Familles de graphes de présentation finie, propriétés et applications

Christophe Morvan

UNIVERSITÉ DE PARIS-EST MARNE-LA-VALLÉE
INSTITUT GASPARD MONGE

HABILITATION À DIRIGER LES RECHERCHES

présentée par

Christophe MORVAN

Familles de graphes de présentation finie, propriétés et applications

Soutenue publiquement le 19 novembre 2014

devant le jury composé de

Ahmed Bouajjani	Professeur, Université Paris-Diderot, Président
Alain Finkel	Professeur, Ens Cachan, Rapporteur
Thierry Massart	Professeur, Université Libre de Bruxelles, Examinateur
Dominique Perrin	Professeur, Université Paris-Est Marne-la-Vallée, Examinateur
Sophie Tison	Professeure, Université de Lille 1, Rapportrice
Wolfgang Thomas	Professeur, RWTH Aachen, Rapporteur

Table des matières

Introduction

Ce document présente la synthèse d'une part importante des travaux que j'ai réalisés depuis l'obtention de mon doctorat en 2001. L'ambition de ce manuscrit est de décrire un certain nombre de familles de graphes infinis, et mettre en évidence la diversité de leurs présentations, tout en soulignant la communauté des questions qui se posent pour ces objets.

I Graphes et logique

Le travail de Büchi établissant la décidabilité de la théorie du second ordre monadique (MSO) sur la structure des entiers munis de la relation *successeur* est pionnier [27]. Il a ouvert la voie à une multitude de travaux ultérieurs qui ont étendu cette décidabilité à des familles de graphes de plus en plus complexes. Le résultat de Rabin [126] étend ce résultat aux arbres complets. Plusieurs résultats portant sur des arbres particuliers ont été obtenus par la suite. En 1985 Muller et Schupp [119] ont réussi à généraliser ce résultat à la famille des graphes des transitions des automates à pile. En 1996 Caucal s'est appuyé sur le résultat de Rabin, ainsi que sur une opération *simple* préservant la décidabilité de MSO : la substitution rationnelle inverse, pour étendre le résultat de Muller et Schupp à une famille plus générale : les graphes *préfixe-reconnaissables* [35]. Ce dernier a encore approfondi ce résultat en s'appuyant sur une seconde opération préservant MSO : le dépliage [40] ; ainsi il a défini une hiérarchie (stricte) de familles de graphes possédant une MSO-théorie décidable. Par la suite, Carayol et Wöhrle ont [31] démontré que cette hiérarchie correspondait, par niveau, aux graphes des transitions des automates à pile d'ordre supérieur. L'ensemble de ces familles constitue une hiérarchie très générale de graphes (souvent désignée sous le nom de *hiérarchie de Caucal*) dont la MSO-théorie est décidable. Il existe également des graphes dont la MSO-théorie est décidable mais qui ne sont pas dans la hiérarchie de Caucal, par exemple un peigne dont chaque nœud atteint par a^n donne naissance à une branche $b \uparrow\uparrow (n-1)$ [1], voir l'article de Montanari et Puppis [112].

Lorsqu'on s'intéresse à des familles plus générales, plus expressives, le prix à payer est l'indécidabilité des MSO-théories. La situation devient alors plus complexe : de nombreuses restrictions de la logique MSO existent. La logique du premier ordre est *une* de ces restrictions permettant d'obtenir la décidabilité des théories d'une famille. Les logiques temporelles ou encore les logiques modales offrent une autre approche pour obtenir des propriétés décidables. On peut également se limiter à des propriétés telles que l'accessibilité de certains états ou ensembles d'états. À minima on exigera la décidabilité de l'existence d'un chemin étiqueté par une trace particulière, en d'autres termes, la récursivité de l'ensemble des étiquettes de chemins dans un graphe.

Au cours de ma thèse j'ai étudié une famille très générale de graphes : les graphes rationnels ; leurs relations de transition sont définies par les transducteurs rationnels. La logique du premier ordre (notée FO) est indécidable, en général, pour les graphes de cette famille [113]. Néanmoins, avec Colin Stirling, nous avons montré que les traces de ces graphes étaient exactement les langages contextuels [118]. Avec Arnaud Carayol, nous sommes allés plus loin : nous avons établi que les graphes rationnels qui sont

1. La notation *double-flèche de Knuth*, $\uparrow\uparrow$, représente l'itération de l'exponentiation, ainsi $2 \uparrow\uparrow 3$ représente 2^{2^2}

des arbres (des graphes connexes, dont chaque sommet possède au plus un prédécesseur, et qui a une racine, c'est-à-dire un sommet sans aucun prédécesseur) ont une FO-théorie décidable [30]. Résultat qui cesse d'être vrai si on considère les graphes rationnels acycliques ou si on s'intéresse, sur les arbres rationnels, à la logique FO avec un prédicat binaire exprimant l'accessibilité. Les graphes automatiques sont une restriction importante des graphes rationnels, leurs relations de transitions sont définies par des transducteurs automatiques ou synchrones (lettre-à-lettre). Cette restriction a pour conséquence la décidabilité des FO-théories [17]. Les algorithmes généraux de décision des FO-théories des graphes automatiques ont une complexité non-élémentaire, Dietrich Kuske et Markus Lohrey ont montré en 2009 qu'elle avait une borne supérieure doublement exponentielle en espace pour les structures de degré borné [98].

Un autre type de modèle met particulièrement l'accent sur le parallélisme : les réseaux de Petri, ce modèle est ancien, et a été largement étudié depuis sa définition en 1962 [123]. On pourra se référer par exemple à l'article de synthèse de Javier Esparza de 1997 [57]. Lorsqu'on s'intéresse à ces objets en tant que représentation finie de systèmes infinis, on considère en général que le graphe engendré est formé par la composante accessible (le graphe d'accessibilité) depuis un marquage donné. Le problème de l'accessibilité est central pour les réseaux de Petri, il a été résolu en 1981 [108, 95, 109]. Pour ce modèle, de nombreux autres problèmes sont décidables :

— être borné [91, 124] ;
— être sans interblocage [72] ;
— être persistant [70] ;
— avoir un ensemble semi-linéaire de marquages [75].

La thèse de Hack [72] présente une collection de problèmes qui sont équivalents à celui de l'accessibilité. Hack y démontre également que l'égalité des langages est indécidable pour les réseaux étiquetés [73, 3], néanmoins, ce problème est décidable dès que l'étiquetage est *injectif* où encore pour les réseaux étiquetés déterministes [121] (là encore par une réduction au problème d'accessibilité). Le problème de l'égalité des ensembles de marquages accessibles, étudié par Rabin [10] et par Hack [73], est également indécidable. Ce dernier résultat joue un rôle fondamental dans le résultat d'indécidabilité de FO pour les graphes d'accessibilité des réseaux de Petri obtenu en collaboration avec Philippe Darondeau, Stéphane Demri et Roland Meyer [52].

Globalement l'ensemble de ces familles forment l'arrière plan fondamental de ce document, nous les présenterons ainsi qu'un ensemble de résultats obtenus en les étudiant.

II Applications : diagnostic, opacité, génération formelle de tests et éléments quantitatifs

Dans ce paragraphe nous présenterons les contextes dans lesquels nos travaux ont été appliqués.

Diagnostic. Une branche de la vérification formelle, la *supervision* s'intéresse à la construction de superviseurs (appelés également moniteurs) qui sont exécutés conjointement au système. Ces outils, fournissent des indications sur l'état réel du système au cours de son exécution. En général, dans ce contexte, le modèle contient une partition des événements qui se produisent dans le système en événements observables d'une part et événements inobservables d'autre part. Ainsi le diagnostic, qui sera largement considéré dans ce document, vise à la construction automatique d'un superviseur qui détecte la survenue d'une faute. À l'utilisation, un tel superviseur peut émettre trois signaux : *Oui* (la défaillance s'est produite avec certitude), *Non* (il est impossible que la défaillance se soit produite), *Peut-être* (il est possible, mais pas certain que la défaillance se soit produite). La formalisation de ce type de problème a été faite par Sampath *et al.* [134]. Une observation importante est que, pour certains systèmes, il est possible que la réponse *Peut-être* soit émise arbitrairement longtemps, alors même qu'une

défaillance s'est effectivement produite. Dans un tel cas de figure le superviseur du diagnostic (appelé diagnostiqueur) a un intérêt très limité. Le *problème de la diagnosticabilité* consiste à déterminer si le meilleur diagnostiqueur risque de répondre perpétuellement *Peut-être* lorsqu'une défaillance s'est effectivement produite. Nous considérerons de façon précise ce problème pour des systèmes ayant un comportement infini. Cette question de la diagnosticabilité a été considérée pour les systèmes finis, avec un algorithme polynomial pour le résoudre [32, 150, 89]. Elle a également été étudiée pour les systèmes temporisés, ainsi Tripakis démontre que la diagnosticabilité est équivalente à l'absence de comportement Zénon dans un automate dérivé de l'original, ce qui induit une borne dans PSPACE [145]. Ces résultats ont été étendus : alors que Tripakis considère le problème du diagnostic dans toute sa généralité, il ne s'intéresse pas au diagnostiqueur, qui est construit *à la volée*, Bouyer *et al.* examinent un problème légèrement différent : celui de l'existence d'un diagnostiqueur dans une classe d'automate particulière [20]. Ainsi, lorsqu'on souhaite obtenir un automate temporisé déterministe, le problème est complet pour la classe 2EXPTIME. Et il est complet pour PSPACE pour la sous-classe d'automates temporisés déterministes *à enregistrement d'événements* [1]. Le problème de la diagnosticabilité pour les réseaux de Petri est indécidable, mais il est possible de construire un diagnostiqueur [146]. Baldan *et al.* ont étendu ce contexte [11] à l'aide de techniques de transformations de graphe, ce qui permet à ce modèle de représenter des systèmes avec de la mobilité, et une évolution de la topologie. Les travaux présentés dans ce document sont dans la continuité de ceux que j'ai réalisés avec Pinchinat [116] où nous avons considéré le problème de la diagnosticabilité pour les automates à pile visible, et des extensions d'ordres supérieurs. Nous y établissons l'indécidabilité de ce problème dans le cas général, ce qui nous a conduit à définir une restriction simple pour laquelle le problème devient décidable. Dans cette ligne, [93] établit des résultats similaires pour une classe légèrement différente.

Opacité. Concernant la confidentialité, un problème naturel consiste à déterminer si un observateur de l'exécution d'un système sera en mesure de déterminer avec certitude que le système a atteint l'un des états d'un ensemble distingué, considéré comme *secret*. Cette question a été étudiée par plusieurs auteurs [26, 7, 53]. Il a été démontré que le problème de l'opacité est complet pour PSPACE. Cassez a démontré l'indécidabilité dans le cas général des systèmes infinis [33]. Ce problème ne semble pas avoir été abordé pour les réseaux de Petri. Une approche nouvelle consiste à ne plus apporter à cette question une réponse binaire, mais plutôt probabiliste [12]. Ainsi, le problème devient : un observateur peut-il établir avec certitude que la probabilité que le système ait atteint un état secret est supérieur à une certaine valeur.

Génération formelle de tests de conformité. La génération formelle de tests de conformité consiste à dériver des séries de tests (qu'on appelle *suite de tests*) à partir d'une spécification du système. Chaque test (appelé *cas de test*) est ensuite exécuté sur une *implémentation* du système (par exemple un programme, ou un ensemble de programmes). Chaque cas de test fournit un *verdict* qui reflète la détection d'une non-conformité. Ce mode de fonctionnement a été introduit dans les travaux de Brinksma *et al.* [25]. Pour pouvoir engendrer une suite de tests pertinents à partir d'une modélisation, il est indispensable de pouvoir exprimer avec précision la notion de conformité. En 1996, Tretmans a défini la relation *ioco* [143]. Cette relation formalise la conformité, et permet ensuite de certifier des propriétés de la suite de tests (tels que la correction ou la sévérité). La relation *ioco* est aujourd'hui largement utilisée dans le contexte du test de conformité des systèmes réactifs. Ces objets sont modélisés par des *systèmes à événements discrets* où l'alphabet de ces événements est une partition entre les entrées qui sont fournies au système par un élément extérieur et les sorties qui sont produites par le système lui-même. Il est également possible qu'un système possède des événements internes qui ne sont pas observables de l'extérieur. Lorsque ces systèmes sont à états finis, de nombreux travaux ont traité de la génération de suites de tests, ainsi que de leurs propriétés [87, 144]. Des approches générales ont été étudiées pour des systèmes à états infinis [88, 63]. depuis 2004, plusieurs travaux ont été conduits pour étendre ces techniques à des systèmes modélisés par des automates temporisés [101, 96, 14]. Pour ce qui est des systèmes modélisés par des automates à pile,

Constant *et al.* en ont considéré une restriction où la clôture d'un tel automate est déterministe [48].

Graphes réguliers probabilistes. Lorsqu'on cherche à vérifier et prouver la qualité d'un système réel, les modèles discrets peuvent se heurter à certaines limites. De la même façon, lorsqu'on cherche à affiner un verdict les réponses de nature binaire peuvent se révéler frustrantes. Pour améliorer les résultats fournis par les méthodes discrètes l'approche probabiliste (et donc quantitative) peut se révéler intéressante.

Les algorithmes opérant sur les chaînes de Markov ou sur les processus de décision markovien produisent des résultats de nature quantitative (le plus souvent en donnant une probabilité de survenue d'un événement). Dans ce contexte, Hansson et Jonsson, en 1994 ont défini la logique PCTL [74], cette logique est une extension de CTL où certaines formules possèdent une garde (formée d'un opérateur de comparaison et d'une valeur rationnelle) exprimant qu'elles sont vraies avec une probabilité conforme à la garde. Cette forme de quantification se substitue à celle qui quantifie sur les chemins dans CTL. La variante qualitative de cette logique n'autorise que les rationnels 0 et 1 dans les gardes.

Le *model-checking* de cette logique a été étudié pour les chaînes de Markov définies par des automates à pile probabilistes [24, 59, 99, 23]. Les résultats les plus marquants sont certainement l'indécidabilité du *model-checking* pour PCTL sur les automates à pile, ainsi que la décidabilité de sa restriction au cas qualitatif.

III Plan du document et éléments de bibliographie personnelle

Le chapitre 1 présente les notations et les notions fondamentales de ce document. Plus précisément, il introduit la terminologie employée pour les graphes, des éléments de théorie des langages (l'outil de mesure d'expressivité) et enfin des éléments de logique, dont le langage de la logique du premier ordre, et la logique modale. Ce chapitre se conclus par quelques mots sur les logiques temporelles (LTL et CTL).

Le chapitre 2 contient une présentation ainsi que les résultats les plus fondamentaux des familles de graphes infinis auxquelles nous nous intéressons. Nous considérons d'abord les graphes des transitions des automates à pile, puis les graphes dits réguliers qui sont engendrés par les grammaires déterministes de graphe. Ensuite nous aborderons les graphes des transitions des automates à pile d'ordre supérieur. Nous présentons ensuite les graphes rationnels, les arbres rationnels, et les graphes d'accessibilité des réseaux de Petri. Le chapitre 2 fait la synthèse d'éléments contenus dans plusieurs articles écrits entre 2005 et 2012, seul ou en collaboration avec Rispal, Darondeau, Demri et Meyer [117, 115, 52].

Le chapitre 3 énonce, et démontre un ensemble de résultats autour de la décidabilité ou de l'indécidabilité de diverses logiques sur les familles introduites au chapitre précédent. En premier lieu, on aborde la décidabilité la logique du premier ordre pour les arbres rationnels, nous montrons également la maximalité de ce résultat tant sur le plan de l'expressivité de la famille que sur celle de la logique. Nous considérons ensuite les graphes d'accessibilité des réseaux de Petri, nous établissons l'indécidabilité de le logique du premier ordre (sans qu'elle ne permette de spécifier le contenu des places) nous examinons également un ensemble de variantes en modulant la classe de réseaux ou des restrictions sur le langage logique. Ce chapitre repose sur des articles de 2006 et 2012 co-écrit avec Carayol, Darondeau, Demri et Meyer [30, 52].

Le chapitre 4 assemble une série de résultats qui peuvent être vus comme des applications par opposition aux résultats plus théorique des chapitres précédents. Ainsi on aborde dans un premier temps des problèmes liés à l'observation partielle d'un objet modélisé par un graphe régulier : diagnosticabilité et opacité, puis la génération de tests de conformité. La dernière partie de ce document est consacrée aux modèles quantitatifs. On y utilise un graphe régulier pour définir une chaîne de Markov et on montre la décidabilité du problème *model-checking* qualitatif de PCTL (extension probabiliste de CTL). On donne, en outre, un exemple qui prouve l'indécidabilité du problème quantitatif. Ces travaux sont pour une part les résultats obtenus durant la thèse de Chédor (en collaboration avec Jéron, co-directeur de la thèse, Marchand et Pinchinat) [46, 44]. La dernière partie est le fruit d'une collaboration avec Bertrand [15].

Chapitre 1

Graphes, logiques et expressivité

Dans ce chapitre, nous allons présenter le contexte formel de ce travail. On l'a déjà vu, le but de ce mémoire est d'examiner quelques familles de graphes ayant un nombre infini de sommets, qui possèdent une présentation finie, et de donner quelques unes de leurs propriétés. Dans un premier temps, nous présenterons les graphes, ainsi que la terminologie mathématique élémentaire permettant de traiter ces objets. Dans un second temps nous examinerons, avec quelques détails, les bases de la théorie des langages. Dans un troisième temps, nous passerons en revue plusieurs logiques. Ces dernières sont utilisées pour formuler des propriétés qui ensuite peuvent être vérifiées par un graphe particulier. Un schéma général pour les problèmes qui nous intéressent est la décidabilité de théories logiques, ce qui se formule ainsi : « *Étant donnée une formule φ d'une logique X, et un graphe G d'une famille de graphe Y, la formule φ est-elle vérifiée par G ?* » Toute paire formée d'une logique X, et d'une famille de graphe Y définit un problème, dont une instance est une formule φ, et un graphe G. La décidabilité de ces problèmes constitue une part importante de ce travail.

I Graphes

Un graphe étiqueté est un ensemble de sommets reliés par des arcs portant des étiquettes. Cette section va formaliser cette notion afin de pouvoir définir et raisonner précisément sur ces objets.

I.1 Préliminaires mathématiques

Monoïdes et langages. L'ensemble des entiers naturels est noté $\mathbb{N}$. Si n est un entier, on note $[n] \stackrel{\text{def}}{=} \{1, 2, \cdots, n\}$ l'ensemble des entiers strictement positifs inférieurs ou égaux à n. L'ensemble vide est noté $\emptyset$. Si E est un ensemble, on note $|E|$ son *cardinal*, il s'agit d'un entier lorsque l'ensemble est fini, et c'est le symbole $+\infty$ dans le cas contraire. On utilise les symboles $\in$ et $\subseteq$ pour désigner respectivement l'appartenance et l'inclusion. La notation 2^E désigne l'ensemble des parties d'un ensemble E. Si E et F sont deux ensembles, on note $E - F$ la *différence* entre ces deux ensembles : $E - F \stackrel{\text{def}}{=} \{x \in E \mid x \notin F\}$. Si F est une partie d'un ensemble E, on note $\overline{F}$ son *complémentaire* dans E, c'est à dire $E - F$. Si E et F sont des ensembles arbitraires, on note $E \times F$ le *produit cartésien* de E et F : l'ensemble des couples (x, y) pour tout x élément de E et tout y élément de F. On peut étendre ce produit à une collection finie d'ensembles, les éléments de l'ensemble construit sont alors des n-uplets.

Soient E et E' deux ensembles. On appelle *relation* toute partie de $E \times E'$. Si R est une relation, on note R^{-1} la relation $\{(v, u) \mid (u, v) \in R\}$ dans $E' \times E$, dite *relation inverse*. Si u est un élément de E, l'ensemble $R(u) \stackrel{\text{def}}{=} \{v \mid (u, v) \in R\}$ est l'image de u par R. On note souvent $u\, R\, v$ pour $(u, v) \in R$. Le *domaine* (*resp. l'image*) d'une relation est l'ensemble $Dom(R)$ (*resp.* $Im(R)$), défini par : $Dom(R) \stackrel{\text{def}}{=} \{u \mid \exists v \in E', u\, R\, v\}$ (*resp.* $Im(R) \stackrel{\text{def}}{=} \{v \mid \exists u \in E, u\, R\, v\}$). Un *relation d'équivalence*, $R \subseteq E \times E$, est une relation *réflexive* (pour tout élément $a \in E$, on a $a\, R\, a$), *symétrique* (pour toute paire d'éléments a, b

de E, on a : $a\,R\,b$ entraîne $b\,R\,a$) et enfin *transitive* (pour tout triplet d'éléments a, b, c de E, on a $a\,R\,b$ et $b\,R\,c$ entraîne $a\,R\,c$). Lorsque R est une relation d'équivalence sur E, la classe $[a]_R$ d'un élément a de E est l'ensemble des éléments qui sont en relation avec a : $[a]_R \stackrel{\text{def}}{=} \{b \mid a\,R\,b\}$. On peut alors définir le *quotient* d'un ensemble par une relation : $E/R \stackrel{\text{def}}{=} \{[a]_R \mid a \in E\}$.

Une *fonction* est une relation f de $E \times F$, notée $f : E \to F$, telle que pour tout élément e de E, on ait $|f(e)| \leqslant 1$. Une *application* est une fonction f telle que $Dom(f) = E$. L'ensemble des applications de E dans F est noté F^E. Une fonction est dite *surjective* si $Im(f) = F$, *injective* si pour tout u et v dans $Dom(f)$, $f(u) = f(v) \Rightarrow u = v$, et enfin *bijective* si elle est à la fois injective et surjective. Les ensembles que nous utiliserons seront, sauf précision contraire, finis ou *dénombrables* autrement dit, en bijection [1] avec $\mathbb{N}$.

Soit M un ensemble, et $\cdot$ une opération binaire (une application entre $M \times M$ et M) sur cet ensemble, on note $(M, \cdot)$, l'ensemble M muni de $\cdot$. On dit que $(M, \cdot)$ est un *monoïde* si l'opération $\cdot$ est associative et possède un élément neutre (noté 1_M) : pour tout triplet d'éléments u, v, w dans M, on a $(u \cdot v) \cdot w = u \cdot (v \cdot w)$ et, pour tout élément u dans M, $u \cdot 1_M = 1_M \cdot u = u$. Une partie $A \subseteq M$ d'un monoïde M est appelée *sous-monoïde* lorsque, $1_M \in A$ et, pour tous u et v éléments de A, $u \cdot v \in A$.

A présent, on définit quelques opérations sur les monoïdes. L'opération d'un monoïde peut s'étendre à l'ensemble de ses parties : on note $A \cdot B$ le produit de deux parties A et B défini par $A \cdot B \stackrel{\text{def}}{=} \{u \cdot v \mid u \in A \wedge v \in B\}$; la *puissance* d'une partie A est définie par récurrence, on pose $A^0 \stackrel{\text{def}}{=} \{1_M\}$, ensuite, pour n strictement positif, $A^n \stackrel{\text{def}}{=} A \cdot A^{n-1}$. Le *plus du produit* est défini à partir de la puissance : $A^+ \stackrel{\text{def}}{=} \bigcup_{i=1}^{\infty} A^i$. On définit également *l'étoile du produit* : $A^* \stackrel{\text{def}}{=} \bigcup_{i=0}^{\infty} A^i$. On peut remarquer que A^* est le plus petit sous-monoïde de M contenant A, il est appelé monoïde *engendré* par A. Un monoïde M est dit *libre* s'il existe une partie $P \subseteq M$ ne contenant pas ε, telle que $P^* = M$ et, pour tout élément $u \in M$ il existe une unique suite finie $(u_i)_{i \in [n]}$ d'éléments de P telle que $u = u_1 \cdot u_2 \cdots u_n$ (l'élément neutre du monoïde est le produit de zéro élément du P).

Étant donnés $(M_1, \cdot_1)$ et $(M_2, \cdot_2)$ deux monoïdes, le produit cartésien $M_1 \times M_2$ est un monoïde pour l'opération $\cdot$ définie par : $(u_1, u_2) \cdot (u_1', u_2') \stackrel{\text{def}}{=} (u_1 \cdot_1 u_1', u_2 \cdot_2 u_2')$. L'élément neutre est $(1_{M_1}, 1_{M_2})$

Un *morphisme (de monoïde)* est une application entre deux monoïdes $(E_1, \cdot_1)$ et $(E_2, \cdot_2)$, telle que, pour tous éléments u et v de E_1, $f(u \cdot_1 v) = f(u) \cdot_2 f(v)$. Un isomorphisme est un morphisme bijectif. Un endomorphisme est un morphisme d'un ensemble dans lui-même.

Pour conclure, un *alphabet* Σ est un ensemble fini non-vide de symboles appelés *lettres*. Un *mot* u sur Σ, de *longueur* $|u| \in \mathbb{N}$ est une application de $[|u|]$ dans Σ, ou de façon équivalente est un $|u|$-uplet $(u(1), \ldots, u(|u|))$ de lettres de Σ noté $u(1) \ldots u(|u|)$. On note ε le mot de longueur nulle, appelé *mot vide*. On note $\tilde{u}$ le *miroir* de u, définit par récurrence : $\tilde{\varepsilon} = \varepsilon$, et $\widetilde{au} = \tilde{u}a$ (où a est une lettre, et u un mot). L'ensemble Σ^* des mots sur Σ est un monoïde pour le produit de *concaténation* défini par $u \cdot v = u(1) \ldots u(|u|)v(1) \ldots v(|v|)$ pour tous mots u et v sur Σ. Ainsi Σ^* est le monoïde libre engendré par Σ. Tout sous-ensemble d'un monoïde libre engendré par un alphabet est appelée un *langage*.

I.2 Graphes

Soit V un ensemble (dénombrable ou fini) et soit Σ un ensemble (le plus souvent fini). Un Σ-*graphe* (simplement *graphe* lorsque Σ est clair) est une partie G de $V \times \Sigma \times V$. Un élément (u, a, v) d'un graphe G est un *arc*, de *source* u, de *but* v et d'*étiquette* a. Chaque arc (u, a, v) est noté $u \xrightarrow[G]{a} v$, ou plus simplement $u \xrightarrow{a} v$ s'il n'y a pas d'ambiguïté sur G. Comme pour les relations, on note respectivement $Dom(G)$ et $Im(G)$ l'ensemble des sources et des buts, et $V_G \stackrel{\text{def}}{=} Dom(G) \cup Im(G)$ l'ensemble des *sommets* de G. Soit $\Sigma' \subseteq \Sigma$, on note $G_{|\Sigma'}$ le graphe restreint aux transitions étiquetées par des éléments de Σ' : $G \cap (V \times \Sigma' \times V)$.

1. Ensembles tels qu'il existe une application bijective avec $\mathbb{N}$.

Remarque 1. Il est important de souligner que cette définition n'est pas tout à fait classique dans le sens où l'ensemble effectif des sommets (V_G) est dérivé de celui des transitions qui, pour sa part, est défini explicitement. Ainsi il ne peut pas exister de sommets isolés. Il est naturellement possible d'autoriser ce type de sommets en utilisant un nouveau symbole, par exemple $\#$, dans Σ et en ajoutant une boucle étiquetée par $\#$ sur chaque sommet du graphe. Ainsi, un tel graphe peut contenir des sommets qui ne sont connectés à aucun autre sommet, et dont toute boucle est étiquetée par le symbole $\#$.

Pour tout graphe G, on définit les relations $\xrightarrow[G]{d} \stackrel{\text{def}}{=} \{(u,v) \mid u \xrightarrow[G]{d} v\}$ pour tout d dans Σ, et aussi $\xrightarrow[G]{} \stackrel{\text{def}}{=} \bigcup_{d\in\Sigma} \xrightarrow[G]{d}$ la relation de *transition* de G qui ignore les étiquettes. On note de façon habituelle $\xrightarrow[G]{d}(u)$ (resp. $\xrightarrow[G]{}(u)$) l'image du mot u par la relation $\xrightarrow[G]{d}$ (resp. $\xrightarrow[G]{}$) ; cette notation s'étend de façon naturelle aux parties de V. Le *degré sortant* (*resp. entrant*) d'un sommet u est le nombre $|\xrightarrow[G]{}(u)|$ (*resp.* $|(\xrightarrow[G]{})^{-1}(u)|$) d'arcs issus de (*resp.* aboutissant à) ce sommet. Le *degré* $deg(u) \stackrel{\text{def}}{=} |\xrightarrow[G]{}(u)| + |(\xrightarrow[G]{})^{-1}(u)|$ d'un sommet u est la somme de ses degrés entrant et sortant. On dit qu'un graphe est de degré fini si tous ses sommets sont de degré fini. On dit qu'il est de degré borné si il existe un entier majorant le degré de chacun de ses sommets, formellement : $\exists M, \forall u,\ deg(u) \leqslant M$. Enfin, on note $d(G) = |\{deg(u) \mid u \in V_G\}|$ le *nombre de degrés* d'un graphe G.

Un graphe est *déterministe* si deux arcs distincts de même source ont des étiquettes distinctes : $r \xrightarrow{a} s \wedge r \xrightarrow{a} t \Rightarrow s = t$. Un graphe G est dit *complet* si, pour chaque étiquette a, tout sommet est source d'un arc étiqueté a : $\forall a \in \Sigma\ \forall u \in V_G\ \exists v\ u \xrightarrow[G]{a} v$. On note, G^+ est le Σ^+-graphe obtenu par fermeture transitive du graphe G. Cette fermeture est définie de façon récursive : $G_1 \stackrel{\text{def}}{=} G$, et pour tout $n \geqslant 1$, $G_{n+1} \stackrel{\text{def}}{=} \left\{ r \xrightarrow{va} t \mid r \xrightarrow[G_n]{v} s \wedge s \xrightarrow[G]{a} t \right\}$. La relation $\xrightarrow[G^+]{u}$ est notée $\xRightarrow[G]{u}$ ou plus simplement $\xRightarrow{u}$ lorsqu'il n'y a pas ambiguïté sur G. Cette relation est formée des couples d'extrémités des *chemins* étiqueté par u (élément de Σ^+) dans G. La relation $\xRightarrow[G]{\varepsilon}$ est définie par l'identité sur l'ensemble des sommets du graphe G : $\xRightarrow[G]{\varepsilon} \stackrel{\text{def}}{=} \{(u,u) \mid u \in V_G\}$. Soit $p \in V_G$, on dit qu'un sommet $q \in V_G$ est accessible depuis p lorsque il existe $u \in \Sigma^+$, $p \xRightarrow[G]{u} q$. On définit les notations suivantes : la relation d'accessibilité : $\xRightarrow[G]{} \stackrel{\text{def}}{=} \cup_{u\in\Sigma^+} \xRightarrow[G]{u}$; l'accessibilité d'un ensemble de sommets $P \in V_G$, $acc_G(P) \stackrel{\text{def}}{=} \xRightarrow[G]{}(P)$ et l'accessibilité restreinte à un sous-alphabet $acc_G(P, \Sigma') \stackrel{\text{def}}{=} \xRightarrow[G_{|\Sigma'}]{}(P)$.

Pour tout langage L de Σ^* et pour tout Σ-graphe G, on note $s \xRightarrow[G]{L} t$ l'existence de u dans L tel que $s \xRightarrow[G]{u} t$.

La *trace* (ou ensemble des étiquettes des chemins), $L(G, E, F)$, de G allant de l'ensemble de sommets E à l'ensemble de sommets F est la partie de Σ^* suivante :

$$L(G, E, F) \stackrel{\text{def}}{=} \left\{ u \in \Sigma^* \mid \exists\, s \in E,\ \exists\, t \in F,\ s \xRightarrow[G]{u} t \right\}$$

On appelle Σ-*automate* un Σ-graphe A dont les sommets sont appelés *états*, ayant un état initial i, et une partie, F de V_G d'états finals ; cet automate *reconnaît* l'ensemble $L(A) \stackrel{\text{def}}{=} L(G, \{i\}, F)$. Un automate est fini (*resp.* déterministe, complet) si son graphe associé est fini (*resp.* déterministe, complet). Cette notion d'automate est essentielle pour l'étude des langages. Enfin, un graphe G est dit *globalement déterministe* si le graphe G^+ est déterministe, autrement dit, si $\xRightarrow[G]{u}$ est une fonction pour tout u dans Σ^+ : $r \xRightarrow{u} s \wedge r \xRightarrow{u} t \Rightarrow s = t$. Lorsqu'un graphe G est globalement déterministe et complet, on note su l'unique état vérifiant $s \xRightarrow{u} su$; en particulier $s(u \cdot v) = (su)v$. Pour ce qui concerne la théorie des automates dans un contexte très général, on peut consulter l'ouvrage de Sakarovitch [132].

L'*isomorphisme* de graphe correspond à l'idée d'abstraire le nom des sommets d'un graphe. Précisément, deux graphes G et G' sont dits *isomorphes*, s'il existe une bijection h de V_G sur $V_{G'}$ telle que : $\forall a \in \Sigma,\ \forall u, v \in V_G, u \xrightarrow[G]{a} v \Leftrightarrow h(u) \xrightarrow[G']{a} h(v)$.

La *bisimulation* correspond à l'isomorphisme « à redondance de structure près ». Elle a été introduite initialement par Park [120] et reprise par Milner [111]. Tout comme l'isomorphisme, il s'agit d'une relation définie entre les sommets de deux graphes. Une relation $R \subseteq V_G \times V_{G'}$ est une *bisimulation* (forte), si, pour tout $s\, R\, s'$, on a :

1. dès que $s \xrightarrow[G]{a} t$, alors il existe t' dans G' tel que $s' \xrightarrow[G']{a} t'$, et $t\, R\, t'$
2. dès que $s' \xrightarrow[G']{a} t'$, alors il existe t dans G tel que $s \xrightarrow[G]{a} t$, et $t\, R\, t'$

II Familles de Langages

Les familles de langages constituent une mesure usuelle de l'expressivité d'un modèle. L'échelle utilisée le plus souvent est constituée par la hiérarchie de Chomsky [41, 42].

Nous allons donc dans un premier temps nous attacher à rappeler quelques points clés de cette hiérarchie. Dans un second, nous présenterons quelques familles qui sont définies à l'intérieur de la hiérarchie.

II.1 Hiérarchie de Chomsky

La *hiérarchie de Chomsky* distingue quatre grandes familles de langages. Ces familles sont les langages *rationnels*, *algébriques*, *contextuels* et *récursivement énumérables*, notées respectivement, $Rat(\Sigma^*)$, $Alg(\Sigma^*)$, $CS(\Sigma^*)$, $RE(\Sigma^*)$. Chacune de ces familles correspond à un système de récriture particulier. L'objectif de ce paragraphe est de faire un survol rapide de ces différentes familles et d'en donner des caractérisations. De nombreux ouvrages de synthèse couvrent ce domaine de façon complète, le lecteur intéressé pourra s'y référer [6, 81, 107].

Grammaires de Chomsky

L'approche de Chomsky pour définir des familles de langages est d'utiliser des systèmes de générateurs décrits de façon finie, qui produisent par transformations successives, à partir d'un axiome, chacun des mot d'un langage. Ensuite, en limitant les systèmes de générateurs on limite la richesse des familles considérées.

Ces systèmes de générateurs sont des grammaires. Pour engendrer des mots de Σ^* ces dernières ont recours à un second alphabet, dit de non-terminaux (par exemple N). Une grammaire est un quadruplet (Σ, N, A, R) où Σ est un alphabet, N est un alphabet de terminaux ($\Sigma \cap N = \emptyset$), $A \in N$ est l'*axiome* et R est un ensemble fini de *règles (de récriture)* qui sont des couples de mots de $(N \cup \Sigma)^*$. Une règle de la forme (U, V) avec $U, V \in (N \cup \Sigma)^*$, sera souvent notée $U \to V$. On peut observer que le membre gauche d'une telle règle peut ne pas contenir de non-terminaux, ces derniers sont cependant utilisés par certains types de grammaires. Ensuite, si R est un ensemble de règles de récriture définie sur un alphabet Σ, et un ensemble de non-terminaux N, la *récriture suivant* R est la relation suivante :

$$\xrightarrow[R]{} := \left\{(u, v) \in (N \cup \Sigma)^* \times (N \cup \Sigma)^* \mid u = w_0 u' w_1 \wedge v = w_0 v' w_1 \wedge (u', v') \in R\right\}.$$

La *dérivation suivant* R, notée $\xrightarrow[R]{}^*$ est la fermeture réflexive et transitive de la relation de récriture, c'est à dire $\xrightarrow[R]{}^* := \bigcup_{n\in\mathbb{N}} \xrightarrow[R]{}^n$, où $\xrightarrow[R]{}^0 := \{(u, u) \in \Sigma^* \times \Sigma^*\}$, et $\xrightarrow[R]{}^{n+1} := \xrightarrow[R]{} \circ \xrightarrow[R]{}^n$. Étant donné une grammaire $G = (\Sigma, N, A, R)$, un langage est engendré par G s'il est composé de tous les mots de Σ^* qui sont dans l'image de A, par la dérivation suivant R. Formellement, on a $L(G) = \xrightarrow[R]{}^*(A) \cap \Sigma^*$.

La hiérarchie de Chomsky est définie de la façon suivante : pour chacun des types (0, 1, 2 et 3), on limite la structure des règles. Le liste ci-dessous résume ces limitations.

Type 0 **:** Les règles sont quelconques.

Type 1 **:** Les règles sont de la forme $UAV \rightarrow UWV$, avec $A \in N$, $W \in (N \cup \Sigma)^+$ [2] et $U, V \in (N \cup \Sigma)^*$.

Type 2 **:** Les règles sont de la forme $A \rightarrow V$, avec $A \in N$ et $V \in (N \cup \Sigma)^*$.

Type 3 **:** Les règles sont de la forme $A \rightarrow uB$, avec $A, B \in N$ et $u \in \Sigma^*$.

A présent nous allons examiner plus en détail chacun de ces niveaux en commençant par les grammaires de type 3.

II.2 Langages rationnels

Les langages engendrés par les grammaires de type 3 possèdent de très nombreuses caractérisations. Pour les besoin de ce document nous en examinerons quelques unes.

On peut définir les langages de type 3 sur Σ^* comme *la plus petite famille* de langages contenant l'ensemble vide, les singletons, et qui est close par union, concaténation, et étoile de la concaténation. Cette propriété de clôture étant appelée clôture rationnelle, ces langage sont dit rationnels. On parle de la famille des langages rationnels.

Remarque 2. Il est usuel, en particulier chez les auteurs anglo-saxons de privilégier le terme *régulier* plutôt que rationnel pour les langages. Hors, dans le cas des transducteurs, ou des séries, on privilégie en général le terme rationnel. C'est la raison pour laquelle, ici nous préférerons cette terminologie.

Kleene [92], puis Rabin et Scott [127] ont démontré le résultat suivant.

Théorème 1 (Kleene)**.** *Les langages rationnels sur Σ^* sont exactement les langages reconnus par les Σ-automates finis.*

Une des conséquences du Théorème 1 est qu'il établit une connexion forte entre les automates, donc les graphes, et les langages. Il délimite de façon exacte l'expressivité des graphes finis : ils reconnaissent les langages rationnels. Ce qui a également pour conséquence que les graphes qui reconnaissent des langages plus riches sont nécessairement infinis.

II.3 Langages algébriques

Les grammaires de type 2 engendrent les langages algébriques. Elles sont donc appelées grammaires algébriques. L'ouvrage de Berstel s'intéresse particulièrement à cette famille de langages (sans toutefois dresser une liste exhaustive des multiples formalismes permettant de les caractériser) [13].

Une autre caractérisation, très opérationnelle, des langages algébrique est donnée par les automates à pile (qui ne sont pas des Σ-automates). Un automate à pile est un quintuplet $(\Sigma, \Gamma, Q, q_0, \Delta)$ où Σ est l'alphabet d'entrée, Γ l'alphabet de pile, Q un ensemble fini d'états, $q_0 \in Q$ l'état initial et Δ un ensemble de règles de récriture de mots de la forme $pA \xrightarrow{a} qV$, où $p, q \in Q$, $A \in \Gamma$, $V \in \Gamma^*$ et a une lettre de Σ.

Il existe de nombreuses variantes de ces automates, ainsi qu'une vision « graphe », que nous présenterons dans la deuxième partie du deuxième chapitre.

2. Il est autorisé d'avoir la règle $A \rightarrow \varepsilon$ lorsque A n'apparaît pas en membre droit d'un règle.

II.4 Langages contextuels

La famille des langages contextuels a suscité une activité intense entre le début des années 60 et le milieu des années 70 [42, 97, 66, 122]. Cet engouement a permis la résolution de problèmes importants pour cette famille. En particulier des résultats de normalisation de grammaires engendrant les langages contextuels ou encore des propriétés de fermeture. Plus tard, à la fin des années 80, il a été démontré que cette famille est close pour l'opération de complémentation, de façon indépendante par Immerman [85] et Szelepcényi [139]. Depuis le début des années 2000, quelques familles de langages contextuels ont été l'objet d'une attention particulière : les langages reconnus par les automates à pile d'ordre supérieur. Nous y reviendrons ultérieurement.

Nous allons présenter quelques aspects marquant des langages contextuels. Une grammaire de type 1 est dite *contextuelle*. Selon la définition syntaxique, les langages engendrés par de telles grammaires ne contiennent pas le mot vide. En fait, on autorise la règle $A \to \varepsilon$ lorsque A est l'axiome de la grammaire, sous réserve que ce non-terminal n'apparaisse pas dans le membre droit d'une règle de la grammaire. Une telle grammaire sera également dite contextuelle.

Définition 1. Un langage de Σ^* est *contextuel* si il existe une grammaire contextuelle qui l'engendre.

Une grammaire $G = (\Sigma, N, A, R)$ est dite *croissante*, si pour toute règle $u \to v \in R$ on a $|u| \leqslant |v|$. Lorsque A n'apparaît dans le membre droit d'aucune règle de la grammaire, la règle $A \to \varepsilon$ est autorisée.

Théorème 2 (Chomsky 59)**.** *Tout langage engendré par une grammaire croissante est contextuel.*

Voici un exemple (emprunté à Mateescu *et al.* [107]) de langage contextuel.

Exemple 1. On va définir une grammaire croissante $G = (\Sigma, N, A, R)$ telle que :

$$L(G) = \left\{ ww \mid w \in \Sigma^+ \right\}.$$

L'alphabet de non-terminaux N est ainsi défini :

$$N = \{A\} \cup \{X_a \mid a \in \Sigma\} \cup \{Y_a \mid a \in \Sigma\} \cup \{Z_a \mid a \in \Sigma\}.$$

Voici les règles de G (déclinées pour couple (a, b) dans Σ) :

$$\begin{cases} A \to aAX_a & (1_a) \\ A \to Y_a Z_a & (2_a) \\ Z_a X_b \to X_b Z_a & (3_{ab}) \\ Z_a \to a & (4_a) \\ Y_a X_b \to Y_a Z_b & (5_{ab}) \\ Y_a \to a & (6_a) \end{cases}$$

Cette grammaire commence par produire, en utilisant les règles de type (1_a) et (2_a), un mot de la forme $a_1 a_2 \dots a_m Y_c Z_c X_{a_m} X_{a_{m-1}} \dots X_{a_1}$. À partir de là, le non-terminal Z_c va migrer à la fin du mot, en utilisant les règles de type (3_{cb})) puis être récrit en c grâce a la règle (4_c). Si jamais cette récriture intervient alors que le non-terminal n'a pas atteint l'extrémité du mot, les non-terminaux X_{a_i} situé à sa droite ne seront jamais récrits, et donc on n'obtiendra pas un mot de Σ^*. On obtient donc $a_1 a_2 \dots a_m Y_c X_{a_m} X_{a_{m-1}} \dots X_{a_1} c$. On peut alors appliquer (5_{ca_m}) ce qui produit $a_1 a_2 \dots a_m Y_c Z_{a_m} X_{a_{m-1}} \dots X_{a_1} c$. A nouveau, on peut faire migrer Z_{a_m} jusqu'à la fin du mot pour obtenir $a_1 a_2 \dots a_m Y_c X_{a_{m-1}} \dots X_{a_1} a_m c$. En répétant ce processus, on finit par atteindre le mot $a_1 a_2 \dots a_m Y_c a_1 a_2 \dots a_m c$. Il ne reste plus qu'à appliquer alors la règle (6_c) pour avoir $a_1 a_2 \dots a_m c a_1 a_2 \dots a_m c$, qui est bien un mot de la forme ww.

Voici quelques exemples classiques de langages contextuels.

Exemple 2. Les langages suivants sont contextuels :
— $\{a^n b^n c^n \mid n \in \mathbb{N}\}$
— $\{a^{2^n} \mid n \in \mathbb{N}\}$
— $\{a^p \mid p$ est un nombre premier$\}$

Le modèle le plus classique pour reconnaître les langages contextuels est certainement celui des machines de Turing linéairement bornées (LBM). Plus précisément, une LBM est une machine de Turing non déterministe M (voir paragraphe suivant) pour laquelle il existe une fonction linéaire f de $\mathbb{N}$ dans $\mathbb{N}$ telle que lorsque M a terminé son exécution à partir d'un mot de longueur n, elle a utilisé au plus $f(n)$ cases de la bande.

Théorème 3 (Kuroda 64)**.** *Tout langage contextuel est reconnu par une machine de Turing linéairement bornée.*

Quelques Résultats. La famille des langages contextuels possède aussi quelques propriétés importantes que nous allons énoncer à présent.

Il est remarquable d'observer qu'en utilisant un morphisme effaçant, il est possible d'obtenir tous les langages récursivement énumérables à partir des langages contextuels [60, 67].

