. In *JELIA'04*, volume 3229 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 2004.

[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.

[Boo93] G. Boolos. *The Logic of Provability*. Cambridge University Press, 1993.

[Bou02] P. Bouyer. A logical characterization of data languages. *Information Processing Letters*, 84(2):75–85, 2002.

[BP97] B. Bérard and C. Picaronny. Accepting Zeno words: a way toward timed refinements. In *MFCS'97*, volume 1295 of *Lecture Notes in Computer Science*, pages 149–158. Springer, 1997.

[BP04] P. Bonatti and A. Peron. On the undecidability of logics with converse, nominals, recursion and counting. *Artificial Intelligence*, 158(1):75–96, 2004.

[BPT03] P. Bouyer, A. Petit, and D. Thérien. An algebraic approach to data languages and timed languages. *Information and Computation*, 182(2):137–162, 2003.

[Bro03] J. Broersen. *Modal action logics for Reasoning about Reactive systems*. PhD thesis, Dutch Research School for Information and Knowledge Systems, 2003.

[BS73] R. Büchi and D. Siefkes. *The monadic second order theory of all countable ordinals*, volume 328 of *Lecture Notes in Mathematics*. Springer, 1973.

[BS95] P. Blackburn and J. Seligman. Hybrid languages. *Journal of Logic, Language, and Information*, 4:251–272, 1995.

[BT05] I. Boneva and J.M. Talbot. Automata and logics for unranked and unordered trees. In *RTA'05*, volume 3467 of *Lecture Notes in Computer Science*, pages 500–515. Springer, 2005.

[Büc62] R. Büchi. On a decision method in restricted second-order arithmetic. In *International Congress on Logic, Method and Philosophical Science'60*, pages 1–11, 1962.

[Büc64] R. Büchi. Transfinite automata recursions and weak second order theory of ordinals. In *Int. Cong. Logic, Methodology and Philosophy of Science, Jerusalem*, pages 3–23, 1964.

[Büc65] R. Büchi. Decision methods in the theory of ordinals. *Bulletin of the AMS*, 71:767–770, 1965.

[Cac06] T. Cachat. Controller synthesis and ordinal automata. In *ATVA'06*, volume 4218 of *Lecture Notes in Computer Science*, pages 215–228. Springer, 2006.

[Car01] O. Carton. Automates et mots infinis. Université de Marne-la-Vallée, 2001. Habilitation thesis.

[Car02] O. Carton. Accessibility in automata on scattered linear orderings. In *MFCS'02*, volume 2420 of *Lecture Notes in Computer Science*, pages 155–164. Springer, 2002.

[Cau96] D. Caucal. On infinite transition graphs having a decidable monadic theory. In *ICALP'96*, volume 1099 of *Lecture Notes in Computer Science*, pages 194–205. Springer, 1996.

[Cau03] D. Caucal. On infinite transition graphs having a decidable monadic theory. *Theoretical Computer Science*, 290:79–115, 2003.

[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.

[CCDF97] L. Cai, J. Chen, R. G. Downey, and M. R. Fellows. On the parameterized complexity of short computation and factorization. *Archive for Mathematical Logic*, 36(4/5):321–337, 1997.

[Cer90] C. Cerrato. General canonical models for graded normal logics. *Studia Logica*, 49(2):242–252, 1990.

[Čer94a] K. Čerāns. Deciding properties of integral relational automata. In *ICALP'94*, volume 820 of *Lecture Notes in Computer Science*, pages 35–46. Springer, 1994.

[Cer94b] C. Cerrato. Decidability by filtrations for graded normal logics (graded modalities V). *Studia Logica*, 53(1):61–73, 1994.

[Ces03] M. Cesati. The Turing way to parameterized intractability. *Journal of Computer and System Sciences*, 67(4):654–685, 2003.

[CF03] Y. Chen and J. Flum. Machine characterization of the classes of the W-hierarchy. In *CSL'03*, volume 2803 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2003.

[CFP02] C. Combi, M. Franceschet, and A. Peron. A logical approach to represent and reason about calendars. In *Int. Symposium on Temporal Representation and Reasoning*, pages 134–140. IEEE, 2002.

[CG05] D. Calvanese and G. De Giacomo. Expressive description logics. In *Description Logics Handbook*, pages 178–218. Cambridge University Press, 2005.

[CGL98] D. Calvanese, G. De Giacomo, and M. Lenzerini. What can knowledge representation do for semi-structured data. In *AAAI'98*, pages 205–210. MIT Press, 1998.

[CGL99] D. Calvanese, G. De Giacomo, and M. Lenzerini. Representing and Reasoning on XML documents: A Description Logic Approach. *Journal of Logic and Computation*, 9(3):295–318, 1999.

[CH96] M. Castilho and A. Herzig. An alternative to the iteration operator of propositional dynamic logic. Technical Report 96-05-R, IRIT- Université Paul Sabatier, 1996.

[Cho78] Y. Choueka. Finite automata, definable sets, and regular expressions over ω^n-tapes. *Journal of Computer and System Sciences*, 17:81–97, 1978.

[Chu36] A. Church. A note on Entscheidungsproblem. *The Journal of Symbolic Logic*, 1:40–41, 1936. Correction ibid 1 (1936), 101–102.

[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.

[CL94] C. Chen and I. Lin. The complexity of propositional modal theories and the complexity of consistency of propositional modal theories. In *LFCS'94*, volume 813 of *Lecture Notes in Computer Science*, pages 69–80. Springer, 1994.

[CMP99] S. Cerrito, M. Cialdea Mayer, and S. Praud. First-order linear temporal logic over finite time structures. In *LPAR'99*, volume 1705 of *Lecture Notes in Computer Science*, pages 62–76. Springer, 1999.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *STOC'71*, pages 151–158, 1971.

[Cop02] J. Copeland. The genesis of possible worlds semantics. *Journal of Philosophical Logic*, 31(1):99–137, 2002.

[Cor02] V. Cortier. About the decision of reachability for register machines. *Theoretical Informatics and Applications*, 36(4):341–358, 2002.

[CR04] C. Chitic and D. Rosu. On validation of XML streams using finite state machines. In *WebDB, Paris*, pages 85–90, 2004.