Théorème 4 (Evey 63)**.** *Soit L un langage dans $RE(\Sigma^*)$ et soit c un symbole n'appartenant pas à Σ. Il existe un langage contextuel L' de $\Sigma^* c^*$ tels que : $L = \pi_c(L')$ où π_c est le morphisme qui efface c.*

Ce résultat devient faux s'il existe une borne linéaire (vis-à-vis de la longueur du mot reconnu).

Théorème 5 (Ginsburg et Greibach 66)**.** *Soit Σ un alphabet, soit c une lettre de Σ, et soit $L \in CS(\Sigma^*)$. S'il existe un entier k tel que, pour tout mot $w \in L$, $|w| \leqslant k|\pi_c(w)|$ alors $\pi_c(L) \in CS((\Sigma - \{c\})^*)$.*

En 1988, indépendamment, Immerman et Szelepcsényi ont apporté une réponse positive [85, 139] à la question de clôture des langage contextuels vis-à-vis de l'opération de complémentation (posée en 1963 [100]).

Théorème 6 (Immerman et Szelepcsényi 88)**.** *Le complémentaire d'un langage contextuel est un langage contextuel.*

Pour conclure cet aperçu, voici quelques propriétés de décidabilité.

Théorème 7.
Problèmes décidables *: appartenance d'un mot à un langage contextuel.*
Problèmes indécidables *: la vacuité et la finitude d'un langage contextuel, l'égalité et l'inclusion de deux langages contextuels.*

II.5 Langages récursivement énumérables

Les langages de types 0, également appelés *récursivement énumérables*, sont ceux pour lesquels le système de récriture est le plus libre. La caractérisation la plus usuelle est constituée par les machines de Turing. Il existe de nombreuses définitions équivalentes de ces machines. L'une d'elles est donnée par une bande de travail bi-infinie, contenant des cases initialement blanches, chacune des cases pouvant contenir un symbole. Une tête de lecture se déplace sur cette bande vers la droite ou la gauche, en modifiant éventuellement ce qui y est inscrit. La machine est définie par un ensemble de règles qui indiquent, en fonction de l'état, du symbole placé sous la tête de lecture, dans quelle direction la tête doit se déplacer, et ce qu'elle doit inscrire là où elle se trouvait. Formellement, une machine de Turing M est un quintuplet $(Q, \Sigma, q_0, F, \mathcal{R})$, où Q est l'alphabet des états, Σ est l'alphabet de bande, q_0 est l'état initial, $F \subseteq Q$ est l'ensemble (éventuellement vide) des états accepteurs et $\mathcal{R}$ un sous ensemble de $Q \times \widehat{\Sigma} \times Q \times \widehat{\Sigma} \times m$, où

le caractère blanc est noté $\square$ (lorsqu'il n'y a rien dans une case de la machine), $\widehat{\Sigma}$ est l'ensemble $\Sigma \cup \{\square\}$, et m est l'ensemble $m := \{\triangleright, \triangleleft\}$; $\mathcal{R}$ est l'ensemble des transitions de la machine. Si $(p, A, q, B, \diamond)$ est une transition, p représente l'état dans lequel la machine se trouve, A la lettre de bande sur laquelle porte la tête de lecture, q représente l'état dans lequel abouti la machine après avoir effectué la transition, B la lettre qui doit remplacer A, enfin, $\diamond$ indique la direction dans laquelle la tête de lecture doit se déplacer, d'une case vers la droite si c'est $\triangleright$, d'une case vers la gauche sinon. Une machine de Turing est dite *déterministe* si pour tout état $p \in Q$, toute lettre $\widehat{X} \in \widehat{\Sigma}$, il existe *au plus* une transition du type $(p, \widehat{X}, q, \widehat{X}', \diamond)$; dans le cas contraire, on dit que la machine est *non-déterministe*.

Dans la suite, par souci de simplicité, on note une transition de la forme $(p, A, q, B, \triangleright)$ par la règle de récriture de mots $pA \to Bq$ (le mot pA est remplacé par le mot Bq). Dans le cas d'une transition du type $(p, A, q, B, \triangleleft)$), on notera, pour chaque lettre C dans $\widehat{\Sigma}$, $CpA \to qCB$. On appelle *configuration* de la machine, la donnée du contenu de la bande, de la position de la tête de lecture et de l'état dans lequel se trouve la machine. Ainsi si il n'y a qu'un mot ABC sur la bande (entouré de cases vides), que la tête de lecture pointe sur la case contenant B, et que la machine est dans l'état p, on présente la configuration ainsi : $ApBC$. Notez bien que cette représentation ignore les cases vides qui sont à gauche de A et à droite de B. On dit finalement qu'un mot u est *reconnu* par une machine $M = (Q, \Sigma, q_0, F, \mathcal{R})$ si on a $q_0 u \underset{\mathcal{R}}{\to}^* v f w$, avec $v, w \in \widehat{\Sigma}^*$ et $f \in F$. Le *langage reconnu* par une machine de Turing M, noté $L(M)$, est formé par les mots qui sont reconnus par cette machine. Lorsque la machine abouti à une configuration dont l'état est un élément de F on dit que le calcul est un *succès*, lorsque la machine est dans une configuration où plus aucune transition n'est possible et que l'état n'est pas dans F, on dit que le calcul est un *échec*. Il est également possible que la machine puisse poursuivre son calcul perpétuellement, sans passer par une configuration succès, dans ce cas on dit que la machine *boucle* ou plus simplement ne s'arrête pas.

Les machines de Turing ont une très grande expressivité. Jusqu'à présent, aucun modèle de calcul n'en a montré plus. En fait, la thèse de Church (voir par exemple Salomaa [133, pp 77–78]) conjecture que les machines de Turing forment un modèle universel de calcul. Ces machines, en dehors d'être utilisées pour caractériser les langages récursivement énumérables servent donc de modèle général pour la *calculabilité*, c'est à dire, pour caractériser le fait qu'une famille de problèmes puisse être résolue par un algorithme (voir, entre autres, l'ouvrage introductif de Wolper [148]). Plus formellement, et pour un alphabet Σ, on appelle *problème* (noté (P_+, P_-)) une partition de Σ^* en deux ensembles P_+ et P_-. Dans ce contexte, un élément u de Σ^* est appelé une *instance*, *positive* si elle est dans P_+ *négative* sinon. On dit alors qu'un problème est *décidable* si il existe une machine de Turing qui à partir de toute instance de Σ^* arrive dans une configuration de succès si cette instance est positive, échec dans le cas contraire. Un problème qui n'est pas décidable est dit *indécidable*. Le plus souvent on formulera un problème sous forme d'une donnée, et d'une question ayant deux réponses possibles. Ainsi le problème *de la vacuité d'un langage algébrique*, noté $\mathrm{Vide(Alg)}$, se formule de la façon suivante :

Donnée : Une grammaire algébrique G.

Question : Le langage $L(G)$ est-il vide ?

Ce problème est décidable, en d'autre terme, on peut construire une machine de Turing qui prend en entrée une grammaire algébrique, cette machine accepte la grammaire si le langage qu'elle engendre est vide, et la rejette sinon. Cette machine s'arrête sur toute entrée.

Pour illustrer les problèmes indécidable, on examine précisément le problème de correspondance de Post (noté PCP). Soit Σ un alphabet ayant au moins deux lettres, PCP est définit comme suit :

Donnée : Soient (u_0, v_0), (u_1, v_1),..., (u_n, v_n) des éléments de $\Sigma^* \times \Sigma^*$.

Question : Existe-t-il une suite d'indices $0 \leqslant i_1, i_2, \ldots, i_m \leqslant n$, telle que $u_{i_1} \cdots u_{i_m} = v_{i_1} \cdots v_{i_m}$?

Le PCP est indécidable. En d'autres termes, il n'existe pas de machine de Turing qui s'arrête tout le temps, et qui accepte les instances ayant une solution et refusant les autres.

Pour conclure, les machines de Turing elles-mêmes sont la source de nombreux problèmes indécidables. Par exemple, l'arrêt (une machine de Turing s'arrête-t-elle toujours ?), le vide (le langage reconnu par une machine de Turing est-il vide ?) sont indécidables. Plus généralement, le Théorème de Rice énonce que toute propriété non triviale des langages récursivement énumérables (autrement dit toute propriété qui est satisfaite par au moins un langage récursivement énumérable, et qu'au moins un de ces langages ne satisfait pas) est indécidable.

II.6 Sous-familles contextuelles

Les automates à pile d'ordre supérieur étendent les automates à pile et permettent de caractériser des langages non-algébriques [106]. On présente ici une définition minimale reposant sur les notations de Carayol [28].

Soit Γ un alphabet de pile. Pour tout entier $k \geq 1$, les piles de *niveau* k, or simplement *k-piles*, (sur Γ) sont définies par récurrence : Une 1-pile est de la forme $[U]_1$, où $U \in \Gamma^*$, et la pile vide est notée $[]_1$; les 1-piles coïncident avec les piles des automates à pile. Pour $k > 1$, une k-pile est une suite finie de $(k-1)$-piles ; la k-pile vide est notée $[]_k$. Une *opération de niveau* k agit sur la k-pile la plus haute d'une $(k+1)$-pile ; les opérations sur les piles (de tout niveau) préservent leur niveau. Les opérations de niveau 1 correspondent aux opérations classiques de $push_X$ et pop_X, pour tout $X \in \Gamma$: $push_X([U]_1) = [UX]_1$ et $pop_X([UX]_1) = [U]_1$. Les opérations de niveau $k > 1$ sont copy_k et $\overline{\mathrm{copy}}_k$, ces opérations agissent sur les $(k+1)$-piles de la façon suivante ($S_1, \ldots, S_n$ sont des k-piles).

$$\mathrm{copy}_k([S_1, \ldots, S_n]_{k+1}) := [S_1, \ldots, S_n, S_n]_{k+1}$$
$$\overline{\mathrm{copy}}_k([S_1, \ldots, S_n, S_n]_{k+1}) := [S_1, S_2, \ldots, S_n]_{k+1}$$

Toute opération ρ de niveau k s'étend au piles de niveau quelconque ainsi : $\rho([S_1, \ldots, S_n]_\ell) = [S_1, \ldots, \rho(S_n)]_\ell$, pour $\ell > k+1$.

Un *automate d'ordre supérieur (HPDA) d'ordre* k est une 6-uplet $\mathcal{A} = (\Sigma, \Gamma, Q, q_0, F, \Delta)$, où Σ est un alphabet, Γ un alphabet de pile, Q un ensemble d'états, q_0 l'état initial, F un ensemble d'états finals et Δ décrit les transitions définies en terme d'opérations sur les k-piles de l'automate. La thèse de Carayol propose une étude approfondie de ce modèle [29]. Les ensembles rationnels de piles d'ordre k sont ceux qui sont obtenus par un ensemble rationnel d'opérations de pile (au sens des langages rationnels tels que définis en § II.2).

III Logiques

Dans cette section, on va définir un certains nombre de logiques, avec pour but d'établir la décidabilité de certains problèmes. Dans ce paragraphe, on s'appuie principalement sur les formalismes tels qu'ils sont présentés dans l'ouvrage de Ebbinghaus *et al.* [55].

III.1 Logiques du premier et second ordre

Il y a deux terminologies équivalentes pour la décidabilité de logiques pour des familles de structures relationnelles. La première parle de *décidabilité de théories* pour certaines familles de structure. Précisément, la théorie d'une structure est constitué de l'ensemble des formules satisfaite par une structure. La seconde terminologie parle de décidabilité du problème du *model-checking*. Dans ce dernier cas, les données du problème sont un graphe et une formule, et le résultat est la validité de la formule sur le graphe.

Ces deux approches sont équivalentes. Pour des question d'homogénéité avec d'autres problèmes considérés dans ce mémoire, on retiendra la présentation *model checking*.

Logique du premier ordre. On s'intéresse d'abord à la logique du premier ordre sur les structures relationnelles.

Une *signature relationnelle* σ est un alphabet gradué. Pour chaque symbole R de σ, on note $|R| \geq 1$ l'arité de R (c'est-à-dire son nombre d'arguments). Une structure relationnelle, $\mathcal{M}$, sur σ est donnée par un quadruplet $(M, (R^{\mathcal{M}})_{R\in\sigma})$ où M est *l'univers* de $\mathcal{M}$ et où pour tout $R \in \sigma$, on a $R^{\mathcal{M}} \subseteq M^{|R|}$.

Soit $\mathcal{V}$ un ensemble dénombrable de variables du premier ordre. On utilise $x, y, z \ldots$ pour des éléments de $\mathcal{V}$ et $\bar{x}, \bar{y}, \bar{z}, \ldots$ pour identifier des n-uplets de variables. Une formule atomique sur σ est soit $R(x_1, \ldots, x_{|R|})$ pour $R \in \sigma$ et $x_1, \ldots, x_{|R|} \in \mathcal{V}$ ou $x = y$ pour $x, y \in \mathcal{V}$. Les formules sur σ (σ-formules) sont obtenues à partir des σ-formules par clôture par conjonction $\wedge$, négation $\neg$ et quantification existentielle $\exists$. Les variables libres et liées sont définies de façon classique. On appelle *énoncé* une formule sans variable libre. On écrit $\varphi(\bar{x})$ pour exprimer que les variables libres de φ appartiennent à $\bar{x}$.

Pour toute structure relationnelle $\mathcal{M}$, toute formule $\varphi(x_1, \ldots, x_n)$ et $a_1, \ldots, a_n$ dans M, on écrit $\mathcal{M} \models \varphi[a_1, \ldots, a_n]$ si $\mathcal{M}$ satisfait la formule φ lorsque x_i est interprété par a_i. Si φ est un énoncé, on écrit simplement $\mathcal{M} \models \varphi$. Deux énoncé φ et ψ sont *logiquement équivalents* si pour toutes structures $\mathcal{M}$, $\mathcal{M} \models \varphi$ si et seulement si $\mathcal{M} \models \psi$.

Le *rang de quantification* $\mathrm{qr}(\varphi)$ d'une formule φ est défini par récurrence sur la structure de φ en posant $\mathrm{qr}(\varphi) = 0$ pour φ atomique, $\mathrm{qr}(\varphi \wedge \psi) = \max\{\mathrm{qr}(\varphi), \mathrm{qr}(\psi)\}$, $\mathrm{qr}(\neg\varphi) = \mathrm{qr}(\varphi)$ et $\mathrm{qr}(\exists x\, \varphi) = \mathrm{qr}(\varphi) + 1$. Pour une signature fixée, σ, il y a un ensemble dénombrable de σ-énoncés ayant un rang de quantification donné. Néanmoins, à équivalence logique prêt, il n'y a qu'un nombre fini de tels énoncés, mais cette équivalence est indécidable. La méthode classique pour surmonter ce problème consiste à définir une équivalence (décidable) syntaxique sur les formules (y compris ci ces dernières possèdent des variables libres) de façons à ce que, à cette équivalence prêt, il n'y a qu'un ensemble fini de formule pour chaque rang de quantification, et chaque nombre de variable libre (voir par exemple [54]).

Nous définissons pour tout rang $k \geq 0$ un ensemble fini Norm_k^σ de σ-énoncés normalisés de façon à ce que pour tout σ-énoncé φ il soit possible de calculer, de façon effective, un énoncé équivalent, $\mathrm{Norm}(\varphi)$, dans Norm_k^σ. Cet ensemble est fini, calculable, et définit par récurrence à partir des ensembles suivants :

$$\begin{array}{lcl}
\mathrm{ANA}^\sigma(\bar{x}) & = & \{\, \varphi, \neg\varphi \mid \varphi \text{ atomique sur } \sigma \text{ avec des variables libres dans } \bar{x} \,\} \\
\mathrm{Norm}_0^\sigma(\bar{x}) & = & \left\{ \bigvee_{R\in\mathcal{R}} \bigwedge_{\varphi\in R} \varphi \mid \mathcal{R} \subseteq 2^{\mathrm{ANA}^\sigma(\bar{x})} \right\} \\
\mathrm{Norm}_{k+1}^\sigma(\bar{x}) & = & \left\{ \bigvee_{R\in\mathcal{R}} \bigwedge_{\varphi\in R} \varphi \mid \mathcal{R} \subseteq 2^{\left\{ \exists y\varphi, \forall y\varphi \mid \varphi\in\mathrm{Norm}_k^\sigma(\bar{x},y) \right\}} \right\}
\end{array}$$

où $y \notin \bar{x}$. Notez bien que les ensembles Norm_k^σ ne contiennent que des énoncés, qui par définition ne possèdent pas de variable libre.

La k-théorie de la structure $\mathcal{M}$ sur σ est l'ensemble fini :

$$\mathrm{Thm}_k(\mathcal{M}) \stackrel{\text{def}}{\equiv} \{\, \varphi \mid \varphi \in \mathrm{Norm}_k^\sigma \text{ et } \mathcal{M} \models \varphi \,\}.$$

Nous notons $\mathrm{Thm}_k^\sigma = 2^{\mathrm{Norm}_k^\sigma}$ l'ensemble de toutes les k-théories possibles [3].

Nous appellerons FO la logique du premier ordre, dans certains cas, on précisera $\mathrm{FO}(\sigma)$ pour une signature σ donnée. Et nous serons particulièrement intéressés par la décidabilité de FO pour certaines structures relationnelles. On s'intéressera particulièrement au *model-checking* pour des ensembles particuliers de structures.

Logique du second ordre monadique. Il est possible d'étendre la logique du premier ordre en autorisant la quantification sur les ensembles d'objets. Ce n'est pas l'intégralité de la logique du second ordre, mais une logique du second ordre monadique, noté MSO.

3. Notez que Thm_k contient des éléments qui ne sont la k-théorie d'aucune structure. Par exemple, un élément de Thm_k peut contenir à la fois φ et $\neg\varphi$.

La logique MSO est soumise à la même syntaxe que FO. Les ensembles de signatures seront les mêmes. En revanche, on dispose de $\mathcal{V}_2$, ensemble de variables du second ordre. Ces variables définissent des ensembles d'objets du premier ordre (pour des questions de lisibilité, nous adopterons la convention que les variables du premier ordre seront en minuscule, et du second en majuscule). Et la quantification existentielle sera possible pour les variables du second ordre. On ajoute également les prédicats d'appartenance et d'inclusion. Du fait de ces ajouts, l'égalité des variables est un prédicat dérivé.

Exemple 3. Quelques exemples de formules de MSO.
— $=_1 (x, y) \stackrel{\text{def}}{=} \forall X((x \in X) \Leftrightarrow (y \in X))$
— $=_2 (X, Y) \stackrel{\text{def}}{=} (X \subseteq Y) \wedge (Y \subseteq X)$
— $\text{Singleton}(X) \stackrel{\text{def}}{=} \forall x \forall Y(((x \in X) \wedge (x \in Y)) \Rightarrow X \subseteq Y)$

De la même façon on définit les problèmes de model checking pour MSO et certaines familles de structures.

III.2 Logiques modales

En renonçant à l'usage explicite des quantificateurs et en se limitant à une portée locale, restreinte à des voisinages sur une structure, les logiques modales définissent un sous-ensemble très utile de la logique du premier ordre. L'expressivité est moindre, mais en contre-partie, ces logiques sont décidables sur un plus grand nombre de familles. Il y a de multiples variantes de logiques modales, tels que des logiques épistémiques, ou encore temporelles. Nous reviendrons dans un prochain paragraphe sur les logiques temporelles, mais à présent nous allons nous focaliser sur une logique modale, fragment de la logique du premier ordre sur les structures relationnelles. Le langage, ML que nous définissons par la suite ne possède pas de variable propositionnelle (comme celui de Henessy-Milner [79] mais à l'opposé de la logique modale classique, K, [16]). En outre, les opérateurs modaux que nous définissons ne possèdent pas d'étiquette (ce qui n'est pas la façon de faire usuelle lorsque les systèmes sont eux-même étiquetés). On précisera toutefois lorsque certains résultats de décidabilité s'étendent à des structures étiquetées. Les formules modales de ML sont définies par la syntaxe suivante :

$$\varphi ::= \bot \mid \top \mid \neg\varphi \mid \varphi \wedge \psi \mid \Box\varphi \mid \Diamond\varphi \mid \Box^{-1}\varphi \mid \Diamond^{-1}\varphi.$$

Ce langage est relativement pauvre vis-à-vis de la logique du premier ordre sur les structures relationnelles. Néanmoins, nous verrons des familles de graphes pour lesquels même le *model-checking* de cette logique est indécidable.

Étant donnée une formule modale, φ, le *degré modal* de cette formule est la plus grande valeur d'occurrences imbriquées d'opérateurs modaux dans φ. On note $\text{ML}(\Box)$ la restriction de ML aux opérateurs $\Box$ et $\Diamond$. Les formules de ML seront interprétées dans des graphes orientés, relativement à une signature relationnelle d'arité maximale 2. On définit la notion de satisfaction, $\models$, relativement à un graphe orienté arbitraire $\mathcal{M} = (W, R)$ (avec $w \in W$). Les clauses pour les opérateurs booléens sont classiques et nous les omettrons ; pour les opérateurs modaux, on pose :
— $\mathcal{M}, w \models \Box\varphi \stackrel{\text{def}}{\Leftrightarrow}$ pour tout $w' \in W$ tel que $(w, w') \in R$, on a $\mathcal{M}, w' \models \varphi$.
— $\mathcal{M}, w \models \Diamond\varphi \stackrel{\text{def}}{\Leftrightarrow}$ il existe $w' \in W$ tel que $(w, w') \in R$ et $\mathcal{M}, w' \models \varphi$.
— $\mathcal{M}, w \models \Box^{-1}\varphi \stackrel{\text{def}}{\Leftrightarrow}$ pour tout $w' \in W$ tel que $(w', w) \in R$, on a $\mathcal{M}, w' \models \varphi$.
— $\mathcal{M}, w \models \Diamond^{-1}\varphi \stackrel{\text{def}}{\Leftrightarrow}$ il existe $w' \in W$ tel que $(w', w) \in R$ et $\mathcal{M}, w' \models \varphi$.
On notera que $\Box$ et $\Diamond$ sont duaux, de même que leurs réciproques $\Box^{-1}$ et $\Diamond^{-1}$, ces opérateurs peuvent être mutuellement dérivés dès lors que la négation est présente dans le langage.

Le problème de model checking, $\text{MC}^{\mathcal{S}}(\text{ML})$, est défini comme suit pour les éléments d'une structure relationnelle $\mathcal{S}$:

Donnée : $S \in \mathcal{S}$, un élément de son domaine $u_0 \in Dom(S)$ et $\varphi \in \text{ML}$.

Question : $S, u_0 \models \varphi$?

Dans le cas de la logique ML, un second problème proche est celui de la validité d'une formule. On définit le *problème de validité,* $\mathrm{VAL}^{\mathcal{S}}(\mathrm{ML})$, appelé parfois *model-checking global*, comme ceci :

Donnée : $S \in \mathcal{S}$ et $\varphi \in \mathrm{ML}$.

Question : $S, u \models \varphi$ pour tout élément $u \in Dom(S)$?

Nous l'avons déjà souligné, les formules de $\mathrm{ML}(\Box, \Box^{-1})$ peuvent être vues comme des formules de FO. Ainsi modéliser des propriétés à l'aide de formules modales revient à utiliser des fragments de FO. Plus précisément, étant donnée une formule modale φ de $\mathrm{ML}(\Box, \Box^{-1})$, il est possible, en temps linéaire, de calculer une formule du premier ordre φ' ayant simplement deux variables (voir, entre autres, Blackburn *et al.* [16]) telle que, pour toute structure relationnelle S on ait $S \models \varphi'$ si et seulement si $S, u \models \varphi$ pour tout sommet u de S. De cette façon, le problème de validité, $\mathrm{VAL}^{\mathcal{S}}(\mathrm{ML})$, est le pendant naturel du model-checking de FO sur la famille $\mathcal{S}$.

III.3 Logiques temporelles

Nous allons conclure ce chapitre par un bref survol des logiques temporelles. On l'a déjà vu, ces dernières sont des logiques modales avec des opérateurs modaux différents. Les deux logiques temporelles principales sont LTL (*linear temporal logic*) et CTL (*computation tree logic*). Elles possèdent en commun les opérateurs classiques, l'opérateur suivant, et jusqu'à, mais diffèrent par leurs portée, LTL est relative à une structure, un chemin et une position sur ce chemin, alors que CTL porte sur la structure, un sommet et des chemins issus de ce sommet. La logique CTL possède de plus deux opérateurs de quantification sur les chemins permettant de considérer *tous* les chemins ou bien *au moins un* chemin.

Les formules de LTL sont définies par la syntaxe suivante :

$$\varphi ::= \mathtt{tt} \mid a \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathbf{X}\,\varphi \mid \varphi\,\mathbf{U}\,\psi$$

La sémantique d'une formule de LTL est donc définie par rapport à un graphe étiqueté et un chemin, l'opérateur $\mathbf{X}$ exprimant que la sous-formule φ est satisfaite par le successeur de l'état courant vis-à-vis du chemin. L'opérateur $\mathbf{U}$ exprimant que la première sous-formule est satisfaite par tous les états du chemin jusqu'à ce que la seconde sous-formule soit satisfaite. Cette dernière étant nécessairement satisfaite en un état donné.

Les formules de CTL sont définies par la syntaxe suivante :

$$\varphi ::= \mathtt{tt} \mid a \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathbf{A}\,\psi \mid \mathbf{E}\,\psi$$

$$\psi ::= \mathbf{X}\,\varphi \mid \varphi_1\,\mathbf{U}\,\varphi_2$$

La sémantique est définie par rapport à un graphe étiqueté et un état. Les formules $\mathbf{A}$ et $\mathbf{E}$ imposant que la sous-formule soit vérifiée sur *tous* (*resp. au moins un*) les chemins issus de l'état. Les formules de chemin $\mathbf{X}$ et $\mathbf{U}$ étant relative à un chemin par rapport à la quantification les précédant. Une présentation détaillé de ces logiques est faite, par exemple dans l'ouvrage de Baier et Katoen [9]

Chapitre 2

Graphes de présentation finie

Dans ce chapitre, nous nous intéressons aux graphes possédant des présentations finies. De façon à classer ces différentes familles nous nous appuierons sur la hiérarchie de Chomsky telle qu'elle a été présentée dans le chapitre précédent.

Les résultats de ce chapitre reposent d'une part sur les travaux de Caucal [40, 37, 39, 38] (principalement § II) et d'autre part sur des travaux personnels [117, 115].

I Présentations de graphes

Comme il a été vu dans le premier chapitre, les graphes sont des ensembles de sommets connectés par des arcs. L'ensemble de sommets lui-même est, en général arbitraire. Dès qu'on s'intéresse à des graphes infinis vient la question de la façon de les présenter de façon finie.

Réciproquement, un graphe fini peut être simplement défini par un ensemble explicite de transitions. Néanmoins, même lorsqu'on s'intéresse aux graphes finis, la plupart du temps, les propriétés intéressantes sont indépendantes de la nature exact, ou du choix des sommets. Ainsi la plupart du temps les graphes finis sont donnés *à isomorphisme prêt*. Plus précisément, les noms des sommets sont fixés pour réaliser le travail effectif, mais le résultat vaut pour l'ensemble des graphes isomorphes.

En ce qui concerne les graphes infinis de présentation finie, on distinguera deux sorte de présentation : les présentations internes qui donnent de façon totalement explicite, et fixée, le nom des sommets du graphe, et les présentations externes qui opèrent par transformation de graphe, et qui ne fixent donc pas le nom des sommets. La seconde approche permet de mettre plus en avant la structure.

Présentation interne. L'objectif d'une telle présentation est de donner une vision opérationnelle du graphe dont il est question. Ainsi, une machine de Turing définit un graphe : chaque sommet est une configuration, et les arcs sont les transitions de la machine.

Dans un tel graphe, la moindre modification de la machine produit un graphe distinct. Que ce soit en changeant un symbole, ou en dupliquant un état.

En revanche ce type de présentation permet de manipuler de façon très efficace un tel graphe (en particulier pour écrire des preuves). A titre d'exemple, il est souvent très simple de déterminer si il existe une transition entre deux configurations données, ou encore d'obtenir une description explicite de l'ensemble des configurations.

Présentation externe. Il est plus délicat de donner des exemples précis de présentation externe sans entrer dans des détails. Ce qu'on peut observer c'est qu'il est possible de définir des graphes par transformation à partir d'un graphe donné. Ainsi, la demi-droite (graphe connexe, ayant une racine, et où chaque sommet possède un unique successeur) est un graphe. On peut en donner de multiples

caractérisations explicites. On peut utiliser ce graphe, pour créer d'autres graphes par transformation. Par exemple ajouter un arc partout où deux sommets sont à distance 2, exactement. Ou encore, ajouter un segment de n arcs à partir du sommet à distance n de la racine. Ce type de définition est externe. on ne donne pas explicitement le nom des différents sommets obtenus dans ce graphe. Pourtant, il est possible de raisonner sur ce graphe, ou de prouver certaines de ses propriétés.

II Familles d'expressivité algébrique

Comme nous l'avons vu dans le premier chapitre, les langages algébriques sont caractérisés par les grammaires du même nom. Néanmoins les grammaires de mots ne sont pas l'outil le plus naturel pour définir une famille de graphes ayant une expressivité algébrique. (Il est néanmoins possible de définir des graphes à l'aide de grammaires de mots, mais nous n'aborderons pas ce sujet directement dans ce document).

En pratique ce paragraphe présentera en détail les *graphes réguliers* qui sont engendrés par les *grammaires déterministes de graphes* définies ci-dessous.

II.1 Grammaires déterministes de graphes

Le champs d'application des grammaires de graphe est très général. Ces grammaires ont été définies sur le modèle des grammaires de mots (voir, par exemple, le *Handbook of Graph Grammars and Computing by Graph Transformations* [129]). À partir d'un axiome, une telle grammaire engendre un ensemble infini de graphes. Les travaux de Courcelle [49] utilisent des ensembles déterministes de règles pour construire un unique graphe (éventuellement infini) à partir d'un axiome. En 2008, Caucal a réalisé une étude approfondie de ce type de grammaires [37]. Il en donne une présentation homogène, et propose un catalogue important de techniques de normalisation et de transformations. Caucal fait un usage novateur des couleurs sur les sommets pour faciliter les manipulation des graphes qu'ils engendrent.

Les grammaires déterministes de graphes, même lorsque elles engendrent des *graphes* ordinaires, nécessitent la notion d'hyperarc et d'hypergraphe : étant donné un ensemble de sommets V et un alphabet gradué $N = \cup_{k\in\mathbb{N}} N_k$ avec $\varrho : N \to \mathbb{N}$ est la fonction qui associe à chaque symbole son arité, un *hyperarc* sur N, est un élément $Av_1 \dots v_{\varrho(A)}$ de $\cup_{k\in\mathbb{N}} N_k V^k$ (on assimile un hyperarc d'arité deux avec un arc). Un *hypergraphe* est un ensemble d'hyperarcs. Une grammaire de graphes est principalement constituée d'un ensemble de règles dont le membre gauche est constitué d'un hyperarc, et le membre droit d'un hypergraphe.

Définition 2 (Grammaire d'hyperarcs)**.** Une *grammaire d'hyperarcs* (HR-grammaire) $\mathcal{G}$, est un quadruplet (N, T, H_0, R), où N et T sont deux alphabets gradués de symboles *non-terminaux* et *terminaux* respectivement ; H_0 est l'axiome, un hypergraphe fini formé d'hyperarcs étiquetés par $N \cup T$, et R est un ensemble fini de règles de la forme $A\ x_1\ \cdots\ x_{\varrho(A)} \to H_A$ où $A\ x_1\ \cdots\ x_{\varrho(A)}$ est un hyperarc qui connecte des sommet finis, et H_A est un hypergraphe fini.

En règle générale, les sommets qui sont présents dans le membre gauche d'une règle apparaissent dans le membre droit, mais ce n'est pas imposé par la définition. Comme indiqué dans la définition, on notera le plus souvent H_A le membre droit d'une règle dont le non-terminal étiquetant le membre gauche est A.

Remarque 3. Comme déjà précisé, nous considérons des graphes dans ce document, l'alphabet des symboles terminaux se limitera donc à ceux d'ordre un ou deux. Les graphes sont de simples sous-ensembles de $T_2VV \cup T_1V$.

Une grammaire est dite *déterministe* lorsqu'elle ne possède qu'une seule règle de récriture par non-terminal :

$$(X_1, H_1), (X_2, H_2) \in R \wedge X_1(1) = X_2(1) \Rightarrow (X_1, H_1) = (X_2, H_2)$$

On définit à présent, pour un ensemble R de règles, la relation de *récriture* $\underset{R}{\rightarrow}$: M se récrit en N, noté $M \underset{R}{\rightarrow} N$, si il y a un hyperarc non-terminal $X = Av_1v_2 \dots v_p$ dans M et une règle $Ax_1x_2 \dots x_p \rightarrow H_A$ dans R tels que N est obtenu en remplaçant X par H_A dans M : $N = (M - X) \cup h(H_A)$ pour une injection h, qui associe v_i à x_i pour tout i, et tous les autres sommets de H_A à de nouveaux sommets qui ne sont pas dans M. Une telle récriture est notée $M \underset{R,X}{\longrightarrow} N$. Cette notion de récriture s'étend aux ensembles de non-terminaux, soit E un tel ensemble, on note la récriture par E, $M \underset{R,E}{\longrightarrow} N$. La récriture complète et parallèle, notée $\underset{R}{\longrightarrow}$ est l'opération de récriture relative à l'ensemble de tous les arcs non-terminaux de R.

II.2 Graphes réguliers

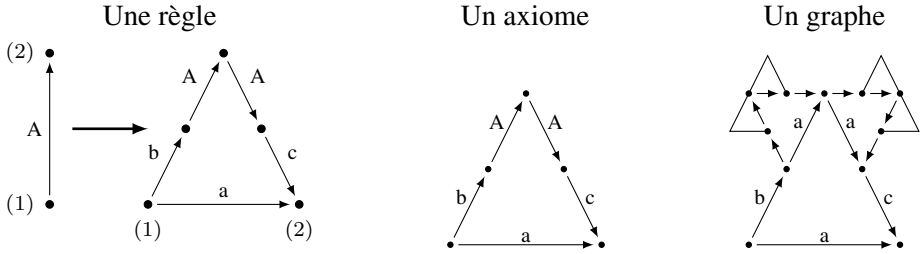
Étant donnés une grammaire déterministe de graphe, $\mathcal{G} = (N, T, H_0, R)$, et un hypergraphe H, on note $[H] := H \cap (T\ V_H\ V_H \cup T\ V_H)$ l'ensemble des arcs terminaux et des couleurs de H. On définit, $\mathcal{G}^\omega$, le graphe limite de $\mathcal{G}$, comme étant l'ensemble des unions dénombrables de $[H_n]$, obtenu à partir de H_0 (il y a une infinité de graphes H_n pour chaque n) :

$$\mathcal{G}^\omega = \left\{ \cup_{n \geqslant 0}[H_n] \mid \forall n \geqslant 0, H_n \underset{R}{\longrightarrow} H_{n+1} \right\}$$

Formellement, $\mathcal{G}^\omega$ est un ensemble de graphes, et on dit qu'un graphe H est engendré par $\mathcal{G}$ lorsqu'il appartient à $\mathcal{G}^\omega$. Dans la mesure où ces graphes sont isomorphes entre eux, on dira par abus de langage que $\mathcal{G}^\omega$ est *le* graphe engendré par $\mathcal{G}$.

Pour tout sommet v d'un graphe $H \in \mathcal{G}^\omega$ on note $\mathsf{Can}(v)$ le sommet de H_A, tel que v n'est pas un sommet de H_k, v est un sommet de $\rightarrow R_A(H_k)$, et $v = h(\mathsf{Can}(v))$. En d'autres termes v est engendré par le sommet $\mathsf{Can}(v)$ du membre droit de la règle A. De plus si un sommet v appartient à H_n, mais pas à H_{n-1}, on dit que v est de *niveau* n, noté $L(n)$.

Exemple 4. Voici un exemple simple de grammaire déterministe de graphes, et une représentation (fractale) du graphe engendré.



Ces grammaires déterministes de graphe définissent les *graphes réguliers*. Cette caractérisation est efficace pour étendre à des graphes infinis certaines techniques employées pour les graphes finis. Ainsi, pour calculer les composantes connexes d'un graphe régulier, il suffit de réaliser ce calcul dans chacun des membre droit des règles. À une réserve près : il est possible que certaines règles soient dupliquées suivant le contexte dans lequel elles apparaissent. Globalement, le principe est exactement le même que pour des graphes finis. Il en va de même pour l'accessibilité.

Ces graphes peuvent avoir des sommets de degré infini, néanmoins, pour chacun de ces graphes l'ensemble des degrés est fini, et calculable.

Exemple 5. La figure 2.1 illustre une règle ainsi que le graphe de degré sortant infini engendré par cette règle.

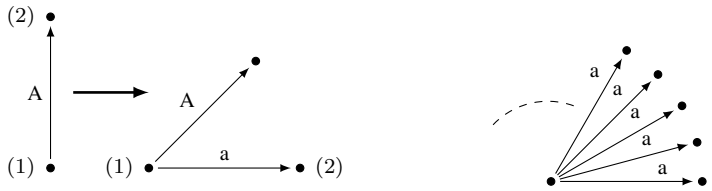


FIGURE 2.1 – Degré infini : règle et graphe engendré.

Lorsqu'on se restreint au graphes de degré fini, les graphes réguliers correspondent exactement aux graphes des transition des automates à pile [40]. Le plus souvent, l'algorithmique de ces automates s'appuie sur des restrictions syntaxiques, des propriétés particulières des états ou encore la structure de la pile. Or les propriétés mises en évidences par ces algorithmes sont, le plus souvent, de nature structurelles et sont indépendantes de la forme particulière de l'automate.

Réciproquement, de nombreuses transformations syntaxiques sur les automates peuvent préserver le langage reconnu par ce dernier, mais modifier radicalement le graphe des transitions. Dans les cas des transformations de grammaires, elle sont clairement de deux sortes : celles qui ne modifient pas le graphe engendré (mais transforment les règles dans un but particulier), et celles qui transforment le graphe pour atteindre un but précis, tel que restreindre à une composante connexe, ou déterminiser le graphe.

Dans la suite, nous verrons des caractérisations similaires pour des familles plus générales de graphes (dans la section III.2).

II.3 Propriétés élémentaires et formes normales pour les graphes réguliers

Pour toute règle $(Av_1 \dots v_{\rho(A)}, H_A)$, on dit que les sommets $v_1, \dots, v_{\rho(A)}$ de H_A sont les *entrées* de H_A, ensuite, les sommets $\bigcup_{Y \in H_A \wedge Y(1) \in N} V_Y$ sont les *sorties* de H_A. Les sorties appartiennent aux hyperarcs non-terminaux.

Étant donné un non terminal $A \in N$, on note $\mathsf{Succ}(A)$ l'ensemble des non-terminaux qui apparaissent dans H_A.

Étant donnés une HR-grammaire $\mathcal{G} = (N, T, H_0, R)$ et un hyperarc non-terminal $X = Av_1v_2 \dots v_p$, on note R^ω (resp. $R^\omega[X]$) un graphe particulier dans l'ensemble $\mathcal{G}^\omega$ (resp. in $(\mathcal{G}[X])^\omega$ avec $\mathcal{G}[X] := (N, T, R, X)$).

Étant données deux HR-grammaires $\mathcal{G} = (N, T, H_0, R)$ et $\mathcal{G}' = (N', T', H'_0, R')$, on dit que $\mathcal{G}'$ est un *coloriage* de $\mathcal{G}$ si, pour tous graphes $H \in \mathcal{G}^\omega$ et $H' \in \mathcal{G}'^\omega$, il existe un isomorphisme entre H et H' qui préserve les couleurs de H, et s'il existe un couleur de T'_1 qui n'existe pas dans T_1 (qui marque ainsi certains sommets).

Pour une partie de notre utilisation des HR-grammaires nous utiliserons la forme normale qui suit : une HR-grammaire est dite *complètement extérieure* si les seules entrées qui sont aussi des sorties sont des sommets de degré infini.

Toutes les grammaires peuvent être mise en forme complètement extérieure.

Théorème 8. *[37] Tout graphe régulier est engendré par une grammaire complètement extérieure. Cette transformation est effective.*

L'intérêt de ce résultat est qu'on peut faire la supposition que les HR-grammaires qui engendrent les graphes qui nous intéressent sont dans cette forme. Il est donc simple d'identifier les sommets de degré infini.

Accessibilité pour les graphes réguliers. Le calcul des ensembles d'accessibilité est central pour les problèmes de vérification qui seront abordés dans le chapitre 4.

Proposition 9 ([37]). *Étant donnés une* HR*-grammaire* $\mathcal{G} = (N, T, H_0, R)$ *et un sous-alphabet d'étiquettes d'arcs* $T' \subseteq T_2$*, une couleur* $c \in T_1$*, et une couleur* $r_c \notin T_1$*, il est possible de calculer une* HR*-grammaire* $\mathcal{G}' = (N', T \cup \{r_c\}, H'_0, R')$ *de sorte que* $\mathcal{G}'$ *soit un coloriage de* $\mathcal{G}$ *et de plus tous les états accessibles depuis la couleur* c *en utilisant des arcs étiquetés par* T' *portent la couleur* r_c*.*

La proposition 9 produit une grammaire qui est de taille polynomiale par rapport à la grammaire initiale.

Toujours dans le chapitre 4 nous aurons besoin de détecter les chemins infinis. Ces derniers peuvent être de deux sortes : soit un circuit dans le graphe (où un nombre fini de sommets est vu un nombre infini de fois) ou bien un chemin *divergeant*, c'est à dire visitant un nombre infini de sommets.

Proposition 10. *Étant donnée une* HR*-grammaire* $\mathcal{G} = (N, T, H_0, R)$*, une couleur* $c \in T_1$*, il est possible de décider en temps polynomial de l'existence d'un chemin infini dans le graphe engendré par* $\mathcal{G}$*, de sorte qu'après une position (dans le chemin)* $i_0 \in \mathbb{N}$ *tous les sommets placés après la position* i_0 *possèdent la couleur* c*.*

La proposition 10 est démontrée dans certains de nos travaux [46, 44], cette démonstration est extrêmement classique et nous ne la rappellerons pas ici. L'idée est d'adapter le calcul d'accessibilité avec une forme d'exclusivité, et en vérifiant une propriété garantissant un prolongement perpétuel d'un tel chemin.

II.4 Connexion entre graphes réguliers et automates à pile

On commence par rappeler une définition des automates à pile. Soit $\mathcal{A} = (\Sigma, \Gamma, Q, q_0, \Delta)$ un automate à pile. Une configuration de cet automate est formée par un état de Q, et un mot de pile dans Γ^*. Le *graphe des transitions*, $G^{trans}_{\mathcal{A}}$ d'un automate à pile est un sous-ensemble de $Q\Gamma^* \times \Sigma \times Q\Gamma^*$, où on a $(pUw, a, qVw) \in G^{trans}_{\mathcal{A}}$ si on a la transition $(pU, a, qV) \in \Delta$. Le *graphe des configurations* de l'automate $\mathcal{A}$, noté $G^{conf}_{\mathcal{A}}$, est la restriction de $G^{trans}_{\mathcal{A}}$ aux configurations accessibles depuis la configuration initiale : q_0.

Il y a une forte connexion entre les graphes des configurations d'un automate à pile et les graphes réguliers. Lorsqu'on se restreint au graphes réguliers de degré fini (à la fois sortant et entrant), on obtient exactement les graphes des configurations des automates à pile (voir Caucal [37, Théorème 5.11]).

Ce qui est intéressant avec la plupart des transformations qui modifient les grammaires engendrant les graphes réguliers c'est que le graphe produit est toujours le même (à isomorphisme près). Cette propriété n'est en générale pas vérifiée par les transformations sur les automates à pile qui ne préservent que le langage engendré.

Dans le chapitre 4 nous présenterons quelques applications ayant pour modèle les graphes réguliers.

III Familles d'expressivité contextuelles

Nous allons à présent aborder des éléments plus personnels de ce travail. Ceux-ci portent sur les graphes rationnels. Dans un premier temps nous rappellerons les résultats les plus anciens et les définitions fondamentales. Ensuite, nous présenterons deux lignes de résultats : sur des systèmes de récriture de graphes (des grammaires de graphes généralisées), puis sur les arbres rationnels.

III.1 Graphes rationnels

On présente, ici, les éléments clés définissant les graphes rationnels. Plus de détails sont disponibles dans quelques articles antérieurs à 2005 [113, 118, 117].

Nous avons déjà vu que la famille des ensembles *rationnels* d'un monoïde $(M, \cdot)$ est la plus petite famille contenant les singletons de M, et qui soit close pour les opération d'union, concaténation et itération.

Il est possible d'étendre cette notion de rationalité à des monoïdes partiels où l'opération de concaténation est définie de façon partielle (dans le cas des graphes rationnels en tenant compte de l'étiquette des arcs), voir par exemple le second chapitre de ma thèse de doctorat [114]. Dans le présent document, on se focalise sur une définition opérationnelle de ces graphes en utilisant les transducteurs finis.

Un *transducteur* est un automate fini étiqueté par des paires de mots sur un alphabet fini, X (consulter, par exemple, les ouvrages de référence de Berstel ou de Sakarovitch [13, 132]). Un tel transducteur accepte une relation rationnelle dans $X^* \times X^*$. Ces relations sont les ensembles rationnels du monoïde produit $(X^* \times X^*, \cdot)$.

A présent, on considère les graphes de $\Sigma \times X^* \times X^*$, et en particulier les *graphes rationnels*, qui sont l'extension des relations rationnelles, et définis par les *transducteurs étiquetés*.

Définition 3. Un *transducteur étiqueté* $T = (Q, I, F, E, L)$ sur X et Σ, est formé d'un ensemble d'états Q, un ensemble d'états initiaux $I \subseteq Q$, un ensemble d'états finals $F \subseteq Q$, un ensemble fini de transitions $E \subseteq Q \times X^* \times X^* \times Q$ et d'une application L de F vers 2^{Σ}.

Un arc $u \xrightarrow{a} v$ est *accepté* par un transducteur étiqueté T s'il existe un chemin entre un état de I vers un état f de F étiqueté (u, v) et tel que $a \in L(f)$.

Définition 4. L'ensemble des *graphes rationnels* sur $\Sigma \times X^* \times X^*$, noté $Rat(\Sigma \times X^* \times X^*)$, est constitué de l'ensemble des graphes acceptés par les transducteurs étiquetés.

Les *graphes automatiques* sont définis par les transducteurs automatiques qui forment une sous-classe stricte des transducteurs généraux. Très brièvement, ces transducteurs sont *synchrones* (c'est-à-dire les étiquettes des transitions sont des couples de symboles), et chacun des états finals est associé à une relation de la forme soit $\{\varepsilon\} \times L$, soit $L \times \{\varepsilon\}$ pour L un langage rationnel donné, ainsi un couple est reconnu par un tel transducteur lorsqu'il est obtenu par un chemin dans le transducteur concaténé avec un couple appartenant à la relation associé à l'état final atteint. Blumensath *et al.* [17] ont étudié les structures automatiques, qui ont une logique du premier ordre décidable.

Exemple 6. Le graphe représenté sur la droite de la figure 2.2, est engendré par le transducteur étiqueté représenté sur la gauche de cette même figure.

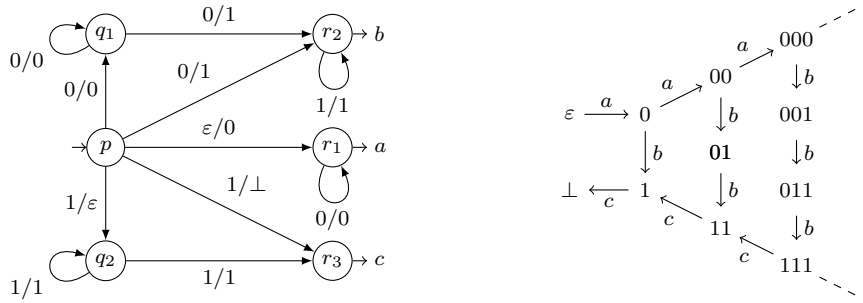


FIGURE 2.2 – Graphe rationnel, et transducteur l'engendrant

Le chemin, $p \xrightarrow{0/0} q_1 \xrightarrow{0/1} r_2 \xrightarrow{1/1} r_2$, accepte le couple $(001, 011)$, l'état final, r_2, est étiqueté par b donc l'arc $001 \xrightarrow{b} 011$ est présent dans le graphe.

Les graphes rationnels définissent une famille très générale de graphes. Ils constituent une généralisation naturelle des graphes réguliers. La plupart de leurs caractéristiques sont indécidables ce qui est résumé dans l'énoncé qui suit.

Théorème 11 ([113]). *Les problèmes suivants sont indécidables pour les graphes rationnels :*

— *(pour un graphe et deux sommets) Accessibilité ;*
— *(pour un graphe et une formule) Logique du premier ordre ;*
— *(pour deux graphes) Isomorphisme, bisimulation, inclusion.*

Si on note $TR(\Sigma^*)$ la famille de langages qui sont traces de graphes rationnels, $Rat(\Sigma \times X^* \times X^*)$, entre un singleton et un ensemble rationnel de sommets, formellement,

$$TR(\Sigma^*) \stackrel{\text{def}}{=} \{L(G, \{v\}, L) \mid G \in Rat(\Sigma \times X^* \times X^*) \wedge v \in X^* \wedge L \in Rat(X^*)\}.$$

On peut formuler le résultat suivant.

Théorème 12 ([118]). *Les traces des graphes rationnels sont les langages contextuels :*

$$CS(\Sigma^*) = TR(\Sigma^*)$$

De plus cette égalité reste valide si on se restreint aux graphes de degré fini (mais non-borné).

On peut observer que la caractérisation des graphes rationnels à l'aide de transducteurs est interne. Cette définition a un effet de bord notable : l'ensemble des sommets d'un graphe rationnel est un ensemble rationnel de mots. Ainsi il est possible de définir des graphes non-rationnels simplement en utilisant un codage non-rationnel, sans pour autant que leur structure soit intrinsèquement non rationnelle. Comme, en outre, le théorème 11 indique que l'isomorphisme de deux graphes rationnels est indécidable, c'est une contrainte pour déterminer les limites de cette famille en terme d'expressivité structurelle.

Dans la suite, nous verrons une définition externe des graphes rationnels. Elle s'appuie sur des systèmes contextuels de récriture de graphe.

III.2 Systèmes contextuels de graphes

Ce paragraphe assemble des résultats publiés dans un travail de 2010 [115]. Il existe de nombreux systèmes contextuels de récriture de graphes, voir, par exemple, les travaux de Schurr [137, § 4]). Cependant, comme pour les HR-grammaires, ces systèmes n'ont pas été utilisés pour définir des familles de graphes (infinis), mais plutôt pour définir des familles d'ensembles de graphes finis.

Comme nous allons le voir, ces systèmes, en l'absence de contraintes fortes, engendrent des graphes non-récursifs.

Systèmes généraux

Étant donné un système de génération, la récursivité est interprétée différemment pour un ensemble de graphes finis, et pour un graphe infini : dans le premier cas on s'intéresse à l'existence d'un graphe donné dans l'ensemble, alors que dans le second on s'intéresse à l'existence d'un arc entre deux sommets donnés. Dans le cas d'un système de récriture de graphes, dès lors qu'il ne peut pas supprimer d'arcs terminaux, la première interprétation détermine la récursivité du modèle. Dans le second cas, en revanche, ce n'est pas suffisant : un arc entre deux sommets donnés peut apparaître après un nombre arbitraire (non borné, en général) de récriture.

Soit N_R un ensemble gradué, fini, de *non-terminaux*, et T_R un ensemble gradué, fini, de *terminaux*.

Voici donc une définition naturelle de système contextuel de récriture de graphe.

Définition 5 (Système contextuel de récriture de graphe). Un *système contextuel de récriture de graphe* S, est constitué d'un ensemble fini de règles de la forme $H_c \cup f\ x_1\ \cdots\ x_{\varrho(f)} \to H_c \cup H$ où $f\ x_1\ \cdots\ x_{\varrho(f)}$ est un hyperarc non-terminal, H_c un graphe fini de *contexte*, et H un hypergraphe fini partageant certains sommets avec H_c et $\left\{x_1, \cdots, x_{\varrho(f)}\right\}$. L'hypergraphe H_c n'est formé que d'hyperarcs terminaux, et $H_c \cup f\ x_1\ \cdots\ x_{\varrho(f)}$ constitue un hypergraphe connexe.

Dans le preuve de la proposition 13 nous définissons un exemple explicite de système contextuel de récriture de graphe (défini par les règles r_0, r_1, r_Δ).

À présent, étant donnée une règle $H_c \cup f\ x_1\ \cdots\ x_{\varrho(f)} \to H_c \cup H$, une étape de récriture dans un graphe G consiste à identifier une occurrence du non-terminal f dans G, de façon à ce qu'il existe un morphisme de graphe [1], h, de $H_c \cup f\ x_1\ \cdots\ x_{\varrho(f)}$ dans G, ensuite l'hyperarc f est retiré de G, puis, H est ajouté à G, selon la règle, et une extension du morphisme h qui envoie les sommets de H qui ne sont pas dans H_c sur de nouveaux sommets de G. Étant donnés un système contextuel de récriture de graphe et un graphe fini H, on définit $S^\omega(H)$ de la même façon que G^ω pour une HR-grammaire G. Il est utile de remarquer que le graphe obtenu est la restriction aux symboles terminaux, en particulier, dans notre contexte, les hyperarcs d'arité 1 ou 2.

On dit qu'un système contextuel de récriture de graphe est *déterministe* lorsqu'il n'existe qu'une seule règle pour chaque non-terminal. Cette restriction ne fait que *limiter* le non-déterminisme, dans la mesure où il reste possible que, dans le voisinage d'un hyperarc non-terminal, le contexte d'une règle puisse être trouvé plus d'une fois. Dans un tel cas, le système peut, éventuellement, engendrer plusieurs graphes non-isomorphes.

Cette généralisation naturelle des HR-grammaires est en pratique trop générale. Même si elle est limitée aux systèmes qui garantissent l'unicité du contexte tout au long des récritures. Les systèmes contextuels de récriture de graphe engendrent des graphes non-récursifs.

En guise de préambule à la preuve de cette propriété, on peut observer que les graphes des transitions d'une machine de Turing sont récursifs. En effet, étant données deux configurations d'une machine de Turing, ils est trivial de vérifier si elles définissent une transition de la machine. Une seconde observation est que l'ensemble des configurations accessibles depuis une configuration donnée est non-récursif, en vertu de l'indécidabilité de l'accessibilité pour les machines de Turing.

Proposition 13. *Soient M une machine de Turing, u_0, et u_1 deux configurations, il existe, de façon effective, un système contextuel de récriture de graphe, S, qui engendre, à partir d'un axiome fini, A, un ensemble de graphes dont chacun est isomorphe à la composante connexe de M à partir de sa configuration initiale. De plus dans chacun de ces graphes, le sommet en bijection avec u_0 (*resp. *u_1) est marqué par la couleur c_1 (*resp. *c_2).*

Démonstration. Supposons que la machine M utilise l'alphabet de bande $\Sigma = \{0, 1\}$, et possède un ensemble Q d'états. Pour simplifier, on suppose que la bande est finie à gauche, et initialement remplie de symboles 0. On suppose également qu'une configuration est terminée par le symbole 1, ou un état.

On rappelle que les transitions de M, sont de la forme $(p, A, q, B, \diamond)$ où $p, q \in Q$ sont respectivement l'état actuel et futur de M, $A \in \Sigma$ est le symbole qui suit l'état sur la bande de lecture et $\diamond \in \{\triangleright, \triangleleft\}$ détermine le mouvement de la tête de lecture vers la droite ou la gauche.

Soit $\Delta = (p, q, 1, 0, \triangleleft)$ une transition de M. On représente la règle de récriture qui lui correspond sur la figure 2.3.

Les règles r_0 et r_1 engendrent un arbre complet qui correspond aux configurations de M. Chaque chemin depuis la racine de cet arbre, jusqu'à un arc étiqueté $\#$ représente une configuration de la machine. Pour chaque transition Δ de M, il y a une ou deux règles [2] similaires à $r_{\Delta,0}$ qui simulent une étape de récriture pour la machine. Le contexte de ces règles permet de se déplacer dans l'arbre des configurations. Ici, on observe les arcs $0, p, 1$ et $q, 0, 1$ l'opération de récriture remplace le non-terminal Δ_0 et produit **nxt**. Il y a d'autres règles pour se déplacer jusqu'à ces situations intéressantes (en utilisant un non-terminal **test**). la règle associée à **test** progresse suivant des chemins identiques et engendre les non-terminaux $\Delta_?$ pour toutes les règles Δ.

1. Il pourrait être tentant d'ajouter l'injectivité pour ce morphisme, il n'en est rien : la plupart du temps il est souhaitable que certains sommets puissent être identifiées dans le graphe de contexte.

2. Il y a deux règles lorsque le déplacement se produit vers la gauche ($\diamond = \triangleleft$) qui dépendent du symbole précédent la tête de lecture.

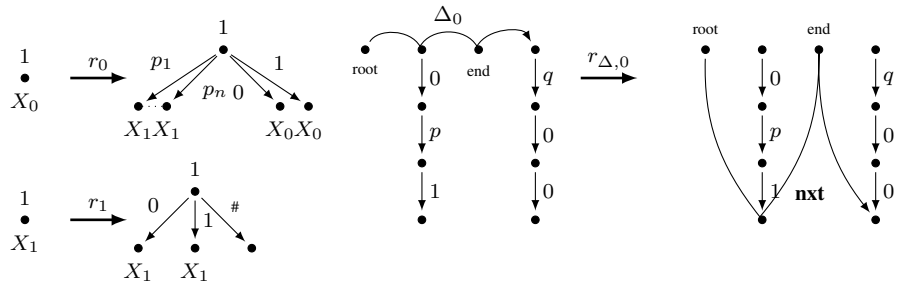


FIGURE 2.3 – Quelques règles pour la démonstration de la proposition 13

La règle pour **nxt** se reproduit en suivant des chemins identiques.

La règle qui assure la terminaison, et donc la restriction à la composante connexe de la configuration initiale est représentée sur la figure 2.4.

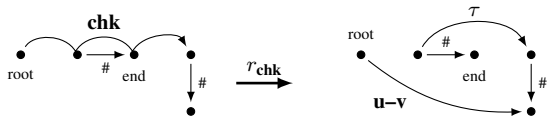


FIGURE 2.4 – Règle pour le non-terminal **chk**

Le membre gauche de la règle $r_{\mathbf{chk}}$ (figure 2.4) valide que le calcul a atteint la fin de la configuration de départ (dont l'extrémité était *enregistrée* par le sommet *end* des différents hyperarcs intermédiaires). Le non-terminal **u–v** est produit. Il amorcera le calcul de la configuration suivante. Cet arc non-terminal enregistre à la fois la racine des configuration ainsi que l'extrémité de la nouvelle configuration. Un arc terminal τ est produit entre les deux extrémités des configurations.

L'axiome est décrit sur la figure 2.5.

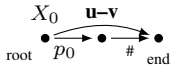


FIGURE 2.5 – Règle pour le non-terminal **u–v**

Les couleurs c_0 et c_1 sont simplement ajoutées à l'aide de deux règles ayant comme contexte précisément les configurations respectives leur correspondant.

Ceci démontre l'existence d'un système contextuel de récriture de graphe qui engendre un graphe isomorphe, et qui possède les couleurs c_0 et c_1 sur les images des configurations u_0 et u_1. □

Ainsi, s'il existait un algorithme permettant de déterminer l'existence d'un τ-arc entre les deux sommets colorés c_0 et c_1, il serait possible de décider l'accessibilité d'une configuration pour la machine M.

Ainsi, on peut reformuler la proposition 13 en le corollaire suivant.

Corollaire 14. *Les systèmes contextuels de récriture de graphes engendrent des graphes non-récursifs.*

Grammaires contextuelles de graphes

Maintenant, nous allons ajouter des contraintes aux systèmes contextuels de récriture de graphes pour permettre de caractériser exactement les graphes rationnels.

Définition 6. Une *grammaire contextuelle d'hyperarcs* (notée CHR-grammaire) est un quintuplet (C, N, T, H_0, R_c), où C, N et T sont des alphabets gradués finis de symboles respectivement *contextuels*, *non-terminaux* et *terminaux* ; R_c est un *système contextuel de récriture de graphes déterministe* où, pour chaque règle $H_c \cup f x_1 \dots x_{\varrho(f)} \to H_c \cup H$, le graphe H_c n'est composé que d'hyperarcs étiquetés dans C, et H par des hyperarcs étiquetés dans $T \cup N$, et H_0, l'axiome est un graphe régulier (engendré par une grammaire déterministe de graphes) *déterministe* dont les arcs sont étiqueté dans C, et possédant un unique hyperarc étiqueté dans N.

Étant donnée une CHR-grammaire $G = (C, N, T, H_0, R_c)$, l'ensemble G^ω est définit par $R_c^\omega(H_0)$, de façon similaire au graphe d'une HR-grammaire.

On peut noter que pour chaque règle de R_c, le graphe de contexte (H_c) n'est pas modifié : aucun arc est ajouté ni retiré, les récritures successives préservent l'axiome H_0. D'autre part, comme l'axiome est un graphe régulier *déterministe*, il est aisé de déterminer si un non-terminal peut apparaître entouré de plusieurs occurrence de son contexte.

Vérifier si un graphe régulier est de déterministe est décidable, voir le travail de Caucal [37, proposition 3.13].

Dans la suite, nous allons introduire des restrictions plus fortes sur cet axiome et sur les règles et voir qu'il est difficile d'autoriser des graphes qui ne sont pas des arbres.

Dans un premier temps, nous allons voir qu'il est suffisant d'utiliser un arbre $|X|$-aire pour obtenir tous les graphes de $Rat(\Sigma \times X^* \times X^*)$.

Proposition 15. *Chaque graphe de $Rat(\Sigma \times X^* \times X^*)$ est isomorphe à un graphe engendré par une* CHR*-grammaire, et dont l'axiome est un arbre.*

Démonstration. Pour illustrer cette démonstration, on utilisera le graphe définit dans l'exemple 6.

Soit G un graphe rationnel de $\Sigma \times X^* \times X^*$ (et $T = (Q, \{q_0\}, F, E, L)$ un transducteur qui le réalise), soit Γ_X l'arbre $|X|$-aire complet étiqueté par X .

On définit la CHR-grammaire $G_\Gamma = (X, Q, \Sigma, R_c, \Gamma_X \cup \{q_0\ v_\varepsilon\ v_\varepsilon\})$ de la façon suivante : X est l'alphabet de contexte (les étiquettes de Γ_X), l'ensemble des états du transducteur (Q) forme l'ensemble des non-terminaux, chacun d'eux est d'arité 2, le sommet v_ε est la racine de l'arbre Γ_X. De plus, pour tout $u \in X^*$, on note v_u le sommet de Γ_X atteint par le chemin étiqueté par u).

Pour chaque état de T on définit la règle r_p qui est représentée schématiquement sur la figure 2.6.

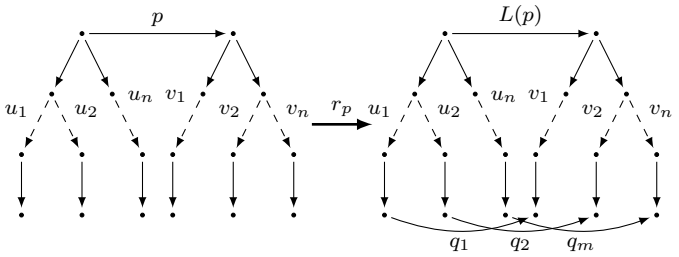


FIGURE 2.6 – Règle relative à l'état p de T

Précisément, le membre gauche de la règle est formée par un graphe ayant un arc $p\ v_s\ v_g$ entre deux sommets (v_s et v_g) qui sont respectivement racines de deux arbres déterministes finis t_1 et t_2. Pour chaque transition $p \xrightarrow{U_i/V_i} q_i$ dans T, les arbres t_1 et t_2 possèdent respectivement les chemin U_i et V_i, d'extrémité respectives u_i et v_i. Le membre droit de la règle ne possède plus l'arc $p\ v_s\ v_g$ mais possède un arc $q_i\ v_{u_i}\ v_{v_i}$ pour chaque q_i (comme Γ_X est déterministe les contexte sont déterminés de façon unique).

De plus, si $p \in F$, pour chaque $a \in L(p)$ il y a un arc $a\ v_s\ v_g$ dans le membre gauche de la règle r_p.

Ainsi, la CHR-grammaire G_Γ produit un arc étiqueté a entre des sommets v_u et $v_{u'}$ si et seulement si il existe un chemin étiqueté u/u' dans le transducteur T, et qui se termine par un sommets final produisant le terminal a. Ainsi tout graphe dérivé de l'axiome, en utilisant G_Γ est isomorphe à G. □

Les arbre-grammaires contextuelles séparées engendrent les graphes rationnels

Dans ce paragraphe, nous définissons une restrictions aux CHR-grammaire qui permet une réciproque à la proposition 15.

Une CHR-grammaire (C, N, T, H_0, R_c) est dite *arbre-grammaire contextuelle* (notée CHR-arbre-grammaire) si l'axiome H_0 est un arbre, et le membre gauche de chaque règle de R_c est formé par des arbres dont les racines sont les sommets de l'hyperarc non-terminal (il est possible que ce dernier possède d'autres sommets *dans* ces arbres). De plus si chacun des arbres possède un unique sommet commun avec l'hyperarc non-terminal, (nécessairement sa racine) cette grammaire est dite *arbre-grammaire contextuelle séparée* (notée CHR-arbre-grammaire-séparée). Les graphes engendrés par cette dernière sont tous rationnels. On peut également noté que le système utilisé dans la preuve de la proposition 15 est de ce type, le schéma de ses règles est sur la figure 2.6.

Proposition 16. *Tout graphe engendré par une* CHR*-arbre-grammaire-séparée, est, de façon effective, isomorphe à un graphe de* $Rat(\Sigma \times X^* \times X^*)$.

Il y a plusieurs difficultés pour établir cette réciproque : en premier lieu, l'axiome peut ne pas être un arbre complet, il est donc nécessaire de vérifier qu'aucun arc ne peut être créé si sa source et son but ne sont pas présent dans l'axiome. Il est également nécessaire de prendre en compte les hyperarcs d'arité supérieure à deux. Enfin, des croisements peuvent se produire : l'arbre associé à la source du non-terminal du membre gauche peut contenir la cible d'un non-terminal du membre droit et réciproquement (de façon identique pour les arcs terminaux, ou les hyperarcs d'arité supérieur).

Pour démontrer la proposition 16 nous établissons d'abord deux lemmes techniques.

Lemme 17. *L'ensemble des étiquettes des chemins entre la racine d'un arbre régulier menant à une certaine couleur (étiquettes d'arité* 1*) forme un ensemble régulier de mots.*

Lemme 18. *Toute* CHR*-arbre-grammaire-séparée* G *peut être transformée, de façon effective en une* CHR*-arbre-grammaire-séparée* G' *qui engendre le même ensemble de graphes, et telles que, pour toute règle,* r *de* G'*, tous les non-terminaux qui apparaissent dans le membre droit de* r *ont au plus un sommet dans chaque sous arbre.*

A présent, il nous est possible de démontrer la proposition 16.

Démonstration. Soit H engendré par une CHR-arbre-grammaire-séparée $G = (X, N, T, \Gamma \cup \left\{A_0 v_1 \cdots v_{\varrho(A_0)}\right\}, R_c)$, où Γ est un arbre *déterministe*, et les arcs non-terminaux de R_c sont placés aux racines d'arbres distincts. D'après le lemme 18, sans perte de généralité, on peut supposer que pour chaque règle, tous les non-termiaux qui apparaissent dans les membres droits y ont au plus un unique sommet par arbre de contexte. De plus au prix de quelques règles supplémentaire, on suppose également que chaque arc terminal, ou couleur terminal est produit entre deux sommets racine du membre droit de la règle associée à un non-terminal.

Dans un premier temps, nous allons caractériser l'ensemble d'arcs qui forme le graphe G^ω, pour les couleurs (les terminaux d'arité 1), l'argument est le même, mais se limite à un seul chemin.

On peut observer que la production de chaque arc terminal ne nécessite que de suivre 2 chemins dans l'arbre Γ. On construit donc un transducteur $T_0 = (Q_0, I_0, F_0, E_0, L_0)$. Pour chaque non-terminal d'arité ϱ supérieur ou égale à 2, on définit un état $q_{(A,a,b)} \in Q_0$ pour chaque couple (a, b) de sommets distincts

du graphe de la règle du non-terminal A. Plus précisément, on définit :

$$Q_0 = \Big\{ q_{(A,a,b)} \mid A \in N \wedge (A\ v_1\ \cdots\ a\ \cdots\ b\ \cdots\ v_{\varrho(A)} \in Dom(R_c) \\ \vee A\ v_1\ \cdots\ b\ \cdots\ a\ \cdots\ v_{\varrho(A)} \in Dom(R_c)) \Big\} \cup \{S_\varepsilon\}$$

avec $Dom(R_c)$ qui représente l'ensemble des membre gauche des règles de R_c. La présence des sommets a et b dans l'indice associé à chaque état est de pouvoir prendre en compte les permutations tout a long du chemin entre la racine et le sommet d'arrivée où est produit l'arc terminal. Lorsque le sommet $q_{A,a,b}$ est atteint dans le transducteur, cela signifie que les sommets respectifs a et b ont été atteint par le chemin gauche et droit dans l'arbre de contexte.

Nous donnons à présent quelques détails supplémentaires sur la construction du transducteur. L'ensemble I_0 possède un unique élément : S_ε. L'ensemble des états finals est ainsi constitué :

$$F = \Big\{ q_{(A,a,b)} \mid q_{(A,a,b)} \in Q_0 \wedge \exists c \in T, c\ a\ b \in R_c(A) \Big\}$$

et où $R_c(A)$ représente le membre droit de la règle associée à A dans R_c. Suivant ce principe, la fonction d'étiquetage associe à chaque état $q_{(A,a,b)}$ de F l'ensemble des arcs terminaux $c \in T$ tels que $c\ a\ b \in R_c(A)$.

À présent, l'ensemble des transitions est défini ainsi : soit A et B deux non-terminaux tels que $B\ v_1\ \cdots\ v_{\varrho(B)} \in R_c(A)$, pour tout couple de sommets distincts (a, b) de A tels qu'il existe un chemin vers le couple (a', b') de sommets de B, on définit la transition $q_{(A,a,b)} \xrightarrow{u/v} q_{(B,a',b')}$, où u est le chemin de a vers a' et v celui de b vers b' (Les suppositions faites en début de démonstration assurent l'unicité de ces chemins lorsqu'ils existent). Les transitions à partir de S_ε sont définies de la même façon : pour chaque couple de sommets (a, b) de A_0 il existe une transition $S_\varepsilon \xrightarrow{u/v} A_{0(a,b)}$ où u est l'étiquette du chemin depuis la racine jusqu'au sommet a, et v celle du chemin jusqu'au sommet b.

Maintenant, supposons que Γ, est Γ_X l'arbre $|X|$-aire complet. Ainsi le contexte de chaque règle peut toujours être satisfait. Dans ce cas, tout chemin dans T_0 représente un couple de chemins dans Γ_X. Et donc, selon la définition de L_0, tout tel couple est étiqueté par le terminal correct.

Lorsque Γ n'est pas un arbre complet, d'après le lemme 17 on dispose d'un ensemble régulier d'ensemble des chemins, posons L_Γ pour cet ensemble et $\mathcal{A}_\Gamma$ l'automate fini déterministe qui le reconnaisse. Pour obtenir exactement le graphe engendré par la grammaire, *il n'est pas suffisant de se restreindre aux sommets de* L_Γ. En effet, si on considère un arc d'arité 3, il est possible qu'un arc soit créé entre le premier et le second sommet, mais si ces deux sommets existent, mais que le troisième n'existe pas, l'arc terminal ne pourra être produit, car le non-terminal ne le sera pas.

Il est possible de résoudre ce problème en effectuant une synchronisation de T_0 avec $\mathcal{A}_\Gamma$. Le nouveau transducteur, T_1, est construit, à partir de T_0 de la façon suivante (Sans pour autant détailler précisément les états où les transitions, il s'agit d'une construction classique). Chaque état de T_1 est un n-uplet composé d'un état de T_0 (par exemple $q_{(A,a,b)}$ et, pour chaque sommet de l'hyperarc A, d'un état de $\mathcal{A}_\Gamma$ qui représente l'avancement dans cet automate sur le chemin qui atteint le sommet en question du non-terminal. Il est capital que l'état permette de suivre tous les sommets et pas simplement les sommets a et b. On peut noter un tel état ainsi : $(q_{(A,a,b)}, q_1, \ldots, q_{\varrho(A)})$. Les transitions sont dérivées de celles de T_0, en ajoutant la mise-à-jour des états de $\mathcal{A}_\Gamma$: si on a $q_{A,a,b} \xrightarrow{u/v} q_{A',a',b'}$ dans T_0, on a la transition :

$$(q_{(A,a,b)}, q_1, \ldots, q_{\varrho(A)}) \xrightarrow{u/v} (q_{(A',a',b')}, \mathcal{A}_\Gamma(q_1, \ldots, q_{\varrho(A)}, A \to A')).$$

où la notation $\mathcal{A}_\Gamma(q_1, \ldots, q_{\varrho(A)}, A \to A')$ représente les états obtenus dans l'automate $\mathcal{A}_\Gamma$, avec les étiquettes des chemins, dans le contexte de la règle de A, pour atteindre les sommets de A' depuis ceux de A (en partant dans $\mathcal{A}_\Gamma$ des états $q_1, \ldots, q_{\varrho(A)}$). Tous les états de T_1 qui ne sont pas composés que d'états

terminaux de $\mathcal{A}_\Gamma$ sont supprimés de Q_1 l'ensemble des états de T_1, les états finals de T_1 sont ceux dont la première composante est un état final de T_0. Le transducteur T_1 reconnais un sous-ensemble de celui reconnu par T_0, il élimine toutes les transitions qui passeraient par des sommets qui n'existent pas dans Γ.

Il reconnaît bien un graphe isomorphe à ceux de G^ω. Ce qui démontre le résultat énoncé. □

Il est maintenant possible de combiner les propositions 16 et 15 en remarquant que la grammaire qui permet de démontrer la proposition 15 est CHR-arbre-grammaire-séparée, on obtient le théorème suivant.

Théorème 19. *La famille des graphes rationnels et celle des graphes engendrés par les* CHR*-arbre-grammaires-séparées coïncident.*

Il est également possible de modifier très légèrement les règles présentés dans la preuve de la proposition 13 pour l'étendre : les règles r_0 et r_1 peuvent engendrer un axiome, et la seule règle qui ne soit pas satisfaisante pour les CHR-arbre-grammaires est $r_{\mathbf{chk}}$. Cette remarque permet d'établir le résultat qui suit.

Proposition 20. *Les* CHR*-arbre-grammaires engendrent des graphes non-récursifs.*

La proposition 20 établit en quelque sorte que les CHR-arbre-grammaires-séparées sont les systèmes contextuels de récriture de graphe les plus généraux qui capturent les graphes rationnels et rien de plus.

Applications

Il est à présent possible de reformuler le théorème 12 pour les CHR-arbre-graphes-séparés.

Théorème 21 ([118])**.** *L'ensemble des étiquettes des chemins entre deux couleurs dans les* CHR*-arbre-graphes-séparés coïncide avec les langages contextuels.*

L'intérêt majeur d'avoir une caractérisation par automates (y compris infinis) des langages est de pouvoir réaliser des démonstrations plus simples. Nous allons donc présenter quelques applications de notre approche pour les langages contextuels. On définit les opérations de concaténation et d'itération pour des graphes, non-plus comme conduit page 2, mais avec l'objectif de réaliser ces opérations pour les langages qu'ils reconnaissent.

Ainsi, la concaténation relative à deux couleurs (c_1 et c_2) consiste à relier les prédécesseurs des sommets colorés par c_1 dans le premier graphe à ceux colorés c_2 dans le second. L'itération remplit la même fonction pour un graphe, vis-à-vis des couleurs *sommet final* et *sommet initial*.

Proposition 22. *Soient G_1 et G_2 deux* CHR*-arbre-graphes-séparés : l'itération de G_1, la concaténation, et le produit de synchronisation de G_1 et G_2 sont des* CHR*-arbre-graphes-séparés.*

Démonstration. Pour la concaténation, on suppose que les graphes possèdent deux ensembles distincts de sommets. On pose c_1 et c_2 pour les couleurs vis-à-vis desquelles la concaténation va être faite. Dans chaque règle on va ajouter un marquer qui permet de situer la position de éventuelle de c_1 de façon à repérer tous les sommets pouvant avoir cette couleur. Ensuite à chaque fois qu'un arc a pour cible un sommet étiqueté par c_1 (que cette couleur soit produite à ce moment là ou pas), on crée un nouveau non-terminal qui aura marquera le sommet source, et suivra un parcours à partir de la racine de l'axiome de G_2, pour créer un arc vers chacun des sommets coloré c_2.

Le processus pour la concaténation est identique.

Pour le produit, le construction est plus subtile, elle nécessite de réaliser une sorte de codage des couples de sommets. En pratique, étant données deux graphes G_1 et G_2 engendrés à partir des axiomes H_1 et H_2, et des ensembles de règles contextuelles R_1 et R_2. Chaque couple de sommet (u_1, u_2), sommet de $G_1 \times G_2$, se trouve dans un arbre $H_1 \# H_2$. Cet arbre est formé par l'arbre H_1, et chaque sommet de cet arbre possède un arc supplémentaire étiqueté par $\# \notin X$, le sommet atteint par ces arc est la

racine d'un arbre isomorphe à H_2. Ainsi, le couple (u_1, u_2) est associé à l'unique sommet de $H_1 \# H_2$ atteint par le chemin $u_1 \# u_2$. À présent, l'ensemble des règle permettant de produire $G_1 \times G_2$ sont en premier lieu les règles de R_1, modifiées de façon à produire la configuration initiale de R_2 au lieu de produire un arc terminal (par exemple a). Les modifications des règles de R_2 sont plus complexes : pour chaque non-terminal A d'arité n, on produit une règle pour chaque symbole a de Σ, on définit un nouveau symbole non-terminal $(A - A)_a$ d'arité double de celle de A. La règle pour $(A - A)_a$ est dérivée de celle de A par duplication, une partie gauche qui agit comme A, pour déterminer la source de l'arc, et la partie droite qui détermine la cible. De par son synchronisme, cette règle produirait, éventuellement, simultanément un arc a pour chacune des deux parties. la production effective produirait un arc entre la source contenue dans la partie gauche, et le but dans la partie droite.

□

Pour conclure ces applications, le théorème 21, conjugué à la proposition 22 permet de démontrer le résultat suivant.

Corollaire 23. *Les* CSL *sont clos par concaténation et étoile de Kleene ; l'intersection de deux* CSL *est un* CSL.

Un des résultat les plus étonnant pour les CSL est que ces langages sont clos par l'opération de complémentation [139, 85]. Il semblerait naturel de pouvoir démontrer ce résultat en utilisant une approche *automate*. Cependant, pour réaliser une telle démonstration, il est en général nécessaire d'avoir accès à un modèle déterministe de graphe (pas toujours comme pour les automates de Büchi). Ici, on sait que les langages reconnus par les CHR-arbre-graphes-séparés sont des CSL déterministes. Mais nous conjecturons que certains CSL déterministes ne peuvent être reconnus par des CHR-arbre-graphes-séparés. Et même si tous ces langages étaient obtenus par ces automates déterministes, il serait encore nécessaire d'établir la conjecture de Kuroda [97], sur l'équivalence des CSL déterministes et non-déterministes, pour obtenir la clôture par complémentation.

III.3 Arbres rationnels

Dans ce paragraphe, nous allons présenter les *arbres rationnels*. Ici, ce terme est entendu au sens des graphes rationnels qui sont également des arbres. Ces résultats pour la plupart été présentés dans un travail conduit avec Carayol [30]. Les résultats liés à la logique du premier ordre qui y sont présentés seront exposés dans le chapitre suivant.

Les arbres sont des structure naturelles qui sont utilisés de façon très fréquente en informatique. De nombreuses classes d'arbres ont été étudiés sans même les aborder sous l'angle d'un graphe. Les arbres réguliers, par exemple, sont les arbres qui sont ceux qui ont un nombre fini de sous-arbre à isomorphisme près. Les arbres algébriques, qui sont les dépliages des graphes réguliers (voir Caucal [36]). Des familles encore plus générales : les termes qui sont solution des schémas de programme d'ordre supérieurs [50, 31, 94].

Définition 7. Un *arbre rationnel* est un graphe rationnel qui satisfait les propriétés suivante :

(i) connexité ;

(ii) co-fonctionalité (tout sommet est cible d'au plus un arc) ;

(iii) possède un unique sommet de degré entrant 1 (appelé *racine*).

Chaque sommet d'un arbre est appelé *nœud*. Les *feuilles*sont les sommets qui ne sont source d'aucun arc ;

Résultats élémentaires

Les propriétés *(ii)* et *(iii)* de la définition 7 sont simple à vérifier : *(ii)* il suffit de vérifier que la relation $\bigcup_{a\in\Sigma}(\xrightarrow{a})^{-1}$ est fonctionnelle. Pour réaliser cette vérification, on utilise le théorème de Shutzenberger, voir, entre autres, le livre de Berstel [13]. La condition *(iii)* est testée en vérifiant que l'ensemble régulier $Dom(T) \setminus Im(T)$ ne possède qu'un élément.

Pour démontrer que le problème de l'arborescence pour un arbre rationnel est indécidable on utilise de nouveau une variante du problème d'arrêt uniforme pour les machines de Turing.

Proposition 24. *Étant donnée une machine de Turing déterministe M il est possible de construire M' telle que M s'arrête sur ε si et seulement si M' s'arrête à partir de n'importe quelle configuration.*

La machine M' est construite comme suit. M' possède deux modes : vérification ou calcul. Lorsqu'elle est en mode vérification, elle s'assure que la machine ne possède sur sa bande que des configurations successives de M. Au cours de la vérification, si elle détecte une erreur, elle s'arrête. Le mode calcul consiste simplement à ajouter une configuration de plus à la fin de la bande. Chaque configuration est délimitée par deux symboles spécifiques. Ainsi pour qu'une telle machine ait une suite infinie de calcul il faut et il suffit que la machine M en possède une à partir du mot vide.

Proposition 25. *Étant donnée une machine de Turing déterministe M, il est possible de construire un graphe rationnel (non-étiqueté), G_M tel que M s'arrête à partir de n'importe quelle configuration si et seulement si $G(M)$ est un arbre.*

Démonstration. Considérons la machine de Turing déterministe $M = (Q, T, \delta, q_0)$, où Q est l'ensemble d'états (avec $q_0 \in Q$ état initial), T est l'ensemble des symboles de bande (avec les deux symboles spéciaux \$ et $\#$ qui délimitent les extrémités de la bande) et $\delta : Q \times T \to Q \times T \times \{l, r, p\}$ est la fonction de transition.

Comme habituellement, les configurations sont notées : uqv, avec $q \in Q, u \in \$(T + \square)^*, v \in (T + \square)^*\#$, ($\square$ désigne une case vide).

Les sommets de $G(M)$ sont ainsi définis : ce sont les configurations de la machine, ainsi que le sommet $\$\#$.

Les *arcs* sont les transitions de la machine inversés (pour assurer la co-fonctionnalité du graphe), on ajoute également les arcs de cet ensemble : $\{\$\#\} \times \{\$uqAv\# \mid (q, A) \notin Dom(\delta) \wedge u, v \in (T + \square)^*\}$.

Le sommet $\$\#$ est le seul à n'être la cible d'aucun arc (condition *(iii)*), et comme la machine est déterministe, que les arcs sont inversés par rapport aux transitions, ce graphe satisfait la condition *(ii)*. Ce graphe est connexe, si et seulement si la machine M atteint, à partir de toute configuration, une configuration où plus aucune transition n'est possible. □

En conjugant les propositions 24 et 25, on obtient l'indécidabilité de l'arborescence d'un graphe rationnel.

Proposition 26. *Il est indécidable de savoir si un graphe rationnel est un arbre.*

Avant d'énoncer un résultat sur l'accessibilité dans les arbres rationnels. Nous rappelons la définition de *l'accessibilité rationnelle*. Étant donné un graphe G, étiqueté sur Σ, et un langage rationnel $L \subseteq Rat(\Sigma^*)$, on dit que deux sommets v et w de G sont *accessibles rationnellement par L* si il existe un mot $u \in L$ tel que $v \xrightarrow[G]{u} w$.

Proposition 27. *Les problèmes d'accessibilité et d'accessibilité rationnelle sont décidables pour les arbres rationnels.*

Démonstration. On observe d'abord que si deux sommets sont accessibles dans un arbre il existe un unique chemin (orienté) dont ils sont les extrémités. Soient u et v deux sommets d'un arbre rationnel, et soit T le transducteur qui reconnaît cet arbre.

On calcule successivement $(T^{-1})^n(u)$ et $(T^{-1})^n(v)$ jusqu'à obtenir la racine, v ou u. En cas de succès on obtient un chemin pour connecter les deux sommets.

Pour vérifier l'accessibilité rationnelle, le processus est le même et ultimement on vérifie que le chemin appartient aux langage rationnel voulu. □

Le 2^n-arbre

Nous donnons à présent un premier exemple d'arbre rationnel. Cet arbre est en fait automatique. Il est définit précisément par une demi-droite d'arcs étiquetés pas a, et après n arcs étiquetés a le nœud est également racine d'une séquence de 2^n arc étiquetés par b.

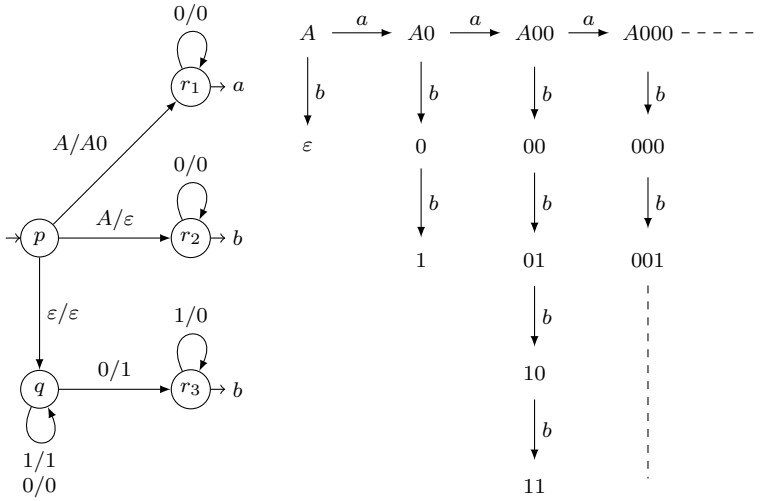


FIGURE 2.7 – 2^n-arbre avec le transducteur qui le reconnaît.

Le codage des sommets de cet arbre repose sur le fait qu'il y a précisément 2^n n-uplets sur $\{0, 1\}$. Le transducteur réalise un incrément de 1.

Un arbre rationnel non-automatique

A présent nous présentons un arbre de degré fini, mais non-borné qui est rationnel, mais non-automatique, à isomorphisme près.

Cet arbre est obtenu à partir d'une forêt rationnelle en ajoutant une ligne qui connecte les racines de chacune des composantes connexes. Comme ces racines constituent un ensemble rationnel de mot, le lemme qui suit permet de réaliser une telle ligne.

Lemme 28. *Étant donné un langage rationnel L, le graphe formé exclusivement des mots de L connectés sous forme d'une demi-droite, en ordre militaire*[3]*, constitue un graphe automatique.*

Démonstration. On note $<_{\text{mil}}$ l'ordre militaire *strict* sur Σ^*. Le graphe $G_{\text{mil}} := \{(u, v) \mid u <_{\text{mil}} v\}$ est automatique.

Les opérations d'intersection et de complémentation préservent l'automaticité. Ainsi les graphes suivants sont automatiques.

$$G_1 := G_{\text{mil}} \cap L \times L$$

3. L'ordre militaire consiste à ordonner les éléments d'un monoïde libre, par longueur, puis suivant l'ordre lexicographique pour les mots de même longueur.

$$(G_1)^2 := \left\{ u \to v \mid \exists w, u \xrightarrow[G_1]{} w \wedge w \xrightarrow[G_1]{} v \right\}$$
$$G := G_1 \setminus (G_1)^2 \quad (= \overline{(G_1)^2} \cap G_1)$$

Soit (u, v) un élément de G, par définition, $u <_{\text{mil}} v$ de plus, il n'y a aucun w de L tel que $u <_{\text{mil}} w <_{\text{mil}} v$ (car (u, v) n'est pas dans $(G_1)^2$. Comme la réciproque est également vraie, G est précisément le graphe recherché. □

L'exemple qui suit s'appuie sur la limite de croissance des graphes automatiques de degré fini. Pour un tel arbre automatique, un simple argument de dénombrement établit que il existe des entiers p, q et s tels qu'il y ait au plus p^{qn+s} sommets à distance n de la racine.

Exemple 7. On appelle *simplexp* l'arbre dont chaque sommet de profondeur n a précisément 2^n fils, il y a donc, au total $2^{n(n-1)/2}$ sommets de profondeur n, ainsi ce graphe n'est pas automatique.

Il n'est pas clair que simplexp soit un graphe rationnel. Cependant, le transducteur présenté sur la figure 2.8 définit une forêt dont la composante connexe du sommet ε est simplexp. Ainsi, en utilisant le lemme 28 on construit un arbre de même croissance, et qui n'est donc pas automatique, à isomorphisme près. Pour des question de simplicité, la figure 2.8 ne réalise que le transducteur de la forêt.

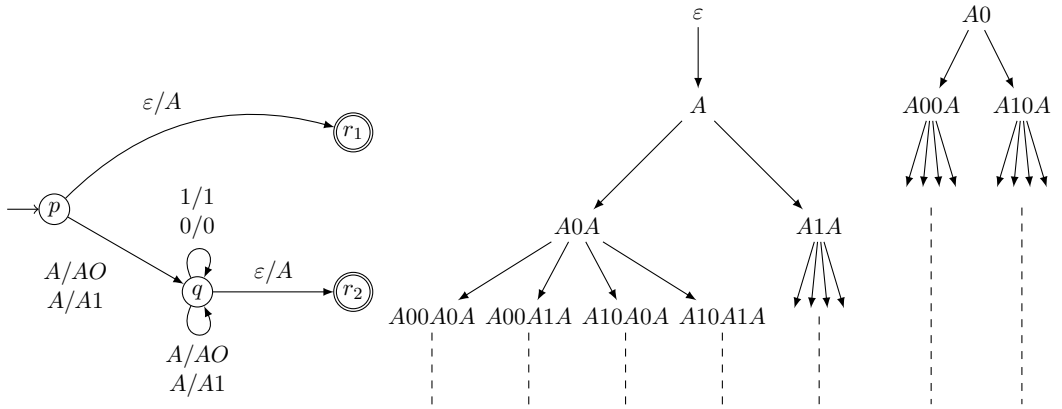


FIGURE 2.8 – Transducteur de la forêt contenant simplexp, ainsi qu'une partie de la forêt.