[CS91] R. Cleaveland and B. Steffen. A linear-time model-checking algorithm for the alternation-free modal mu-calculus. In *CAV'91*, volume 575 of *Lecture Notes in Computer Science*, pages 48–58. Springer, 1991.

[CS94] A. Chagrov and V. Shehtman. Algorithmic aspects of propositional tense logics. In *CSL'94*, volume 933 of *Lecture Notes in Computer Science*, pages 442–455. Springer, 1994.

[dA01] L. de Alfaro. Model checking the world wide web. In *CAV'01*, volume 2102 of *Lecture Notes in Computer Science*, pages 337–349. Springer, 2001.

[Dam99] D. Dams. Flat fragments of CTL and CTL*: separating the expressive and distinguishing powers. *Logic Journal of the IGPL*, 7(1):55–78, 1999.

[Dan84] R. Danecki. Nondeterministic propositional dynamic logic with intersection is decidable. In *5th Symposium on Computation Theory, Zaborów, Poland*, pages 34–53. Lecture Notes in Computer Science, Vol. 208. Springer, Berlin, 1984.

[Dav04] C. David. Mots et données infinies. Master's thesis, LIAFA, 2004.

[dCG99] L. Fariñas del Cerro and O. Gasquet. Tableaux based decision procedures for modal logics of confluence and density. *Fundamenta Informaticae*, 40:317–333, 1999.

[dCG02] L. Fariñas del Cerro and O. Gasquet. A general framework for pattern-driven modal tableaux. *Logic Journal of the IGPL*, 10(1):51–83, 2002.

[dCP88] L. Fariñas del Cerro and M. Penttonen. Grammar logics. *Logique et Analyse*, 121–122:123–134, 1988.

[DD02] S. Demri and D. D'Souza. An automata-theoretic approach to constraint LTL. In *FST&TCS'02*, volume 2556 of *Lecture Notes in Computer Science*, pages 121–132. Springer, 2002.

[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.

[DdN03] S. Demri and H. de Nivelle. Deciding regular grammar logics with converse through first-order logic. Technical Report LSV-03-4, LSV, February 2003. 29 pages.

[DdN04] S. Demri and H. de Nivelle. Deciding regular grammar logics with converse through first-order logic. arXiv:cs.LO/0306117, February 2004.

[DdN05] S. Demri and H. de Nivelle. Deciding regular grammar logics with converse through first-order logic. *Journal of Logic, Language, and Information*, 14(3):289–329, 2005. Special issue dedicated to guarded logics.

[Deb05] D. Debarbieux. *Modélisation et requêtes des documents semi-structurés: exploitation de la structure de graphe*. PhD thesis, Université des Sciences et Technologies de Lille, 2005.

[Dec92] R. Dechter. From local to global consistency. *Artificial Intelligence*, 55:87–107, 1992.

[Dem96a] S. Demri. A class of information logics with a decidable validity problem. In *MFCS'96*, volume 1113 of *Lecture Notes in Computer Science*, pages 291–302. Springer, 1996.

[Dem96b] S. Demri. A simple tableau system for the logic of elsewhere. In *TABLEAUX'96*, volume 1071 of *Lecture Notes in Computer Science*, pages 177–192. Springer, 1996.

[Dem96c] S. Demri. The validity problem for the logic DALLA is decidable. *Bulletin of the Polish Academy of Sciences, Mathematics*, 44(1):79–86, 1996.

[Dem97a] S. Demri. A completeness proof for a logic with an alternative necessity operator. *Studia Logica*, 58(1):99–112, 1997.

[Dem97b] S. Demri. Extensions of modal logic S5 preserving NP-completeness. *Bulletin of the Section of Logic*, 26(2):73–84, 1997.

[Dem98] S. Demri. A class of decidable information logics. *Theoretical Computer Science*, 195(1):33–60, 1998.

[Dem99a] S. Demri. A logic with relative knowledge operators. *Journal of Logic, Language and Information*, 8(2):167–185, 1999.

[Dem99b] S. Demri. Sequent calculi for nominal tense logics: a step towards mechanization? In *TABLEAUX'99*, volume 1617 of *Lecture Notes in Computer Science*, pages 140–154. Springer, 1999.

[Dem00a] S. Demri. Complexity of simple dependent bimodal logics. In *TABLEAUX'00*, volume 1847 of *Lecture Notes in Artificial Intelligence*, pages 190–204. Springer, 2000.

[Dem00b] S. Demri. The nondeterministic information logic NIL is PSPACE-complete. *Fundamenta Informaticae*, 42(3–4):211–234, 2000.

[Dem00c] S. Demri. A simple modal encoding of propositional finite many-valued logics. *International Journal on Many-Valued Logics*, 6:443–461, 2000.

[Dem01a] S. Demri. The complexity of regularity in grammar logics and related modal logics. *Journal of Logic and Computation*, 11(6):933–960, 2001.

[Dem01b] S. Demri. Coping with semilattices of relations in logics with relative accessibility relations. In *[OSe01]*, pages 163–181, 2001.

[Dem02] S. Demri. Modal logics with weak forms of recursion: PSPACE specimens. In *Advances in Modal Logics, selected papers from 3rd Workshop on Advances in Modal Logics (AIML'2000), Leipzig, Germany, Oct. 2000*, pages 113–138. World Scientific, 2002.

[Dem03] S. Demri. A polynomial space construction of tree-like models for logics with local chains of modal connectives. *Theoretical Computer Science*, 300(1–3):235–258, 2003.

[Dem04] S. Demri. LTL over integer periodicity constraints. Technical Report LSV-04-6, LSV, February 2004. 35 pages.

[Dem05] S. Demri. A reduction from DLP to PDL. *Journal of Logic and Computation*, 15(5):767–785, 2005.

[Dem06a] S. Demri. Linear-time temporal logics with Presburger constraints: An overview. *Journal of Applied Non-Classical Logics*, 16(3–4):311–347, 2006.

[Dem06b] S. Demri. LTL over integer periodicity constraints. *Theoretical Computer Science*, 360(1003):96–123, 2006.

[DF99] R. Downey and M. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.

[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.

[dG95] G. de Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms*. PhD thesis, Universita Degli Studi Di Roma "La Sapienza", Dipartimento Di Informatica E Sistemistica, 1995.

[DG00a] S. Demri and D. Gabbay. On modal logics characterized by models with relative accessibility relations: Part I. *Studia Logica*, 65(3):323–353, 2000.

[DG00b] S. Demri and D. Gabbay. On modal logics characterized by models with relative accessibility relations: Part II. *Studia Logica*, 66(3):349–384, 2000.

[DG00c] S. Demri and R. Goré. Display calculi for logics with relative accessibility relations. *Journal of Logic, Language and Information*, 9(2):213–236, 2000.

[DG00d] S. Demri and R. Goré. An $\mathrm{O}((n.log\ n)^3)$-time transformation from Grz into decidable fragments of classical first-order logic. In *Automated Deduction in Classical and Non Classical Logics*, volume 1761 of *Lecture Notes in Artificial Intelligence*, pages 153–167. Springer, 2000.

[DG02a] S. Demri and R. Goré. Display calculi for nominal tense logics. *Journal of Logic and Computation*, 12(6):993–1016, 2002.

[DG02b] S. Demri and R. Goré. Theoremhood preserving maps characterising cut elimination for modal provability logics. *Journal of Logic and Computation*, 12(5):861–884, 2002.

[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.

[DG06] S. Demri and R. Gascon. The effects of bounding syntactic resources on presburger LTL. Technical Report LSV-06-5, LSV, 2006. 36 pages. Short version under submission.

[DG07] S. Demri and R. Gascon. The effects of bounding syntactic resources on presburger ltl (extended abstract). In *TIME'07*. IEEE, 2007. to appear.

[dGL94] G. de Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *AAAI'94*, pages 205–212, 1994.

[DK98] S. Demri and B. Konikowska. Relative similarity logics are decidable: reduction to FO^2 with equality. In *JELIA'98*, volume 1489 of *Lecture Notes in Artificial Intelligence*, pages 279–293. Springer, 1998.

[DL06a] S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. In *LICS'06*, pages 17–26. IEEE, 2006.

[DL06b] S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. arXiv:cs.LO/0610027, October 2006. 35 pages, completed version of [DL06a], under submission.

[DL06c] S. Demri and D. Lugiez. Complexity of modal logics with Presburger constraints. Technical Report LSV-06-15, LSV, 2006. 31 pages. Corrected version of [DL06d].

[DL06d] S. Demri and D. Lugiez. Presburger Modal Logic is PSPACE-complete. In *IJCAR'06*, volume 4130 of *Lecture Notes in Computer Science*, pages 541–556. Springer, 2006.

[DLN05] S. Demri, R. Lazić, and D. Nowak. On the freeze quantifier in constraint LTL: decidability and complexity. In *TIME'05*, pages 113–121. IEEE Computer Society Press, 2005.

[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.

[DLS02] S. Demri, F. Laroussinie, and Ph. Schnoebelen. A parametric analysis of the state explosion problem in model checking (extended abstract). In *STACS'02*, volume 2285 of *Lecture Notes in Computer Science*, pages 620–631. Springer, 2002.

[DLS06] S. Demri, F. Laroussinie, and Ph. Schnoebelen. A parametric analysis of the state explosion problem in model checking. *Journal of Computer and System Sciences*, 72(4):547–575, 2006.

[DMP95] G. D'Agostino, A. Montanari, and A. Policriti. A set-theoretical translation method for (poly)modal logics. In *STACS'95*, volume 900 of *Lecture Notes in Computer Science*, pages 217–228. Springer, 1995.

[dN98] H. de Nivelle. A resolution decision procedure for the guarded fragment. In *CADE'98*, volume 1421 of *Lecture Notes in Artificial Intelligence*, pages 191–204. Springer, 1998.

[dN99] H. de Nivelle. Translation of S4 and K5 into GF and 2VAR, 1999. Manuscript, available from `http://www.mpi-sb.mpg.de/~nivelle`.

[dN01] H. de Nivelle. Translation of S4 and K5 into GF and 2VAR, April 2001. Slides available from `http://www.mpi-sb.mpg.de/~nivelle/` on WWW.

[DN07] S. Demri and D. Nowak. Reasoning about transfinite sequences. *International Journal of Foundations of Computer Science*, 18(1):87–112, 2007. Special issue dedicated to selected papers from ATVA'05.

[DO98] S. Demri and E. Orłowska. Logical analysis of indiscernibility. In *[Oe98]*, pages 347–380, 1998.

[DO02] S. Demri and E. Orłowska. *Incomplete Information: Structure, Inference, Complexity*. EATCS Monographs. Springer, 2002.

[DO07] S. Demri and E. Orłowska. Relative nondeterministic information logic is EXPTIME-complete. *Fundamenta Informaticae*, 75(1–4):163–178, 2007.

[DPK03] Z. Dang, P. San Pietro, and R. Kemmerer. Presburger liveness verification of discrete timed automata. *Theoretical Computer Science*, 299:413–438, 2003.

[dR92] M. de Rijke. The modal logic of inequality. *The Journal of Symbolic Logic*, 57(2):566–584, 1992.

[DS00] S. Demri and J. Stepaniuk. Computational complexity of multimodal logics based on rough sets. *Fundamenta Informaticae*, 44(4):373–396, 2000.

[DS02a] S. Demri and U. Sattler. Automata-theoretic decision procedures for information logics. *Fundamenta Informaticae*, 53(1):1–22, 2002.

[DS02b] S. Demri and Ph. Schnoebelen. The complexity of propositional linear temporal logics in simple cases. *Information and Computation*, 174(1):84–103, 2002.

[DSV04] A. Deutsch, L. Sui, and V. Vianu. Specification and verification of data-driven web services. In *PODS'04, Paris*, pages 71–82, 2004.

[DT90] M. Dauchet and S. Tison. The theory of ground rewrite systems is decidable. In *LICS'90*, pages 242–256, 1990.

[EFM99] J. Esparza, A. Finkel, and R. Mayr. On the verification of broadcast protocols. In *LICS'99*, pages 352–359, 1999.

[FdR06] M. Franceschet and M. de Rijke. Model checking hybrid logics (with an application to semistructured data). *Journal of Applied Logic*, 4(3):279–304, 2006.

[FdRS03] M. Franceschet, M. de Rijke, and B.-H. Schlingloff. Hybrid logics on linear structures: Expressivity and complexity. In *10th Int. Symp. Temporal Representation and Reasoning and 4th Int. Conf. Temporal Logic (TIME-ICTL)*, pages 164–171. IEEE, 2003.