On observe que les sommets de profondeur n ont précisément n occurrence du symbole A, et possède donc 2^n fils. De plus, ce transducteur, et co-fonctionnel, et strictement croissant, donc chaque composante connexe est un arbre (avec une racine).

Avec le transducteur qui relie toutes les racines, on obtient donc un arbre rationnel qui n'est pas automatique, à isomorphisme près.

Dans la première section du chapitre 3, nous dessinerons avec précision les frontières de la décidabilité des théories logiques des arbres rationnels.

IV Graphes d'accessibilité des réseaux de Petri

Dans ce paragraphe nous en définissons dans un premier temps les réseaux de Petri, puis nous définissons la semi-linéarité.

IV.1 Réseau de Petri et leurs graphes

Nous résumons, ici, quelques éléments fondamentaux sur les réseaux de Petri, ainsi que sur les ensembles semi-linéaires d'entiers. Nous rappelons également quelques un des résultats utiles pour la suite.

Un *réseau de Petri* est un graphe bi-partite noté $N = (P, T, F, M_0)$, où :
— P et T sont des ensembles finis respectifs de *places* et de *transitions* (avec $P \cap T = \emptyset$) ;
— $F : (P \times T) \cup (T \times P) \to \mathbb{N}$;
— $M_0 : P \to \mathbb{N}$.
F représente l'ensemble des arêtes orientées, ces dernières portent des poids entiers, toute fonction $M : P \to \mathbb{N}$ définit un *marquage* du réseau, et M_0 est un marquage distingué, appelé *marquage initial* de N.

Une transition $t \in T$ est *tirable* pour un marquage M, noté $M[t\rangle$, si $M(p) \geq F(p,t)$ pour toutes les places $p \in P$. Si t est tirable pour M alors elle peut être *tirée*. Le marquage M' atteint, par t, depuis M, est défini ainsi : $M'(p) = M(p) + F(t,p) - F(p,t)$ pour tout $p \in P$. Le tirage d'une transition est noté de la façon suivante $M[t\rangle M'$. Cette définition est étendue au suites de transitions $s \in T^*$ de la façon naturelle. Un marquage M' est *accessible* depuis un marquage M dès lors qu'il existe $M[s\rangle M'$ pour une suite $s \in T^*$ quelconque. Une transition t est *en auto-boucle* avec une place p si et seulement si $F(p,t) = F(t,p) > 0$. Une transition est *neutre* si elle n'a aucun effet sur aucune place. L'*ensemble d'accessibilité* $\mathrm{Reach}(N)$ du réseau N est l'ensemble de tous les marquages accessibles depuis le marquage initial.

Théorème 29. *[95, 109] Étant donné un réseau de Petri N, deux marquages M et M', on peut décider si M' est accessible depuis M.*

On peut formuler ainsi les problèmes respectifs d'égalité, $\mathrm{Eg}_{\mathrm{reach}}(PN)$, et d'inclusion, $\mathrm{Inc}_{\mathrm{reach}}(PN)$, des ensembles de marquages :

Donnée : Deux réseaux de Petri, N et N'
Question : $\mathrm{Reach}(N) = \mathrm{Reach}(N')$ (*resp.* $\mathrm{Reach}(N) \subseteq \mathrm{Reach}(N')$)

Théorème 30. *[10, 73] Les problèmes $\mathrm{Eg}_{\mathrm{reach}}(PN)$ et $\mathrm{Inc}_{\mathrm{reach}}(PN)$ sont indécidables.*

Une version plus forte du théorème 30 a été établie dans le travail de Jančar [86]. Il y est démontré que ces problèmes sont indécidables dès que les réseaux ont au moins cinq places, même lorsqu'un des deux réseaux est fixé (pour un réseau particulier).

Il est possible de définir plusieurs structures relationnelles à partir d'un réseau de Petri $N = (P, T, F, M_0)$. On pourra ensuite interpréter la logique FO sur l'une de ces structures.

Le *graphe d'accessibilité non-étiqueté* d'un réseau N est défini par la structure $\mathrm{URG}(N) = (D, init, \to, \xrightarrow{*}, \xrightarrow{+}, =)$ où $init = \{M_0\}$, et les relations respectives $\xrightarrow{*}$ et $\xrightarrow{+}$ sont les clôtures transitive, respectivement transitive stricte de la relation $\to$. Cette structure correspond à l'idée intuitive du graphe de réseau de Petri où le marquage initial est singularisé.

On appelle *graphe d'accessibillité non-étiqueté, ordinaire* d'un réseau N la structure $\mathrm{PURG}(N) = (D, \to)$ où $D = \mathrm{Reach}(N)$ et $\to$ est la relation binaire sur D induite par la relation de tirage : $M \to M'$ lorsqu'on a $M[t\rangle M'$ pour une transition $t \in T$. On notera que $M_0 \in D$ mais, à priori, aucun prédicat ne distingue ce marquage dans D. Cette structure, bien que définie à partir d'un marquage initial, ne permet par d'isoler ce marquage, et donc bride l'expressivité des logiques que nous considérons.

Le *graphe des transitions non-étiqueté* de N est constitué de la structure $\mathrm{UG}(N) = (\mathbb{N}^P, init, \to, \xrightarrow{*}, \xrightarrow{+}, =)$ où $M \to M'$ lorsque $M[t\rangle M'$ pour une transition $t \in T$ quelconque. On peut observer que l'accessibilité n'est pas prise en compte dans la définition de la structure $\mathrm{UG}(N)$. Cette dernière structure repose simplement sur la définition des transitions du réseau. En particulier, contrairement à $\mathrm{PURG}(N)$ et $\mathrm{URG}(N)$ elle n'est pas affecté (hormis pour le prédicat $init$) par le marquage initial de N.

Ces trois structures sont importante pour la suite du document, elle définiront la nature des objets sur lesquels seront interprétées les logiques.

Dans la suite, par défaut, $\mathrm{card}(P) = n$ et on identifie les ensembles $\mathbb{N}^P$ et $\mathbb{N}^n$. On appelle également 1-*boucle* une transition $M \to M'$ avec $M = M'$.

IV.2 Réseaux de Petri et ensembles semi-linéaires

Il est possible de voir l'ensembles des marquages d'un réseau de Petri à n places comme un sous-ensemble de $\mathbb{N}^n$. Ceci nous conduit à considérer des sous-ensembles intéressants que sont les parties semi-linéaires. La structure $(\mathbb{N}^n, +)$ est un monoïde commutatif où l'opération produit est l'addition composante par composante des n-vecteurs et l'élément neutre est le n-vecteur nul.

Un sous-ensemble $E \subseteq \mathbb{N}^n$ est dit *linéaire* s'il est possible de le définir ainsi $x + \{y_1, \ldots, y_m\}^*$ avec des vecteurs $x \in \mathbb{N}^n$ et $y_1, \ldots, y_m \in \mathbb{N}^n$. L'étoile de Kleene, $\{y_1, \ldots, y_m\}^*$, est une abréviation pour $k_1 y_1 + \ldots + k_m y_m$ pour un ensemble de valeurs $k_1, \ldots, k_m \in \mathbb{N}$. Une partie $E \subseteq \mathbb{N}^n$ est *semi-linéaire* si elle est égale à une union finie d'ensembles linéaires. Du fait de la commutativité de l'opération $+$, les parties semi-linéaires de $\mathbb{N}^n$ en sont également les parties rationnelles. Elles sont donc reconnues par les automates sur $\mathbb{N}^n$. Il est même possible de se limiter aux automates étiquetés par les générateurs, autrement dit, les n-vecteurs qui possèdent exactement une composante non-nulle, égale à 1. Les ensembles semi-linéraires de $\mathbb{N}^n$ forment une algèbre de Boole, effective [68], il est donc possible, par exemple de décider du vide d'une telle partie. Dans un article de 1966, Ginsburg et Spanier ont donné une correspondance effective entre les ensembles semi-linéaires et les ensembles définissables dans $\mathbb{N}^n$ en arithmétique de Presburger, [69]. L'arithmétique de Presburger peut être décidée en temps triplement exponentiel [19].

Proposition 31. *Étant donné un réseau de Petri $N = (P, T, F, M_0)$ et un ensemble semi-linéaire de marquages $E \subseteq \mathbb{N}^n$, on peut décider si un marquage de E est atteint depuis M_0.*

En 1976, Hack avait réduit le problème d'accessibilité d'un ensemble semi-linéaire à celui de l'accessibilité des réseaux de Petri, [73, Lemme 4.3]. La proposition 31 est une conséquence du résultat de Hack, et du théorème 29. On déduit de la proposition 31 que pour un marquage $M \in \mathbb{N}^{|P|}$ quelconque, on peut décider si un marquage supérieur ou égal à M est accessible.

Déterminer si un réseau de Petri a un ensemble d'accessibilité semi-linéaire est décidable (il existe deux preuves de ce résultat, l'une de Hauschildt [75], la seconde de Lambert, présenté dans sa thèse, aucune des deux n'a été complètement validé par le processus classique de publication). On peut observer que la semi-linéarité de l'ensemble d'accessibilité, $\mathrm{Reach}(N)$, n'induit pas celle de la *relation* d'accessibilité $\xrightarrow{*} \subseteq \mathrm{Reach}(N) \times \mathrm{Reach}(N) \subseteq \mathbb{N}^{n+n}$. Voici quelques exemples de classes de réseaux de Petri pour lesquelles la relation $\xrightarrow{*}$ est de façon effective semi-linéaire :

— les réseaux cycliques, voir, par exemple, [4, 21, 103].
— les réseaux de Petri sans communication [57].
— les systèmes d'addition de vecteurs à états [4] de dimension 2 [80, 104].
— les réseaux *Single-path* [82].
— réseaux de Petri régulliers [147].
— Les systèmes à compteur affines, plats, avec la propriété du monoïde fini [18, 62].
— Les systèmes à compteurs plats, relationnels [47, 22].
— Les systèmes à compteurs *reversal-bounded* [84].

Certains de ces résultats sont complexes, néanmoins, la semi-linéarité est un outil puissant pour les preuves de décidabilité.

4. les systèmes d'addition de vecteurs à états sont un formalisme équivalent aux réseaux de Petri.

Chapitre 3

Résultats de logique

Ce chapitre contient les résultats d'un ensemble de travaux réalisés avec Carayol d'une part, et Darondeau, Demri et Meyer d'autre part [30, 51, 52]. Ils portent sur la décidabilité du problème de *model-checking* de la logique du premier ordre, et de certaines extensions et restrictions pour les arbres rationnels et les graphes d'accessibilité des réseaux de Petri.

I Logique du premier ordre et arbres rationnels

Dans ce paragraphe nous examinons la décidabilité du *model-checking* de la logique du premier ordre pour les graphes rationnels qui sont des arbres. Nous considérons aussi plusieurs extensions possibles pour ce résultat, plus précisément, nous envisageons l'extension aux graphes qui ne possèdent pas de circuits, et aussi une logique du premier ordre étendu d'un prédicat d'accessibilité.

I.1 Théorème de Gaifman pour les graphes

Tout graphe étiqueté par Σ est une Σ-structure. On dit qu'une structure est arborescente lorsque son graphe sous-jacent est un arbre. Pour toute structure arborescente, $\mathcal{T}$, on écrit $r(\mathcal{T}) \in T$ pour désigner sa racine. Pour tout $u \in T$, on écrit $\mathcal{T}_{/u}$ pour désigner le sous-arbre de $\mathcal{T}$ dont la racine est u ; pour tout entier $n \geq 0$, $\mathcal{T}^n_{/u}$ est le sous-arbre de $\mathcal{T}_{/u}$ restreint aux nœuds de profondeur inférieure à n.

Le théorème de Gaifman énonce que toute formule du premier ordre sur un graphe est logiquement équivalente à une formule dite *locale*. Pour définir ces formules locales, il est d'abord nécessaire de disposer de la notion de *distance*. Dans la suite, on écrit $d(x, y) \leqslant n$ (*resp.* $d(x, y) < n$) pour représenter la formule du premier ordre qui exprime la distance, sans prendre en compte l'orientation des arcs, entre x et y inférieure ou égale à n (*resp.* strictement inférieure à n).

On note par $S(r, x)$ la boule de rayon r centrée sur x : $\{y \mid d(x, y) \leqslant r\}$

On restreint une formule $\varphi(x)$ à la boule de rayon r centrée en x, notée $\varphi^{S(r,x)}$. Cette notation est définie en renommant chaque occurrence liée de x dans φ par une nouvelle variable, et en localisant chaque quantification à x :

$$[\exists z \varphi]^{S(r,x)} := \exists z (d(x, z) \leqslant r \wedge \varphi^{S(r,x)})$$

Une formule locale de base est de la forme suivante :

$$\exists x_1 \ldots \exists x_n \bigwedge_{1 \leqslant i < j \leqslant n} (d(x_i, x_j) > 2r \wedge \psi^{S(r,x_i)}(x_i))$$

Un énoncé local est une combinaison booléenne de formules locales de base.

Théorème 32 (Gaifman)**.** *Tout énoncé du premier ordre est logiquement équivalent à un énoncé local.*

Notez bien que l'équivalence affirmée par ce théorème est effective.

I.2 Résultats compositionnels pour les arbres

On s'intéresse maintenant aux principaux résultats compositionnels pour les arbres qui vont nous permettre de caractériser les centres des boules impliquées dans la définition des formules locales de base.

La méthode compositionnelle est une technique puissante pour établir la décidabilité de logiques. Elle a été principalement développée par Shelah [138] (Le lecteur intéressé peut aussi se référer aux travaux de Rabinovich et de Zeitman pour un aperçu global [128, 151]). Les résultats présentés ici pourraient être dérivés directement de canevas généraux pour la méthode compositionnelle [128, 151].

Pour chaque structure arborescente $\mathcal{T}$ sur la signature $\Sigma = \{E_1, \ldots, E_\ell\}$ et pour tout $k \geq 1$, on définit *l'arbre réduit de* $\mathcal{T}$, il s'agit de la structure $< \mathcal{T} >_k$ sur la signature monadique $< \Sigma >_k \stackrel{\text{def}}{=} \{S_1, \ldots, S_\ell\} \cup \{P_M\}\, M \in \mathrm{Thm}_k^\Sigma$. L'univers de $< \mathcal{T} >_k$ est l'ensemble des successeurs de la racine de $\mathcal{T}$. Les prédicats dans $< \Sigma >_k$ sont interprétés de la façon suivante : pour tout $i \in [\ell]$, $u \in S_i^{<\Sigma>_k}$ si et seulement si $(r(\mathcal{T}), u) \in E_i^{\mathcal{T}}$ et pour tout $M \in \mathrm{Thm}_k^\Sigma$, $u \in P_M$ si et seulement si $\mathrm{Thm}(\mathcal{T}_{/u}) = M$.

Exemple 8. La figure 3.1 représente un arbre et son équivalent réduit.

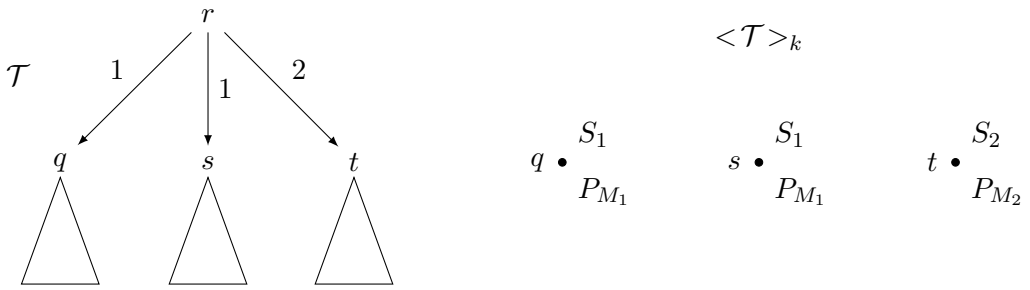


FIGURE 3.1 – Arbre $\mathcal{T}$, et son arbre réduit $< \mathcal{T} >_k$.

L'arbre représenté sur la gauche de la figure 3.1 est définit sur $\Sigma = \{E_1, E_2\}$, l'arbre réduit $< \mathcal{T} >_k$ est définit sur $< \Sigma >_k \stackrel{\text{def}}{=} \{S_1, S_2\} \cup \left\{ P_M \mid M \in \mathrm{Thm}_k^\Sigma \right\}$; et $\mathrm{Thm}_k(\mathcal{T}_{/q}) = \mathrm{Thm}_k(\mathcal{T}_{/s}) = M_1, \mathrm{Thm}_k(\mathcal{T}_{/t}) = M_2$

Lemme 33. *Pour toute structure arborescente $\mathcal{T}$ sur $\Sigma = \{E_1, \ldots, E_\ell\}$ et tout $k \geq 1$, $\mathrm{Thm}_k(\mathcal{T})$ peut être calculé de façon effective à partir de $\mathrm{Thm}_k(< \mathcal{T} >_{k+1})$.*

Remarque 4. Comme la signature de $< \mathcal{T} >_k$ est monadique, toute formule est équivalente à une combinaison booléenne de formules de la forme suivante :

$$\exists x_1, \ldots, x_\ell \bigwedge_{i \neq j} x_i \neq x_j \ \wedge \bigwedge_{P \in R, i \in [\ell]} P(x_i) \ \wedge \bigwedge_{P \notin R, i \in [\ell]} \neg P(x_i).$$

où $P \subseteq < \Sigma >_k$. voir, par exemple le livre de cours de Ebbinghaus *et al.* [54, Exercice 2.3.12].

Le lemme suivant permet de calculer la théorie d'une boule dans un arbre à partir des théories de certains des sous-arbres de cette boule.

Lemme 34. *Pour tout arbre $\mathcal{T}$ de signature $\Sigma = \{E_1, \ldots, E_\ell\}$ et tout sommet $u \in T$ atteint par un chemin $u_0 a_1 u_1 \ldots u_m$ (avec $u_m = u$) depuis la racine de T et pour tout rang $k \geq 1$ et toute profondeur $n \geq 0$, il existe une constante p calculable de façon effective à partir de m, n et k telle que toute formule $\varphi(x)$ avec $\mathrm{qr}(\varphi) = k$, il est possible de décider si $T \models \varphi^{S(n,x)}[u]$ à partir de la suite d'étiquettes $a_1 \ldots a_m$ et à partir de $(\mathrm{Thm}_p(< \mathcal{T}^n_{/u_i} >_p))_{i \in [0,m]}$.*

I.3 Théorie du premier ordre des arbres rationnels

A présent nous abordons la démonstration de la décidabilité des théories du premier ordre des arbres rationnels.

Pour ce faire, on s'appuie d'abord sur les résultats compositionnels pour les arbres rappelés précédemment pour établir que pour tout $r \geq 1$ et pour toute formule $\varphi(x)$, l'ensemble des centres des boules de rayon r qui satisfont $\varphi(x)$ (où x est interprété comme le centre de la boule) est un langage rationnel.

Ensuite, on établit que l'ensemble des racines des sous-arbres d'une profondeur fixée ayant une k-théorie donnée forme un ensemble rationnel de mots. Pour pouvoir appliquer le lemme 33, il est d'abord nécessaire d'établir le lemme suivant.

Lemme 35. *Pour tout arbre rationnel T, étiqueté par Σ et sur X^*, et pour tout $i \in \Sigma$ et $L \in \mathbb{Q}(X^*)$, l'ensemble des $u \in \mathrm{Dom}(T)$ ayant au moins ℓ successeurs par i dans L est rationnel et constructible.*

Schéma. La démonstration repose sur le fait que le degré entrant de tous les sommets d'un arbre est au plus 1. On applique l'uniformisation des relations rationnelles [56, 13] qui énonce que pour tout transducteur[1] H il existe un transducteur fonctionnel $\overrightarrow{H}$ tel que $\overrightarrow{H} \subseteq H$ et $\mathrm{Dom}(H) = \mathrm{Dom}(\overrightarrow{H})$. Comme le degré entrant de T est au plus 1, si on restreint H_i (le transducteur qui accepte les i-arcs de T restreint en image à L) à l'ensemble rationnel $X^* \setminus \Im(\overrightarrow{H_i})$ on obtient un transducteur H_i', ce dernier a diminué, précisément de 1, le degré sortant de H_i. Ainsi l'ensemble des sommets ayant au moins deux successeurs par i sont $\mathrm{Dom}(H_i')$. La fin de la démonstration repose sur une simple récurrence. □

Remarque 5. Il est important de souligner que ce résultat ne tient pas lorsque le degré entrant est plus grand que 1.

Le transducteur H représenté sur la figure 3.2 l'ensemble des sommets ayant précisément 1 image est formé des mots ayant le même nombre de a et de b. Ce dernier n'est naturellement pas rationnel.

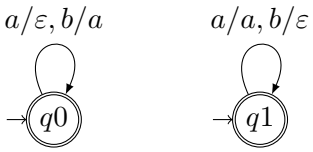


FIGURE 3.2 – Transducteur H.

Lemme 36. *Pour tout arbre rationnel T, étiqueté par $\Sigma = [\ell]$, tout $k \geqslant 1$, $n \geqslant 1$, et tout énoncé φ sur la signature $\Sigma = \{E_1, \ldots, E_\ell\}$ ou $<\Sigma>_k$, les ensembles suivants sont rationnels, de façon effective :*

— $L_\varphi^{n,k} \stackrel{def}{\equiv} \{\, u \in \mathrm{Dom}(T) \mid <\mathcal{T}_{/u}^n>_k \models \varphi \,\}$

— $L_\varphi^{n} \stackrel{def}{\equiv} \{\, u \in \mathrm{Dom}(T) \mid \mathcal{T}_{/u}^n \models \varphi \,\}$

Schéma. On démontre simultanément les deux propriétés par récurrence sur la profondeur n.

Pour le cas de base, $n = 0$, on remarque que pour tout arbre rationnel T, T^0 est réduit à un unique sommet, et pour tout entier $k \geqslant 1$, $<T^0>_k$ est vide. Comme ces structures sont finies, on peut décider pour toute formule φ si elle est satisfaite sur la structure. Ensuite, on établit que L_φ^0 et $L_\varphi^{0,k}$ sont soit l'ensemble vide, soit $\mathrm{Dom}(T)$.

Pour l'étape de récurrence $n+1$: soit l'entier $k \geqslant 1$ un rang et φ un $<\Sigma>_k$-énoncé. D'après la remarque 4, on peut se limiter aux formules qui expriment l'existence d'au moins m éléments dans $S_i^{<T>_k}$ et $P_M^{<T>_k}$ pour une certaine valeur de $i \in [\ell]$ et $M \in \mathrm{Thm}_k^\Sigma$.

1. On fait volontairement l'amalgame entre le transducteur et la relation qu'il reconnaît.

Soit l'entier $m \geqslant 0$, $i \in [\ell]$, $M \in \mathrm{Thm}_k^\Sigma$ et ψ la formule correspondante. D'après l'hypothèse de récurrence, l'ensemble de sommets $X := \left\{ u \in \mathrm{Dom}(T) \mid \mathrm{Thm}_k(\mathcal{T}^n_{/u}) = M \right\}$ est rationnel et calculable. Il est simple de vérifier que pour tout élément $u \in \mathrm{Dom}(T)$, la structure $<T^{n+1}_{/u}>_k$ vérifie ψ si et seulement si u possède m i-successeurs dans X. D'après le lemme 35, l'ensemble $L^{n+1,k}_\psi$ est aussi rationnel.

La seconde propriété se déduit du lemme 33. □

Le résultat suivant est une conséquence des lemmes 34 et 36 :

Lemme 37. *Pour tout arbre rationnel T étiqueté par $\Sigma = [\ell]$, toute formule $\varphi(x)$ sur $\Sigma = \{E_1, \ldots, E_\ell\}$ et tout entier $n \geqslant 1$, l'ensemble $\left\{ u \in \mathrm{Dom}(T) \mid \mathcal{T} \models \varphi^{S(n,x)}[u] \right\}$ est rationnel et calculable de façon effective.*

Avant de pouvoir appliquer le théorème de Gaifman une dernière propriété des arbres rationnels est nécessaire.

Lemme 38. *Pour tout arbre rationnel T dont les sommets sont dans X^*, $L \subseteq \mathrm{Dom}(T) \in Rat(X^*)$ et pour tout $r \geq 1$, on peut décider de l'existence d'éléments $u_1, \ldots, u_m \in L$ tels que pour tout entier $i \neq j \in [m]$ $d(u_i, u_j) > r$.*

Il est désormais possible d'appliquer le théorème de Gaifman pour obtenir la décidabilité de la théorie du premier ordre des arbres rationnels.

Théorème 39. *La théorie du premier ordre de tout arbre rationnel est décidable.*

Démonstration. D'après le théorème de Gaifman, 32, il est possible de se limiter aux formules locales de base. Soit T un arbre rationnel et $\varphi = \exists x_1 \ldots \exists x_n \bigwedge_{1 \leq i < j \leq n} (d(x_i, x_j) > 2r \wedge \psi^{S(r,x_i)}(x_i))$ une formule locale de base.

En vertu du lemme 37, l'ensemble $L = \left\{ u \in \mathrm{Dom}(T) \mid \mathcal{T} \models \psi^{S(r,x)}[u] \right\}$ est rationnel.

Le lemme 38 permet de décider si il existe des sommets $u_1, \ldots, u_n \in L$ tels que pour tout $i \neq j \in [n]$, on ait $d(u_i, u_j) > 2r$.

En combinant ces deux résultats, on peut décider si T satisfait la formule φ.

□

L'utilisation du théorème de Gaifman induit une complexité non-élémentaire pour ce processus de décision. En fait, dès lors qu'on se limite aux arbres rationnels de degré fini, il est possible d'avoir un procédé de décision élémentaire en appliquant les même techniques que celles utilisées pour les graphes automatiques de degré borné, introduites par Lohrey [105].

I.4 Extensions possibles de ce résultat

Nous allons à présent établir la maximalité de la proposition 39. Plus précisément, nous établirons, dans un premier temps, l'indécidabilité de la logique du premier ordre avec accessibilité des arbres rationnels. Dans un second temps, nous construirons un graphe orienté acyclique dont la théorie du premier ordre sera indécidable.

Trouver une logique plus expressive décidable sur les arbres rationnels

Une extension simple de la logique du premier ordre est constituée de la logique du premier ordre avec accessibilité. Cette dernière consiste simplement en l'adjonction d'un prédicat exprimant la clôture transitive de la structure considérée. Une extension légèrement plus riche consiste à moduler ce prédicat d'accessibilité par un langage rationnel : précisément, pour tout langage rationnel $L \in Rat(\Sigma^*)$, on étend

le vocabulaire de FO à l'aide du prédicat binaire $\mathbf{reach}_L$ exprimant que le premier sommet est connecté au second par un chemin étiqueté par un élément de L.

Théorème 40. *IL existe un arbre rationnel dont la théorie du premier ordre avec accessibilité rationnelle est indécidable.*

Une conséquence immédiate de ce résultat est l'indécidabilité du problème de *model-checking* de cette logique pour les arbres rationnels.

Corollaire 41. *Le problème du* model-checking *de la logique du premier ordre avec accessibilité rationnelle est indécidable pour les arbres rationnels.*

Pour démontrer le théorème 40, on utilise la grille (le quart de plan), avec des arcs de retour. Il s'agit d'un graphe rationnel, illustré sur la figure 3.3.

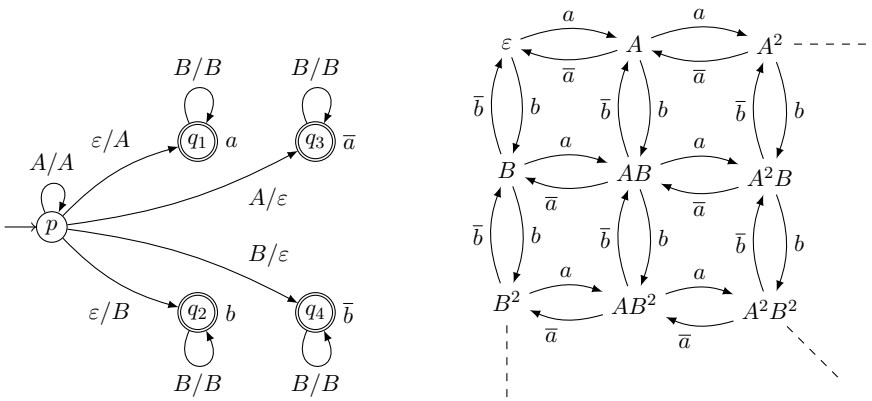


FIGURE 3.3 – Transducteur et illustration de la grille avec retours.

On simule les machines à deux compteurs sur un dépliage de ce graphe (on le verra, il s'agit d'un arbre rationnel). Comme ces machines peuvent réaliser un test à zéro on ajoute une boucle sur les sommets pour exprimer que l'un, l'autre, ou les deux compteurs sont nuls (respectivement notés par les symboles $\#_a, \#_b, \#_{ab}, \#$).

Pour réaliser le dépliage de ce graphe, on transforme le transducteur pour ajouter en préfixe l'unique chemin qui conduit au sommet. Comme cette grille est à la fois déterministe et co-déterministe, on obtient une forêt rationnelle. Cette forêt est formée de composantes connexes ayant une racine, celle du sommet ε est isomorphe au dépliage de la grille avec retour (figure 3.3). C'est également le cas de toutes les composantes dont la racine est dans $\left\{a, b, \overline{a}, \overline{b}\right\}^*$. Le transducteur des arcs a de cette forêt est illustré sur la figure 3.4.

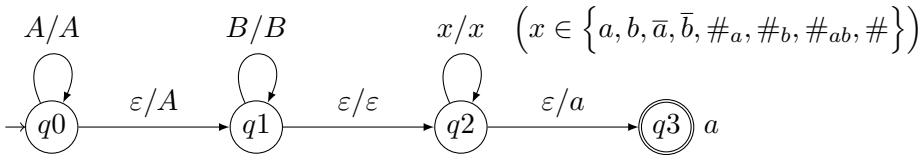


FIGURE 3.4 – Transducteur des arcs étiquetés a du dépliage de la grille avec retour.

Les transducteurs pour les arcs $b, \overline{a}$ et $\overline{b}$ sont exactement similaires. Ceux qui reconnaissent les arcs $\#_a, \#_b, \#_{ab}, \#$ réalisent initialement l'identité, puis vérifie l'absence de A, B, des deux ou de aucun.

En utilisant le lemme 28 on transforme la forêt en un arbre rationnel.

Ensuite pour toute machine de Minsky, M, on définit le langage rationnel L_M de son comportement, et on utilise une formule du premier ordre pour vérifier si on atteint un configuration avec les compteurs vides (propriété indécidable) :

$$\exists u \exists v (\mathbf{reach}_{L_M}(u,v) \wedge \mathbf{root}(u) \wedge \neg(\exists w (v \xrightarrow{\overline{a}} w \vee v \xrightarrow{\overline{b}} w))).$$

Ainsi, nous avons défini un arbre rationnel avec une théorie du premier ordre avec accessibilité rationnelle indécidable.

Extension de la famille de graphes

La théorie du premier ordre des graphes rationnels est indécidable. Il est même possible de construire un tel graphe dont la théorie du premier ordre soit indécidable. Ici, nous allons construire un graphe orienté acyclique (abrégé *dag*) rationnel dont la théorie du premier ordre est indécidable. Ceci met en évidence la profonde connexion entre le résultat de décidabilité et la structure d'arbre.

Théorème 42. *Il existe un dag rationnel dont la théorie du premier ordre est indécidable.*

Schéma. La construction de ce dag (noté G_{pcp}) s'appuie sur un codage de toutes les instances du problème de correspondance de Post (pcp).

La construction détaillée de G_{pcp} est très complexe. Thomas en décrit une similaire [141], il construit un graphe rationnel dont la théorie du premier ordre est indécidable. Ce graphe repose sur un codage d'une machine de Turing universelle, et repose sur la détection d'une boucle marquant l'arrêt de la machine simulée ; ce qui clairement ne se transpose pas aux dag.

Une instance du pcp est une suite de n couples de la forme $((u_i, v_i))_{i\in[n]}$, où les u_i et les v_i sont des mots sur un alphabet donné. Le problème consiste à déterminer si il existe une suite d'indices $(i_k)_{k\in[N]}$, pour un certain $N \in \mathbb{N}$, telle que $u_{i_1}u_{i_2}\dots u_{i_N} = v_{i_1}v_{i_2}\dots v_{i_N} = w$ pour un certain mot w.

Le graphe G_{pcp} est orienté de façon à éviter la formation de circuits. Il est constitué de trois « *composantes* ». La première procède à une forme d'initialisation qui construit toutes les séquences possibles d'indices. La seconde partie, en suivant deux chemins distincts, effectue la substitution de chaque occurrence de l'indice k (dans la séquence) par une série de symboles correspondant à la première composante (u_k) et suivant une autre ligne, effectue la substitution simultanée avec la seconde composante (v_k). Ces deux chemins sont suivis indépendamment. La troisième et dernière partie procède en fusionnant ces deux branches si elles ont abouti au même mot.

A présent, toute instance de pcp sera traduite par une formule de FO dont la satisfaction induit une réponse positive à l'instance codée. La formule assure une initialisation correcte, la substitution des u_i et des v_i convenablement, elle assure également que les deux chemins aboutissent ultimement à un sommet commun.

□

Ceci clos la partie consacrée aux arbres rationnels. Nous avons établi la décidabilité de leur logique du premier ordre. Nous avons également établi l'absence d'extension simple pour ce résultat, que ce soit en modulant l'expressivité de la logique ou en autorisant des graphes plus généraux.

II Propriétés structurelles des graphes de réseaux de Petri

Ce nouveau paragraphe considère un ensemble de résultats de logique pour les graphes d'accessibilité des réseaux de Petri, ils ont été présentés dans le journal *Logical Methods in Computer Science* [52]. Auparavant, on savait que le *model-checking* des formules de CTL, avec proposition atomiques de la forme $p > 0$ (la place p au moins un jeton) pour les réseaux de Petri est indécidable [58]. Ce résultat

négatif se propage à tout fragment de CTL qui contient les modalités EF ou AF. Le problème du *model-checking* de CTL sans proposition atomique, mais avec une modalité **suivant** indicé par une étiquette de transition est également indécidable [58]. À contrario, le problème du *model-checking* de formules LTL est décidable, EXPSPACE-complet [71] lorsque les propositions font référence aux états de contrôle.

En dépit de ces résultats négatifs, il est possible de rechercher des fragments décidables de la logique du premier ordre décrivant simplement des aspects *structurels* liés à la forme du voisinage de certaines configurations dans le graphe d'accessibilité. C'est pourquoi, nous allons délibérément nous abstraire des étiquettes des transitions ainsi que de proposition atomiques portant sur les marquages. En règle générale, on se placera dans la structure $(\mathbb{N}^n, \to)$ définie par un réseau de Petri N possédant n places et tel que $M \to M'$ si et seulement si il est possible de passer du marquage M à M' en tirant une transition de N. Pour tout réseau N, la structure $(\mathbb{N}^n, \to)$ qui lui est associée est automatique, et donc sa théorie du premier ordre est décidable pour les prédicats $\to$ et $=$ (voir, e.g., les travaux de Blumensath et Grädel [17]). Il est possible d'étendre l'algorithme de décision aux graphes des transitions des réseaux de Petri, avec des prédicats Presburger-définissables sur les marquages et des étiquettes sur les transitions.

Un second exemple de résultat lié à notre travail : étant donnée une formule φ de $\mathrm{FO}(\to, =)$ avec comme variables libres $\mathsf{x}_1, \ldots, \mathsf{x}_m$, il est possible de construire une formule de Presburger qui caractérise exactement les marquages satisfaisant φ de $(\mathbb{N}^n, \to)$. En revanche, lorsqu'on considère la structure définie par le graphe d'accessibilité du réseau N : $(\mathrm{Reach}(N), \to)$, la décidabilité de FO est plus incertaine.

Ce paragraphe considère ce problème avec une approche très générale. Ainsi, nous allons étudier la décidabilité d'un ensemble de dialectes de FO sur une variété de structures définies à partir des réseaux de Petri. Tout comme les travaux de Schulz [136], nous examinerons des extensions empruntant à MSO des prédicats d'accessibilité. À l'opposé de la plupart des travaux sur les réseaux de Petri, les propriétés auxquels nous nous intéresserons seront liées à la structure du graphe d'accessibilité et feront donc abstraction des jetons ou des étiquettes des transitions. Elles seront également locales, dans le sens où les dialectes faisant appel aux prédicats d'accessibilité seront minoritaires. Le tableau 3.1 présente une vue d'ensemble des résultats présentés. Ce travail est, à notre connaissance, le seul ayant une telle approche. Ainsi, les travaux récents de Atig et Habermehl [5] s'intéressent à des valeurs quantitatives pour les marquages et les transitions. Les formules sont évaluées sur des exécutions.

II.1 Logique du premier ordre

On va définir un dialecte spécifique de la logique du premier ordre pour spécifier les propriétés des structures $\mathrm{URG}(N)$, $\mathrm{PURG}(N)$ ainsi que $\mathrm{UG}(N)$ (voir, page 30) résultant du réseau de Petri N. On définit la signature relationnelle de FO avec les prédicats atomiques $\mathsf{x} \to \mathsf{y}$, $\mathsf{x} \xrightarrow{*} \mathsf{y}$, $\mathsf{x} \xrightarrow{+} \mathsf{y}$ et $init(\mathsf{x})$. On rappelle que pour une signature relationnelle σ, on note la *restriction de* FO *à* σ par $\mathrm{FO}(\sigma)$. En général, FO fait référence au langage comprenant tous ses symboles relationnels. Les formules sont interprétées sur l'une des structures suivantes : $\mathrm{PURG}(N)$, $\mathrm{URG}(N)$ ou $\mathrm{UG}(N)$. On peut souligner que FO sur $\mathrm{UG}(N)$ permet, en utilisant les prédicats $init$ et d'accessibilité, de relativiser une formule à la structure $\mathrm{URG}(N)$, néanmoins, la présence de langages plus restreints motive l'existence des deux structures. Par abus de notation, nous utiliserons la même notation pour les symboles de prédicats et leur interprétation. On peut noter que, vis-à-vis des interprétations, $\xrightarrow{*} = (= \cup \xrightarrow{+})$ et $\xrightarrow{+} = (\to \circ \xrightarrow{*})$, donc, $\mathrm{FO}(init, \to, \xrightarrow{+}, =)$, $\mathrm{FO}(init, \to, \xrightarrow{*}, =)$, et $\mathrm{FO}(init, \to, \xrightarrow{+}, \xrightarrow{*}, =)$ ont la même expressivité. Dans la logique FO la quantification porte sur les marquages individuels (d'où le premier ordre), néanmoins, les prédicats binaires $\xrightarrow{+}$ ou $\xrightarrow{*}$ entraînent une expressivité plus importante que la logique du premier ordre ordinaire. Nous omettrons la définition classique de la relation de satisfaction $\mathcal{U}, \mathbf{v} \models \varphi$ avec $\mathcal{U}$ une structure ($\mathrm{PURG}(N)$, $\mathrm{URG}(N)$ ou $\mathrm{UG}(N)$) et $\mathbf{v}$ une valuation des variables libres dans φ. A titre d'exemple, $\forall \mathsf{x}\ \varphi$ est vérifiée dès lors que la formule φ est vérifiée pour tous les éléments (marquages) de la structure considérée. On rappelle que les *énoncés* sont les formules closes, i.e., sans variable libre. Si $\mathcal{U} \models \varphi$ on dit alors que $\mathcal{U}$ est un modèle de φ.

On peut observer que, excepté pour l'égalité, FO ne peut exprimer que des propriétés de *théorie des graphes* sur les structures $\mathcal{U}$. Les relations binaires n'utilisent par les étiquettes des transitions ou encore des propositions atomiques faisant référence au marquage. Ainsi, les propriétés quantitatives sur les marquages ne peuvent être exprimés par FO, tout au moins de façon simple. Les contraintes relatives au tirage de transitions particulières ne peut pas être exprimé non-plus. On peut enfin remarquer que FO n'est pas minimale pour l'expressivité. Cette redondance nous permet de considérer des fragments logiques intéressants.

Dans la suite, nous allons considérer un ensemble de problèmes de *model-checking*. Le premier problème, noté $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO})$, se formule ainsi :

Donnée : un réseau de Petri $N = (P, T, F, M_0)$ et un énoncé $\varphi \in \mathrm{FO}$

Question : $\mathrm{URG}(N) \models \varphi$?

La variante, $\mathrm{MC}^{\mathrm{UG}}(\mathrm{FO})$ s'exprime ainsi :

Donnée : un réseau de Petri $N = (P, T, F, M_0)$ et un énoncé $\varphi \in \mathrm{FO}$

Question : $\mathrm{UG}(N) \models \varphi$?

Les logiques $\mathrm{FO}(\sigma)$ (dont les formules atomiques sont limités à σ) permettent de définir des variantes des deux problèmes de *model-checking* qui sont notés respectivement $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\sigma))$ et $\mathrm{MC}^{\mathrm{UG}}(\mathrm{FO}(\sigma))$. Les formules de FO permettent d'exprimer des propriétés structurelles classiques, telles que, l'absence de blocage $\forall \mathsf{x}\ \exists \mathsf{y}\ \mathsf{x} \to \mathsf{y}$, l'existence d'une 1-boucle avec $\exists \mathsf{x}\ \mathsf{x} \to \mathsf{x}$, ou encore que le réseau est cyclique : $\forall \mathsf{x} \forall \mathsf{y}\ \mathsf{x} \xrightarrow{*} \mathsf{y} \Rightarrow \mathsf{y} \xrightarrow{*} \mathsf{x}$.

On l'a déjà vu (Chapitre 2, § III.1, p 18), les structures automatiques forment une vaste collection d'objets pour lesquels le problème de *model-checking* de FO est décidable. Ces structures ont des présentations qui sont des relations d'ordre k définies par des transducteurs synchrones (le lecteur intéressé peut se référer aux travaux de Bluhmensath et Grädel [17]). On note $\mathrm{MC}(\mathrm{FO}, \mathcal{S})$ le problème de *model-checking* d'une formule de FO pour une structure particulière $\mathcal{S}$.

Théorème 43. *[17] Soit $\mathcal{S}$ une structure automatique, alors le problème $\mathrm{MC}(\mathrm{FO}(\to, =), \mathcal{S})$ est décidable.*

Les résultats de Ginsburg et Spanier [68] établissent que les ensembles et les relations semi-linéaires sont automatiques. En particulier, ceci signifie que les structures $(\mathbb{N}^n, \to, =)$ sont automatiques (pour toute valeur entière de n). Les propositions 44, 45 et 46 sont des conséquences du théorème 43 ; on les indique ci-dessous pour présenter un état complet des connaissances à ce jour.

Proposition 44. $\mathrm{MC}^{\mathrm{UG}}(\mathrm{FO}(\to, =))$ *est décidable.*

Il est important de noter que pour une formule φ dans $\mathrm{FO}(\to, =)$, on peut construire de façon effective une formule de Presburger qui caractérise précisément les valuations qui satisfont φ dans $\mathrm{UG}(N)$. La décidabilité est préservée en ajoutant des propriétés Presburger-définissables sur les marquages et en ayant des transitions étiquetées $[t\rangle$. En revanche, avoir comme domaine l'ensemble $\mathbb{N}^n$ n'est pas une garantie de décidabilité. Ainsi, Schulz [136, Theorem 2] considère la structure dont le domaine est $\mathbb{N}^n$ mais équipé de la relation successeur dans toutes les dimensions, et avec des prédicats d'accessibilité rationnelle. De façon similaire, les sous-problèmes de $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO})$ nécessitent des contraintes supplémentaires pour garantir la décidabilité, par exemple la semi-linéarité supposée dans l'énoncé plus haut. Le résultat qui suit est une nouvelle conséquence du théorème 43.

Proposition 45. *Soit $\mathcal{C}$ une classe de réseaux de Petri pour laquelle la restriction de la relation d'accessibilité, $\mathsf{x} \xrightarrow{*} \mathsf{y}$, à l'ensemble des marquages accessibles est de façon effective semi-linéaire. Le problème $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO})$ restreint aux éléments de $\mathcal{C}$ est décidable.*

Démonstration. Soit $N = (P, T, F, M_0)$ un réseau de Petri de $\mathcal{C}$ avec $\mathrm{card}(P) = n$. Ses marquages sont représentés par des vecteurs $M \in \mathbb{N}^n$. Par hypothèse, $\mathrm{Reach}(N)$ et les ensembles $\{(M, M') \mid M, M' \in \mathrm{Reach}(N) \text{ et } M \xrightarrow{*} M'\}$ sont de façon effective semi-linéaires. De même l'ensemble $\{(M, M) \mid M \in \mathrm{Reach}(N)\}$ est également semi-linéaire de façon effective. Considérons la relation $\Delta = \{(M, M') \mid M, M' \in \mathrm{Reach}(N) \text{ et } M \xrightarrow{*} M',\ M \neq M'\}$. Ainsi Δ est semi-linéaire de façon effective. Soit la relation $\Delta^2 = \{(M, M') \mid (\exists M'')\, (M, M'') \in \Delta \text{ et } (M'', M') \in \Delta\}$. Or, les ensemble semi-linéaires sont clos pour la projection (élimination des quantificateurs dans l'arithmétique de Presburger), Δ^2 est donc une relation semi-linéaire de façon effective. À présent, $\{(M, M') \mid M \in \mathrm{Reach}(N) \text{ et } M \xrightarrow{+} M'\}$ est égal à l'intersection $\Delta \cup \Delta^2$. Il s'agit donc encore d'un ensemble semi-linéaire. Il est donc possible de réaliser une traduction effective entre les ensembles définissables en arithmétique de Presburger et les énoncés φ de FO. Une telle formule se traduit en un énoncé φ' de l'arithmétique de Presburger, de sorte que $\mathrm{URG}(N) \models \varphi$ si et seulement si φ' est vraie. La proposition résulte de la décidabilité de l'arithmétique de Presburger [125]. □

Lorsque l'ensemble d'accessibilité est de façon effective semi-linéaire, mais que la relation ne l'est pas, la logique strictement moins expressive $\mathrm{FO}(\rightarrow, =)$ reste décidable, d'après le théorème 43.