[FHMV95] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. The MIT Press, 1995.

[FI87] M. Fischer and N. Immerman. Interpreting logics of knowledge in propositional dynamic logic with converse. *Information Processing Letters*, 25:175–181, 1987.

[Fin72] K. Fine. In so many possible worlds. *Notre Dame Journal of Formal Logic*, 13(4):516–520, 1972.

[Fin75] K. Fine. Some connections between elementary and modal logic. In *3rd Scandinavian Logic Symposium*, pages 15–31. North-Holland, 1975.

[FIS03] A. Finkel, S. Purushothaman Iyer, and G. Sutre. Well-abstracted transition systems: Applications to FIFO automata. *Information and Computation*, 181(1):1–31, 2003.

[Fit83] M. Fitting. *Proof Methods for Modal and Intuitionistic Logics*. D. Reidel Publishing Co., 1983.

[Fit02] M. Fitting. Modal logic between propositional and first-order. *Journal of Logic and Computation*, 12(6):1017–1026, 2002.

[FL79] M. Fischer and R. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18:194–211, 1979.

[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, Berlin, 2002.

[FO97] L. Fribourg and H. Olsén. Proving safety properties of infinite state systems by compilation into presburger arithmetic. In *CONCUR'97*, volume 1243 of *Lecture Notes in Computer Science*, pages 213–227. Springer, 1997.

[FR74] M. Fischer and M. Rabin. Super-exponential complexity of Presburger arithmetic. In *Complexity of Computation*, volume 7 of *SIAM-AMS proceedings*, pages 27–42. American Mathematical Society, 1974.

[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.

[FtC05] M. Franceschet and B. ten Cate. Guarded fragments with constants. *Journal of Logic, Language, and Information*, 14(3):281–288, 2005.

[Gab75] D. Gabbay. Decidability results in non-classical logics. *Annals of Mathematical Logic*, 8:237–295, 1975.

[Gab76] D. Gabbay. *Investigations in Modal and Tense Logics with Applications*. D. Reidel, 1976.

[Gab81] D. Gabbay. Expressive functional completeness in tense logic. In *Aspects of Philosophical Logic*, pages 91–117. Reidel, 1981.

[Gar79] J.L. Gardies. *Essai sur la logique des modalités*. Philosophie d'aujourd'hui. P.U.F., 1979.

[Gar86] G. Gargov. Two completeness theorems in the logic for data analysis. Technical Report 581, Institute of Computer Science, Polish Academy of Sciences, Warsaw, 1986.

[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.

[GdN99] H. Ganzinger and H. de Nivelle. A superposition decision procedure for the guarded fragment with equality. In *LICS'99*, pages 295–305, 1999.

[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, St Augustine, Florida*, pages 460–464, 2003.

[GKV97] E. Grädel, Ph. Kolaitis, and M. Vardi. On the decision problem for two-variable first-order logic. *The Bulletin of Symbolic Logic*, 3(1):53–69, 1997.

[GKWZ03] D. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-dimensional modal logics: theory and practice*. Cambridge University Press, 2003.

[GMV99] H. Ganzinger, C. Meyer, and M. Veanes. The two-variable guarded fragment with transitive relations (extended abstract). In *LICS'99*, pages 24–34. IEEE Computer Society Press, 1999.

[GN05] R. Goré and L. Nguyen. A tableau calculus with automaton-labelled formulae for regular grammar logics. In *TABLEAUX'05*, volume 3702 of *Lecture Notes in Artificial Intelligence*, pages 138–152. Springer, 2005.

[Göd33] K. Gödel. Zur intuitionistischen arithmetik und zahlentheorie. *Ergebnisse eines mathematischen Kolloquiums*, 4:34–38, 1933. English translation in [Göd86], pp 286–295.

[Göd86] K. Gödel. *Collected works, Volume I*. Oxford University Press, Oxford, 1986.

[Göl07] S. Göller. On the complexity of reasoning about dynamic policies. In *CSL'07*, 2007. To appear.

[Gor91] R. Goré. Semi-analytic tableaux for propositional modal logics with applications to nonmonotonicity. *Logique et Analyse*, 133–134:73–104, 1991.

[Gor96] V. Goranko. Hierarchies of modal and temporal logics with references pointers. *Journal of Logic, Language, and Information*, 5:1–24, 1996.

[Gor99] R. Goré. Tableaux methods for modal and temporal logics. In *Handbook of Tableaux Methods*, pages 297–396. Kluwer, 1999.

[Gou02] J. Goubault-Larrecq. Vérification de protocoles cryptographiques: la logique à la rescousse! In *Actes du 1er Workshop International sur la Sécurité des Communications sur Internet (SECI'02)*, pages 119–152, Tunis, Tunisia, 2002. INRIA.

[GP92] V. Goranko and S. Passy. Using the universal modality: gains and questions. *Journal of Logic and Computation*, 2(1):5–30, 1992.

[Grä99a] E. Grädel. Decision procedures for guarded logics. In *CADE'99*, volume 1632 of *Lecture Notes in Artificial Intelligence*, pages 31–51. Springer, 1999.

[Grä99b] E. Grädel. On the restraining power of guards. *The Journal of Symbolic Logic*, 64(4):1719–1742, 1999.

[Gre01] D. Gregory. Completeness and decidability results for some propositional modal logics containing "actually" operators. *Journal of Philosophical Logic*, 30(1):57–78, 2001.

[Grz69] A. Grzegorczyk. A philosophical plausible formal interpretation of intuitionistic logic. *Indapationes Mathematicae*, 26:596–601, 1969.

[GS66] S. Ginsburg and E. Spanier. Semigroups, Presburger formulas and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966.

[GW94] P. Godefroid and P. Wolper. A partial approach to model checking. *Information and Computation*, 110(2):305–326, 1994.

[GW99] E. Grädel and I. Walukiewicz. Guarded fixed point logic. In *LICS'99*, pages 45–54, 1999.

[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.

[Har83] D. Harel. Recurring dominoes: making the highly undecidable highly understandable. In *Foundations of Computing Theory*, volume 158 of *Lecture Notes in Computer Science*, pages 177–194. Springer-Verlag, 1983.

[Har85] D. Harel. Recurring dominoes: making the highly undecidable highly understandable. *Annals of Discrete Mathematics*, 24:51–72, 1985.

[HB91] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *KR'91*, pages 335–346, 1991.

[Hem94] E. Hemaspaandra. Complexity transfer for modal logic (extended abstract). In *LICS'94*, pages 164–173, 1994.

[Hem96] E. Hemaspaandra. The price of universality. *Notre Dame Journal of Formal Logic*, 37(2):173–203, 1996.

[Hen90] Th. Henzinger. Half-order modal logic: how to prove real-time properties. In *PODC'90*, pages 281–296. ACM, 1990.

[Her89] A. Herzig. *Raisonnement automatique en logique modale et algorithmes d'unification*. PhD thesis, Université P. Sabatier, Toulouse, 1989.

[HHI+01] J. Halpern, R. Harper, N. Immerman, Ph. Kolaitis, M. Vardi, and V. Vianu. On the unusual effectiveness of logic in computer science. *Bulletin of Symbolic Logic*, 7(2):213–236, 2001.

[HKT00] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, Cambridge, MA, 2000.

[Hla02] J. Hladik. Implementation and optimisation of a tableau algorithm for the guarded fragment. In *TABLEAUX'02*, volume 2381 of *Lecture Notes in Computer Science*, pages 145–159. Springer Verlag, 2002.

[HLMR02] M. Heymann, F. Lin, G. Meyer, and S. Resmerita. Analysis of Zeno behaviors in hybrid systems. In *Proceedings of the 41st IEEE Conference on Decision and Control, Las Vegas, Nevada, USA*, pages 2379–2384, 2002.

[HM85a] J. Halpern and Y. Moses. A guide to the modal logics of knowledge and belief: preliminary draft. In *IJCAI'85*, pages 480–490. Morgan Kaufmann, 1985.

[HM85b] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the Association for Computing Machinery*, 32(1):137–161, 1985.

[HR83] J. Halpern and H. Reif. The propositional dynamic logic of deterministic, well-structured programs. *Theoretical Computer Science*, 27:127–165, 1983.

[HRS76] H. Hunt, D. Rosenkrantz, and Th. Szymanski. On the equivalence, containment, and covering problems for the regular and context-free languages. *Journal of Computer and System Sciences*, 12:222–268, 1976.

[HS99] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3):385–410, 1999.

[HS03] U. Hustadt and R. Schmidt. A principle for incorporating axioms into the first-order translation of modal formulae. In *CADE'03*, volume 2741 of *Lecture Notes in Artificial Intelligence*, pages 412–426. Springer, 2003.

[HS04] I. Horrocks and U. Sattler. Decidability of SHIQ with complex role inclusion axioms. *Artificial Intelligence*, 160(1–2):79–104, 2004.

[HST00] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3):239–263, 2000.

[HT99] J. Henriksen and P. Thiagarajan. Dynamic linear time temporal logic. *Annals of Pure and Applied Logic*, 96(1–3):187–207, 1999.

[HW91] J.C. Hemmer and P. Wolper. Ordinal finite automata and languages (extended abstract). Technical report, Université of Liège, 1991.

[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.

[Jär06] J. Järvinen. Book review of [DO02]. *Studia Logica*, 84:469–475, 2006.

[JJKS97] J. Jensen, M. Jorgensen, N. Klarlund, and M. Schwartzbach. Automatic verification of pointer programs using monadic second-order logic. In *PLDI'97*, pages 226–236. ACM, 1997.

[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.

[JM96] L. Jategaonkar and A. R. Meyer. Deciding true concurrency equivalences on safe, finite nets. *Theoretical Computer Science*, 154(1):107–143, 1996.

[Jur98] M. Jurdzinski. Deciding the winner in parity games is in UP ∩ co-UP. *Information Processing Letters*, 68(3):119–124, 1998.

[Kam68] J. Kamp. *Tense Logic and the theory of linear order*. PhD thesis, UCLA, USA, 1968.

[Kaz04] Y. Kazakov. A polynomial translation from the two-variable guarded fragment with number restrictions to the guarded fragment. In *JELIA'04*, volume 3229 of *Lecture Notes in Computer Science*, pages 372–384. Springer, 2004.

[KF94] M. Kaminski and N. Francez. Finite-memory automata. *Theoretical Computer Science*, 134(2):329–363, 1994.

[Kie05] E. Kieroński. Results on the guarded fragment with equivalence or transitive guards. In *CSL'05*, volume 3634 of *Lecture Notes in Computer Science*, pages 309–324. Springer, 2005.

[Kie06] E. Kieroński. On the complexity of the two-variable guarded fragment with transitive guards. *Information and Computation*, 204:1663–1703, 2006.

[Kon98] B. Konikowska. A logic for reasoning about similarity. In *[Oe98]*, pages 462–491, 1998.

[Kra96] M. Kracht. Power and weakness of the modal display calculus. In *Proof theory of modal logic*, pages 93–121. Kluwer, 1996.

[KRM05] V. Kuncak, M. Rinard, and B. Manette. On algorithms and complexity for sets with cardinality constraints. In *Deduction and Applications, 23.-28. October 2005. Dagstuhl Seminar Proceedings 05431 Internationales Begegnungs- und Forschungszentrum (IBFI)*, October 2005.

[KSV02] O. Kupferman, U. Sattler, and M.Y. Vardi. The complexity of the graded μ-calculus. In *CADE'02*, volume 2392 of *Lecture Notes in Computer Science*, pages 423–437, 2002.

[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. Vardi. Memoryful Branching-Time Logic. 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 Association for Computing Machinery*, 47(2):312–360, 2000.

[Lad77] R. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal of Computing*, 6(3):467–480, 1977.

[Lar94] F. Laroussinie. *Logique temporelle avec passé pour la spécification et la vérification des systèmes réactifs*. PhD thesis, Institut National Polytechnique de Grenoble, 1994.

[Lar05] F. Laroussinie. *Model checking temporisé — Algorithmes efficaces et complexité*. Mémoire d'habilitation, Université Paris 7, Paris, 2005.

[LAS00] T. Lev-Ami and M. Sagiv. TVLA: A system for implementing static analyses. In *SAS'00*, volume 1824 of *Lecture Notes in Computer Science*, pages 280–301, 2000.