Proposition 46. *Soit $\mathcal{C}$ une classe de réseaux de Petri telle que $\mathrm{Reach}(N)$ est semi-linéaire de façon effective pour tout membre N de $\mathcal{C}$. Le problème de* model-checking $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\rightarrow, =))$*, restreint aux éléments de $\mathcal{C}$ est décidable.*

Démonstration. Soit $N = (P, T, F, M_0)$ un réseau de Petri dans $\mathcal{C}$. Considérons la formule de Presburger suivante $\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ caractérisant les marquages de $\mathrm{Reach}(N)$ (avec $|P| = n$). On peut également définir la formule de Presburger $\varphi'(\mathsf{x}_1, \ldots, \mathsf{x}_n, \mathsf{x}'_1, \ldots, \mathsf{x}'_n)$ qui caractérise la relation binaire $\rightarrow$ dans $\mathrm{UG}(N)$.

A présent, étant donnée un énoncé ψ de $\mathrm{FO}(\rightarrow, =)$, on peut construire un énoncé $f(\psi)$ de l'arithmétique de Presburger arithmetic tel que $\mathrm{URG}(N) \models \psi$ si et seulement si $f(\psi)$ est satisfaisable dans l'arithmétique de Presburger. L'application $f(\cdot)$ est un morphisme pour les opérateurs booléens. De plus,

— $f(\mathsf{z} \rightarrow \mathsf{z}') \stackrel{\text{def}}{=} \varphi'(\mathsf{z}_1, \ldots, \mathsf{z}_n, \mathsf{z}'_1, \ldots, \mathsf{z}'_n)$,
— $f(\mathsf{z} = \mathsf{z}') \stackrel{\text{def}}{=} \bigwedge_{i \in [1,n]} \mathsf{z}_i = \mathsf{z}'_i$,
— $f(\forall \mathsf{z}\ \chi) \stackrel{\text{def}}{=} \forall \mathsf{z}_1, \ldots, \mathsf{z}_n\ (\varphi(\mathsf{z}_1, \ldots, \mathsf{z}_n) \Rightarrow f(\chi))$.

Pour évaluer le prédicat $\rightarrow$, on utilise φ'. La formule φ relativise les quantification aux éléments de $\mathrm{Reach}(N)$. □

A nouveau la décidabilité est préservée avec les propriétés Presburger définissables sur les marquages et les transitions étiquetées. A titre d'illustration de ce résultat, on peut observer que le problème de *model-checking* $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\rightarrow, =))$, restreint aux réseaux de Petri cycliques est décidable. Il s'agit d'une conséquence directe de la proposition 46 combiné avec le résultat de Bouziane et Finkel qui établi la semi-linéarité des réseaux cycliques [21]. La restriction au langage $\mathrm{FO}(\rightarrow, =)$ est cruciale pour la décidabilité de la proposition 46. Comme on le constatera avec la proposition 70, le problème du *model-checking* $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\rightarrow, \xrightarrow{*}))$ est indécidable, même lorsqu'on fait la supposition de la semi-linéarité de l'ensemble des marquages.

II.2 Logique Modale

Nous allons examiner certains problèmes définis sur les logiques modales. Schématiquement, dans ce paragraphe nous nous intéressons aux cas décidables, qui sont les problèmes de *model-checking* pour les logiques $\mathrm{ML}(\Box)$ (restriction aux prédicats modaux "en avant"), et ML, notée $\mathrm{ML}(\Box, \Box^{-1})$, pour être plus explicite. En fin de paragraphe, nous enrichissons cette logique pour ajouter des prédicats arithmétiques

définis à partir du contenu des places (PAML). Ultérieurement nous considérerons les problèmes de validité (intuitivement équivalent au *model-checking* des formules précédés d'une quantification universelle sur les marquages).

On note $\mathrm{MC}^{\mathrm{URG}}(\mathrm{ML}(\Box))$ la restriction du problème $\mathrm{MC}^{\mathrm{URG}}(\mathrm{ML})$ à la logique $\mathrm{ML}(\Box)$. La proposition 47 établi la décidabilité de problème. La technique s'appuie sur le fait qu'une formule modale de degré d ne définit de contrainte que sur les marquages à une distance d du marquage initial. Cet argument est classique [16].

Proposition 47. $\mathrm{MC}^{\mathrm{URG}}(\mathrm{ML}(\Box))$ *est décidable et* PSPACE*-complet.*

Démonstration. Soit $N = (P, T, F, M_0)$ un réseau de Petri, avec $\mathrm{URG}(N) = (D, init, \to, \xrightarrow{*}, \xrightarrow{+}, =)$. Soit φ une formule modale de degré d dans $\mathrm{ML}(\Box)$ (d représente le plus grand nombre d'imbrication d'opérateurs modaux dans φ). On considère le graphe orienté $\mathcal{M} = (W, R)$ défini de la façon suivante

— $W \subseteq \mathbb{N}^P$ et R est la restriction de $\to$ à W.

— pour un marquage $M \in \mathbb{N}^P$ on défini $M \in W \overset{\text{def}}{\Leftrightarrow}$ il existe une suite de transitions s de longueur au plus d tel que $M_0[s\rangle M$.

On peut souligner que $\mathcal{M}$ est fini, et le cardinal de W est au plus exponentiel dans la taille de N et d. On peut démontrer que $\mathcal{M}, M_0 \models \varphi$ si et seulement si $(D, \to), M_0 \models \varphi$. Ainsi, $\mathrm{MC}^{\mathrm{URG}}(\mathrm{ML}(\Box))$ est décidable, puisque le *model-checking* de ML pour les structures finies est décidable (en temps polynomial). La borne supérieure PSPACE est obtenue à l'aide d'un algorithme similaire à celui qui démontre que le *model-checking* de CTL sur les réseaux de Petri 1-saufs est dans PSPACE, voir, par exemple Esparza [58, Section 4.2]. En fait, notre problème est plus simple puisque on peut se restreindre aux opérateurs temporels **A X** et **E X** correspondant respectivement aux opérateurs $\Box$ et $\Diamond$.

La dureté dans PSPACEpeut être établie par une réduction de QBF (satisfiabilité des formules booléennes quantifiées). Soit une formule QBF $\mathcal{Q}_1 \mathrm{p}_1 \ \cdots \ \mathcal{Q}_{2n} \mathrm{p}_{2n} \ \psi$ be où $\mathcal{Q}_1 \cdots \mathcal{Q}_{2n}$ est un suite de quantificateurs qui débute par $\mathcal{Q}_1 = \exists$, avec une alternance stricte des quantificateurs $\exists$ et $\forall$, la formule propositionnelle ψ ne contient aucun quantificateur et porte sur l'ensemble de variables $\{\mathrm{p}_1, \ldots, \mathrm{p}_{2n}\}$. On considère à présent la formule modale φ de la forme $(\Diamond\Box)^n\psi'$ où ψ' est construite à partir de ψ en remplaçant chacune des variable propositionnelle p_i par $\Diamond^i\Box \perp$. On construit ensuite le réseau de Petri $N = (P, T, F, M_0)$ comme suit. L'ensemble des places de P contient un sous-ensemble $\{p_1, \ldots, p_{2n}\}$ en bijection avec les propositions atomiques et initialement vide. Quelques places auxiliaires seront décrites par la suite. À partir du marquage initial, M_0, N produit une suite de $2n$ choix indépendants $(t'_1 + t''_1) \cdot (t'_2 + t''_2) \cdot \ldots \cdot (t'_{2n} + t''_{2n})$ où la présence de i jetons dans une place p_i représente que la variable est vraie. Réciproquement, lorsqu'il n'y a aucun jeton, la valuation de p_i est fausse. Après la séquence de choix binaire pour chacune des variables, N effectue un nouveau choix non-déterministe $(x_1 + \cdots + x_{2n})$ où x_i retire un jeton de la place p_i puis place un jeton dans la place p'_i originellement vide. Chacune des places p'_i est en auto-boucle avec une transition t_i qui retire un jeton à p_i à chaque tirage.

Les quantifications existentielles sont remplacées par $\Diamond$, les universelles par $\Box$. Un chemin relatif à une formule $(\Diamond\Box)^n$ se termine dans une configuration où la valuation a été choisie. Notez que la formule doit être vérifié pour une des continuation à chaque position $\Diamond$ et vérifié pour toutes les continuations au positions $\Box$. La dernière partie de la formule effectue la vérification des valeurs de vérités pour les variables individuelles. Pour chacune des variables p_i, on utilise la formule $\Diamond^i\Box \perp$ qui est vérifiée uniquement par les chemins de longueur précisément i, ce qui coïncide avec notre codage de la valuation. La sélection de chaque variable (individuellement) est réalisée par la transition $(x_1 + \cdots + x_{2n})$. Globalement, on a : $(\mathrm{Reach}(N), \to), M_0 \models (\Diamond\Box)^n\psi'$ si et seulement si la formule $\mathcal{Q}_1 \mathrm{p}_1 \ \cdots \ \mathcal{Q}_{2n} \mathrm{p}_{2n} \psi$ est satisfaisable. On peut remarquer que l'ensemble $\mathrm{Reach}(N)$ est fini. □

Il est très fréquent que l'ajout de l'opérateur $\Box^{-1}$ au langage $\mathrm{ML}(\Box)$ n'ait pas d'impact sur la décidabilité ou la complexité du *model-checking* (par exemple pour les structures finies) voir, par exemple Blackburn *et al.* [16]. En revanche, dans le cas des graphes d'accessibilité des réseaux de Petri, $\mathrm{PURG}(N)$, l'ajout de l'opérateur réciproque $\Box^{-1}$ n'influe pas sur la décidabilité, mais impose des tests d'accessibilité.

Proposition 48. *Le problème* $\mathrm{MC}^{\mathrm{URG}}(\mathrm{ML}(\Box, \Box^{-1}))$ *est décidable.*

Démonstration. Étant donné un réseau de Petri $N = (P, T, F, M_0)$ avec $\mathrm{URG}(N) = (D, init, \to, \xrightarrow{*}, \xrightarrow{+}, =)$. Soit une formule modale φ dans $\mathrm{ML}(\Box, \Box^{-1})$ dont le degré modale soit d. On définit le réseau $\overline{N} = (P, T \cup T^{-1}, F, M_0)$ où T^{-1} est l'ensemble des inverses formels des transitions de T, *i.e.*, $F(p, t^{-1}) = F(t, p)$ et $F(t^{-1}, p) = F(p, t)$ pour tout $t \in T$. Pour vérifier la formule φ vis-à-vis de $\mathrm{URG}(N)$, on considère une expansion de profondeur d de la structure $\mathrm{URG}(\overline{N})$. Cependant, suivre les transitions inverses $M'[t^{-1}\rangle M$, impose de faire des test d'accessibilité pour les marquages M et vérifier qu'ils appartiennent bien au domaine D de la structure $\mathrm{URG}(N)$. Ces tests sont effectifs d'après le théorème 29 repris des classiques [109, 95, 102]. De façon plus formelle on définit le graphe $\mathcal{M}' = (W', R')$ de la façon suivante :

— $W' \subseteq \mathbb{N}^P$ et R' est la restriction de la relation $\to$ à W'.

— Un marquage $M \in \mathbb{N}^P$ appartient à W' dès lors que :

1. $M \in D$,
2. il y a une suite de transitions de longueur au plus d $s \in (T \cup T^{-1})^*$ telle que $M_0[s\rangle M$.

Il est simple de vérifier si $M_0[s\rangle M$ alors que un test d'accessibilité est nécessaire pour $M \in D$. On peut remarquer que le graphe $\mathcal{M}'$ est fini, et conctructibke de façon effective. Le cardinal de W' est exponentiel en d. On peut montrer que $\mathcal{M}', M_0 \models \varphi$ si et seulement si $(D, \to), M_0 \models \varphi$. Ainsi, $\mathrm{MC}^{\mathrm{URG}}(\mathrm{ML}(\Box, \Box^{-1}))$ est décidable car le *model-checking* de ML sur les structures finies est polynomial. □

On sait que les meilleurs algorithmes pour l'accessibilité des réseaux de Petri ne sont pas primitifs récursifs, ce qui nous donne au pire la même complexité pour l'algorithme de la proposition 48.

Il est difficile de déterminer si il s'agit d'une borne supérieure optimale. En revanche, §II.6 nous établirons que le problème d'accessibilité dans les réseaux de Petri peut se réduire à celui du *model-checking* $\mathrm{MC}^{\mathrm{URG}}(\mathrm{ML}(\Box, \Box^{-1}))$.

Nous examinons à présent un autre problème pour ML qui est proche du *model-checking* de FO pour les graphes d'accessibilité des réseaux de Petri. Le *problème de la validité* $\mathrm{VAL}^{\mathrm{URG}}(\mathrm{ML})$, parfois appelé model-checking *global*, s'exprime de la façon suivante :

Donnée : Un réseau de Petri $N = (P, T, F, M_0)$ qui définit la structure $\mathrm{URG}(N) = (D, init, \to, \xrightarrow{*}, \xrightarrow{+}, =)$, et une formule modale $\varphi \in \mathrm{ML}$.

Question : $(D, \to), M \models \varphi$ pour chaque marquage $M \in D$?

Comme on l'a déjà observé, les formules de $\mathrm{ML}(\Box, \Box^{-1})$ peuvent être vues comme des formules du premier ordre de $\mathrm{FO}(\to)$. Considérer des langages modaux revient à considérer un fragment particulier de $\mathrm{FO}(\to)$. En effet, étant donnée une formule modale φ de $\mathrm{ML}(\Box, \Box^{-1})$, il est possible de calculer, en temps linéaire, une formule (possédant seulement deux variables [16]) φ' qui satisfait l'énoncé suivant : pour tout réseau de Petri N on a $\mathrm{PURG}(N) \models \varphi'$ si et seulement si $\mathrm{PURG}(N), M \models \varphi$ pour tout marquage M in $\mathrm{Reach}(N)$. Ainsi, le problème de validité $\mathrm{VAL}^{\mathrm{URG}}(\mathrm{ML})$ se révèle être un équivalent naturel au problème de *model-checking* de FO sur les graphes d'accessibilité dess réseaux de Petri non-étiquetés. Nous verrons dans le paragraphe suivant que ces deux problèmes sont indécidables.

Nous conclurons ce paragraphe en définissant une extension de ML qui autorise des formules de l'arithmétique de Presburger sans quantification comme proposition atomique. L'idée est d'imposer des contraintes arithmétiques sur le nombre de jetons dans les places et ainsi augmenter l'expressivité de ML. Nous appelons cette logique PAML et elle sera principalement utilisé pour les résultats de décidabilité §II.5. Le domaine de la structure PAML doit être de la forme $\mathbb{N}^P$. Plus précisément, on utilise les termes de la forme $t ::= a \times p \mid t + t$ où p est une place et $a \in \mathbb{Z}$ on définit PAML à partir de ML en ajoutant des formules atomiques ψ définies par

$$\psi ::= \top \mid t \leq k \mid t \geq k \mid t \equiv_c k' \mid \psi \wedge \psi \mid \neg\psi.$$

Ici, le symbole $\top$ représente la constante **vrai**, $c \in \mathbb{N} \setminus \{0,1\}$, $k \in \mathbb{Z}$ et $k' \in \mathbb{N}$. La définition de $(\mathrm{Reach}(N), M) \models \psi$ dépend de celle de la satisfaction d'une formule ψ et un n-uplet M dans l'arithmétique de Presburger. Nous omettrons les détails qui sont classiques. On peut montrer que le problème $\mathrm{MC}^{\mathrm{URG}}(\mathrm{PAML}(\Box, \Box^{-1}))$ est décidable en utilisant une démonstration similaire à celle de la proposition 48.

II.3 Schéma de preuve pour l'indécidabilité de $\mathrm{FO}(\rightarrow)$

Pour démontrer l'indécidabilité du problème $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\rightarrow))$, autrement dit le *model-checking* des spécifications du premier ordre pour les graphes d'accessibilité des réseaux de Petri, nous utilisons une réduction du problème de l'égalité des ensembles d'accessibilité. Soient deux réseaux N_1 et N_2 avec des ensembles de places identiques, Hack a démontré l'indécidabilité de l'égalité des ensembles de marquages $\mathrm{Reach}(N_1)$ et $\mathrm{Reach}(N_2)$ (le théorème 30 rappel ce résultat [73]). Pour réaliser cette réduction, on construit un réseau de Petri $\overline{N}$ à partir de deux réseaux quelconques N_1 et N_2. La construction du réseau $\overline{N}$ garantit qu'on a la propriété $\mathrm{Reach}(N_1) = \mathrm{Reach}(N_2)$ si et seulement si $\mathrm{PURG}(\overline{N}) \models \varphi$. On peut souligner que cette formule, φ, est indépendante des réseaux d'entrée N_1 et N_2. Avant de regarder le détail de la construction, nous allons présenter les idées sous-jacentes. La figure 3.5 illustre une partie du comportement du réseau $\overline{N}$: un choix initial le conduit à simuler soit N_1 ou N_2 avec une place désigné pour distingué le réseau choisi, ensuite, après chaque configuration « originale » le réseau peut oublier sont origine (et ne plus avoir de transition tirable) ce sont les transitions étiquetées t^1_{end} ou t^2_{end} suivant le réseau d'origine. On peut remarquer que cette construction entraîne que les seules marquage en blocage sont ceux qui suivent les transitions t^1_{end} et t^2_{end}. Ainsi les deux réseaux auront les mêmes ensembles de marquages $\mathrm{Reach}(N_1)$ et $\mathrm{Reach}(N_2)$ si et seulement si toutes les configurations accessibles qui sont en blocages ont précisément deux ancêtres distincts. Cependant pour limiter les expressivité de la logique nécessaire pour le résultat on ajoute également les transitions t^1_{dl}, t^2_{dl} et t^1_{end} pour permettre de distinguer l'ancêtre issu de N_1 de celui issu de N_2.

Ensuite on peut construire une formule de FO qui permette d'étudier le voisinage des marquages en blocage. Cette formule utilisera une transition réciproque et plusieurs transitions pour effectuer la vérification voulue : existence d'un ancêtre avec un successeur qui porte une 1-boucle, et un ancêtre sans successeur ayant une 1-boucle.

La force de cette construction repose sur la combinaison de deux idées. Les réseaux de Petri on les capacités suivantes :

(i) mémoriser des choix tout au long d'une trajectoire ;

(ii) exprimer ces choix à l'aide de structures locales qui peuvent être détectées à l'aide de formules du premier ordre.

Construction. Soient deux réseaux N_1 et N_2 possédant les mêmes ensemples de places (dont on souhaite vérifier l'égalité des ensembles de marquages). Le réseau qu'on définit, $\overline{N}$, possède le même ensemble de place ainsi que les places suivantes :

— une place d'initialisation p ;
— deux places de contrôle p_1 et p_2 ;
— trois places supplémentaires p'_1, p''_1, et p'_2 dont on détaillera les fonctions plus tard.

Le marquage initial est constitué d'un seul jeton dans la place d'initialisation.

Les transitions du réseau $\overline{N}$ sont formées par celles des réseaux N_1 et N_2, ainsi qu'un ensemble d'autre transitions dont nous allons donner la définition. En premier lieu les transitions sont en auto-boucle avec leur place de contrôle respectives. Ensuite, il y a les deux transitions en concurrence t^1_c, t^2_c qui ont pour effet de consommer le jeton initial et de réaliser respectivement le marquage initial de N_1 ou N_2 en plaçant un jeton dans la place p_1 ou p_2. Tirer la transition t^1_c amorce la simulation de N_1, et de façon similaire t^2_c amorce la simulation du second réseau. Les transitions t^1_{end} et t^2_{end} on pour effet d'interrompre

la simulation en déplaçant le jeton des places p_1 ou p_2 vers la place p'_1 ou p'_2, respectivement. Ceci interdit toute modification ultérieur des marquage des places correspondant aux réseaux N_1 et N_2.

Une fois les transitions t^1_{end} et t^2_{end} tirées, $\overline{N}$ réalise le comportement représenté sur la figure 3.5, à la suite des marquages M_1 et M_2. Plus précisément, une fois au marquage M_1, la place p'_1 autorise une transition t^1_ℓ qui place un jeton dans la place p''_1, représentée par $M_\circlearrowleft$ sur la figure. Enfin, les deux transitions t^1_{dl} et t^2_{dl} (depuis le marquage M_1 vers M_ℓ et depuis M_2 vers M_r) vident les places respectives p'_1 et p'_2. Comme il a déjà été observé, les marquages atteints par ces transitions sont les seuls qui sont en blocage. Lorsque les transitions t^1_{dl} ou t^2_{dl} sont tirées le réseau $\overline{N}$ oublie la valeur 1 ou 2 du réseau qui était simulé, on a donc la relation suivante : lorsqu'un marquage M est atteint à la fois dans N_1 et N_2, le marquage correspondant dans $\overline{N}$ conduit au marquage $M_\ell = M_r$

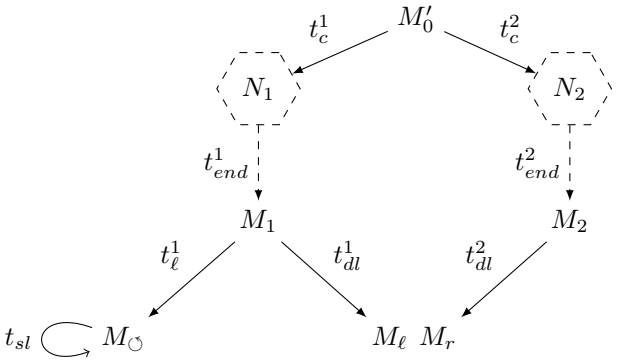


FIGURE 3.5 – Graphe d'accessibilité du réseau de Petri $\overline{N}$

A présent, nous sommes en mesure de présenter une formule (notée φ_{indec}) qui exprime légalité des ensembles de marquages $\mathrm{Reach}(N_1)$ et $\mathrm{Reach}(N_2)$ (sans recycler les variables) :

$$\varphi_{indec} \stackrel{\text{def}}{=} \forall\, \mathsf{z}\ (\neg\exists \mathsf{z}'\ \mathsf{z} \to \mathsf{z}') \Rightarrow (\exists \mathsf{z}_1\ \mathsf{z}_1 \to \mathsf{z} \wedge \varphi_l(\mathsf{z}_1)) \wedge (\exists \mathsf{z}_2\ \mathsf{z}_2 \to \mathsf{z} \wedge \neg\varphi_l(\mathsf{z}_2))$$

La formule $\varphi_l(\mathsf{x}) \stackrel{\text{def}}{=} \exists\, \mathsf{y}\ (\mathsf{x} \to \mathsf{y} \wedge \mathsf{y} \to \mathsf{y})$ révèle que le marquage x a un successeur qui porte une 1-boucle.

Lemme 49. $\mathrm{Reach}(N_1) = \mathrm{Reach}(N_2)$ *si et seulement si* $\mathrm{PURG}(\overline{N}) \models \varphi_{indec}$.

Démonstration. Pour l'implication de la gauche vers la droite, considérons une configuration M en blocage. Le marquage M est uniquement accessible depuis la configuration t^1_{dl} ou t^2_{dl}, disons $M_1[t^1_{dl}\rangle M$. Alors le marquage M_1 satisfait φ_l et est issu d'un marquage $M'_1[t^1_{end}\rangle M_1$ de N_1. D'après l'hypothèse d'égalité des ensembles de marquages on a un marquage M'_2 de N_2 qui conduit par la transition t^2_{end} au marquage M_2 qui satisfait $\neg\varphi_l$ comme on le souhaite.

Réciproquement, si la formule φ_{indec} est valide on peut démontrer les deux inclusion. Considérons d'abord $\mathrm{Reach}(N_1) \subseteq \mathrm{Reach}(N_2)$, soit le marquage M'_1 accessible par une suite de transitions s_1 dans N_1. Dans le réseau $\overline{N}$, il est possible de prolonger la suite s_1 pour atteindre un marquage en blocage M, avec $M'_0[t^1_c\rangle M^1_0[s_1\rangle M'_1[t^1_{end}\rangle M_1[t^1_{dl}\rangle M$. Ici, le marquage M_1 satisfait φ_l. La formule φ_{indec} permet d'avoir un autre prédécesseur M_2 de M avec $M_2 \neq M_1$. Comme le marquage M_2 ne possède pas de 1-boucle, il doit être obtenu par une suite de transition $M'_0[t^2_c\rangle M^2_0[s_2\rangle M'_2[t^2_{end}\rangle M_2[t^2_{dl}\rangle M$. Or, on sait que les marquages M'_1 et M'_2 coïncident en dehors de la place p_i. Donc, on a $M'_1 \in \mathrm{Reach}(N_2)$, et donc $\mathrm{Reach}(N_1) \subseteq \mathrm{Reach}(N_2)$, par symétrie de la construction on peut démontrer de façon équivalente la réciproque. □

Corollaire 50. *Le problème* $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\rightarrow))$ *est indécidable. Le problème de vérifier la validité de la formule* φ_{indec} *est également indécidable pour les graphes* $\mathrm{URG}(N)$ *des réseaux de Petri.*

Pour obtenir le résultat le plus précis, on peut recycler les variables de φ_{indec} ceci nous permet de délimiter avec précision les frontières de la décidabilité du *model-checking* de $\mathrm{FO}(\rightarrow)$: indécidable lorsque la formule possède au moins deux variables, décidable, lorsqu'il n'y en a plus qu'une.

Théorème 51. *Il existe une formule* φ'_{indec} *de* $\mathrm{FO}(\rightarrow)$ *ayant précisément deux variables, pour laquelle le problème de validité de la formule* φ'_{indec} *est indécidable pour les graphes* $\mathrm{URG}(N)$ *des réseaux de Petri.*

Démonstration. Il suffit d'observer que la formule φ_{indec} ci-dessous :

$$\forall \mathsf{z}\ (\neg\exists \mathsf{z}'\ \mathsf{z} \rightarrow \mathsf{z}') \Rightarrow (\exists \mathsf{z}_1\ \mathsf{z}_1 \rightarrow \mathsf{z} \wedge \varphi_l(\mathsf{z}_1)) \wedge (\exists \mathsf{z}_2\ \mathsf{z}_2 \rightarrow \mathsf{z} \wedge \neg\varphi_l(\mathsf{z}_2))$$

avec $\varphi_l(\mathsf{x}) \stackrel{\text{def}}{=} \exists \mathsf{y}\ (\mathsf{x} \rightarrow \mathsf{y} \wedge \mathsf{y} \rightarrow \mathsf{y})$ est logiquement équivalente à

$$\forall \mathsf{z}\ (\neg\exists \mathsf{z}'\ \mathsf{z} \rightarrow \mathsf{z}') \Rightarrow (\exists \mathsf{z}'\ \mathsf{z}' \rightarrow \mathsf{z} \wedge \varphi'_l(\mathsf{z}')) \wedge (\exists \mathsf{z}'\ \mathsf{z}' \rightarrow \mathsf{z} \wedge \neg\varphi'_l(\mathsf{z}'))$$

avec $\varphi'_l(\mathsf{z}') \stackrel{\text{def}}{=} \exists\ \mathsf{z}\ (\mathsf{z}' \rightarrow \mathsf{z} \wedge \mathsf{z} \rightarrow \mathsf{z})$. Le recyclage de variables est une méthode classique décrite par exemple dans Gabbay [64]. □

En s'appuyant sur le théorème 51 qui énonce que le *model-checking* de la logique du premier ordre sur les structures automatiques est décidable, on déduit le résultat suivant.

Corollaire 52. *Pour la classe des graphes d'accessibilité des réseaux de Petri, il n'existe pas de d'algorithme qui à partir de tout graphe d'accessibilité construit un graphe automatique qui lui est isomorphe.*

On peut souligner que ce résultat ne peut en aucun cas être obtenu par de simple considération de complexité. En effet, même si tout graphe d'accessibilité pouvait être représenté par un graphe automatique cette représentation ne pourrait être utilisé pour résoudre l'accessibilité de l'état initial dans le réseau de Petri (simple problème d'appartenance à un langage rationnel dans le cas du graphe automatique, EXPSPACE-dur dans le cas d'un réseau de Petri) à moins d'une représentation effective de la bijection entre les ensembles de sommets des deux graphes.

Le problème de *model-checking* de $\mathrm{FO}(\rightarrow)$ devient décidable lorsqu'on se limite à une seule variable.

Proposition 53. *Le problème* $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\rightarrow))$ *restreint à l'utilisation d'une seule variable est décidable.*

Démonstration. Tout énoncé de $\mathrm{FO}(\rightarrow)$ limité à une seule variable est logiquement équivalent soit à $\bot$, ou à $\top$, ou encore à une formule booléenne positive avec des formules atomiques de l'une des formes suivantes :

1. $\exists \mathsf{x}\ (\mathsf{x} \rightarrow \mathsf{x})$
2. $\exists \mathsf{x}\ \neg(\mathsf{x} \rightarrow \mathsf{x})$
3. $\forall \mathsf{x}\ (\mathsf{x} \rightarrow \mathsf{x})$
4. $\forall \mathsf{x}\ \neg(\mathsf{x} \rightarrow \mathsf{x})$.

Comme, (2) est la négation de (3) et (1) celle de (4), on obtient la décidabilité en évaluant (1) $\mathrm{PURG}(N) \models \exists \mathsf{x}\ (\mathsf{x} \rightarrow \mathsf{x})$ et (3) $\mathrm{PURG}(N) \models \forall \mathsf{x}\ (\mathsf{x} \rightarrow \mathsf{x})$. On peut résoudre (1) par la résolution d'un instance du problème de couverture pour chacune des transitions neutres. Pour (3) il faut résoudre une instance du problème d'accessibilité. En effet, soit $T_\circlearrowleft$ le sous-réseau composé des transitions neutres, l'ensemble de marquage qui suit est semi-linéaire :

$$Z \stackrel{\text{def}}{=} \{M :\ \text{not } M[t\rangle \text{ for all } t \in T_\circlearrowleft\}$$

On n'a pas $\mathrm{PURG}(N) \models \forall\, \mathsf{x}\ (\mathsf{x} \rightarrow \mathsf{x})$ si et seulement si il y a un marquage $M \in Z$ accessible, $M_0 \stackrel{*}{\rightarrow} M$. D'après le lemme 4.3 du travail de Hack [73] ceci se réduit au problème d'accessibilité. □

Il est encore possible d'aller plus loin dans la modulation des paramètres. Ainsi, notre démonstration d'indécidabilité vu précédemment repose sur une variabilité du graphe d'accessibilité avec une formule constante. Il est ouvert de savoir si il existe un réseau de Petri donné dont le graphe d'accessibilité a une théorie du premier ordre indécidable.

II.4 Robustesse du schéma

En s'appuyant sur le schéma de la preuve du théorème 51 nous allons à présent présenter d'autres résultats d'indécidabilité pour des variantes de problèmes de *model-checking*. Plus précisément, nous allons considérer le fragment positif, le fragment *en avant*, celui où l'orientation des arcs est perdue et enfin $\mathrm{ML}(\Box, \Box^{-1})$.

Ignorer l'orientation

Posons $\lambda(\mathsf{x}, \mathsf{x}') \stackrel{\text{def}}{=} (\mathsf{x} \to \mathsf{x}') \vee (\mathsf{x}' \to \mathsf{x})$. Exprimer des propriétés de $\mathrm{FO}(\lambda)$ sur $\mathrm{PURG}(N)$ revient à oublier l'orientation des arcs dans le graphe. En dépit de cet affaiblissement, le *model-checking* de cette logique est toujours indécidable. Pour exploiter un raisonnement similaire à celui décrit dans la preuve du théorème 51 on va s'attacher, à nouveau, à identifier les marquages en blocage et analyser leur environnement. Pour $\mathrm{FO}(\lambda)$, on va incorporer les marquages rencontrés dans la simulation par des 3-cycles. Ainsi, l'absence de 3-cycles ainsi qu'un environnement sans aucun tel cycle caractérise ces marquages en blocages.

Proposition 54. $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\lambda))$ *est indécidable.*

Démonstration. On exploite le fait que la logique $\mathrm{FO}(\lambda)$ permet d'exprimer qu'un état x appartient à un cycle non-orienté de logueur trois. Voici une formule possible pour exprimer une telle propriété :

$$3cycle(\mathsf{x}) \stackrel{\text{def}}{=} \exists \mathsf{y} \exists \mathsf{z}\ (\lambda(\mathsf{x},\mathsf{y}) \wedge \lambda(\mathsf{y},\mathsf{z}) \wedge \lambda(\mathsf{z},\mathsf{x})) \wedge \neg(\lambda(\mathsf{x},\mathsf{x}) \vee \lambda(\mathsf{y},\mathsf{y}) \vee \lambda(\mathsf{z},\mathsf{z}))$$

À présent, soient N_1 et N_2 deux réseaux de Petri avec des ensembles de places identiques. Pour tout entier i, $1 \leq i \leq 3$, on ajoute à chacun des réseau une nouvelle place p_i et des transitions t_i telle que p_1 contient initialement un unique jeton, et les places p_2 et p_3 sont vides. Les transition t_i consomment un jeton de p_i et en place un dans la place $p_{i+1 \mod 3}$. Les réseaux résultant ont les même ensembles d'accessibilité si et seulement si les réseaux N_1 et N_2 ont les même ensembles d'accessibilité. Ainsi l'égalité des ensemble d'accessibilité est indécidable pour les réseaux pour lesquelles chaque marquage accessible appartient à un 3-cycle. À présent supposons que N_1 et N_2 aient cette propriété, soit $\overline{N}$ le réseau obtenu à partir de N_1 et N_2 comme dans la démonstration du théorème 51 (voir également la figure 3.5). On peut supposer sans perte de généralité que chaque transition de N_1 et N_2 modifient le marquage (les autres transitions n'affectent pas l'ensemble d'accessibilité et peuvent être identifiées et supprimées). Ainsi le graphe d'accessibilité des réseaux modifiés N_1 et N_2 n'ont pas de 1-boucles, ce qui est indispensable pour l'efficacité de la formule $3cycle(\mathsf{x})$. Ainsi, dans le graphe d'accessibilité de $\overline{N}$, les marquages en blocage sont exactement ceux qui ne portent pas de 3-cycle, ni de 1-boucle, et qui sont entourés d'états sans 3-cycle :

$$bloq(\mathsf{z}) \stackrel{\text{def}}{=} \neg\lambda(\mathsf{z},\mathsf{z}) \ \wedge\ \neg 3cycle(\mathsf{z}) \wedge\ \forall \mathsf{x}\ \lambda(\mathsf{z},\mathsf{x}) \Rightarrow \neg 3cycle(\mathsf{x}).$$

L'égalité des ensembles d'accessibilité de N_1 et N_2 s'exprime alors par la formule, φ_{indec}, suivante :

$$\forall \mathsf{z}\ bloq(\mathsf{z}) \Rightarrow (\exists \mathsf{z}_1\ \lambda(\mathsf{z},\mathsf{z}_1) \wedge \varphi_l(\mathsf{z}_1)) \wedge (\exists \mathsf{z}_2\ \lambda(\mathsf{z},\mathsf{z}_2) \wedge \neg\varphi_l(\mathsf{z}_2))$$

avec $\varphi_l(\mathsf{z}) \stackrel{\text{def}}{=} \exists \mathsf{y}\ \lambda(\mathsf{z},\mathsf{y}) \wedge \lambda(\mathsf{y},\mathsf{y})$. On a $\mathrm{Reach}(N_1) = \mathrm{Reach}(N_2)$ si et seulement si $\overline{N} \models \varphi_{indec}$. d'après le théorème 30, le problème $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\lambda))$ est indécidable. □

Le fragment $\mathrm{ML}(\Box, \Box^{-1})$

Pour établir l'indécidabilité du problème $\mathrm{VAL}^{\mathrm{URG}}(\mathrm{ML}(\Box, \Box^{-1}))$, on effectue à nouveau une réduction du problème de l'égalité des ensembles d'accessibilité pour les réseaux de Petri.

Proposition 55. $\mathrm{VAL}^{\mathrm{URG}}(\mathrm{ML}(\Box, \Box^{-1}))$ *est indécidable.*

Démonstration. On considère à nouveau deux réseaux N_1 et N_2 ayant les mêmes ensembles de places. On réalise à nouveau la construction du réseau $\overline{N}$ faite en § II.3, mais on donne cette fois une formule modale φ_{indec} (indépendante des réseaux N_1 et N_2) qui, de la même façon, entraîne l'équivalence suivante : N_1 et N_2 ont le même ensembles de marquages si et seulement si $\mathrm{PURG}(\overline{N}), M \models \varphi_{indec}$ pour tout marquage M dans $\mathrm{Reach}(\overline{N})$. Cette formule exprime que pour tout état en blocage, il existe un prédécesseur (issu de N_1) qui peut réaliser deux pas en avant, et un autre prédécesseur qui ne peut pas : $\varphi_{indec} \stackrel{\text{def}}{=} \Box\, \bot \Rightarrow\ (\Diamond^{-1}\Diamond\Diamond\top\ \wedge\ \Diamond^{-1}\Box\Box\, \bot)$. La formule φ_{indec} est sémantiquement équivalente à la formule φ_{fo} définie ainsi :

$$\forall \mathsf{z}\ (\neg\exists \mathsf{z}'\ \mathsf{z} \to \mathsf{z}') \Rightarrow (\exists \mathsf{z}_1, \mathsf{z}_2, \mathsf{z}_3\ (\mathsf{z}_1 \to \mathsf{z}) \wedge (\mathsf{z}_1 \to \mathsf{z}_2) \wedge (\mathsf{z}_2 \to \mathsf{z}_3)) \wedge (\exists \mathsf{z}_1\ (\mathsf{z}_1 \to \mathsf{z}) \wedge \forall \mathsf{z}_2, \mathsf{z}_3\ \neg((\mathsf{z}_1 \to \mathsf{z}_2) \wedge (\mathsf{z}_2 \to \mathsf{z}_3))).$$

□

Ce résultat d'indécidabilité est minimal : en § II.5, on démontre la décidabilité d'une variante étendue de $\mathrm{VAL}^{\mathrm{URG}}(\mathrm{ML}(\Box))$ où la modalité inversée $\Box^{-1}$ n'est pas présente. D'autre part, en traduisant les formules de $\mathrm{ML}(\Box, \Box^{-1})$ en $\mathrm{FO}(\to)$ restreint à deux variables, on obtient une nouvelle preuve de l'indécidabilité du problème $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\to))$ restreint à seulement deux variables.

$\mathrm{FO}(\to)$ **restreint aux formules positives ou en avant**

Bien que les problèmes $\mathrm{VAL}^{\mathrm{URG}}(\mathrm{ML}(\Box, \Box^{-1}))$ et $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\to))$ soient indécidables en général, dans § II.2 nous avons isolé des fragments décidables. De façon analogue, on pourrait obtenir des problèmes décidables en considérant des fragments de la logique du premier ordre similaires. Nous allons démontrer, à présent, qu'il n'en est rien. Nous considérons les restrictions *en avant*, et *positif* de $\mathrm{FO}(\to)$ et nous montrons que leurs problèmes respectifs de *model-checking* sont indécidables. Les *formules positives*, sont telles que les propositions atomiques ne peuvent aparaître que dans le champs d'un nombre *pair* de l'opérateur de négation. On note $\mathrm{FO}^{+}(\sigma)$ l'ensemble de formules positives de FO dont les prédicats sont dans σ.

Proposition 56. $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}^{+}(\to))$ *est indécidable.*

Démonstration. Nous utilisons à nouveau le schéma précédemment introduit. Soit N_1 et N_2 deux réseaux et $\overline{N}$ le réseau obtenu par la combinaison représentée figure 3.5. Nous introduisons une formule positive, φ_{indec}, telle que l'inclusion $\mathrm{Reach}(N_2) \subseteq \mathrm{Reach}(N_1)$ est vérifiées si et seulement si $\mathrm{PURG}(\overline{N}) \models \varphi_{indec}$:

$$\varphi_{indec} \stackrel{\text{def}}{=} \forall \mathsf{z}\ \exists \mathsf{z}_1\ \exists \mathsf{y}_\ell\ \exists \mathsf{z}'\ (\mathsf{z} \to \mathsf{z}') \vee ((\mathsf{z}_1 \to \mathsf{z}) \wedge (\mathsf{z}_1 \to \mathsf{y}_\ell) \wedge (\mathsf{y}_\ell \to \mathsf{y}_\ell))$$

La formule considère un marquage arbitraire M. Si M n'est pas en blocage, rien n'est exigé par la formule. En revanche, lorsque M est en blocage, alors φ_{indec} impose aux états M_1 et $M_{\circlearrowleft}$ que M_1 est un prédécesseur immédiat de M et $M_{\circlearrowleft}$ ensuite, $M_{\circlearrowleft}$ porte une 1-boucle.

Par construction du réseau $\overline{N}$, la formule φ_{indec} est vérifiée si et seulement si tous les états en blocage, accessibles dans $\overline{N}$ (en particulier les simulations de N_2) peuvent être atteints dans N_1. Ce qui est exactement $\mathrm{Reach}(N_2) \subseteq \mathrm{Reach}(N_1)$. □

Il peut sembler paradoxale que nous n'ayons pas de résultat pour le problème suivant.

Problème ouvert 1. Décidabilité de $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}^{+}(\xrightarrow{*}))$.

Une *formule en avant* est une formule où toute occurrence de deux variables comme ceci : $\mathsf{x} \to \mathsf{y}$ est dans la portée d'une suite de quantification de la forme $Q_1\,\mathsf{x} \ldots Q_2\,\mathsf{y}$ où x est liée avant y. Soit $\mathrm{FO}_f(\sigma)$ l'ensemble des formules en avant de FO dont les prédicats sont dans σ.

Proposition 57. $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}_f(\to))$ *est indécidable*

Démonstration. Nous utilisons à nouveau le schéma précédemment introduit (et le problème d'égalité des ensembles d'accessibilité). Soit N_1 et N_2 deux réseaux et $\overline{N}$ le réseau obtenu par la combinaison représentée figure 3.5. Nous introduisons une formule en avant, φ_{indec} telle que $\mathrm{Reach}(N_2) = \mathrm{Reach}(N_1)$ si et seulement si $\mathrm{PURG}(\overline{N}) \models \varphi_{indec}$:

$$\varphi_{indec} \stackrel{\text{def}}{=} \forall \mathsf{z}_2\, \exists \mathsf{z}_1\, \forall \mathsf{z}\, \exists\, \mathsf{y}_\ell\ \ \exists \mathsf{z}'\, (\mathsf{z}_2 \to \mathsf{z}) \Rightarrow ((\mathsf{z} \to \mathsf{z}') \vee \psi(\mathsf{z}_1, \mathsf{z}_2, \mathsf{z}, \mathsf{y}_\ell))$$

$$\psi(\mathsf{z}_1, \mathsf{z}_2, \mathsf{z}, \mathsf{y}_\ell) \stackrel{\text{def}}{=} (\mathsf{z}_1 \to \mathsf{z}) \wedge (\mathsf{y}_\ell \to \mathsf{y}_\ell)\ \wedge\ ((\mathsf{z}_1 \to \mathsf{y}_\ell) \Leftrightarrow \neg(\mathsf{z}_2 \to \mathsf{y}_\ell))$$

Les formules en avant rendent plus délicate la quantification sur les marquages en blocage. M. Avant de discuter sur la façon dont la formule φ_{indec} permet la réduction, quelques précisions sur les quantifications : cette formule cherche à quantifier la variable z, mais la contrainte « en avant » impose de commencer par quantifier la variable z_2, puis z_1, et seulement ensuite z. Ceci ne constitue pas un problème dans la mesure où, une fois la variable z_2 fixée, la variable z_1 peut à son tour être fixée, enfin, z, peut être choisie. La formule φ_{indec} a pour but de capturer la situation décrite dans la figure 3.5, avec potentiellement les rôles de M_1 et M_2 échangés. Plus précisément, la formule considère un marquage quelconque M_2, un marquage correspondant M_1 (lorsqu'il existe), et un autre marquage quelconque M. Si M_2 et M ne sont pas liés, la formule φ_{indec} n'exige rien. Si les états M_2 et M sont connectés et que l'état M n'est pas en blocage, à nouveau il n'y a aucune exigeance. Dans les autres cas, lorsque M_2 et M sont connectés et que M est en blocage. Dans ce cas, il doit exister un marquage $M_{\circlearrowleft}$ (qui est une valuation pour y_ℓ) de sorte que la formule ψ est vérifiée pour $(M_1, M_2, M, M_{\circlearrowleft})$. La sous-formule ψ vérifie que l'état en blocage, M, est accessible à la fois dans N_1 et N_2, voir la figure 3.5. Ainsi, on a $\mathrm{Reach}(N_1) = \mathrm{Reach}(N_2)$ si et seulement si $\mathrm{PURG}(\overline{N}) \models \varphi_{indec}$. Ceci démontre l'affirmation recherchée. □

À nouveau, on ignore ce qu'il advient dans le cas où seul le prédicat d'accessibilité est autorisé.