[Laz06] R. Lazić. Safely freezing LTL. In *FSTTCS'06*, volume 4337, pages 381–392. LNCS, 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, Berlin, 2001.

[LM05] C. Lutz and M. Miličić. 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.

[LMP03] U. Dal Lago, A. Montanari, and G. Puppis. Towards compact and tractable automaton-based representations of time granularities. In *ICTCS 2003*, volume 2841 of *Lecture Notes in Computer Science*, pages 72–85. Springer, Berlin, 2003.

[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.

[LP00] O. Lichtenstein and A. Pnueli. Propositional temporal logics: Decidability and completeness. *Logic Journal of the IGPL*, 8(1):55–85, 2000.

[LP05] A. Lisitsa and I. Potapov. Temporal logic with predicate λ-abstraction. In *TIME'05*, pages 147–155. IEEE, 2005.

[LR03] Ch. Löding and Ph. Rohde. Model checking and satisfiability for sabotage modal logic. In *FST&TCS'03*, volume 2914 of *Lecture Notes in Computer Science*, pages 302–313. Springer, 2003.

[LS00] F. Laroussinie and Ph. Schnoebelen. The state explosion problem from trace to bisimulation equivalence. In *FOSSACS'00*, volume 1784 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2000.

[LS05] J. Leroux and G. Sutre. Flat counter systems are everywhere! In *ATVA'05*, volume 3707 of *Lecture Notes in Computer Science*, pages 489–503. Springer, 2005.

[LS06] Ch. Löding and O. Serre. Propositional dynamic logic with recursive programs. In *FOSSACS'06*, volume 3921 of *Lecture Notes in Computer Science*, pages 292–306. Springer, 2006.

[Lut03] C. Lutz. Description logics with concrete domains—a survey. In *Advances in Modal Logics Volume 4*, pages 265–296. King's College Publications, 2003.

[Lut04] C. Lutz. NEXPTIME-complete description logics with concrete domains. *ACM Transactions on Computational Logic*, 5(4):669–705, 2004.

[Lut05] C. Lutz. PDL with intersection and converse is decidable. In *CSL'05*, volume 3634 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2005.

[LW04] C. Lutz and D. Walther. PDL with negation of atomic programs. In D. Basin and M. Rusinowitch, editors, *IJCAR'04*, volume 3097 of *Lecture Notes in Computer Science*, pages 259–273. Springer, 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.

[Lyn77] N. Lynch. Log Space recognition and translation of parentheses languages. *Journal of the Association for Computing Machinery*, 24(4):583–590, 1977.

[Mar02] M. Marx. Narcissists, stepmothers and spies. In *2002 International Workshop on Description Logics*, volume 53 of *CEUR Workshop Proceedings*, 2002.

[Mar03a] N. Markey. *Logiques temporelles pour la vérification : Expressivité, complexité, algorithmes*. PhD thesis, Université d'Orléans, 2003.

[Mar03b] M. Marx. XPath and Modal Logics of finite DAG's. In *TABLEAUX'03*, volume 2796 of *Lecture Notes in Artificial Intelligence*, pages 150–164. Springer, 2003.

[Mas97] F. Massacci. Tableaux methods for access control in distributed systems. In *TABLEAUX'97*, volume 1227 of *Lecture Notes in Artificial Intelligence*, pages 246–260. Springer, 1997.

[Mas00] F. Massacci. Single steps tableaux for modal logics. *Journal of Automated Reasoning*, 24(3):319–364, 2000.

[Mey73] A.R. Meyer. Weak second order theory of successor is not elementary-recursive. Technical Report MAC TM-38, MIT, 1973.

[Mey88] J.J. Ch. Meyer. A different approach to deontic logic: deontic logic viewed as a variant of dynamic logic. *Notre Dame Journal of Formal Logic*, 29(1):109–136, 1988.

[Min67] M. Minsky. *Computation: Finite and Infinite Machines*. Prentice Hall, Englewood Cliffs, NJ, 1967.

[MMS99] M. Marx, Sz. Mikulas, and S. Schlobach. Tableau calculus for local cubic modal logic and its implementation. *Logic Journal of the IGPL*, 7(6):755–778, 1999.

[Moo77] R. Moore. Reasoning about knowledge and action. In *IJCAI-5*, pages 223–227, 1977.

[Mor76] Ch. Morgan. Methods for automated theorem proving in non classical logics. *IEEE Transactions on Computers*, 25(8):852–862, 1976.

[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.

[MP97] A. Montanari and A. Policriti. A set-theoretic approach to automated deduction in graded modal logics. In *IJCAI'97*, pages 196–201, 1997.

[MS97] A. Mateescu and A. Salomaa. Formal languages: an introduction and a synopsis. In *Handbook of Formal Languages - Volume 1: Word, Language and Grammar*, pages 1–40. Springer, 1997.

[MS03] N. Markey and Ph. Schnoebelen. Model checking a path. In *CONCUR'03*, volume 2761 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2003.

[MS06] N. Markey and Ph. Schnoebelen. Mu-calculus path checking. *Information Processing Letters*, 97(6):225–230, 2006.

[MW95] A. Mendelzon and P. Wood. Finding regular simple paths in graph databases. *SIAM Journal of Computing*, 24(6):1235–1258, 1995.

[MW04] A. Muscholl and I. Walukiewicz. An NP-complete fragment of LTL. In *DLT'04*, volume 3340 of *Lecture Notes in Computer Science*, pages 334–344. Springer, 2004.

[Nak93] A. Nakamura. On a logic based on graded modalities. *IEICE Transactions*, E76-D(5):527–532, 1993.

[NO80] A. Nakamura and H. Ono. On the size of refutation Kripke models for some linear modal and tense logics. *Studia Logica*, 39(4):325–333, 1980.

[NS92] M. Niezette and J. Stevenne. An efficient symbolic representation of periodic time. In *Proc. of the International Conference on Information and Knowledge Management, Baltimore, Maryland*, volume 752 of *Lecture Notes in Computer Science*, pages 161–168. Springer, 1992.

[NSV04] F. Neven, T. Schwentick, and V. Vianu. Finite state machines for strings over infinite alphabets. *ACM Transactions on Computational Logic*, 5(3):403–435, 2004.

[Oe98] E. Orłowska (ed.). *Incomplete Information: Rough Set Analysis*, volume 13 of *Studies in Fuzziness and Soft Computing*. Physica, Heidelberg, 1998.

[ONdRG01] H.J. Ohlbach, A. Nonnengart, M. de Rijke, and D. Gabbay. Encoding two-valued non-classical logics in classical logic. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, pages 1403–1486. Elsevier Science Publishers B.V., 2001.

[OSe01] E. Orłowska and A. Szałas (eds.). *Relational Methods for Computer Science Applications*, volume 65 of *Studies in Fuzziness and Soft Computing*. Physica, Heidelberg, 2001.

[OSH96] H.J. Ohlbach, R. Schmidt, and U. Hustadt. Translating graded modalities into predicate logics. In H. Wansing, editor, *Proof theory of modal logic*, pages 253–291. Kluwer, 1996.

[OTTR05] H. Ohsaki, J.M. Talbot, S. Tison, and Y. Roos. Monotone AC-tree automata. In *LPAR'05*, volume 3835 of *Lecture Notes in Artificial Intelligence*, pages 337–351. Springer, 2005.

[OW05] J. Ouaknine and J. Worrell. On the decidability of metric temporal logic. In *LICS'05*, pages 188–197. IEEE, 2005.

[Pap81] Chr. Papadimitriou. On the complexity of integer programming. *Journal of the Association for Computing Machinery*, 28(4):765–768, 1981.

[Pap94] Ch. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.

[PF77] A. Prior and K. Fine. *Worlds, Times and Selves*. University of Massachusetts Press, 1977.

[Pnu84] A. Pnueli. The temporal logic of programs. In *FOCS'77*, pages 46–57. IEEE Computer Society Press, 1984.

[Pot04] I. Potapov. From Post systems to the reachability problems for matrix semigroups and multicounter automata. In *DTL'04*, volume 3340 of *Lecture Notes in Computer Science*, pages 345–356. Springer, 2004.

[Pra79] V. Pratt. Models of program logics. In *FOCS'79*, pages 115–122, 1979.

[Pra81] V. Pratt. Using graphs to understand PDL. In *Workshop on Logics of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 387–396. Springer, 1981.

[Pre29] M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. *Comptes Rendus du premier congrès de mathématiciens des Pays Slaves, Warszawa*, pages 92–101, 1929.

[Pri57] A. Prior. *Time and Modality*. Clarendon Press, Oxford, 1957.

[Pri67] A. Prior. *Past, Present and Future*. Oxford University Press, 1967.

[PS04] E. Pacuit and S. Salame. Majority logic. In *KR'04*, pages 598–605. AAAI Press, 2004.

[Pup06] G. Puppis. *Automata for branching and layered temporal structures*. PhD thesis, Universitá Degli Studi Di Udine, 2006.

[PW04] R. Pucella and V. Weissman. Reasoning about dynamic policies. In *FOSSACS'04*, volume 2987 of *Lecture Notes in Computer Science*, pages 453–467, 2004.

[Rab97] A. Rabinovich. Complexity of equivalence problems for concurrent systems of finite agents. *Information and Computation*, 139:111–129, 1997.

[Rab00] A. Rabinovich. Symbolic model checking for μ-calculus requires exponential time. *Theoretical Computer Science*, 243(1–2):467–475, 2000.

[Rab06] A. Rabinovich. Private communication, August 2006.

[Rev02] P. Revesz. *Introduction to Constraint Databases*. Springer, 2002.

[Rey02] J.C. Reynolds. Separation logic: a logic for shared mutable data structures. In *LICS'02*, pages 55–74. IEEE, 2002.

[Roh97] S. Rohde. *Alternating Automata and The Temporal Logic of Ordinals*. PhD thesis, University of Illinois, 1997.

[Ros82] J.G. Rosenstein. *Linear Ordering*. Academic Press, 1982.

[RU71] N. Rescher and A. Urquhart. *Temporal Logic*. Springer, 1971.

[Sah75] H. Sahlqvist. Completeness and correspondence in the first and second order semantics for modal logics. In *3rd Scandinavian Logic Symposium, Uppsala, Sweden, 1973*, pages 110–143. North Holland, 1975.

[Sat96] U. Sattler. A concept language extended with different kinds of transitive roles. In *20. Deutsche Jahrestagung für Künstliche Intelligenz*. LNM 1137, Springer, 1996.

[SC85] P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the Association for Computing Machinery*, 32(3):733–749, 1985.

[Sch91] K. Schild. A correspondence theory for terminological logics: preliminary report. In *IJCAI'91*, pages 466–471, 1991.

[Sch02] Ph. Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters*, 83(5):251–261, 2002.

[Sch04] K. Schneider. *Verification of Reactive Systems: Formal Methods and Algorithms*. Springer, 2004.

[SCM03] U. Sattler, D. Calvanese, and R. Molitor. Relationships with other formalisms. In *Handbook of Description Logics*, pages 142–183. Cambridge University Press, 2003. to appear.

[Seg71] K. Segerberg. An essay in classical modal logic (three vols.). Technical Report Filosofiska Studier nr 13, Uppsala Universitet, 1971.

[Seg81] K. Segerberg. A note on the logic of elsewhere. *Theoria*, 47:183–187, 1981.

[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.

[Ser04] O. Serre. *Contribution à l'étude des jeux sur des graphes de processus de pile*. PhD thesis, Université Paris 7 - Denis Diderot, 2004.

[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.

[SH04] R. Schmidt and U. Hustadt. A principle for incorporating axioms into first-order translation of modal formulae. *ACM Transactions on Computational Logic*, 2004. to appear.

[SI00] H. Sakamoto and D. Ikeda. Intractability of decision problems for finite-memory automata. *Theoretical Computer Science*, 231(2):297–308, 2000.

[Sol76] R. Solovay. Provability interpretations of modal logics. *Israel Journal of Mathematics*, 25:287–304, 1976.

[Spa93a] E. Spaan. *Complexity of Modal Logics*. PhD thesis, ILLC, Amsterdam University, 1993.

[Spa93b] E. Spaan. The complexity of propositional tense logics. In *Diamonds and Defaults*, pages 287–309. Kluwer Academic Dordrecht, Series Studies in Pure and Applied Intensional Logic, Vol. 229, 1993.