Problème ouvert 2. Décidabilité du problème $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}_f(\xrightarrow{*}))$.

II.5 Décidabilité de fragments

Dans ce paragraphe, on s'intéresse à des restrictions de $\mathrm{FO}(\to)$ pour lesquelles les problèmes de *model-checking* ou de validité sont décidables.

Fragment exsistentiel

Les résultats d'indécidabilité que nous avons présenté reposent sur un principe commun : identifier un motif local qui doit apparaître universellement dans le graphe d'accessibilité. Ici, nous énonçons le résultat suivant : la décidabilité de $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\to))$ restreint au fragment existentiel. Cela illustre le caractère universelle et pas seulement local de la propriété qui entraîne l'indécidabilité. On note le fragment existentiel de FO, $\exists\mathrm{FO}$, il est formé par les formules quantifiées en forme prénexe où n'apparaisse que des quantificateurs existentiels.

Proposition 58. $\mathrm{MC}^{\mathrm{URG}}(\exists\mathrm{FO}(\to, =))$ *est décidable.*

Démonstration. Soit un réseau de Petri, $N = (P, T, F, M_0)$, avec pour ensemble d'accessibilité, $\mathrm{Reach}(N)$, et $|P| = n$. La décidabilité provient de deux propriétés clés :

(1) Étant donné une formule de Presburger $\varphi(\vec{\mathsf{x}}_1, \ldots, \vec{\mathsf{x}}_\alpha)$ avec l'entier $n \times \alpha$ qui correspond aux variables libres, et tel que chaque vecteur $\vec{\mathsf{x}}_i$ est une formé de n variables distinctes, interprété comme un marquage de N, il est possible de décider si $\varphi(M_1, \ldots, M_\alpha)$ est vérifié pour des marquages (non-nécessairement distincts) $M_1, \ldots, M_\alpha$ in $\mathrm{Reach}(N)$. La proposition 31 correspond au cas $\alpha = 1$.

(2) Il est possible de construire une formule de Presburger close, $\varphi_{\rightarrow}(\vec{\mathsf{x}}_1, \vec{\mathsf{x}}_2)$, telle que, pour chaque couple de marquages M_1, M_2, la formule $\varphi_{\rightarrow}(M_1, M_2)$ est vérifié si et seulement si il existe une transition $t \in T$, telle que $M_1[t\rangle M_2$.

Avant de démontrer les points (1) et (2), on décrit en quoi ces points induisent la décidabilité de $\mathrm{MC}^{\mathrm{URG}}(\exists\mathrm{FO}(\rightarrow, =))$. Considérons la formule $\psi = \exists\, \mathsf{x}_1, \ldots, \mathsf{x}_\alpha\ \psi'$ où ψ' est une formule sans quantification, avec des formules atomiques de la forme $\mathsf{x}_i \rightarrow \mathsf{x}_j$ et $\mathsf{x}_i = \mathsf{x}_j$. En utilisant (2), on produit une formule de Presburger sans quantification $\varphi(\vec{\mathsf{x}}_1, \ldots, \vec{\mathsf{x}}_\alpha)$ de sorte que pour tous marquages $M_1, \ldots, M_\alpha$ dans $\mathrm{Reach}(N)$, la formule $\varphi(M_1, \ldots, M_\alpha)$ est vrai si et seulement si $\mathrm{PURG}(N), \mathbf{v} \models \psi'$ où $\mathbf{v}(\vec{\mathsf{x}}_i) = M_i$ pour les entiers i, $1 \leq i \leq \alpha$. Ensuite, d'après (1), on peut décider si la formule $\varphi(M_1, \ldots, M_\alpha)$ est vérifiée pour certains marquages $M_1, \ldots, M_\alpha \in \mathrm{Reach}(N)$. Ceci est équivalent à $\mathrm{URG}(N) \models \psi$.

Reste à établir les points (1) et (2). Pour le second, la formule $\varphi_{\rightarrow}(\vec{x}_1, \vec{x}_2)$ exprime un codage des contraintes pour le tirage des transitions, $M[t\rangle M'$:

$$\bigvee_{t \in T} \Big(\bigwedge_{p \in P} \vec{\mathsf{x}}_1(p) \geq F(p,t) \Big) \wedge \Big(\bigwedge_{p \in P} \vec{\mathsf{x}}_2(p) = \vec{\mathsf{x}}_1(p) - F(p,t) + F(t,p) \Big).$$

En ce qui concerne le point (1), on adapte la démonstration de la proposition 31. On construit un réseau de Petri N' qui simule α répliques du réseau N. Formellement, N' est défini par l'union disjointe de α exemplaires de N. Son marquage initial est le marquage M_0 dupliqué α fois. Pour tous marquages $M_1, \ldots, M_\alpha$ on dispose à présent de l'équivalence suivante : ces marqueges sont accessibles dans N et satisfont la formule $\varphi(M_1, \ldots, M_\alpha)$ si et seulement si $(M_1, \ldots, M_\alpha)$ peut être atteint dans le réseau N' et la formule $\varphi(M_1, \ldots, M_\alpha)$ est vérifiée. C'est une application immédiate de la proposition 31 sur le réseau N' et la formule φ. □

À nouveau, la décidabilité est préservée lorsqu'on autorise des propriétés Presburger définissables sur les marquages ainsi que les transitions étiquetées de la forme $\xrightarrow{t}$.

Corollaire 59. *Le problème* $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\rightarrow, =))$ *restreint aux combinaisons booléennes de formules existentielles est décidable.*

Une conséquence directe de ce résultat est la décidabilité du problème suivant :

Donnée : Un graphe fini orienté $\mathcal{G} = (V, E)$ et un réseau de Petri N

Question : Existe-t-il un sous-graphe de $(\mathrm{Reach}(N), \rightarrow)$ isomorphe à $\mathcal{G}$?

On peut noter que la décidabilité de ce problème ne permet pas pour autant d'appliquer le théorème de Gaifman (théorème 32). En effet, on ne peut par énumérer précisément les voisinages possibles, uniquement de façon positive (il existe des points dont le voisinage contient *au moins* telle structure).

Un nouvelle fois, on ne sait pas ce qu'il en est pour les variantes incluant un prédicat d'accessibilité.

Problème ouvert 3. Décidabilité des problèmes $\mathrm{MC}^{\mathrm{URG}}(\exists\mathrm{FO}(\xrightarrow{*}))$ et $\mathrm{MC}^{\mathrm{URG}}(\exists\mathrm{FO}(\xrightarrow{*}, \rightarrow))$.

Logique $\mathrm{ML}(\Box)$ avec contraintes arithmétiques

En § II.4 on a démontré l'indécidabilité de $\mathrm{VAL}^{\mathrm{URG}}(\mathrm{ML}(\Box, \Box^{-1}))$. À l'opposé du résultat d'indécidabilité de la logique *en avant*, ce résultat requière la modalité inverse. La proposition 60, ci-dessous, démontre la décidabilité du problème de validité pour $\mathrm{ML}(\Box)$, même en présence de prédicats atomiques arithmétiques.

Proposition 60. *Le problème de validité,* $\mathrm{VAL}^{\mathrm{URG}}(\mathrm{PAML}(\Box))$*, est décidable.*

Démonstration. Soit un réseau de Petri, N, et une formule φ de $\mathrm{PAML}(\Box)$. D'après le lemme 61 (introduit ci-après) l'ensemble de marquages qui satisfont la formule $\neg\varphi$ est de façon effective semi-linéaire. Soit $X_{\neg\varphi}$ cet ensemble. Prouver la validité de φ revient à vérifier qu'aucun élément de $X_{\neg\varphi}$ n'est accessible dans N. Ceci est décidable grâce une nouvelle fois à la proposition 31. □

Lemme 61. *Étant donné un réseau N avec n places et une formule φ de* $\mathrm{PAML}(\Box)$*, l'ensemble des marquages dans $\mathbb{N}^n$ qui satisfont la formule φ dans la structure* $\mathrm{UG}(N)$ *est semi-linéaire, de façon effective.*

Démonstration. Cette propriété est démontrée par récurrence sur la structure de la formule φ, et s'appuie sur le fait que les ensembles semi-linéaires forment une algèbre de Boole effective. On se repose également sur l'observation que si un ensemble X est semi-linéaire, alors l'ensemble $pre(X) = \{M \in \mathbb{N}^n : \exists\, M' \in X,\ M \to M'\}$ est semi-linéaire de façon effective. Ce dernier contient tous les marquages dont l'image immédiate par une transition de N se trouve dans l'ensemble X.

Chacune des formules atomiques est une formule de Presburger sans quantification, et définit donc un ensemble semi-linéaire de marquages. Durant la récurrence sur la structure de la formule φ, les sous-formules externes sont traitées à l'aide des opérateurs booléens sur les ensembles semi-linéaires de façon naturelle.

On doit ultimement démontrer que la formule $\Box\psi$ définit un ensemble semi-linéaire dès lors que la formule ψ le fait. L'hypothèse de récurrence énonce que l'ensemble X_ψ des marquages qui satisfont la formule ψ est semi-linéaire. L'ensemble qui satisfait $\Box\psi$ est l'ensemble : $\mathbb{N}^n \setminus pre(\mathbb{N}^n \setminus X_\psi)$, qui est donc lui aussi semi-linéaire. □

Ce résultat peut être étendu en autorisant les étiquettes sur les transitions.

II.6 Dureté des problèmes décidables

Une partie des algorithmes de décision que nous proposons reposent sur des tests d'accessibilité dans les réseaux de Petri. Or, même si on ne connaît pas encore avec précision la complexité de ce problème, les algorithmes actuel les plus performants ne sont pas primitif récursifs. Dans ce paragraphe, nous établissons la dureté de nos problèmes vis-à-vis de l'accessibilité dans les réseaux de Petri. Précisément, ce dernier problème peut être réduit aux problèmes suivants : $\mathrm{MC}^{\mathrm{URG}}(\mathrm{ML}(\Box, \Box^{-1}))$ et $\mathrm{MC}^{\mathrm{URG}}(\exists\mathrm{FO}(\to))$.

En outre, nos algorithmes font également usage de la semi-linéarité effective des ensembles d'accessibilité ou des relations (voir la proposition 46, par exemple). La proposition qui suit illustre que, même dans le cas des réseaux de Petri bornés, le problème $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\to))$ est d'une grande complexité algorithmique.

Proposition 62. *Le problème* $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\to))$ *limité aux réseaux de Petri bornés est décidable, mais de complexité non primitive-récursive.*

Démonstration. On effectue une réduction du problème d'inclusion finie pour les réseaux de Petri. Mayr et Meyer on montré que ce problème a une complexité non primitive récursive [110]. Soient N_1 et N_2 deux réseaux bornés ayant les mêmes ensembles de places. Soit à présent le réseau $\overline{N}$ construit de façon

similaire au § II.3. Ce réseau est également borné. La formule φ de $\mathrm{FO}(\rightarrow)$ qui vérifie l'inclusion est dérivée de celle présentée en § II.3 :

$$\forall \mathsf{z}\ (\neg\exists \mathsf{z}'\ \mathsf{z} \rightarrow \mathsf{z}') \Rightarrow (\exists \mathsf{z}_2\ \mathsf{z}_2 \rightarrow \mathsf{z} \wedge \neg\varphi_l(\mathsf{z}_2))$$

avec $\varphi_l(\mathsf{x}) \stackrel{\text{def}}{=} \exists \mathsf{y}\ (\mathsf{x} \rightarrow \mathsf{y} \wedge \mathsf{y} \rightarrow \mathsf{y})$. La construction assure que $\mathrm{Reach}(N_1) \subseteq \mathrm{Reach}(N_2)$ si et seulement si $\mathrm{URG}(\overline{N}) \models \varphi$. En fait, un marquage en blocage est atteint dans N_2 ou dans N_1. Pour satisfaire la formule il est nécessaire de satisfaire à la fois l'accessibilité dans les deux réseaux. À nouveau, on peut remarquer que la formule est indépendante des réseaux. □

Nous avons vu que le problème $\mathrm{VAL}^{\mathrm{URG}}(\mathrm{ML}(\Box)$ est décidable par une réduction au problème d'accessibilité des réseaux de Petri (proposition 60). À présent, nous donnons une réduction réciproque de la non-accessibilité à $\mathrm{VAL}^{\mathrm{URG}}(\mathrm{ML}(\Box)$.

Proposition 63. *Il y a une réduction logarithmique en espace du problème de non-accessibilité des réseaux de Petri vers le problème* $\mathrm{VAL}^{\mathrm{URG}}(\mathrm{ML}(\Box))$.

Démonstration. Sans perte de généralité, on peut considérer que le problème de non-accessibilité est limité au marquage $\vec{0}$ (aucun jeton dans aucune place). Considérons le réseau $N = (P, T, F, M_0)$ où on suppose (sans perdre de généralité) que chacune des transitions possède au moins une place dans son près-ensemble. On construit un réseau N' à partir de N en ajoutant une nouvelle transition t_p pour chaque place $p \in P$. Les transitions ainsi ajoutées sont mises en 1-boucle avec leur place, $F'(p, t_p) = 1 = F'(t_p, p)$ et $F'(p', t_p) = 0 = F'(t_p, p')$ pour tous les $p' \in P$ avec $p' \neq p$. De façon intuitive, chaque transition t_p révèle, pour un marquage M, la présence de jetons dans la place p de part la présence d'au moins une transition issue de M dans le graphe d'accessibilité de N. Ainsi, on a $\vec{0} \notin \mathrm{Reach}(N)$ si et seulement si tout marquage $M \in \mathrm{Reach}(N')$, certaines transitions peuvent être tirées : $(D, \rightarrow), M \models \Diamond\top$. À nouveau, notre construction repose sur une formule unique. □

Proposition 64. *Il y a une réduction logarithmique en espace du problème d'accessibilité des réseaux de Petri vers le problème* $\mathrm{MC}^{\mathrm{URG}}(\mathrm{ML}(\Box, \Box^{-1}))$.

Démonstration. Nous allons réduire l'accessibilité, dans le réseau N, du marquage M_2 depuis M_1 à une instance de $\mathrm{MC}^{\mathrm{URG}}(\mathrm{ML}(\Box, \Box^{-1}))$ pour un réseau plus grand, noté $\overline{N}$. Comme on peut le voir sur la figure 3.6, l'idée consiste à introduire un nouveau marquage M_w pour lequel l'existence d'un chemin de longueur supérieure à un révèle l'existence d'un chemin du marquage initial, M_1 vers M_2 dans la structure $\mathrm{PURG}(N)$. De façon à pouvoir atteindre le marquage M_w grâce à une formule de ML, ce dernier est placé près du marquage initial. Schématiquement, le marquage initial, M_0 du réseau $\overline{N}$ contient une place unqieu, notée p_i, pour laquelle les deux transitions t_{try} et t_0 sont en concurrence. La transition t_{try} placel'unique jeton de p_i dans une place p_w et produit ainsi le marquage M_w *où aucune place ne possède de jeton*. La transition t_0 produit le marquage M_1 dans les places du réseau N et déplace le jeton de contrôle de p_i vers une autre place p_c qui est en 1-boucle avec toutes les transitions originales du réseau N. Ceci permet de réaliser une simulation à partir du marquage original. Cette dernière suit sont cours normal et peut à tout moment être interrompue sur n'importe quel marquage qui domine M_2. La transition t_{stop} consomme M_2 depuis les places de N et déplace le jeton de contrôle de la place p_c à la place $p_{w'}$. Ultimement, le jeton de contrôle est déplacé de la place $p_{w'}$ à p_w en tirant t_{win}. Le marquage M_w est atteint après le tirage de $t_{stop}\, t_{win}$, si et seulement si $\overline{M_2}$ est atteint. Ainsi M_2 est accessible depuis M_1 si et seulement si M_w est accessible depuis $\overline{M_1}$ (sa restriction aux places de N correspond à M_1). De façon équivalente, on peut dire que le marquage M_w possède un prédécesseur différent de M_0. La forme du graphe d'accessibilité nous permet de l'exprimer dans une formule locale de $\mathrm{ML}(\Box, \Box^{-1})$:

$$\varphi \ := \ \Diamond(\Box \bot \wedge \Diamond^{-1}\Diamond^{-1}\top).$$

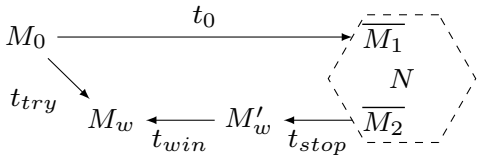


FIGURE 3.6 – Graphe d'accessibilité dans la preuve de dureté du *model-checking* de $\mathrm{ML}(\Box, \Box^{-1})$.

Sans perte de généralité, on peut considérer que le marquage M_1 n'est pas en blocage et aussi que $M_2 \neq M_1$. La formule φ impose que le marquage M_0 possède un successeur en blocage ainsi qu'un chemin qui y abouti de longueur au moins 2. Comme le successeur est en blocage, on sait qu'il n'est pas $\overline{M_1}$ mais M_w résultat du tirage de la transition t_{try}. Lee chemin allant de M_0 à M_w est de longueur exactement 1 et M_0 n'a pas de prédécesseur. Donc, le chemin de longueur 2 qui atteint M_w ne peut provenir de t_{try} et résulte de la transition t_{win}. Ensuite, le marquage M_w est accessible de $\overline{M_1}$, ce qui entraîne que M_2 est accessible depuis M_1 dans le réseau N. □

La preuve de la proposition 64 peut être adaptée à $\exists\mathrm{FO}(\rightarrow)$ pour lequel nous avons montré la décidabilité du *model-checking*. En fait cette réduction est possible avec une unique variable.

Proposition 65. *Il y a une réduction logarithmique en espace du problème d'accessibilité dans les réseaux de Petri vers* $\mathrm{MC}^{\mathrm{URG}}(\exists\mathrm{FO}(\rightarrow))$ *restreint à une variable unique.*

II.7 FO avec un prédicats d'accessibilité

Dans ce paragraphe, on considère un ensemble de variantes de langages du premier ordre comportant un symbole de relation d'accessibilité $\xrightarrow{*}$ ou $\xrightarrow{+}$, principalement sans inclure la relation en un pas $\rightarrow$. Ainsi les résultat d'indécidabilité de ces dialectes ne résultent pas directement du théorème 51. Cependant, les même schémas de preuve s'appliquent. En revanche, lorsque les ensembles d'accessibilité sont semi-linéaires on distingue un cas dont résulte un résultat indécidable (en incluant les deux prédicats, voir la proposition 70). Le dernier résultat établit l'indécidabilité du problème de *model-checking* des graphes des transitions non-étiquetés pour la logique $\mathrm{FO}(\rightarrow, \xrightarrow{*})$.

FO et relations d'accessibilité. On examine dans un premier temps la décidabilité des problèmes de *model-checking* pour l'accessibilité et l'accessibilité stricte.

Indécidabilité de $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\xrightarrow{+}))$

La décidabilité du problème $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\xrightarrow{+}))$ n'est pas directement liée à celle de $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\rightarrow))$. Néanmoins, il est possible d'adapter la construction réalisée en § II.3 mais en utilisant une formule, φ, de $\mathrm{FO}(\xrightarrow{+})$. On utilise le réseau $\overline{N}$ représenté sur la figure 3.5. précisément, la formule φ est définie comme suit :

$$\varphi \stackrel{\text{def}}{=} \forall\, \mathsf{z}\ bl\mathsf{c}(\mathsf{z}) \ \Rightarrow\ (\exists\, \mathsf{z}_1\ (\mathsf{z}_1 \xrightarrow{+} \mathsf{z}) \wedge \varphi_{gauche}(\mathsf{z}_1)) \wedge (\exists\, \mathsf{z}_2\ (\mathsf{z}_2 \xrightarrow{+} \mathsf{z}) \wedge \varphi_{droite}(\mathsf{z}_2))$$

avec

- $bl\mathsf{c}(\mathsf{z}) \stackrel{\text{def}}{=} \neg\exists \mathsf{z}'\ \mathsf{z} \xrightarrow{+} \mathsf{z}'$,
- $bc\mathsf{l}(\mathsf{y}) \stackrel{\text{def}}{=} \mathsf{y} \xrightarrow{+} \mathsf{y}\ \wedge\ \forall \mathsf{w}\ [\mathsf{y} \xrightarrow{+} \mathsf{w} \Rightarrow \mathsf{w} \xrightarrow{+} \mathsf{y}]$,
- $\varphi_{gauche}(\mathsf{z}) \stackrel{\text{def}}{=} [\exists\, \mathsf{y}\ \mathsf{z} \xrightarrow{+} \mathsf{y} \wedge bc\mathsf{l}(\mathsf{y})] \wedge\ [\forall \mathsf{y}\ \mathsf{z} \xrightarrow{+} \mathsf{y} \Rightarrow (bc\mathsf{l}(\mathsf{y})\ \vee\ bl\mathsf{c}(\mathsf{y}))]$,
- $\varphi_{droite}(\mathsf{z}) \stackrel{\text{def}}{=} [\exists \mathsf{y}\ \mathsf{z} \xrightarrow{+} \mathsf{y} \wedge \forall \mathsf{y}\ \mathsf{z} \xrightarrow{+} \mathsf{y} \Rightarrow bl\mathsf{c}(\mathsf{y})]$.

Lemme 66. $\mathrm{Reach}(N_1) = \mathrm{Reach}(N_2)$ *si et seulement si* $\mathrm{PURG}(\overline{N}) \models \varphi$.

La preuve repose une nouvelle fois sur les mêmes arguments, et ne sera pas rappelée ici. On déduit du lemme 66 le résultat recherché.

Corollaire 67. $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\xrightarrow{+}))$ *est indécidable. Ce résultat est vrai pour une formule fixée :* φ*, définie ci-dessus.*

Indécidabilité de $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\xrightarrow{*}))$

Il est encore possible d'adapter le schéma de preuve pour établir l'indécidabilité de $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\xrightarrow{*}))$. En particulier, dans $\mathrm{FO}(\xrightarrow{*})$, il n'est plus possible d'identifier les 1-boucles comme on l'a fait pour $\mathrm{FO}(\xrightarrow{+})$. La figure 3.7 présente le nouveau schéma.

Proposition 68. $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\xrightarrow{*}))$ *est indécidable.*

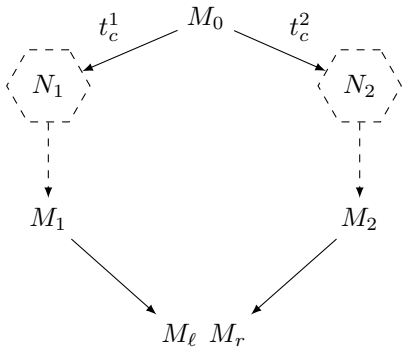


FIGURE 3.7 – Réseau de Petri $\overline{N}$, adapté pour $\mathrm{FO}(\xrightarrow{*})$

Démonstration. À partir des réseaux N_1 et N_2, on construit $\overline{N}$ représenté sur la figure 3.7. On définit également les formules suivantes :

— $blc(\mathsf{z}) \stackrel{\text{def}}{=} \forall \mathsf{w}\ \mathsf{z} \xrightarrow{*} \mathsf{w} \Rightarrow \mathsf{w} \xrightarrow{*} \mathsf{z}$,

— $prdl(\mathsf{z}) \stackrel{\text{def}}{=} \neg blc(\mathsf{z}) \wedge (\forall \mathsf{w}(\mathsf{z} \xrightarrow{*} \mathsf{w} \wedge \neg \mathsf{w} \xrightarrow{*} \mathsf{z}) \Rightarrow blc(\mathsf{w}))$.

Dans la figure 3.7, les marquages M_r et M_ℓ satisfont blc, et les marquages M_1 et M_2 satisfont $prdl$, mais aucun autre marquage ne satisfait ces formules.

La formule φ est définie ainsi :

$$\varphi \stackrel{\text{def}}{=} \forall \mathsf{z}\ (blc(\mathsf{z}) \Rightarrow \exists \mathsf{z}_1, \mathsf{z}_2\ (\mathsf{z}_1 \xrightarrow{*} \mathsf{z} \wedge prdl(\mathsf{z}_1) \wedge \mathsf{z}_2 \xrightarrow{*} \mathsf{z} \wedge prdl(\mathsf{z}_2) \wedge \neg \mathsf{z}_1 \xrightarrow{*} \mathsf{z}_2))$$

On peut remarquer que la proposition $\neg \mathsf{z}_1 \xrightarrow{*} \mathsf{z}_2$ garanti que les instanciations de z_1 et z_2 sont distinctes. Par construction, on a $\mathrm{Reach}(N_1) = \mathrm{Reach}(N_2)$ si et seulement si $\mathrm{PURG}(\overline{N}) \models \varphi$. Et donc l'indécidabilité. □

On a vu que $\mathrm{MC}^{\mathrm{UG}}(\mathrm{FO}(\rightarrow, =))$ est décidable (voir la proposition 44), cependant, remplacer la relation $\rightarrow$ par $\xrightarrow{*}$ et ajouter le prédicat $init$ conduit à l'indécidabilité.

Corollaire 69. $\mathrm{MC}^{\mathrm{UG}}(\mathrm{FO}(init, \xrightarrow{*}))$ *est indécidable.*

En fait, $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\xrightarrow{*}))$ se réduit à $\mathrm{MC}^{\mathrm{UG}}(\mathrm{FO}(init, \xrightarrow{*}))$ par un procédé de relativisation : $\mathrm{URG}(N) \models \varphi$ si et seulement si $\mathrm{UG}(N) \models \exists \mathsf{x}_0\ \ init(\mathsf{x}_0) \wedge f(\varphi)$ avec φ et $f(\varphi)$ dans $\mathrm{FO}(\xrightarrow{*})$, f obtenue par morphisme pour les connecteurs booléens et $f(\forall \mathsf{x}\ \psi) \stackrel{\text{def}}{=} \forall \mathsf{x}\ (\mathsf{x}_0 \xrightarrow{*} \mathsf{x}) \Rightarrow f(\psi)$.

Actuellement, nous n'apportons pas de réponse au problème qui suit.

Problème ouvert 4. Décidabilité de $\mathrm{MC}^{\mathrm{UG}}(\mathrm{FO}(\xrightarrow{*}))$.

Cas semi-linéaire. On a vu que $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\rightarrow,=))$ restreint aux réseaux de Petri dont les ensembles d'accessibilité sont semi-linéaires est décidable en utilisant une traduction en arithmétique de Presburger (voir la proposition 46). Ici, on s'intéresse à ce qui se produit si on ajoute la relation $\xrightarrow{*}$. On peut établir que le problème $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\rightarrow,\xrightarrow{*}))$ restreint aux réseaux dont les ensembles d'accessibilité sont semi-linéaire est indécidable, par une réduction du problème $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\rightarrow))$. Étant donné un réseau N et un énoncé $\varphi \in \mathrm{FO}(\rightarrow)$, on réduit la validité d'une formule φ de $\mathrm{PURG}(N)$ à celle d'une formule $\overline{\varphi}$ de $\mathrm{PURG}(\overline{N})$ où le réseau $\overline{N}$ est obtenu par extension de N et possède un ensemble d'accessibilité semi-linéaire. Le réseau $\overline{N}$ est définit à partir de N en ajoutant les nouvelles places p_0, p_1 et p_2 ; les transitions de N sont mises en auto-boucle avec p_1. Ensuite, on ajoute une nouvelle famille de transitions en auto-boucle avec p_2, chacune ajoute ou retire des jetons de chacune des places du réseau original N (leur contenu est donc modifié arbitrairement). L'ensemble de ses transitions forment le sous-réseau Br. Trois autre transitions sont ajoutées, voir la figure 3.8 pour une représentation schématique de $\overline{N}$ (le marquage initial M_0' de $\overline{N}$ restreint aux places de N est M_0, alors que $M_0'(p_0) = M_0'(p_1) = 1$ et $M_0'(p_2) = 0$). Notre construction force l'ensemble $\mathrm{Reach}(\overline{N})$ à être semi-linéaire tout en étant capable de d'identifier un sous-ensemble de configurations qui sont dans $\mathrm{Reach}(\overline{N})$ en bijection avec $\mathrm{Reach}(N)$; une façon de noyer exactement $\mathrm{Reach}(N)$ dans $\mathrm{Reach}(\overline{N})$. En fait, $\mathrm{Reach}(\overline{N})$ contient tous les marquages tels que la somme de p_1 et p_2 est égale à 1 et p_0 au plus 1. Cependant, si la transition t est tirée en premier, alors tous les marquages accessibles sont exactement ceux du réseau N (excepté que la place p_1 contient un jeton) ; la structure $\mathrm{PURG}(N)$ se plonge de façon isomorphique dans $\mathrm{PURG}(\overline{N})$. Jusqu'à ce que la transition t soit tirée, il est toujours possible de revenir au marquage M_0', en s'appuyant sur le réseau Brownien Br, mais c'est impossible une fois la transition tirée.

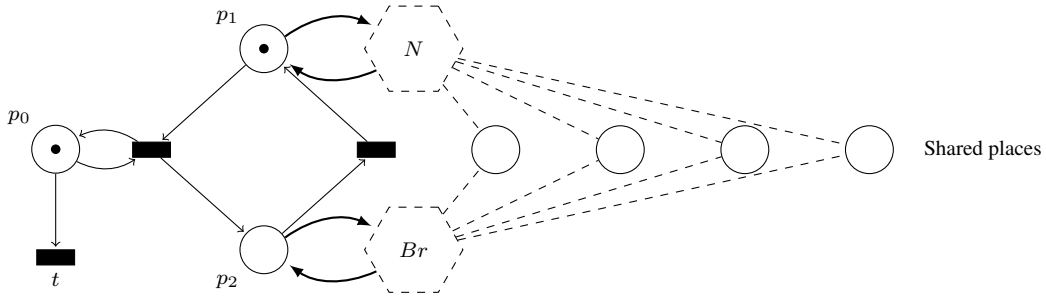


FIGURE 3.8 – Le réseau de Petri $\overline{N}$

Proposition 70. $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\rightarrow,\xrightarrow{*}))$ *restreint aux réseaux de Petri ayant un ensemble d'accessibilité semi-linéaire est indécidable.*

Démonstration. Dans un premier temps on utilise le prédicat $init$ (même s'il ne fait pas parti du langage de $\mathrm{FO}(\rightarrow,\xrightarrow{*})$) Soit la formule $\overline{\varphi} : \exists\, \mathsf{x}_0\, \mathsf{x}_1\ init(\mathsf{x}_0) \wedge \mathsf{x}_0 \rightarrow \mathsf{x}_1 \wedge \neg(\mathsf{x}_1 \xrightarrow{*} \mathsf{x}_0) \wedge f(\varphi)$ où $f(\cdot)$ est un morphisme pour les connecteurs booléens, et $f(\forall \mathsf{x}\ \psi) \stackrel{\text{def}}{=} \forall \mathsf{x}\ (\mathsf{x}_1 \xrightarrow{*} \mathsf{x}) \Rightarrow f(\psi)$ (relativisation). Dans $\overline{\varphi}$, x_0 est interprétée comme le marquage initial M_0', et x_1 est interprétée comme le successeur de x_0 duquel x_0 ne peut plus être atteint à nouveau. Ceci ne peut être réalisé qu'en tirant la transition t depuis M_0'. À présent la relativisation de toutes les autres variable à x_1 dans $\overline{\varphi}$ garantit que $\mathrm{PURG}(N) \models \varphi$ si et seulement si $\mathrm{PURG}(\overline{N}) \models \overline{\varphi}$. Pour s'affranchir de $init$, on construit le réseau de Petri $\overline{N}'$ quasi identique à $\overline{N}$. Ce réseau, $\overline{N}'$, possède une place supplémentaire p_0', marquée initialement par un seul jeton, et une nouvelle transition qui consomme ce jeton, et produit un jetons dans p_0 et p_1, qui sont initialement vides. Par construction, le marquage initial de $\overline{N}'$ est le seul marquage de $\mathrm{PURG}(\overline{N}')$ qui ne possède aucun arc entrant, et un seul arc sortant. Dans ce réseau modifié on utilise la formule $\overline{\varphi}'$ modifiée comme suit :

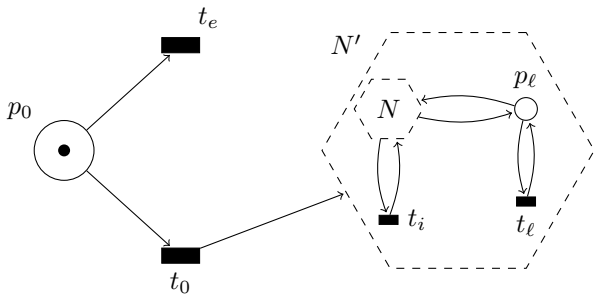
$$\exists\, \mathsf{x}_0'\, \mathsf{x}_0\, \mathsf{x}_1\ (\neg\exists\, \mathsf{y}\ \mathsf{y} \rightarrow \mathsf{x}_0')\ \wedge\ \mathsf{x}_0' \rightarrow \mathsf{x}_0\ \wedge\ \mathsf{x}_0 \rightarrow \mathsf{x}_1 \wedge (\neg \mathsf{x}_1 \xrightarrow{*} \mathsf{x}_0) \wedge f(\varphi)$$

Pour les même raisons que précédemment, $\mathrm{PURG}(N) \models \varphi$ si et seulement si $\mathrm{PURG}(\overline{N'}) \models \overline{\varphi'}$. □

Problème ouvert 5. Décidabilité de $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\xrightarrow{*}))$ restreint aux réseaux de Petri avec un ensemble semi-linéaire d'accessibilité.

La relation d'accessibilité et la structure $\mathrm{UG}(N)$. Le corollaire 69 établit le premier résultat d'indécidabilité pour la structure $\mathrm{UG}(N)$. Ici, on examine deux autres situations où le problème de *model-checking* des formules de $\mathrm{FO}(\rightarrow, \xrightarrow{*})$ est indécidable dans $\mathrm{UG}(N)$.

Proposition 71. $\mathrm{MC}^{\mathrm{UG}}(\mathrm{FO}(\rightarrow, \xrightarrow{*}))$ *est indécidable.*



p_ℓ : nouvelle place en auto-boucle avec chacune des transition de N.

Pour chacune des places p_i de N, il y a une transition t_i en auto-boucle avec elle.

FIGURE 3.9 – Le réseau de Petri $\overline{N}$ pour la démonstration de la proposition 71

Démonstration. On réduit le problème $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\xrightarrow{*}))$ à $\mathrm{MC}^{\mathrm{UG}}(\mathrm{FO}(\rightarrow, \xrightarrow{*}))$. Étant donné un réseau $N = (P, T, F, M_0)$ et une formule φ de $\mathrm{FO}(\xrightarrow{*})$, on construit $\overline{N}$ et φ' telles que $\mathrm{URG}(N) \models \varphi$ si et seulement si $\mathrm{UG}(\overline{N}) \models \varphi'$. La figure 3.9 illustre les points clés de la construction de $\overline{N}$.

On définit dans un premier temps le réseau $N' = (P', T', F', M_0')$ avec $P' = P \cup \{p_\ell\}$, $T' = T \cup \{t_i \mid p_i \in P'\}$, pour tous les couples (p, t) de $P \times T$, $F'(p, t) = F(p, t)$ et $F'(t, p) = F(t, p)$, pour tous les $p_i \in P', F(p_i, t_i) = F(t_i, p_i) = 1$, tous les $t \in T, F(t, p_\ell) = F(p_\ell, t) = 1$, et tous les $p \in P$, $M_0'(p) = M_0(p)$, et $M_0'(p_\ell) = 1$. Lorsqu'on se restreint aux places de P (toutes les places sauf p_ℓ), les marquages accessibles de N' coïncident avec celles de N. Par construction, p_ℓ ne contient, a tout moment, qu'un seul jeton. Dans $\mathrm{URG}(N')$, tout les marquages portent une 1-boucle. De même, tous les marquages de N' dans lesquels au moins une place est positive, possèdent une 1-boucle dans le graphe $UG(N')$. Réciproquement, le quadruplet $(0, 0, \ldots, 0)$, n'autorise aucune transition (la place p_ℓ, lorsqu'elle est vide, inhibe toutes les transitions).

A présent, on construit le réseau $\overline{N}$ à partir de N'. $\overline{N}$ possède les mêmes places et transitions que N', il possède aussi une place supplémentaire, p_0, et ainsi que deux transitions t_e et t_0. La transition t_e consomme un par un les jetons de p_0. La transition t_0 consomme un jeton de p_0 et produit le marquage M_0' dans les places de N'. Le marquage initial, $\overline{M_0}$, de $\overline{N}$ possède un seul jeton dans la place p_0.

On s'appuie d'abord sur les affirmations suivantes qui seront démontrées par la suite :

— Le graphe accessible de $\overline{N}$ est identique à celui de N, sauf pour la première transition, ainsi que les 1-boucles lesquelles n'ont aucune influence sur la validité des formules de $\mathrm{FO}(\xrightarrow{*})$.

— On peut construire une formule noté $\varphi_{init}(x) \in \mathrm{FO}(\rightarrow, \xrightarrow{*})$ qui n'est satisfaite que pour le marquage $\overline{M_0}$ dans $\mathrm{UG}(\overline{N})$.

En partant de ces affirmations, la validité d'une formule de $\mathrm{FO}(\xrightarrow{*})$ vis-à-vis de $\mathrm{URG}(N)$ se réduit à celle d'une formule de $\mathrm{FO}(\rightarrow, \xrightarrow{*})$ vis-à-vis de $\mathrm{UG}(\overline{N})$, en utilisant des techniques similaires à celles employées dans la preuve du corollaire 69. Ainsi, on commence par relativiser une formule donnée de $\mathrm{FO}(\xrightarrow{*})$ aux marquages de $\mathrm{UG}(\overline{N})$ qui sont accessibles depuis le marquage $\overline{M_0}'$ définie par $\overline{M_0}[t_0\rangle\overline{M_0}'$. Ce qui est réalisable dans $\mathrm{FO}(\rightarrow, \xrightarrow{*})$, car $\overline{M_0}'$ est l'unique marquage de $\overline{N}$ qui satisfait la formule

$\exists \mathsf{y}\ \varphi_{init}(\mathsf{y}) \wedge \mathsf{y} \to \mathsf{x} \wedge \mathsf{x} \to \mathsf{x}$. Pour conclure la démonstration, il nous suffit donc de démontrer les deux affirmations faites plus haut.

La première de ces affirmations résulte directement de la construction du réseau $\overline{N}$. La seconde peut être démontrée en posant :

$$\varphi_{init}(\mathsf{x}) \stackrel{\text{def}}{=} (\neg\, \mathsf{x} \to \mathsf{x}) \wedge (\exists\, \mathsf{y} \forall\, \mathsf{z}\ \mathsf{x} \to \mathsf{y} \wedge\ \neg(\mathsf{y} \to \mathsf{z}))$$

Cette formule contient une sous-formule $(\neg\, \mathsf{x} \to \mathsf{x})$ qui exprime simplement l'absence de 1-boucle, donc la formule $\varphi_{init}(\mathsf{x})$ ne peut être vérifiée que pour les marquages dont toutes les places de P' sont vides. De plus, $(\neg\, \mathsf{x} \to \mathsf{x})$ peut être satisfait dans un marquage x où la place p_0 peut contenir un nombre arbitraire de jetons. A présent, on considère les marquages dont les places de P' sont vides et qui possèdent un nombre quelconque de jetons dans la place p_0. Trois cas sont possibles. Tout d'abord, si on suppose que p_0 ne contient qu'un seul jeton (autrement dit, x est interprété par $\overline{M_0}$), alors $(\exists\, \mathsf{y} \forall\, \mathsf{z}\ \mathsf{x} \to \mathsf{y} \wedge\ \neg(\mathsf{y} \to \mathsf{z}))$ est satisfait : x possède un successeur y (atteint en tirant la transition t_e) qui est en blocage. Dans le cas où la place p_0 est vide, alors le marquage x ne possède aucun successeur. Si jamais p_0 contient au moins deux jetons, alors aucun successeur de x n'est en blocage : chacun des marquage atteint par la transition t_0 possède une 1-boucle et t_e peut être tirée au moins deux fois. Ainsi en rassemblant tous ces élément, le seul n-uplet de $\mathbb{N}^n$ qui satisfait $\varphi_{init}(\mathsf{x})$, est le marquage $\overline{M_0} = (1, 0, \ldots, 0)$, ce qui démontre la seconde affirmation. □

La proposition 71 reste vraie lorsque l'ensemble d'accessibilité du réseau est semi-linéaire de façon effective.

Proposition 72. $\mathrm{MC}^{\mathrm{UG}}(\mathrm{FO}(\to, \stackrel{*}{\to}))$ *est indécidable pour la classe des réseaux de Petri qui possèdent, de façon effective, un ensemble d'accessibilité semi-linéaire.*

Démonstration. On peut adapter les preuves des propositions 68, 70, et 71. □

Dans ce paragraphe, nous avons examiné en détail un ensemble de langage du premier ordre comportant un prédicat d'accessibilité. Nous avons obtenu des résultats d'indécidaqbilité, même lorsque les ensembles de marquages étaient semi-linéaires, et même en considérant la structure globale $\mathrm{UG}(N)$ plutôt que $\mathrm{URG}(N)$.

II.8 Synthèse

Nous avons conduit une étude approfondie de la décidabilité de problèmes de *model-checking* de dialectes de logique du premier ordre sur les graphes d'accessibilité, non-étiquetés, des réseaux de Petri. Nous avons mis en évidence la robustesse de l'indécidabilité du problème de base pour $\mathrm{FO}(\to)$, ou des fragments de logique modale dans $\mathrm{ML}(\Box, \Box^{-1})$. Nous avons également examiner la situation lorsque les ensembles d'accessibilité sont semi-linéaires. Le tableau 3.1 présente un résumé de ces résultats (on peut souligner que dès lors que la *relation* $\stackrel{*}{\to}$ est elle-même semi-linéaire, tous ces résultats deviennent décidables). Les résultats écrit en gras sont présentés explicitement dans ce chapitre, alors que les autres en sont des conséquences. On peut également observer que les résultats indécidables tiennent pour une formule fixée. formula. Dans ce sous-chapitre, nous avons étudié un ensemble de problèmes pour délimiter avec précision la frontière entre ceux qui sont décidables et ceux qui ne le sont pas. Par exemple, le problème $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\to))$ restreint à deux variables est indécidable, alors que $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\to))$ restreint au fragment existentiel est décidable (même si dans ce cas, le problème est au moins aussi difficile que l'accessibilité des réseaux de Petri). De même, sur le registre modal, $\mathrm{MC}^{\mathrm{URG}}(\mathrm{ML}(\Box, \Box^{-1}))$ est décidable (également aussi difficile que le problème d'accessibilité des réseaux de Petri) alors que $\mathrm{VAL}^{\mathrm{URG}}(\mathrm{ML}(\Box, \Box^{-1}))$ est indécidable. Ces résultats sont nombreux, mais on peut synthétiser un ensemble de règles de base.

Problème	$\sharp$	Quelconque	$\mathrm{Reach}(N)$ semi-linéaire (effectif)
$\mathrm{MC}^{\sharp}(\mathrm{FO}(\rightarrow))$	URG	**INDEC** (Cor. 50)	**DEC**
$\mathrm{MC}^{\sharp}(\mathrm{FO}(\overset{+}{\rightarrow}))$	URG	**INDEC** (Cor. 67)	ouvert
$\mathrm{MC}^{\sharp}(\mathrm{FO}(\overset{*}{\rightarrow}))$	URG	**INDEC** (Prop. 68)	ouvert
$\mathrm{MC}^{\sharp}(\mathrm{FO}(\rightarrow, \overset{*}{\rightarrow}))$	URG UG	INDEC **INDEC** (Prop. 71)	**INDEC** (Prop. 70) **INDEC** (Prop. 72)
$\mathrm{MC}^{\sharp}(\mathrm{FO}^{+}(\rightarrow))$	URG	**INDEC** (Prop. 56)	DEC
$\mathrm{MC}^{\sharp}(\mathrm{FO}_{f}(\rightarrow))$	URG	**INDEC** (Prop. 57)	DEC
$\mathrm{MC}^{\sharp}(\exists\mathrm{FO}(\rightarrow, =))$	URG	**DEC**† (Prop. 58)	DEC
$\mathrm{MC}^{\sharp}(\mathrm{FO}(\rightarrow))$ avec 1 variable	URG	**DEC**† (Prop. 65)	DEC
$\mathrm{MC}^{\sharp}(\mathrm{FO}(\rightarrow, =))$	UG	**DEC** (Prop. 44)	DEC
$\mathrm{MC}^{\sharp}(\mathrm{ML}(\Box))$	URG	**PSPACE-complet**	PSPACE-complet
$\mathrm{MC}^{\sharp}(\mathrm{ML}(\Box, \Box^{-1}))$	URG	**DEC**† (Prop. 48)	DEC
$\mathrm{VAL}^{\sharp}(\mathrm{ML}(\Box, \Box^{-1}))$	URG	**INDEC** (Prop. 55)	DEC
$\mathrm{VAL}^{\sharp}(\mathrm{PAML}(\Box))$	URG	**DEC**† (Prop. 60)	DEC

TABLE 3.1 – Synthèse des résultats († : équivalence avec l'accessibilité des réseaux de Petri)

1. L'indécidabilité de $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\rightarrow))$ est robuste pour de nombreux fragments de $\mathrm{FO}(\rightarrow)$ qui incluent à la fois la quantification existentielle et universelle (une seule alternance est suffisante).
2. La décidabilité résultant de restriction simple telle que les réseaux bornés, ou encore $\exists\mathrm{FO}(\rightarrow)$ abouti à des problèmes dont la complexité algorithmique est très élevée, parfois non-primitif récursif, ou équivalent au problème d'accessibilité des réseaux de Petri (voir § II.6).
3. Les deux remarques précédentes restent valides pour les variantes modales.

Nous conclurons ce paragraphe par quelques prolongements possibles de cette partie de mon travail. Une première piste pourrait consister à étudier le *model-checking* de fragments de langages du second ordre. Naturellement, dans la mesure où $\mathrm{MC}^{\mathrm{URG}}(\mathrm{FO}(\rightarrow))$ est indécidable, ceci requière à minima d'exclure les quantifications du premier ordre, tout en conservant quelques éléments de quantification du second ordre. Par exemple, une formule atomique pourrait être comme celle-ci : $X \Longrightarrow Y \overset{\text{def}}{\Leftrightarrow}$ pour tout $x \in X$, il existe un $y \in Y$ tel que $x \rightarrow y$ et pour tout $y \in Y$, il existe un $x \in X$ tel que $x \rightarrow y$. Il est en fait simple de voir que $\mathrm{MC}^{\mathrm{URG}}(\mathrm{MSO}(\Rightarrow))$ est indécidable, mais bien d'autres fragments de MSO pourraient être étudiés et comparés aux fragments examinés dans ce chapitre.

Une seconde direction consiste à voir l'ensemble des marquages accessibles comme un simple sous-ensemble de $\mathbb{N}^n$, et s'interroger sur des propriétés locales. Ainsi, il est trivial de déterminer s'il existe au moins un marquage accessible depuis le marquage initial, il est un peu plus complexe de déterminer s'il existe au moins un (un nombre fini de) marquage non-accessible. On peut aussi se questionner sur l'existence de boules homogènes de diamètre arbitraire (pour celles qui sont intégralement composées de marquages accessibles, on peut décider de leur existence). Un ensemble de questions de ce type pourraient être examinées en détail.

Chapitre 4

Applications

Dans ce chapitre nous présentons un ensemble de résultats à la portée plus appliquée. D'une part, il s'agit de travaux portant sur des problèmes liés à l'observation partielle des graphes réguliers, conduits dans le cadre de la thèse de Chédor, codirigée avec Jéron et dont certains résultats ont été obtenus avec Marchand et Pinchinat [43, 44, 45, 46]. On considérera plus particulièrement les problèmes de diagnosticabilité et d'opacité, ainsi que la génération formelle de tests. Dans un second temps nous présenterons des résultats obtenus avec Bertrand sur les graphes réguliers probabilistes [15].

I Problèmes liés à l'observation partielle

I.1 Opacité et diagnostic

Lorsqu'on s'intéresse à la supervision de systèmes, quels que soit leur nature, il est possible que l'ensemble du comportement du système ne soit pas disponible à l'observateur. Dans ce cas, une part importante de l'information sur l'état du système, à un moment donné, doit être déduite. Ces déductions résultent en générale d'une connaissance théorique du mode de fonctionnement du système. Par exemple, lorsqu'il s'agit d'un objet informatique l'observateur peu souvent s'appuyer sur une spécification formelle. Cette dernière comporte fréquemment des actions ou événements qui ne produisent aucun effet visible depuis l'extérieur au système.

Plusieurs grands champs se sont développés pour étudier certaines de ces questions. On peut citer la détection de *fuite d'information* [7, 26, 33], le *diagnostic* [134, 150, 89, 145, 20]. Par ailleurs, la question générale de la supervision a également été étudiée dans de nombreux travaux [76, 77, 53]

Pour ce qui est de la détection de flots d'information, la notion d'*opacité* a été développée, voir, par exemple, les travaux de Bryans *et al.*, Badouel *et al.*, ou encore Dubreil *et al.* [26, 7, 53]. Le *problème de l'opacité* consiste à déterminer si un observateur qui connaît le comportement du système peut déduire des informations *critiques* en observant son exécution (qui ne livre qu'une partie des informations). On peut citer : déterminer si un fichier est verrouillé, déduire la valeur d'une variable donnée, etc. Plus précisément, dans le contexte de l'opacité par état (*state-based opacity*) [34] on se donne un système à états finis, dont l'alphabet d'entrée est partitionné entre événements observables, et inobservables, et dont un ensemble d'états distingués représente un secret. Le problème de l'opacité est alors le suivant : existe-t-il une observation qui permette de conclure que le système est dans un état secret. Il y a plusieurs variantes : opacité par langage (le secret est défini par des ensembles d'exécutions), opacité en k-pas (le système était-il dans un état secret dans l'une des k dernières observations), infinie (le système est-il passé par un état secret), initiale (le système est-il parti d'un état secret) [131, 130, 34, 149].

Le diagnostic pour sa part porte sur les systèmes qui peuvent avoir un comportement fautif (incorrect). Le diagnostiqueur est un superviseur qui révèle la survenue d'une faute à l'exécution. Idéalement, on attend de ce dispositif d'être correct (lorsqu'une faute est détectée elle s'est produite) et et complet (toute

faute sera détectée après un temps fini). Lorsqu'un s'intéresse à un système fini à événement discrets la correction est une conséquence de techniques classiques d'estimation de l'état courant. En revanche, la complétude revient à la propriété de *diagnosticabilité* [134, 150], elle n'est pas toujours garantie et doit donc être démontrée.

Pour ce qui est des complexités de ces problèmes, Cassez *et al.* ont établi que l'opacité par état est PSPACE-complète [34]. Le calcul, hors-ligne d'un superviseur pour détecter la survenue d'une fuite d'information *à l'exécution* a la même complexité. Pour l'opacité par langage et l'opacité initiale il existe une réduction polynomiale (et réciproque) ce qui implique donc la même complexité. Ici, nous nous intéresserons donc à l'opacité par état. Concernant les systèmes infinis, l'opacité est indécidable pour les automates temporisés et les réseaux de Petri en général [33, 26].

Pour le diagnostic, la diagnosticabilité est à présent une propriété bien comprise [134, 32, 89, 90], et les algorithmes les plus efficaces repose sur la recherche de circuits dans un objet de taille quadratique, et sont donc eux-même polynomiaux. Lorsqu'on considère des systèmes infinis, cette question devient plus complexe. Ainsi, Tripakis [145] a montré que la diagnosticabilité pour les automates temporisés était équivalente à la propriété de ne pas être Zénon. D'autres travaux, [20], sont allés plus loin en donnant des bornes précise : 2EXPTIME complet pour les automates temporisés déterministes, et PSPACE complet pour les automates temporisés à enregistrement d'événements (*event-recording*). Les techniques utilisées ne reposent pas sur un auto-produit, mais sur des techniques de jeux. Ushio *et al.* [146] ont considéré les réseaux de Petri et proposé une construction effective du diagnostiqueur. Ils ont également montré l'indécidabilité de la diagnosticabilité. Les travaux de Baldan *et al.* [11] proposent un environnement général qui s'appuie sur des transformations de graphes et qui capturent des systèmes mobiles avec des topologies modifiables. Leur contribution est de calculer un ensemble d'exécutions pour un système suivant une certaine observation. Avec des objectifs similaires, d'autres travaux [78, 83] considère l'observation distribuée d'un système également distribué, représenté par un HMSC (*High Level Message Sequence Chart*). Avec Pinchinat [116], nous avons étudié la diagnosticabilité pour les automates à pile (d'ordre supérieur), et montré la décidabilité pour une sous-classe des automates à pile visible.

Ces deux problèmes ont été résolu en mettant en œuvre des techniques communes : (1) calcul d'une ε-*clôture* (une projection), (2) *déterminisation* en préservant des propriétés d'états, (3) calcul d'un *auto-produit* ou sa simulation par un jeu, (4) détection de *chemins infinis*, et (5) *analyse d'accessibilité*.

Alors que ces techniques sont bien établies pour les systèmes à états finis, leur transfert à des systèmes à états infinis demande une étude approfondie. Dans ce chapitre, nous examinerons diagnosticabilité et opacité pour des systèmes modélisés par des graphes réguliers.

Exemple 9. *Pour motiver l'utilisation de modèles récursifs, nous proposons de modéliser un système de production : il gère deux sortes de ressources (*A *et* B*) avec un principe d'urgence représenté par l'automate de la figure 4.1. La ressource* B *possède une plus grande priorité que* A*.*

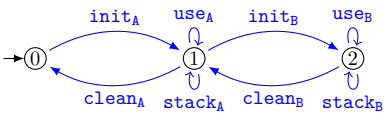


FIGURE 4.1 – Un système à deux ressources.

Dans l'état initial, 0*, le système ne contient aucune ressource et ne peut en traiter. Dans l'état* 1 (resp. *l'état* 2*), le système est configuré pour traîter la ressource* A (resp. B)*. Les transitions* $\texttt{stack}_\texttt{A}$ *et* $\texttt{stack}_\texttt{B}$ *modélisent respectivement la réception des ressources* A *et* B*. De même les transitions* $\texttt{use}_\texttt{A}$ *et* $\texttt{use}_\texttt{B}$ *modélisent la manipulation des ressources (ce qui implique l'existence de telles ressources dans le système). Ce système est très simple, cependant deux éléments ne peuvent être modélisés par un automate fini : dès que la transition* $\texttt{init}_\texttt{B}$ *est franchie, les ressources* A *déjà présentes dans le système y restent, et*

dès que la transition $\texttt{clean}_{\texttt{B}}$ *se produit, les ressources* B *doivent avoir été traitées. On peut souligner qu'un tel système peut être facilement modélisé par une machine à deux compteurs (avec tests à zéro). Cependant, ces machines ont la puissance de celles de Turing et donc la plupart des propriétés sont indécidable. En revanche il est tout a fait possible d'utiliser les graphes réguliers pour modéliser ce type de système. Ceci offre l'avantage de disposer de nombreuses propriétés décidables, en particulier vis-à-vis de l'accessibilité.*

Dans la suite nous allons démontrer l'indécidabilité des problèmes de diagnosticabilité et d'opacité pour les graphes réguliers en général. En revanche nous présenterons une sous-classe qui repose sur les HR-grammaires *pondérées* de Caucal [39] et qui généralise les automates à pile visibles, et assure l'effectivité des constructions nécessaires. Ainsi, nous donnons des caractéristiques décidables pour les HR-grammaires qui garantissent la décidabilité des problèmes de diagnosticabilité et d'opacité. Ces résultats forment une extension d'un travail antérieur avec Pinchinat [116]. Dans ce travail, que ce soit pour la détection de fuite d'information ou de la survenue d'une défaillance, le superviseur est représenté par une *machine à typer*. Il s'agit d'un automate qui prend en entrée les observations de l'exécution du système, et dont l'état courant reflète respectivement l'état du secret ou de la faute. Cet objet est obtenu par une ε-clôture suivie d'une déterminisation. L'application grossière de ces techniques sur des graphes réguliers peut produire des objets qui ne sont pas des graphes réguliers. Ici, on conçoit donc une technique adaptée : d'abord pour le calcul de la clôture, ensuite, on teste si le résultat possède la propriété d'être pondéré. Lorsque c'est le cas, on produit une HR-grammaire qui définit la machine à typer. On a donc une classe : les HR-grammaires à clôture pondérée ($\mathcal{C}$wHRG(Σ_o)) pour lesquelles diagnosticabilité et opacité sont décidable, et pour lesquelles on peut construire de façon effective un superviseur qui est modélisé par une HR-grammaire. En dernier lieu, on examine les complexités de ces problèmes, qui sont résumés dans la table 4.1.

Problème	États finis	HR-grammaires	$\mathcal{C}$wHRG(Σ_o)
Opacité	PSPACE-complet [34]	Indécidable	2-EXPTIME
Diagnosticabilité	PTIME [135]	Indécidable [116]	EXPTIME

TABLE 4.1 – Complexités

Systèmes à événement discrets

Le modèle des *système à événement discrets* (DES) est utilisé de façon usuelle pour décrire le comportement de systèmes à un niveau élevé d'abstraction. Nous les définissons ici comme des automates (avec potentiellement un nombre infini d'états) munis d'un ensemble de couleurs et d'une fonction de typage.

Définition 8. Un *système à événement discrets* (DES) est un sextuplet $\mathcal{A} = (\Sigma, Q, \Lambda, q_0, \Delta, Ty)$ avec Σ l'alphabet des événements, Q l'ensemble des états, Λ l'ensemble des *propositions*, $q_0 \in Q$ l'*état initial*, $\Delta \subseteq Q \times \Sigma \times Q$ l'ensemble des transitions et $Ty : Q \to 2^{\Lambda}$ la *fonction de typage* qui associe à chaque état $q \in Q$ un ensemble de propositions $Ty(q)$ vérifiées en q. On appelle *type* de q la valeur $Ty(q)$.

Pour la terminologie relative aux états et transitions, nous utiliserons celle des automates (vue dans le chapitre 1). Une *exécution* est un chemin issu de l'état initial d'un DES. Le mot étiquetant une exécution est appelé la *trace* de l'exécution. On étend la notion de type pour les exécutions et les traces, dans le sens attendu : le type d'une exécution est le type du sommet atteint par le chemin, et le type d'une trace w est l'union des types de chacune des exécutions ayant pour étiquette w. Pour un DES $\mathcal{A} = (\Sigma, Q, \Lambda, q_0, \Delta, Ty)$, on note $\mathcal{L}(\mathcal{A}) = \{t \in \Sigma^* \mid \exists q(q_o \xrightarrow{t} q \wedge q \in Q)\}$. On notera aussi $\mathcal{A}(w) = \{q \in Q | q_0 \xrightarrow{t} q\}$ l'ensemble des états atteints par une trace w. Ensuite, pour tout sous-ensemble

d'états $P \subseteq Q$, on note $\mathcal{L}_P(\mathcal{A}) = \{t \in \mathcal{L}(\mathcal{A}) \mid \mathcal{A}(t) \subseteq P\}$ l'ensemble des traces qui aboutissent à un des états de P. Pour une trace $w \in \mathcal{L}(\mathcal{A})$ donnée, on note $\mathcal{L}(\mathcal{A})/w = \{u \in \Sigma^* \mid wu \in \mathcal{L}(\mathcal{A})\}$ l'ensemble des mots qui peuvent former un prolongement de la trace w dans le DES $\mathcal{A}$.

On rappelle ici la définition classique du produit synchrone pour les DES.

Définition 9. Le *produit synchrone* pour deux DES $\mathcal{A}_i = (\Sigma, Q_i, \Lambda_i, q_0^i, \Delta_i, Ty_i)$ $(i = 1, 2)$ est le DES $\mathcal{A}_1 \times \mathcal{A}_2 = (\Sigma, Q_1 \times Q_2, \Lambda, (q_0^1, q_0^2), \Delta, Ty)$ avec $((p_1, p_2), a, (q_1, q_2)) \in \Delta$ lorsque $(p_1, a, q_1) \in \Delta_1$ et $(p_2, a, q_2) \in \Delta_2$ et $Ty(p_1, p_2) = Ty_1(p_1) \cup Ty_2(p_2)$.

Étant donnés $F_1 \in Q_1$ et $F_2 \in Q_2$, on a $\mathcal{L}_{F_1 \times F_2}(\mathcal{A}_1 \times \mathcal{A}_2) = \mathcal{L}_{F_1}(\mathcal{A}_1) \cap \mathcal{L}_{F_2}(\mathcal{A}_2)$. On appelle *auto-produit* le produit d'un DES $\mathcal{A}$ avec lui-même, le DES obtenu est noté $\mathcal{A}^2$.

Observation partielle. Un des points clé parmi les questions auxquelles on s'intéresse ici est la capacité, pour un dispositif externe, de déduire l'état interne d'un système en observant un sous-ensemble de ses événements. Ainsi, on fera la supposition que l'alphabet Σ des événements d'un DES $\mathcal{A}$ est partagé en deux sous-ensembles Σ_o et Σ_{no} qui contiennent respectivement les événements *observables* et *inobservables*. La projection canonique π de Σ sur Σ_o est classique : $\pi(\varepsilon) = \varepsilon$, et $\forall w \in \Sigma^*$, $a \in \Sigma$, $\pi(wa) = \pi(w)a$ lorsque $a \in \Sigma_o$, et $\pi(wa) = \pi(w)$ sinon. Cette projection s'étend naturellement aux langages. La projection inverse d'un langage $L \subseteq \Sigma_o^*$ est définie par $\pi^{-1}(L) = \{t \in \Sigma^* \mid \pi(t) \in L\}$. Pour tout mot $\mu \in \Sigma_o^*$, on note $p \overset{\mu}{\rightsquigarrow} q$ si il existe $w \in \Sigma^*$ et $a \in \Sigma_o$ tel que $\pi(w)a$ et $p \overset{wa}{\rightarrow} q$. Une *observation* de $\mathcal{A}$ est un mot $\mu \in \Sigma_o^*$ tel que $\mu = \pi(w)$ pour un $w \in \mathcal{L}(\mathcal{A}) \cap \Sigma^*\Sigma_o$; dans ce cas, w est *associé* à l'observation μ. On note $obs(\mathcal{A})$ l'ensemble des observations de $\mathcal{A}$ et $obs_P(\mathcal{A})$ l'ensemble des observations qui peuvent aboutir à un des états de l'ensemble P. En adoptant à nouveau les notations vues précédemment, on définit $\mathcal{A}(\mu) = \{q \in Q \mid q_0 \overset{\mu}{\rightsquigarrow} q\}$ l'ensemble des états atteint après une exécution d'observation μ. Deux exécutions $\wp$ et $\wp'$ sont *équivalentes* si leurs traces ont la même observation comme projection.

Dans cet environnement, l'information est codée par l'ensemble des propositions qui sont vérifiées dans les états du système. Il est donc nécessaire de définir le type d'une observation.

Définition 10. Étant donnée une observation $\mu \in obs(\mathcal{A})$, son *type* $Ty(\mu)$ est $\bigcup_{q \in \mathcal{A}(\mu)} Ty(q)$.

On peut relever que le type des traces et celui des observations sont liés dans la mesure où on peut définir $Ty(\mu)$ de manière alternative, comme ceci $\bigcup_{w \in \pi^{-1}(\mu) \cap \Sigma^*\Sigma_o} Ty(w)$. En particulier, $Ty(u) \subseteq Ty(\pi(u))$. Intuitivement, le type d'une observation μ correspond à l'ensemble des propositions qui sont vérifiées dans au moins *un* des états qui peut être atteint dans $\mathcal{A}$ par une trace w compatible avec l'observation μ.

Remarque 6. Dans la littérature, en général, le type des états, des exécutions ou des observations d'un DES est défini en terme de sous-ensembles *explicites* de l'ensemble d'état. Notre notion de typage permettra se s'appuyer simplement sur la couleur des états pour définir leur type. Ce qui est plus simple qu'une description explicite dans le cas des systèmes modélisés par un graphe infini. (En particulier une caractérisation externe comme les HR-grammaires.)

Pour calculer le type des observations, on utilise une *machine à typer* :

Définition 11. Une *machine à typer* de $\mathcal{A}$, est une DES déterministe $\mathcal{T}_{\Sigma_o}(\mathcal{A}) = (\Sigma_o, Q', \Lambda, q_0', \Delta', Ty')$ tel que $\forall \mu \in obs(\mathcal{A}), Ty'(\mu) = Ty(\mu)$.

Cette définition d'une machine à typer n'est pas constructive, et il n'y a rien de surprenant à constater que la construction d'un tel objet ne soit pas toujours possible pour un DES arbitraire. La construction, en générale, repose fortement sur une notion de clôture, qu'on rappelle ici pour un système quelconque.

Définition 12. La Σ_{no}-*clôture* d'un DES $\mathcal{A}$ est constituée du DES $\mathcal{A}_o = (\Sigma_o, Q, \Lambda, q_0, \Delta', Ty')$ tel que $\Delta' = \{(p, a, q) \mid a \in \Sigma_o \wedge p \overset{a}{\rightsquigarrow} q\}$ et $Ty'(q) = Ty(q)$ pour tout $q \in Q$.

En d'autres termes toutes les transitions de $\mathcal{A}$ étiquetés par un événement non-observable est supprimée de la clôture, tout en préservant l'ensemble des observations ($obs(\mathcal{A}_o) = obs(\mathcal{A})$) et leurs types ($Ty'(\mu) = Ty(\mu), \forall \mu \in obs(\mathcal{A})$).

Pour conclure, on rappelle que dans le cas fini pour calculer une machine à typer on exécute les opérations suivantes : (1) calcul de la Σ_{no}-*clôture* du DES, et (2) calcul de sa *déterminisation* (voir, par exemple les travaux de Cassandras *et al.* [32]). Si on note $\mathcal{D}(\mathcal{A})$ le déterminisé d'un DES (par exemple par la construction des parties). On peut voir que, si on note $\mathcal{A}_o$ est la Σ_{uo}-clôture d'un DES $\mathcal{A}$, alors $\mathcal{D}(\mathcal{A}_o)$ est la machine à typer de $\mathcal{A}$.

Les problèmes d'opacité et de diagnosticabilité

Soit un DES $\mathcal{A} = (\Sigma, Q, \Lambda, q_0, \Delta, Ty)$ avec $\Lambda = \{\lambda, \overline{\lambda}\}$. Soit une partition de $\Sigma : \Sigma_{uo}, \Sigma_o$. On considère *positifs* (*resp. négatifs*) les états de type $\{\lambda\}$ (*resp.* $\left\{\overline{\lambda}\right\}$). On suppose que les états négatifs et positifs forment une partition de l'ensemble des états Q (i.e. $Q = Q_\lambda \cup Q_{\overline{\lambda}}$). Du fait de l'observation partielle, une observation $\mu \in \Sigma_o^*$ peut correspondre à des exécutions aboutissant à la fois à des états positifs et négatifs. Une telle observation aurait pour type $\left\{\lambda, \overline{\lambda}\right\}$. Pour avoir une terminologie uniforme, une observation de type $\{\lambda\}$ (*resp.* $\{\overline{\lambda}\}$) est dite positive (*resp.* negative). Une observation de type $\{\lambda, \overline{\lambda}\}$ est dite *équivoque*.

Remarque 7. On peut souligner que cette définition des observations positives ou négatives repose exclusivement sur les états du système considéré. Il existe traditionnellement des caractérisations reposant sur des *langages* d'observations. Les deux caractérisations sont interréductibles au prix d'une opération polynomiale dans le cas fini (en s'appuyant sur le produit synchrone). Ce qui illustre l'universalité de ce choix.

Le problème de l'opacité

La notion d'opacité a été définie dans les travaux de Bryans *et al.* [26] en tant que propriété de sécurité. Le problème consiste à déterminer si un attaquant, qui connaît le fonctionnement d'un système et qui l'observe partiellement peut découvrir qu'un comportement secret s'est produit au cours de l'exécution. Ici on fait la supposition que le système est représenté par un DES $\mathcal{A}$, $\Lambda = \{\lambda, \overline{\lambda}\}$ et que le secret est représenté par l'ensemble d'état Q_λ. Dans ce cas, il n'y a pas de fuite d'information (le système est *opaque*) si l'attaquant ne peut pas déterminer avec certitude, à partir de son observation, que le système se trouve dans un état de Q_λ.
Formellement, un ensemble d'états Q_λ est opaque pour $\mathcal{A}$ et Σ_o, si l'équation suivante est satisfaite :

$$\forall \mu \in \Sigma_o^*, Ty(\mu) \neq \{\lambda\} \tag{4.1}$$

Définition 13. Étant donnés un DES $\mathcal{A} = (\Sigma, Q, \Lambda, q_0, \Delta, Ty)$ avec $\Sigma_o \subseteq \Sigma$ et $\Lambda = \left\{\lambda, \overline{\lambda}\right\}$ le problème de l'opacité consiste vérifier si l'équation (4.1) est vérifiée pour $\mathcal{A}$.

Exemple 10. *On considère le DES $\mathcal{A}$ représenté sur la figure 10, avec $\Sigma_o = \{a, b\}$. Le secret est définit par l'ensemble d'états $Q_\lambda = \{q_2, q_5\}$.*
L'ensemble Q_λ n'est pas opaque car après avoir observé une trace dans b^*ab, l'attaquant sait que le système est dans un état secret. Il ne peut pas savoir avec certitude si c'est q_2 ou q_5 mais il sait que le système est dans un état de Q_λ.

Le problème d'universalité pour un langage peut être réduit à celui de l'opacité [34], donc dans le cas général ce problème est indécidable. En revanche, dans le cas fini, l'opacité est décidable.

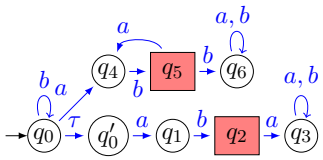


FIGURE 4.2 – Système non-opaque

Proposition 73. *([34]) Le problème de l'opacité est* PSPACE*-complet pour les DES finis non-déterministes.*

Vérifier l'opacité pour un système $\mathcal{A}$ et Σ_o repose sur le calcul de la Σ_{uo}-clôture de $\mathcal{A}$ et celui du déterminisé $\mathcal{D}(\mathcal{A}_o)$. Il suffit ensuite de vérifier s'il existe un état accessible de type $\{\lambda\}$ dans ce DES. Lorsque le système n'est pas opaque, l'étape suivante consiste à construire un moiteur pour détecter la survenue d'une fuite d'information. En considérant que ce moniteur observe les événements Σ_o du système, d'après Dubreil *et al.* [53], on utilise la machine à typer $\mathcal{D}(\mathcal{A}_o)$. On peut remarquer que détecter les fuites d'informations en utilisant un alphabet d'observation différent de celui de l'attaquant pourrait se révéler intéressant, mais plus complexe [53]. Nous y reviendrons ultérieurement.

Exemple 11. *On reprend l'exemple 10, la machine à typer est représentée sur la figure 4.3. L'état* $\{q_2, q_5\}$ *est accessible depuis l'état initial, ce qui confirme la non-opacité de ce système. Pour détecter l'opacité à l'exécution la construction de la machine à typer peut être interrompue lorsque l'état* $\{q_2, q_5\}$ *est atteint. De plus, pour superviser les fuites d'informations à l'exécution, la machine à typer peut être élaguée aux états* $\{q_0\}, \{q_1, q_4\}, \{q_2, q_5\}$ *et une alarme correspondant à l'information révélée peut être levée lorsque l'état* $\{q_2, q_5\}$ *est atteint.*

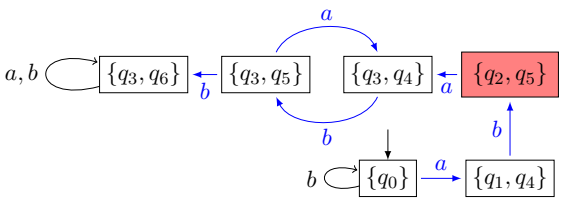


FIGURE 4.3 – La machine à typer du système de la figure 10

Le problème de la diagnosticabilité

Le problème de la diagnosticabilité a été introduit par [134], il consiste à vérifier la satisfaction d'une propriété persistante dans le contexte d'un système partiellement observé (dans l'article de Sampath *et al.* [134], la propriété code la survenue d'une faute dans le système). Comme pour l'opacité, on attribue aux états d'un DES $\mathcal{A}$, exclusivement les types $\{\lambda\}$ ou $\left\{\overline{\lambda}\right\}$ avec l'hypothèse que pour tous les états positifs (qui ont le type $\{\lambda\}$), seuls des états positifs peuvent être atteints. Ainsi un état négatif exprime que la propriété n'a pas encore été vérifiée alors qu'un état positif exprime que la propriété *a été vérifiée* un moment dans le passé ; en conséquence ça restera vérifié.

Le problème du diagnostic consiste à être capable de synthétiser un diagnostiqueur, c'est-à-dire une fonction $Diag : Obs(\mathcal{A}) \to \{yes, no, ?\}$ sur les observations qui fournit comme résultat la réponse à la question de savoir si toutes les exécutions correspondant à l'observation ont atteint Q_λ. Cette fonction

doit vérifier :

$$Diag(\mu) \quad = \begin{cases} oui & \text{lorsque } Ty(\mu) = \{\lambda\} \\ non & \text{lorsque } Ty(\mu) = \left\{\overline{\lambda}\right\} \\ ? & \text{Autrement.} \end{cases}$$

Comme pour la supervision de l'opacité, le diagnostiqueur d'un DES $\mathcal{A}$ peut être construit à partir de son machine à typer $\mathcal{D}(\mathcal{A}_o)$ avec le convention que le verdict oui est associé à un état dont le type est $\{\lambda\}$, non pour ceux de type $\left\{\overline{\lambda}\right\}$ et ? pour ceux de type $\left\{\lambda, \overline{\lambda}\right\}$. Pour que cet outil ait un intérêt pratique, il est nécessaire que lors d'une exécution de type $\{\lambda\}$ la fonction $Diag$ finisse par donner le verdict oui. Ceci est formalisé par la notion de diagnosticabilité : étant donné un DES $\mathcal{A} = (\Sigma, Q, \Lambda, q_0, \Delta, Ty)$ et $\Sigma_o \subseteq \Sigma$, une proposition $\lambda \in \Lambda$ est *diagnosticable dans $\mathcal{A}$ pour Σ_o* lorsque pour tout $u \in \mathcal{L}_{Q_\lambda}(\mathcal{A})$, il existe un entier $n \in \mathbb{N}$ tel que pour tout $v \in \mathcal{L}(\mathcal{A})/u$ avec $|\pi(v)| \geq n$,

$$\pi^{-1}(u.v) \cap \mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}_{Q_\lambda}(\mathcal{A})$$

En d'autres termes, λ est diagnosticable dans $\mathcal{A}$ pour Σ_o lorsqu'il est toujours possible de détecter dans un délais fini (en nombre d'observations) que le système a atteint un état positif.

On peut reformuler cette question de la façon suivante.

Lemme 74. *([116]) Une proposition λ n'est pas diagnosticable dans $\mathcal{A}$ pour Σ_o si et seulement si il existe deux chemins infinis possédant la même observation, l'un atteignant Q_λ et l'autre pas.*

Le *problème de la diagnosticabilité* consiste, étant donné un DES $\mathcal{A}$ avec $\Sigma_o \subseteq \Sigma$ et $\Lambda = \left\{\lambda, \overline{\lambda}\right\}$, la propriété λ est-elle diagnosticable dans $\mathcal{A}$ pour Σ_o ?

Dans le cas général ce problème est indécidable [116] : une simple réduction du problème du vide de l'intersection de deux langages algébriques. En revanche ce problème est décidable pour les DES à états finis.

Proposition 75. *([135]) Le problème de la diagnosticabilité des DES finis est dans* PTIME.

La technique standard consiste à construire un auto-produit de la Σ_o-clôture du système et d'y rechercher un circuit équivoque [89].

I.2 Test de conformité

Dans ce paragraphe, nous décrivons un arrière plan pour les travaux sur le test de conformité considérés en § II (dans une seconde partie). Le test de conformité consiste à vérifier, à partir de série d'expérimentations qu'une implémentation *boîte-noire* se comporte correctement vis-à-vis d'une spécification. Le test est l'une des techniques les plus populaires pour évaluer la qualité d'un logiciel, il représente une part importante du coût du développement logiciel. L'automatisation de la production de jeux de tests est une nécessité pour améliorer à la fois la qualité du processus de test et en diminuer les coûts. Il est, en particulier, intéressant de d'automatiser la production de tests lorsqu'on dispose de spécifications pour le système. Le *test formel fondé sur les modèles* a pour but d'apporter une réponse à ce problème, en fournissant une description formalisée du cadre de test : la spécification, l'implémentation et les cas de test. Ces descriptions mathématiques sont complétées par une définition formelle de la relation de conformité, une modélisation de l'exécution des tests, et les démonstration des propriétés garanties par les résultats des test. La théorie de la conformité *ioco* a été introduite en 1996 par Tretmans [143] est à présent une définition très largement adoptée pour la modélisation formelle du test formel de conformité pour les systèmes de transitions étiquetés à entrées et sorties (IOLTS). Des algorithmes de génération de test ainsi que des outils ont été construit en s'appuyant sur ce modèle [87, 144] et pour les modèles plus généraux dont la sémantique peut s'exprimer par un IOLTS ayant un nombe infini d'états [88, 63]. Des techniques de test ont également été développées pour le modèle des automates temporisés [101, 96, 14].

Globalement, on peut distinguer deux approches pour la génération de tests : *hors-ligne* qui produit des tests les stocks, puis les exécutent parallèlement à l'implémentation. L'approche *en-ligne* vise en général à produire les tests en parallèle à l'exécution de l'implémentation. Cette dernière prend en compte les retours de l'implémentation pour ne calculer que la partie *utile* du cas de test. Dans les deux cas il est possible d'assurer certaines propriétés pour les tests engendrés. Ainsi, la correction exprime qu'aucune implémentation conforme ne peut être rejetée, alors que l' exhaustivité reflète que tout les implémentations non-conformes seront rejetés par l'ensemble des tests.

Dans le cas des systèmes infinis, l'indécidabilité est souvent un problème. Il est aussi possible que la clôture observable d'un système donné ne puisse pas être exprimable dans le même modèle. Pour établir la correction ou l'exhaustivité pour une suite de tests engendrée, il est pratique d'avoir une description formelle pour le système et de pouvoir prouver certaines propriétés des tests engendrés. Entre les modèles à ensemble fini d'états et ceux qui ont la puissance des machines de Turing les systèmes algébriques constituent une fois de plus un moyen terme intéressant entre expressivité et décidabilité. Dans le cas du test, ces systèmes peuvent modéliser des programmes réactifs récursifs tels que celui présenté sur la figure 4.4. Cet exemple utilise une syntaxe similaire à celle de Java, et comporte des exceptions (une forme de raccourci qui est utilisé avec le mot-clé `throw`). Plus précisément, le programme demande de choisir un entier, ensuite appelle la fonction récursive `comp`, qui à son tour demande un booléen, puis, selon sa valeur, poursuit en faisant un appel récursif ou bien s'arrête. Si jamais une exception est levée le programme fait un saut immédiat au niveau du bloc `catch` dans la fonction `main`. Dans la suite de notre propos sur le test, nous nous appuierons sur cet exemple, en faisant abstraction des valeurs entières. Nous utiliserons à nouveau les HR-grammaires pour décrire les spécifications dans le cas du test. Pour exprimer la réactivité du système, l'alphabet est partagé entre les entrées (que le système reçoit), les sorties (qui sont émises par le système) et les actions internes (que le système utilise sans les diffuser). Ainsi une HR-grammaire définie un graphe qui est un IOLTS. L'accessibilité et la co-accessibilité sont des propriétés fondamentales pour la production de suite de test, elles sont à la fois décidables et effectives pour les HR-grammaires. La déterminisation en revanche nécessitera de se restreindre à la classe des HR-grammaires pondérées. On verra qu'en l'absence de possibilité de déterminisation, il nous sera possible de développer des techniques de test *en-ligne*. Dans les travaux de Constant *et al.* [48], les automates à pile déterministes avaient été considéré, et des tests avaient été produits dans des cas très restrictifs. Il s'agit à notre connaissance des seuls travaux pour engendrer des tests pour les systèmes algébriques.

Nous allons, à présent, présenter les bases formelles pour la théorie du test *ioco* telle qu'elle est définie par Tretmans [143] pour les IOLTS. Ceci posera les fondement de la génération de test pour les HR-grammaires. À nouveau nous donnerons une définition pas tout à fait canonique des IOLTS en nous utilisant les couleurs. Nous rappellerons également un ensemble de notions fondamentales : la relation de conformité *ioco*, et une définition formelle des propriétés attendus des suites de test.

Définition et opération pour les IOLTS. La littérature portant sur les IOLTS est abondante. Cette notion se réduit à une structure de Kripke dont l'alphabet est partagé entre les entrées, les sorties et les actions internes. La définition qui suit n'est pas conventionnelle, dans le sens où elle utilise les couleurs pour distinguer des ensembles d'états. Elle est également utilisée pour identifier l'état initial. Il va de soit que cette modification n'est que syntaxique et que cela n'affecte par les propriétés des IOLTS. Ce sera très utile quand nous considérerons les systèmes modélisés par des HR-grammaires.

Définition 14 (IOLTS)**.** Un IOLTS (système de transition étiqueté avec entrées et sorties) est un sextuplet $\mathcal{M} = (Q, \Sigma, \Lambda, \rightarrow_{\mathcal{M}}, \mathcal{C}, \texttt{init})$ avec Q un ensemble d'états ; Σ un alphabet *d'actions* qui formé d'une partition entre l'ensemble des entrées $\Sigma^?$, celui des sorties $\Sigma^!$ et celui des actions internes Σ^τ ; les actions *visibles* sont notées $\Sigma_o \stackrel{\text{def}}{=} \Sigma^? \cup \Sigma^!$ [1] ; Λ un ensemble de couleurs avec $\texttt{init} \in \Lambda$ la couleur des états

1. Dans les exemple pour améliorer la lisibilité, nous noterons l'entrée $a \in \Sigma^?$ ainsi $?a$, et la sortie $x \in \Sigma^!$ sera notée $!x$.

```
static void main(String [] args){
   try{
      // Bloc 1 (entrée)
      int k =in.readInt() ;
      comp(k) ;
      // Bloc 2 (sortie)
      System.out.println("Fin") ;
   }
   catch (Exception e){
      // Bloc 3 (sortie)
      System.out.println(e.getMessage()) ;
   }
}
int comp (int x){
   // Bloc 4 (entrée)
   int res =1 ;
   boolean cont=in.readBoolean() ;
   if (cont){
      if (x==0)throw new Exception("Erreur !") ;
      // Bloc 5 (interne)
      res=x*comp(x-1) ;
      // Bloc 6 (sortie)
      System.out.println("Un texte") ;
      return res ;
   }
   else {
      // Bloc 7 (sortie)
      System.out.println("Vous avez arrêté") ;
      return res ;
   }
}
```

FIGURE 4.4 – Un programme récursif

initiaux ; $\rightarrow_{\mathcal{M}} \subseteq Q \times \Sigma \times Q$ la *relation de transitions* ; $\mathcal{C} \subseteq Q \times \Lambda$ est la relation associant les états aux couleurs.

Dans cette définition inhabituelle des IOLTSs, les couleurs sont utilisées pour marquer les états par la relation $\mathcal{C}$. Pour une couleur $\lambda \in \Lambda$, on note $\mathcal{C}(\lambda) \stackrel{\text{def}}{=} \{q \in Q \mid (q, \lambda) \in \mathcal{C}\}$ et $\overline{\mathcal{C}}(\lambda) \stackrel{\text{def}}{=} Q \setminus \mathcal{C}(\lambda)$ pour les ensembles d'états respectivement colorés et non-colorées par λ. En particulier, $\mathcal{C}(\texttt{init})$ définit les états initiaux.

Les notations pour les transitions et les chemins définies pour les automates et les DES restent valable pour les IOLTS.

On dira, de plus, qu'un chemin est *accepté* par une couleur λ s'il est accepté par un état de $\mathcal{C}(\lambda)$ et on posera $L_\lambda(\mathcal{M})$ pour le langage $L_{\mathcal{C}(\lambda)}(\mathcal{M})$.

Soient $X \subseteq Q$ un sous-ensemble d'états et $\Sigma' \subseteq \Sigma$ un sous-alphabet, on note $post_{\mathcal{M}}(\Sigma', X) = \{q' \in Q \mid \exists q \in X, \exists \mu \in \Sigma' : q \xrightarrow[\mathcal{M}]{\mu} q'\}$ l'ensemble des successeurs d'un état de X par une action de Σ', réciproquement, on note $pre_{\mathcal{M}}(\Sigma', X) = \{q \in Q \mid \exists q' \in X, \exists \mu \in \Sigma' : q \xrightarrow[\mathcal{M}]{\mu} q'\}$ l'ensemble des prédécesseurs de X par une seul action de Σ'. L'ensemble des états *accessibles* par un ensemble d'états

Les actions internes ne porterons pas de symbole particulier.

$P \subseteq Q$ par des actions de Σ' est $\mathrm{Reach}_{\mathcal{M}}(\Sigma', P) \stackrel{\text{def}}{=} \mathrm{lfp}(\lambda X.P \cup post_{\mathcal{M}}(\Sigma', X)$ où lfp désigne l'opérateur de plus petit point fixe. De façon similaire, l'ensemble des états *co-accessible* depuis un ensemble $P \subseteq Q$ (c-à-d l'ensemble des états depuis lesquels P est accessible) est noté $\mathrm{Coreach}_{\mathcal{M}}(\Sigma', P) \stackrel{\text{def}}{=} \mathrm{lfp}(\lambda X.P \cup pre_{\mathcal{M}}(\Sigma', X))$. On étend cette notation aux couleurs, pour $\lambda \in \Lambda$, $\mathrm{Reach}_{\mathcal{M}}(\Sigma', \lambda)$ désigne $\mathrm{Reach}_{\mathcal{M}}(\Sigma', \mathcal{C}(\lambda))$ et $\mathrm{Coreach}_{\mathcal{M}}(\Sigma', \lambda)$ représente $\mathrm{Coreach}_{\mathcal{M}}(\Sigma', \mathcal{C}(\lambda))$.

Pour un état q, on note $\Gamma_{\mathcal{M}}(q) \stackrel{\text{def}}{=} \{\mu \in \Sigma \mid q \xrightarrow[\mathcal{M}]{\mu}\}$ le sous-ensemble d'actions qui sont autorisées en q et respectivement, $Out_{\mathcal{M}}(q) \stackrel{\text{def}}{=} \Gamma_{\mathcal{M}}(q) \cap \Sigma^!$ et $In_{\mathcal{M}}(q) \stackrel{\text{def}}{=} \Gamma_{\mathcal{M}}(q) \cap \Sigma^?$ représente l'ensemble des sorties (*resp.* entrées) autorisées en q. La notation est étendue aux ensembles d'états : pour $P \subseteq Q$, $Out_{\mathcal{M}}(P) \stackrel{\text{def}}{=} \bigcup_{q \in P} Out_{\mathcal{M}}(q)$ et $In_{\mathcal{M}}(P) \stackrel{\text{def}}{=} \bigcup_{q \in P} In_{\mathcal{M}}(q)$.

On note le comportement observable de l'IOLTS $\mathcal{M}$, de façon classique par la relation $\underset{\mathcal{M}}{\Rightarrow} \in Q \times (\{\epsilon\} \cup \Sigma_o) \times Q$ ainsi définie : $q \underset{\mathcal{M}}{\overset{\varepsilon}{\Rightarrow}} q' \stackrel{\text{def}}{=} q = q'$ ou $q \xrightarrow[\mathcal{M}]{\tau_1.\tau_2 \cdots \tau_n}{}^{*} q'$, pour tous les $\tau_i \in \Sigma^\tau$ et pour $a \in \Sigma_o$, $q \underset{\mathcal{M}}{\overset{a}{\Rightarrow}} q' \stackrel{\text{def}}{=} \exists q_1, q_2,\ q \underset{\mathcal{M}}{\overset{\varepsilon}{\Rightarrow}} q_1 \xrightarrow[\mathcal{M}]{a} q_2 \underset{\mathcal{M}}{\overset{\varepsilon}{\Rightarrow}} q'$. La notation est étendue aux mots : pour $\sigma = a_1 \cdots a_n \in (\Sigma_o)^*$ un mots formé d'actions observables, $q \underset{\mathcal{M}}{\overset{\sigma}{\Rightarrow}} q'$ représente $\exists q_0, \ldots, q_n,\ q = q_0 \underset{\mathcal{M}}{\overset{a_1}{\Rightarrow}} q_1 \cdots \underset{\mathcal{M}}{\overset{a_n}{\Rightarrow}} q_n = q'$ et $q \underset{\mathcal{M}}{\overset{\sigma}{\Rightarrow}}$ pour $\exists\, q', q \underset{\mathcal{M}}{\overset{\sigma}{\Rightarrow}} q'$.

La formule q `after` $\sigma \stackrel{\text{def}}{=} \{q' \in Q \mid q \underset{\mathcal{M}}{\overset{\sigma}{\Rightarrow}} q'\}$ représente l'ensemble des états dans lesquels le système $\mathcal{M}$ peut se trouver *après* l'observation du mot σ en partant de l'état q. Cette notation peut s'étendre aux ensembles d'états : pour $P \subseteq Q$, P `after` $\sigma \stackrel{\text{def}}{=} \bigcup_{q \in P} q$ `after` σ.

Pour un état q, la notation $\mathrm{Traces}(q) \stackrel{\text{def}}{=} \{\sigma \in (\Sigma_o)^* \mid q \underset{\mathcal{M}}{\overset{\sigma}{\Rightarrow}}\}$ désigne l'ensemble des traces (comme pour les DES, mots d'actions observables) qui peuvent être observées à partir de q et $\mathrm{Traces}(\mathcal{M}) \stackrel{\text{def}}{=} \bigcup_{q_0 \in \mathcal{C}(\texttt{init})} \mathrm{Traces}(q_0)$ forme l'ensemble des traces qui peuvent être observées à partir de l'état initial. Pour l'ensemble d'états P, on note $\mathrm{Traces}_P(\mathcal{M}) = \{\sigma \in (\Sigma_o)^* \mid (\mathcal{C}(\texttt{init})\ \texttt{after}\, \sigma) \cap P \neq \emptyset\}$ l'ensemble des traces qui aboutissent à P.