[SSMH04] H. Seidl, Th. Schwentick, A. Muscholl, and P. Habermehl. Counting in trees for free. In *ICALP'04*, volume 3142 of *Lecture Notes in Computer Science*, pages 1136–1149. Springer, 2004.

[ST04] W. Szwast and L. Tendera. The guarded fragment with transitive guards. *Annals of Pure and Applied Logic*, 128:227–276, 2004.

[SV01] U. Sattler and M. Vardi. The hybrid mu-calculus. In *IJCAR'01*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 76–91. Springer, 2001.

[SW07] Th. Schwentick and V. Weber. Bounded-variable fragments of hybrid logics. In *STACS'07*, volume 4393 of *Lecture Notes in Computer Science*, pages 561–572. Springer, 2007.

[TC98] D. Toman and J. Chomicki. DATALOG with integer periodicity constraints. *Journal of Logic Programming*, 35(3):263–290, 1998.

[tCF05] B. ten Cate and M. Franceschet. On the complexity of hybrid logics with binders. In *CSL'05*, volume 3634 of *Lecture Notes in Computer Science*, pages 339–354. Springer, 2005.

[Thi04] V. Thion. *Bases de données, contraintes d'intégrité et logiques modales*. PhD thesis, Université Paris XI Orsay, 2004.

[Tob00] S. Tobies. PSPACE reasoning for graded modal logics. *Journal of Logic and Computation*, 10:1–22, 2000.

[Tra63] B. Trahtenbrot. Impossibility of an algorithm for the decision problem in finite classes. *American Mathematical Society - Translation Series*, 23(2):1–5, 1963.

[Tuo90] H. Tuominen. Dynamic logic as a uniform framework for theorem proving in intensional logic. In *CADE'90*, volume 449 of *Lecture Notes in Computer Science*, pages 514–527. Springer, 1990.

[Tza99] M. Tzakova. Tableau calculi for hybrid logics. In *TABLEAUX'99*, volume 1617 of *Lecture Notes in Artificial Intelligence*, pages 278–292. Springer, 1999.

[Vak87] D. Vakarelov. Abstract characterization of some knowledge representation systems and the logic NIL of nondeterministic information. In *AIMSA'87*, pages 255–260. North-Holland, 1987.

[Vak90] D. Vakarelov. Modal characterization of the classes of finite and infinite quasi-ordered sets. In *Mathematical Logic*, pages 373–387. Plenum, New York, 1990.

[Var88] M. Vardi. A temporal fixpoint calculus. In *15th Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, San Diego*, pages 250–259. ACM, 1988.

[Var97] M. Vardi. Why is modal logic so robustly decidable? In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science 31, American Mathematical Society*, pages 149–183, 1997.

[vB76] J. van Benthem. *Modal Correspondence Theory*. PhD thesis, Mathematical Institute, Amsterdam University, 1976.

[vB85] J. van Benthem. *Modal logic and Classical Logic*. Bibliopolis, Naples, 1985.

[vB05] J. van Benthem. An essay on sabotage and obstruction. In *Mechanizing Mathematical Reasoning, Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday*, volume 2605 of *Lecture Notes in Computer Science*, pages 268–276. Springer, 2005.

[vdH92] W. van der Hoek. On the semantics of graded modalities. *Journal of Applied Non-Classsical Logics*, 2(1):81–123, 1992.

[vdHdR95] W. van der Hoek and M. de Rijke. Counting objects. *Journal of Logic and Computation*, 5(3):325–345, 1995.

[vdHM91] W. van der Hoek and J.-J. Meyer. Graded modalities in epistemic logic. *Logique et Analyse*, 133–134:251–270, 1991.

[vdM96] R. van der Meyden. The dynamic logic of permission. *Journal of Logic and Computation*, 6(3):465–479, 1996.

[Vis95] A. Visser. A course on bimodal provability logic. *Annals of Pure and Applied Logic*, 73:109–142, 1995.

[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.

[Wan94] H. Wansing. Sequent calculi for normal modal propositional logics. *Journal of Logic and Computation*, 4(2):125–142, 1994.

[Wan98] H. Wansing. *Displaying Modal Logic*, volume 3 of *Trends in Logic*. Kluwer Academic Publishers, Dordrecht, 1998.

[Wij00] J. Wijsen. A string based-model for infinite granularities. In *AAAI Workshop on Spatial and Temporal Granularity*, pages 9–16. AAAI Press, 2000.

[WM94] R. Wieringa and J.-J. Ch. Meyer. Applications of deontic logic to computer science: a concise overview. In *Deontic Logic in Computer Science: Normative System Specification*, pages 17–40. John Wiley and Sons Ltd, 1994.

[Woj84] J. Wojciechowski. Classes of transfinite sequences accepted by nondeterministic finite automata. *Annales Societatid Mathematicae Polonae*, pages 191–223, 1984.

[Wol83] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56:72–99, 1983.

[WZ97] F. Wolter and M. Zakharyaschev. On the relation between intuitionistic and classical modal logics. *Algebra and Logic*, 36:121–155, 1997.

[WZ98] F. Wolter and M. Zakharyaschev. Intuitionistic modal logics as fragments of classical bimodal logics. In *Logic at Work. Essays Dedicated to the Memory of Helena Rasiowa.*, pages 168–183. Physica Verlag, Berlin, 1998.

[WZ00] F. Wolter and M. Zakharyaschev. Spatio-temporal representation and reasoning based on RCC-8. In *KR'00*, pages 3–14. Morgan Kaufmann, 2000.

[YRSW03] E. Yahav, Th. Reps, M. Sagiv, and R. Wilhelm. Verifying temporal heap properties specified via evolution logic. In *ESOP'03*, volume 2618 of *Lecture Notes in Computer Science*, pages 204–22. Springer, 2003.

[ZL03] S. Dal Zilio and D. Lugiez. XML schema, tree logic and sheaves automata. In *RTA'03*, volume 2706 of *Lecture Notes in Computer Science*, pages 246–263. Springer, 2003.

[ZL06] S. Dal Zilio and D. Lugiez. XML schema, tree logic and sheaves automata. *Applicable Algebra in Engineering, Communication and Computing*, 17(5):337–377, 2006.