Un IOLTS $\mathcal{M}$ est dit *complet en entrée* si dans chaque état toutes les entrées peuvent être acceptées, éventuellement après quelques actions internes, *i.e.* $\forall q \in Q, \forall \mu \in \Sigma^?, \exists q' \in Q,\ q \underset{\mathcal{M}}{\overset{\mu}{\Rightarrow}} q'$.

$\mathcal{M}$ est *complet* en q si toutes les actions sont possibles en q : $\forall q \in Q, \Gamma(q) = \Sigma$. $\mathcal{M}$ est *complet* s'il est complet en tout état de Q

Un IOLTS est *déterministe* si, lorsqu'on le voit comme un automate il est déterministe (En particulier, il possède un unique état initial).

Le produit synchrone des IOLTS est défini de façon identique à celui des DES. Une autre notion de produit s'appuie sur la distinction entre les entrées et les sorties pour synchroniser les sorties d'un premier système avec les entrées d'un second.

Définition 15 (Composition parallèle des IOLTS)**.** Soit $\mathcal{M}_i = (Q_i, \Sigma_i, \Lambda_i, \rightarrow_i, \mathcal{C}_i, \texttt{init}_i)$, $i = 1, 2$ deux IOLTSs avec des alphabets d'entrées et sorties compatibles (*i.e.* $\Sigma_1^! = \Sigma_2^?$ et $\Sigma_1^? = \Sigma_2^!$). Leur produit parallèle est l'IOLTS qui suit : $\mathcal{M}_1 \| \mathcal{M}_2 = (Q_{\mathcal{M}}, \Sigma_{\mathcal{M}}, \Lambda_{\mathcal{M}}, \rightarrow_{\mathcal{M}}, \mathcal{C}_{\mathcal{M}}, \texttt{init}_{\mathcal{M}})$ où $Q_{\mathcal{M}} = Q_1 \times Q_2$, $\Sigma_{\mathcal{M}} = \Sigma_1$, $\Lambda \stackrel{\text{def}}{=} \Lambda_1 \times \Lambda_2$, en particulier $\texttt{init} \stackrel{\text{def}}{=} (\texttt{init}_1, \texttt{init}_2)$, pour tous les $(\lambda_1, \lambda_2) \in \Lambda$ la relation de coloriage est ainsi définie : $\mathcal{C}((\lambda_1, \lambda_2)) \stackrel{\text{def}}{=} \mathcal{C}_1(\lambda_1) \times \mathcal{C}_2(\lambda_2)$, on définit la relation de transition ainsi :

- Si $a \in \Sigma_o$, alors, pour $q_1, q_1' \in Q_1$ et $q_2, q_2' \in Q_2$ tels que $q_1 \xrightarrow[\mathcal{M}_1]{a} q_1'$ et $q_2 \xrightarrow[\mathcal{M}_2]{a} q_2'$, on a la transition $(q_1, q_2) \xrightarrow[\mathcal{M}]{a} (q_1', q_2')$;
- Si $\tau \in \Sigma_{no}$, alors, pour $q_1, q_1' \in Q_1$ et $q_2 \in Q_2$ tels que $q_1 \xrightarrow[\mathcal{M}_1]{a} q_1'$, on a la transition $(q_1, q_2) \xrightarrow[\mathcal{M}]{\tau} (q_1', q_2)$;
- Si $\tau \in \Sigma_{no}$, alors, pour $q_1 \in Q_1$ et $q_2, q_2' \in Q_2$ tels que $q_2 \xrightarrow[\mathcal{M}_2]{a} q_2'$, on a la transition $(q_1, q_2) \xrightarrow[\mathcal{M}]{\tau} (q_1, q_2')$.

Comme la synchronisation porte sur les actions visibles, on a également, pour deux ensembles $P_i \subseteq Q_i, i = 1, 2$, $\text{Traces}_{P_1 \times P_2}(\mathcal{M}) = \text{Traces}_{P_1}(\mathcal{M}_1) \cap \text{Traces}_{P_2}(\mathcal{M}_2)$, et en particulier $\text{Traces}(\mathcal{M}) = \text{Traces}_{P_1}(\mathcal{M}_2) \cap \text{Traces}_{P_2}(\mathcal{M}_2)$. Cette définition n'est pas symétrique pour les entrées et sorties : la nature des actions est donnée par la première opérande.

I.3 La théorie du test *ioco*

Spécification et implémentation. Dans le cadre du test *ioco*, on fait la supposition que le comportement de la spécification est modélisé par un IOLTS $\mathcal{M} = (Q, \Sigma, \Lambda, \rightarrow_{\mathcal{M}}, \mathcal{C}, \texttt{init})$. L'implémentation qui est testée est un système boite-noire avec la même interface observable que la spécification. Pour pouvoir modéliser la conformité, il est souvent supposé que le comportement de l'implémentation est celui d'un IOLTS inconnu (complet), $\mathcal{I} = (Q_{\mathcal{I}}, \Sigma_{\mathcal{I}}, \Lambda_{\mathcal{I}}, \rightarrow_{\mathcal{I}}, \texttt{init}_{\mathcal{I}})$ avec $\Sigma_{\mathcal{I}} = \Sigma^?_{\mathcal{I}} \cup \Sigma^!_{\mathcal{I}} \cup \Sigma^\tau_{\mathcal{I}}$ et $\Sigma^?_{\mathcal{I}} = \Sigma^?$ et $\Sigma^!_{\mathcal{I}} = \Sigma^!$. La supposition de complétude en entrée de l'implémentation reflète l'idée qu'elle est toujours prête à recevoir des données de l'environnement, par exemple du cas de test. Dans la suite, l'ensemble des implémentations ayant un alphabet compatible avec $\mathcal{M}$, est noté $\mathcal{IMP}(\mathcal{M})$.

Quiescence. Il est fondamental pour les tests d'observer les sorties de l'implémentation, mais aussi l'absence de réaction (appelée *quiescence*) en utilisant des temporisations. Les cas de test doivent pouvoir faire la distinction entre une quiescence permise par la spécification, et une qui ne le serait pas. Dans un IOLTS plusieurs sortes de quiescences peuvent survenir : un état q est *quiescent en sortie* si il attend simplement une entrée de l'environnement, *i.e.* $\Gamma(q) \subseteq \Sigma^?$, (un *blocage i.e.* $\Gamma(q) = \emptyset$ est un cas particulier de quiescence en sortie), et un *blocage vif* si une suite infinie peut se produire, *i.e.* $\forall n \in \mathbb{N}, \exists \sigma \in (\Sigma^\tau)^n, \exists q' \in Q,\ q \xrightarrow[\mathcal{M}]{\sigma} q'$ [2]. Lorsque l'état q est quiescent en sortie, ou est un blocage vif on le note $quiescent(q)$. À partir d'un IOLTS $\mathcal{M}$ on peut définir $\Delta(\mathcal{M})$, un nouvel IOLTS où la quiescence est rendue explicite par une nouvelle sortie δ :

Définition 16 (Suspension)**.** Soit $\mathcal{M} = (Q, \Sigma, \Lambda, \rightarrow_{\mathcal{M}}, \mathcal{C}, \texttt{init})$ un IOLTS, la *suspension* de $\mathcal{M}$ est l'IOLTS $\Delta(\mathcal{M}) = (Q, \Sigma_{\Delta(\mathcal{M})}, \Lambda, \rightarrow_{\Delta(\mathcal{M})}, \mathcal{C}, \texttt{init})$ où $\Sigma_{\Delta(\mathcal{M})} = \Sigma \cup \{\delta\}$ avec $\delta \in \Sigma^!_{\Delta(\mathcal{M})}$ (δ est considéré comme une sortie observable), la relation de transition $\rightarrow_{\Delta(\mathcal{M})} \stackrel{\text{def}}{=} \rightarrow_{\mathcal{M}} \cup \{(q, \delta, q) \mid q \in quiescent(M)\}$ est obtenue de $\rightarrow_M$ en ajoutant des boucles δ à chacun des états q quiescents.

On observe que le système $\Delta(\mathcal{M})$ peut ne pas être calculable lorsque l'IOLTSs $\mathcal{M}$ est infini. Dans la suite nous noterons $\Sigma^! \cup \{\delta\}$ et $\Sigma_o \cup \{\delta\}$ par $\Sigma^{!\delta}$ et Σ^δ_o. Les traces de $\Delta(\mathcal{M})$ sont notées $\mathsf{STraces}(\mathcal{M})$ et appelées *traces suspendues* de $\mathcal{M}$. Elle représentent le comportement observable de $\mathcal{M}$, y compris la quiescence, et sont l'élément clé de la relation de conformité *ioco*.

Relation de conformité. Dans la théorie de la conformité *ioco* [143], à partir d'une spécification sous forme d'un IOLTS $\mathcal{M}$, on dit qu'un implémentation $\mathcal{I} \in \mathcal{IMP}(\mathcal{M})$ est conforme à $\mathcal{M}$ si, après toute trace suspendue σ de $\mathcal{M}$, l'implémentation $\mathcal{I}$ ne produit que des sorties ou des quiescences qui sont spécifiées dans $\mathcal{M}$. Plus précisément, on a la définition suivante.

Définition 17 (Relation de conformité)**.** Soit $\mathcal{M}$ un IOLTS et $\mathcal{I} \in \mathcal{IMP}(\mathcal{M})$ un IOLTS complet en entrée possédant le même alphabet observable (*i.e.* $\Sigma^? = \Sigma^?_{\mathcal{I}}$ et $\Sigma^! = \Sigma^!_{\mathcal{I}}$),

$$\mathcal{I}\ ioco\ \mathcal{M} \stackrel{\text{def}}{=} \forall \sigma \in \mathsf{STraces}(\mathcal{M}), Out(\Delta(\mathcal{I})\ \texttt{after}\ \sigma) \subseteq Out(\Delta(\mathcal{M})\ \texttt{after}\ \sigma).$$

2. Un chemin infini, traversant un ensemble infini de sommets distincts et étiqueté exclusivement par des action internes est appelé *divergeant*. La théorie *ioco* originale, se limité aux IOLTS ne possédant pas de chemin divergeant. Dans ce travail, nous autorisons de tels chemins.

On peut démontrer [88] que $\mathcal{I}$ *ioco* $\mathcal{M}$ si et seulement si $\mathsf{STraces}(\mathcal{I}) \cap \mathsf{MinFTraces}(\mathcal{M}) = \emptyset$, avec $\mathsf{MinFTraces}(\mathcal{M}) \stackrel{\text{def}}{=} \mathsf{STraces}(\mathcal{M}).\Sigma^! \setminus \mathsf{STraces}(\mathcal{M})$ un ensemble minimal (pour l'ordre préfixe) de trace suspendues non-conformes. L'ensemble des traces non-conformes est alors $\mathsf{MinFTraces}(\mathcal{M}).\Sigma^*$. Cette caractérisation alternative d'*ioco* sera utilisée dans la suite pour formaliser certaines propriétés des suites de tests.

Cas de test, suite de tests et propriétés. Dans la pratique, un cas de test décrit les interactions à réaliser avec l'implémentation et les verdicts associés selon les sorties observées. Dans notre cas, un cas de test est représenté par un IOLTS dont les couleurs représentent les verdicts.

Définition 18 (Cas de test et suite de tests)**.** Un *cas de test* pour $\mathcal{M}$ est un IOLTS déterministe complet en entrée $\mathcal{TC} = (Q_{\mathcal{TC}}, \Sigma_{\mathcal{TC}}, \Lambda_{\mathcal{TC}}, \rightarrow_{\mathcal{TC}}, \mathcal{C}_{\mathcal{TC}}, \mathtt{init}_{\mathcal{TC}})$ où *Pass*, *Fail*, *Inc*, *None* $\in \Lambda_{\mathcal{TC}}$ sont des couleurs qui caractérisent les verdicts et telles que $\mathcal{C}_{\mathcal{TC}}(\mathit{Pass})$, $\mathcal{C}_{\mathcal{TC}}(\mathit{Fail})$, $\mathcal{C}_{\mathcal{TC}}(\mathit{Inc})$ et $\mathcal{C}_{\mathcal{TC}}(\mathit{None})$ forment une partition de $Q_{\mathcal{TC}}$ et pour tout $\lambda \in \{\mathit{Pass}, \mathit{Fail}\}$, on a $\mathrm{Reach}_{\mathcal{TC}}(\Sigma, \Lambda) \subseteq \mathcal{C}_{\mathcal{TC}}(\lambda)$ et $\mathrm{Reach}_{\mathcal{TC}}(\Sigma, \mathit{Inc}) \subseteq \mathcal{C}_{\mathcal{TC}}(\mathit{Inc}) \cup \mathcal{C}_{\mathcal{TC}}(\mathit{Fail})$. L'alphabet est $\Sigma_{\mathcal{TC}} = \Sigma^?_{\mathcal{TC}} \cup \Sigma^!_{\mathcal{TC}}$ où $\Sigma^?_{\mathcal{TC}} = \Sigma^{!\delta}$ et $\Sigma^!_{\mathcal{TC}} = \Sigma^?$ (les sorties de $\mathcal{TC}$ sont les entrées de $\mathcal{M}$ et vice versa). Une *suite de tests* est un ensemble de cas de tests.

On modélise l'exécution d'un cas de test $\mathcal{TC}$ vis-à-vis d'une implémentation $\mathcal{I}$ par la composition parallèle $\mathcal{TC} \| \Delta(\mathcal{I})$. Ceci a pour effet de réaliser l'intersection de l'ensemble des traces suspendues de $\mathcal{I}$ avec celles de $\mathcal{TC}$ ($\mathrm{Traces}(\mathcal{TC} \| \Delta(\mathcal{I})) = \mathsf{STraces}(\mathcal{I}) \cap \mathrm{Traces}(\mathcal{TC})$). Ainsi, une erreur possible d'une implémentation vis-à-vis d'un cas de test se définit par le fait que l'interaction de $\mathcal{I}$ avec $\mathcal{TC}$ peut aboutir à un état coloré par $Fail$ dans $\mathcal{TC}$. Ceci peut être formalisé par $\mathcal{I}$ *fails* $\mathcal{TC} \stackrel{\text{def}}{=} \mathsf{STraces}(\mathcal{I}) \cap \mathrm{Traces}_{\mathcal{C}_{\mathcal{TC}}(\mathit{Fail})}(\mathcal{TC}) \neq \emptyset$. On peut observer que $\mathcal{I}$ *fails* $\mathcal{TC}$ signifie simplement que $\mathcal{I}$ *peut* être rejetée par $\mathcal{TC}$, selon les choix faits par $\mathcal{I}$ dans ses interaction avec $\mathcal{TC}$. Des définitions analogues peuvent être formulées pour les notations *passes* et *inconc* pour les verdicts *Pass* et *Inc*.

Nous pouvons à présent donner les définitions des propriétés formelles qui devront être satisfaites par les suites de tests pour rendre compte de la conformité d'une implémentation.

Définition 19 (Propriété des suites de tests)**.** Soit $\mathcal{M}$ une spécification, et $\mathcal{TS}$ une suite de tests pour $\mathcal{M}$.

— TS est *correcte* si aucun de ses cas de test ne peut rejeter une implémentation conforme :
$\forall \mathcal{I} \in \mathcal{IMP}(\mathcal{M}), (\mathcal{I}\ \mathit{ioco}\ \mathcal{M} \implies \forall \mathcal{TC} \in \mathcal{TS}, \neg(\mathcal{I}\ \mathit{fails}\ \mathcal{TC}))$.

— $\mathcal{TS}$ est *exhaustive* si elle rejette toutes les implémentations non-conformes :
$\forall \mathcal{I} \in \mathcal{IMP}(\mathcal{M}), (\neg(\mathcal{I}\ \mathit{ioco}\ \mathcal{M}) \implies \exists \mathcal{TC} \in \mathcal{TS}, \mathcal{I}\ \mathit{fails}\ \mathcal{TC})$.

— $\mathcal{TS}$ est *complète* si elle est correcte et exhaustive.

— $\mathcal{TS}$ est *stricte* si une non-conformité est détectée aussi tôt que possible :
$\forall \mathcal{I} \in \mathcal{IMP}(\mathcal{M}), \forall \mathcal{TC} \in \mathcal{TS}, \neg(\mathcal{TC} \| \mathcal{I}\ \mathit{ioco}\ \mathcal{M}) \Rightarrow \mathcal{I}\ \mathit{fails}\ \mathcal{TC}$.

Jeannet *et al.* [88] ont proposé des caractérisations équivalentes des propriétés de correction, exhaustivité et sévérité (être stricte) qui sont adapté pour démontrer qu'une suite de tests les satisfait. Elles sont obtenues en remplaçant la relation *ioco* par sa caractérisation alternative, *fails* par sa définition, en remplaçant la quantification universelle sur $\mathcal{TC}$ par une union, et en supprimant la quantification universelle sur $\mathcal{I}$ tout en remplaçant l'implication par une inclusion.

Proposition 76 ([88])**.** *Soit une suite de tests $\mathcal{TS}$ pour $\mathcal{M}$,*

— *$\mathcal{TS}$ est correcte si* $\bigcup_{\mathcal{TC} \in \mathcal{TS}} \mathrm{Traces}_{\mathcal{C}_{\mathcal{TC}}(\mathit{Fail})}(\mathcal{TC}) \subseteq \mathsf{MinFTraces}(\mathcal{M}).\Sigma^*$,

— *$\mathcal{TS}$ est exhaustive si* $\mathsf{MinFTraces}(\mathcal{M}) \subseteq \bigcup_{\mathcal{TC} \in \mathcal{TS}} \mathrm{Traces}_{\mathcal{C}_{\mathcal{TC}}(\mathit{Fail})}(\mathcal{TC})$,

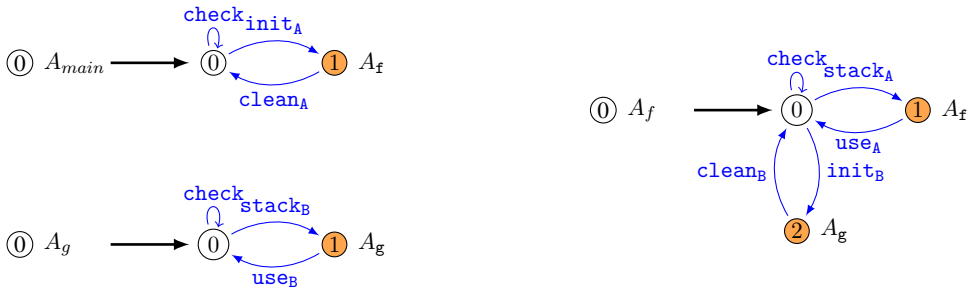


FIGURE 4.5 – Règles de $\mathcal{R}$.

— $\mathcal{TS}$ *est stricte si* $\forall \mathcal{TC} \in \mathcal{TS}, \text{Traces}(\mathcal{TC}) \cap \mathsf{MinFTraces}(\mathcal{M}) \subseteq \text{Traces}_{\mathcal{C}_{\mathcal{TC}}(Fail)}(\mathcal{TC})$.

De façon moins formelle, la correction est caractérisée par le fait que les observations de cas de test conduisant au verdict *Fail* sont non-conformes. L'exhaustivité reflète que toutes les observations non-conformes correspondent à atteindre *Fail* dans certains cas de test. La sévérité va plus loin en exigeant qu'un état *Fail* d'un cas de test correspond à chaque observation minimale non-conforme.

II Systèmes algébriques sous observation partielle

II.1 Un exemple de graphe régulier pour le diagnostic et l'opacité

Dans ce paragraphe, nous allons introduire un exemple qui étend l'exemple 9 de façon à prendre en compte les contraintes sur les ressources qui y sont exprimées.

Exemple 12. *La figure 4.5 présente trois règles pour définir un graphe régulier représentant la propriété qu'on souhaite garantir pour l'exemple 9.*

Chacun des non-terminaux est d'arité 1*, et pour simplifier, la figure 4.5 ne représente que les membres droits des règles.*

La règle de $A_{\texttt{main}}$ *modélise l'initialisation du système pour la ressource* `A`*, et le nettoyage final. La règle de* $A_{\texttt{f}}$ *modélise le stockage des ressources* `A` *ainsi que leur utilisation (le nettoyage intervenant une fois toutes les ressources consommées). Elle permet le passage dans le mode de traitement de la ressource* `B` *(initialisation, et nettoyage spécifique). La règle de* $A_{\texttt{g}}$ *permet le stockage et le traitement des ressources* `B`*. Les boucles* `check` *représentent des inspections de routine qui peuvent survenir à n'importe quel moment. La* HR*-grammaire qui correspond à cette définition est la suivante :* $\mathcal{R} = (N, \Sigma \cup \Lambda, R, \{A_{\texttt{main}} v_0\})$ *avec* $\Sigma_o = \{\texttt{check}, \texttt{stack}_\texttt{A}, \texttt{stack}_\texttt{B}, \texttt{use}_\texttt{A}, \texttt{use}_\texttt{B}\}$, $\Sigma_{uo} = \{\texttt{init}_\texttt{A}, \texttt{init}_\texttt{B}, \texttt{clean}_\texttt{A}, \texttt{clean}_\texttt{B}\}$, $\Lambda = \{\texttt{init}\}$, $N = \{A_{\texttt{main}}, A_{\texttt{f}}, A_{\texttt{g}}\}$*, et les membres droits des règles de* R *pour chacun des non-terminaux sont données sur la figure 4.5*

Le graphe engendré par les premières récritures de la HR*-grammaire* $\mathcal{R}$ *est représenté sur la figure 4.6.*

Comportement observable d'un graphe régulier. Pouvoir supprimer les transitions inobservables est essentiel dans l'étude d'un système sous observation partielle.

La proposition suivante, démontrée dans nos travaux conduits avec Chédor *et al.* [44, 46], est centrale pour la suite.

Proposition 77. *Soit une* HR*-grammaire* $\mathcal{G} = (N, T, R, H_0)$*, avec* $\Sigma_o \subseteq \Sigma$*, les événements observables, il est possible de construire une* HR*-grammaire* $Clo(\mathcal{G})$ *de taille exponentielle et qui engendre un graphe sans transition non-observable, de degré sortant fini, dont les observations conduisant à la couleur* $c \in T_1$*, sont les même que celles du graphe engendrées par* $\mathcal{G}$ *:* $obs_{Q_c}(\mathcal{G}^\omega) = \mathcal{L}_{Q'_c}((Clo(\mathcal{G}))^\omega)$.

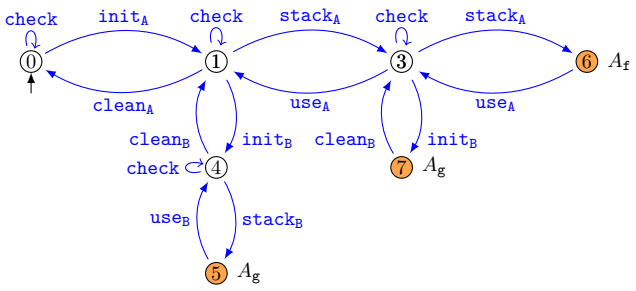


FIGURE 4.6 – Graphe engendré par trois étapes de récriture complète.

Déterminisation des graphes réguliers.

Définition 20. Une HR-grammaire $\mathcal{G}$ est *pondérée pour* `init` (une couleur) si dans le graphe $\mathcal{G}^\omega$ qu'elle engendre, il n'y a qu'un seul état coloré par `init`, et si pour tout mot $u \in T_2^*$ et tous états q, q' de $q_{\texttt{init}} \xrightarrow{u} q$ et $q_{\texttt{init}} \xrightarrow{u} q'$ implique $\ell(q) = \ell(q')$ (les deux sommets sont engendrés au même niveau).

En général, on ne précisera pas la couleur pour laquelle une HR-grammaire sera pondérée.

Remarque 8. Déterminer si un HR-grammaire est pondérée pour une couleur donnée est décidable en temps polynomial d'après un algorithme de Caucal *et al.* [39].

Comme nous l'avons vu précédemment, la construction d'une machine à typer repose principalement sur une déterminisation. Il est immédiat de vérifier si le graphe engendré par une HR-grammaire est déterministe, de plus lorsqu'une telle grammaire est pondérée on peut en produire une équivalente dont le graphe engendré est déterministe.

Proposition 78 ([39])**.** *Étant donnée une* HR*-grammaire $\mathcal{G}$ pondérée, il est possible de produire une grammaire $D(\mathcal{G})$ qui engendre un graphe déterministe, qui possède le même ensemble de traces (pour tout ensemble de couleur cible). La grammaire $D(\mathcal{G})$ est de taille exponentielle par rapport à celle de $\mathcal{G}$.*

Le théorème qui suit énonce une condition suffisante pour laquelle on peut construire une machine à typer pour une HR-grammaire donnée.

Proposition 79. *Soit $\mathcal{G}$ une* HR*-grammaire telle que sa clôture observable, $Clo(\mathcal{G})$, soit pondérée. Alors, la déterminisation de $Clo(\mathcal{G})$ est une machine à typer pour le système engendré par $\mathcal{G}$.*

Produit synchrone. Les graphes réguliers ne sont pas clos par produit synchrone. En effet, l'intersection de deux langages algébriques peut être obtenue comme la trace du produit synchrone de deux graphes réguliers. En fait, le vide de l'intersection de deux langages algébriques interdit ce produit synchrone de se situer à un quelconque niveau de la hiérarchie de Caucal. En revanche, il est tout a fait possible de construire le produit synchrone d'un graphe régulier avec un DES fini. Cette possibilité offre une grande richesse en terme de modélisation. Il est ainsi possible de modéliser un système et une propriété séparément, puis de les combiner, dès lors que l'un des deux (système ou propriété) possède un nombre d'états finis.

Dans le cas des HR-grammaires pondérées il est possible de construire l'auto-produit d'un graphe régulier engendré par une telle grammaire. Le résultat étant engendré par une HR-grammaire. Étant donnée $\mathcal{G} = (N, T, R, H_0)$ une HR-grammaire pondérée, on définit son auto-produit de la façon suivante : $\mathcal{G}^2 = (N^2, T, R^2, H_0^2)$, pour tout couple de non-terminaux $A_1, A_2 \in N$, on définit le non-terminal $A_{1\times 2}$ dans N^2 l'arité de ce non-terminal est le produit des deux arités, les hyperarcs étiquetés par ce non-terminal sont tous les couples de sommets sur lesquels portent chacun des non-terminaux. Les règles

de R^2 sont définies pour les non-terminaux du type $A_{1\times 2}$, il s'agit du produit synchrone des membres droits des règles de A_1 et A_2, et les non-terminaux sont tous les couples possibles $A_{i\times j}$ pour A_i dans le membre droit de A_1, et A_j dans celui de A_2. L'axiome est construit de façon analogue.

En s'appuyant sur cette construction, on obtient le résultat suivant.

Proposition 80. *L'auto-produit d'une* HR*-grammaire pondérée pour* `init`*, est une* HR*-grammaire pondérée pour* `init`*, de taille quadratique.*

II.2 Problèmes d'opacité et de diagnosticabilité pour les graphes réguliers

Nous en venons donc à l'opacité et la diagnosticabilité des systèmes modélisés par des graphes réguliers.

Proposition 81. *Le problème de l'opacité est indécidable pour les systèmes modélisés par des graphes réguliers.*

Démonstration. On peut réduire le problème de l'inclusion pour des langages algébriques de la façon suivante : soient $\mathcal{L}$ et $\mathcal{L}'$ deux langages algébriques sur l'alphabet Σ, soient s et $\#$ deux nouveaux symboles qui n'appartiennent pas à Σ (on note $\Sigma_\#$ l'ensemble $\Sigma \cup \{\#\}$). On considère à présent les langages obtenus en concaténant le symbole $\#$ à la fin de chacun des mots contenu dans les deux langages : $\mathcal{L}\#$ et $\mathcal{L}'\#$ Ces deux langages peuvent être la traces de deux graphes réguliers à partir d'une couleur initiale (voir, par exemple Caucal, § 5 [37]) considérons les HR-grammaires $\mathcal{G} = (N, T, R, H_0)$ et $\mathcal{G}' = (N', T, R', H'_0)$ qui possèdent chacune un unique état de leur axiome, coloré par `init`. Les sommets atteint par un chemin étiqueté par $\mathcal{L}'\#$ seront colorés par λ. Considérons à présent la HR-grammaire définit par : $\mathcal{G}'' = (N \cup N', T, R \cup R', H''_0)$ où H''_0 est formé par l'union disjointe des deux axiomes, auquel on ajoute un sommet supplémentaire, q_0 qui est le seul portant la couleur `init`, il est relié par deux transitions non-observables aux états initiaux présents dans les axiomes H_0 et H'_0. En outre, on fait en sorte que tous les sommets qui ne sont pas colorés par λ possèdent $\overline{\lambda}$. Ainsi, les chemins aboutissant à un chemin secret sont étiquetés par un mot de $\mathcal{L}'\#$, comme toutes les étiquettes des chemins non-secrets qui se terminent par $\#$ sont précisément les mots de $\mathcal{L}\#$, on a donc, $\mathcal{L}' \subseteq \mathcal{L}$ si, et seulement si, l'ensemble des états secrets est opaque dans le graphe engendré par $\mathcal{G}''$. □

En fait, il est possible de reprendre le processus de décision qui permet de résoudre l'opacité pour les DES finis (comme discuté en § I.1), c'est-à-dire en construisant la machine à typer. Dès que la clôture d'une HR-grammaire est pondérée on peut le réaliser.

Définition 21. La classe des HR-grammars à *clôture pondéré*, notée $\mathcal{C}wHRG(\Sigma_o)$, est formée des HR-grammars dont la clôture (sur l'alphabet d'observable Σ_o) est pondérée.

La notion de clôture pondérée est importante puisqu'elle constitue une classe sur laquelle les problèmes d'observation partielle que nous considérons ici deviennent décidables.

Théorème 82. *Le problème de l'opacité est dans* 2EXPTIME *pour la classe des* $\mathcal{C}wHRG(\Sigma_o)$*. En outre, ce problème est* EXPTIME*-difficile.*

Démonstration. Étant donnée une HR-grammaire $\mathcal{G}$, on utilise la proposition 79 pour obtenir la machine à typer $D(Clo(\mathcal{G}))$. Ensuite, il suffit de déterminer l'accessibilité des états $\{\lambda\}$ dans la machine à typer, ce qui est décidable, d'après la proposition 9. Ceci peut être réalisé en temps doublement exponentiel (puisque la taille de la machine est doublement exponentielle, et l'accessibilité polynomiale). Pour la dureté exponentielle, l'universalité est EXPTIME-complète pour les automates à pile visible (voir Alur *et al.* [2]) qui est une sous-classe des HR-grammaires pondérées. Et l'universalité peut être réduite à l'opacité comme nous l'avons déjà vu, nous avons donc que le problème est EXPTIME-difficile. □

La borne supérieure de 2EXPTIME peut sembler très pénalisante par rapport au problème dans le cas fini, qui est *simplement* PSPACE-complet. Néanmoins, cette complexité résulte de la borne supérieure exponentielle de la clôture qui dans les cas ordinaires n'est pas atteinte. En particulier, si on peut borner la longueur des suites d'action non-observables, la clôture devient alors polynomiale, ce qui ensuite induit une borne supérieure EXPTIME pour décider l'opacité.

En ce qui concerne le diagnostic, on a le résultat suivant.

Proposition 83. *Le problème du diagnostic est indécidable*

Ce résultat est une conséquence immédiate d'un résultat analogue que nous avons obtenu avec Pinchinat, pour les automates visibles [116]. Cependant, nous avons un résultat analogue au théorème 82.

Théorème 84. *Le problème de la diagnosticabilité pour la classe des $\mathcal{C}$wHRG(Σ_o) est dans* EXPTIME.

Démonstration. Soit une HR-grammaire $\mathcal{G} \in \mathcal{C}\text{wHRG}(\Sigma_o)$, puisque sa clôture $Clo(\mathcal{R})$ est pondérée, d'après la proposition 80, on peut construire une HR-grammaire engendrant l'autoproduit du graphe qu'elle engendre. À présent, le lemme 74, exprime que la non-diagnosabilité est équivalente à l'existence d'un chemin ambigu dont les états on pour type $\left\{\lambda, \overline{\lambda}\right\}$. Ce qui peut être décidé en s'appuyant sur la proposition 10. Pour ce qui est de la complexité nous utilisons exclusivement des algorithmes polynomiaux, et la clôture d'une HR-grammaire est de taille exponentielle, d'où le résultat. □

Ici, on peut regretter la borne supérieure en temps exponentielle, cependant, comme il a été dit dans le cas de l'opacité, cette borne résulte du calcul de la clôture, et en règle générale, la borne supérieure n'est pas atteinte. Dès lors qu'on identifie une restriction pour laquelle le calcul de la clôture est toujours polynomial (par exemple, l'existence d'une borne sur les suites d'actions non-observables) la diagnosticabilité devient alors polynomiale aussi.

La machine à typer

Que ce soit pour l'opacité et le diagnostic, la machine à typer est constituée d'un graphe régulier obtenu à partir de la HR-grammaire $\mathcal{D}(Clo(\mathcal{G}))$. Dans le cas de l'opacité, cet objet permet de détecter la survenue d'une fuite d'information. Lorsqu'un système est diagnosticable, on a la certitude que, si une défaillance se produit, la machine à typer sera en mesure de la détecter avec certitude. On peut imaginer aussi formuler le problème un peu différemment en donnant un alphabet distinct à l'attaquant et au moniteur. On parle alors de diagnosticabilité de fuite d'information. Ce problème est assez délicat et n'a pas été totalement résolu dans notre travail avec Chédor *et al.* [46].

II.3 Un exemple pour le test

Dans ce paragraphe, nous introduisons un exemple de HR-grammaire pertinent pour le test. Pour cela nous construisons une grammaire qui engendre une représentation du graphe de flot de contrôle du programme illustré sur la figure 4.4.

Exemple 13. Voici une HR-grammaire qui engendre un graphe modélisant une abstraction du programme représenté sur la figure 4.4 : $\mathcal{G} = (\{A_c\}, \Sigma \cup \Lambda, R, H_0)$ où on a $\Sigma^\tau = \{\texttt{try}, \texttt{throw}, \texttt{catch}, \texttt{intern}\}$, $\Sigma^? = \{\texttt{int}, \texttt{true}, \texttt{false}\}$, $\Sigma^! = \{\texttt{m1}, \texttt{m2}, \texttt{m3}, \texttt{m4}\}$, $\Lambda = \{\texttt{init}, \texttt{succ}\}$, une représentation graphique de l'ensemble des règles est donnée sur la figure 4.7). Les actions de sortie correspondent aux messages : `m1` est `Fin`, `m2` est `Erreur !`, `m3` est `Un texte` et `m4` est `Vous avez arrêté`. Le symbole `int` représente l'entrée d'un entier ; on peut souligner que la valeur de cet entier n'a pas d'incidence sur la structure du graphe. Les entrées `true`, `false` représente la saisie d'un booléen du bloc 4. Le symbole d'action interne `intern` correspond au calcul du bloc 5.

La figure 4.8 illustre le graphe produit après deux étapes de récriture.

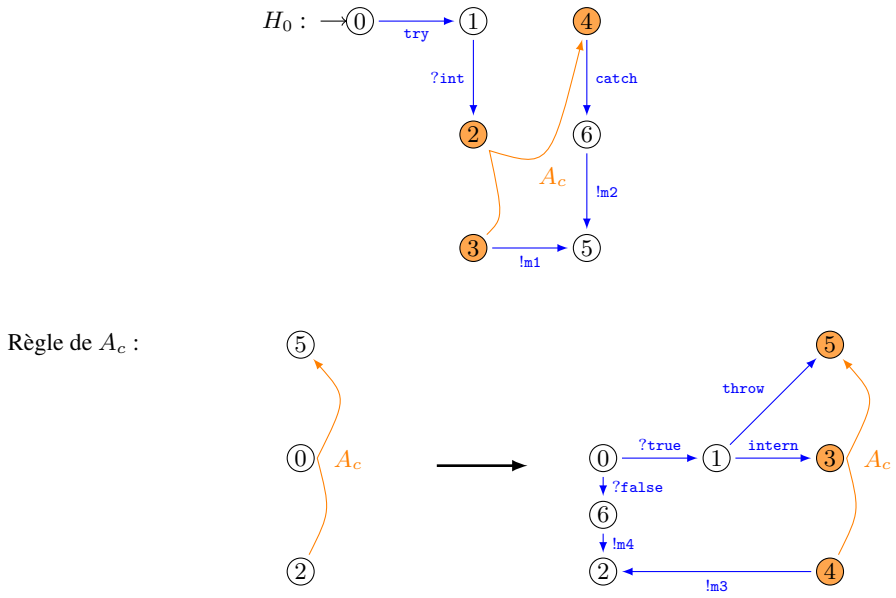


FIGURE 4.7 – Axiome et règle de la HR-grammaire de l'exemple 13

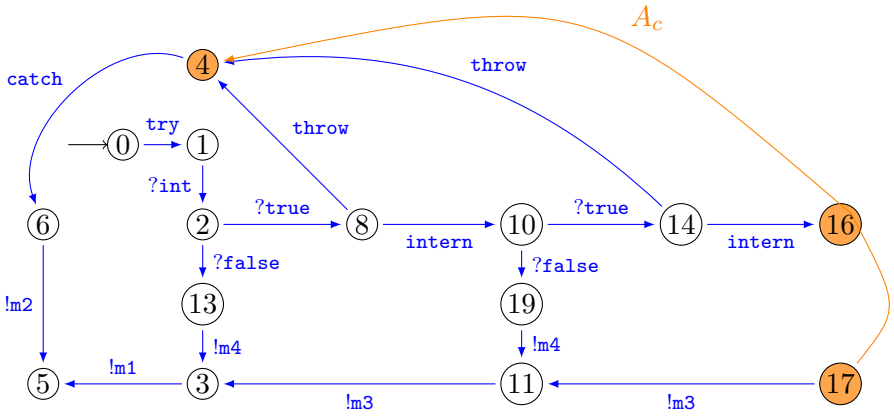


FIGURE 4.8 – Graphe obtenu après deux étapes de récriture.

II.4 Exécutions effectives dans les HR-grammaires

Dans ce paragraphe, nous allons sommairement présenter comment utiliser une HR-grammaire pour suivre le déroulement effectif d'une exécution. Il s'agit d'un point clé pour pouvoir utiliser de telles grammaires dans le contexte du test (qu'il soit hors-ligne ou en-ligne), mais aussi le diagnostic ou plus généralement, la supervision. En particulier, il est important de voir qu'il n'est pas nécessaire de construire l'intégralité du graphe après n étapes de récriture pour déterminer la validité d'une exécution, même si celle-ci visite des sommets à une telle profondeur.

Exécutions dans les graphes réguliers *déterministes*. Soient une HR-grammaire $\mathcal{G} = (N, T, R, H_0)$ engendrant un graphe régulier déterministe, et un mot $w = \mu_0 \cdots \mu_{n-1} \in \Sigma^*$, soit $v_0 \in H_0$ tel que $v_0 \in \texttt{init}$. On appelle *chemin symbolique*, étiqueté par w une suite de la forme suivante : $(v_0, \varepsilon) \xrightarrow{\mu_0} (v_1, U_1) \cdots (v_i, U_i) \xrightarrow{\mu_i} (v_{i+1}, U_{i+1}) \cdots \xrightarrow{\mu_{n-1}} (v_n, U_n)$, où chaque mot U_i est une suite de symboles de N il représente les différents non-terminaux traversés, et chacun des v_i est un sommet du membre droit le la règle du dernier non-terminal de U_i (de l'axiome, si U_i est ε). On fait ici la supposition que dans le membre droit de chacune des règles, au plus une seule occurrence de chaque non-terminal peut exister (ce qui est toujours possible au prix de quelques duplications). Pour chacun des i, la transition $(v_i, U_i) \xrightarrow{\mu_i} (v_{i+1}, U_{i+1})$ (on pose $U_i := U_i' A_i$) correspond à un des trois cas décrits ci-dessous (on peut observer que ces trois opérations correspondent respectivement aux transitions interne, de dépilement et d'empilement des automates à pile).

- (interne) la transition étiquetée par μ_i appartient au membre droit de A_i (mais *pas une des entrées*), alors, v_{i+1} est simplement le sommet cible de la transition dans ce membre droit, et $U_{i+1} = U_i$;
- (dépile) la transition étiquetée par μ_i appartient au membre droit de A_i, mais atteint une des *entrées*, alors on identifie le sommet correspondant dans le membre droit de la règle associée au dernier non-terminal de U_i' (si c'est le mot vide, on utilise l'axiome H_0), ce sommet, est v_{i+1}, et $U_{i+1} := U_i'$;
- (empile) le sommet v_i appartient à un non-terminal A_{i+1}, et la transition étiqueté par μ_i est dans le membre droit de A_{i+1} issue du sommet correspondant à v_i. Alors, le sommet v_{i+1} est la cible de cette transition, et on a $U_{i+1} = U_i A_{i+1}$.

Lorsque le graphe régulier est déterministe, il y a au plus un chemin symbolique pour un mot donné, de plus, la construction du chemin symbolique ne nécessite pas de calculer l'ensemble du graphe traversé par le chemin. Plus encore, seul le dernier couple d'un chemin symbolique est nécessaire pour vérifier la validité d'une étape suivante, ou pour reconstituer à partir de la trace d'une exécution, les étapes successives.

Exécutions dans les graphes réguliers *non-déterministes*. Pour ce qui est des systèmes modélisés par des graphes réguliers non-déterministes, un chemin w de Σ^* peut étiqueter de nombreux chemins symboliques à partir d'un état étiqueté par `init`. Il peut y avoir un nombre exponentiel de tels chemins (vis-à-vis de la longueur du mot w). Plus encore, il n'y a aucune garantie vis-à-vis des mots de N^* qui peuvent évoluer de façon complètement indépendante et avoir des longueurs comprises entre 0 et $|w|$. En revanche, pour les HR-grammaires pondérées, un mot w peut toujours engendrer un ensemble de chemins symboliques de taille exponentielle (c'est déjà le cas pour les systèmes finis non-déterministes), mais chacun de ces mots (dans N^*) possède la même taille (du fait de la pondération de la grammaire). Cette propriété permet une plus grande efficacité dans le stockage et le calcul des continuations.

Une note sur l'implémentation. Pour implémenter avec efficacité les exécutions sur les systèmes modélisés par des HR-grammaires, le programme doit accéder à une unique représentation de chacun des membres droits de règles. Pour chacun des chemins symboliques, seul un couple formé d'un sommet et d'un mot de non-terminaux doit être mémorisé. Cette structure de donnée peut devenir de très grande

taille, mais beaucoup plus réduite que le graphe obtenu après k étapes de récriture, lorsque k est grand (par rapport au nombre de tuiles).

II.5 Génération de tests hors-ligne pour les IOLTS engendrés par des HR-grammaires

Dans ce paragraphe, on s'intéresse à la génération de cas de test pour des IOLTS engendrés par des HR-grammaires, en particulier lorsque ces dernières sont pondérées. Cette dernière caractéristique assure que le système est déterminisable, et permets de produire, hors ligne, des cas de tests. Nos algorithmes permettent en outre de s'appuyer sur un objectif de test, noté $\mathcal{TP}$. Notre processus de construction des cas de tests s'inspire fortement des travaux de Jard *et al.* [87]) : tout d'abord on construit un système suspendu, $\Delta(\mathcal{M})$, ce système est ensuite déterminisé en $deter(\mathcal{M})$. Ce dernier est ensuite complété en ajoutant les transitions pour les sorties non-spécifiées vers de nouveaux états *Fail*. Un inversant les actions on obtient le *testeur canonique* noté $Can(\mathcal{M})$. La avant-dernière étape consiste à réaliser le produit de ce testeur canonique avec l'objectif de test : $Can(\mathcal{M}) \times \mathcal{TP}$ le verdict de succès, *Pass* provient des états accepteurs de l'objectif de test. Le dernier point de ce processus consiste à réaliser une analyse de co-accessibilité depuis les états étiquetés par *Pass*, ceci permet d'étiqueter certains sommets par les verdicts *None* et *Inc* par complémentaire les actions de sorties qui conduisent à des sommets de type *Inc* ainsi que les transitions depuis *Inc* sont retirées du cas de test produit. Chacune de ces opérations sera réalisée sur une HR-grammaire, induisant des propriétés sur le graphe engendré et donc sur les cas de tests ainsi obtenus.

Construction du testeur canonique

Quiescence. Comme il a été vu en § I.2 la quiescence représente l'absence de réaction visible de la part de la spécification. Compte tenu de l'existence de chemins infinis faisant intervenir des ensembles infinis de sommets, la détection de la quiescence est un peu plus subtile pour les graphes réguliers. Néanmoins, en se reposant sur la proposition 10, il est possible d'établir le résultat suivant.

Proposition 85. *Pour toute* HR*-grammaire $\mathcal{G}$, il est possible de construire, de façon effective, une* HR*-grammaire noté $\Delta(\mathcal{G})$ qui engendre un graphe isomorphe à la suspension du graphe engendré par $\mathcal{G}$. On a* $\mathrm{Traces}((\Delta(\mathcal{G}))^\omega) = \mathsf{STraces}(\mathcal{G}^\omega)$.

Complétion en sortie. Une fois qu'on a construit un IOLTS suspendu, en utilisant la proposition 85, il est nécessaire de compléter le graphe engendré. On note la HR-grammaire obtenue à partir de $\Delta(\mathcal{G})$ ainsi $CS(\mathcal{G})$ qui permet de reconnaître $\mathsf{STraces}(\mathcal{G}).\Sigma^{!\delta}$ avec une nouvelle couleur *UnS*.

On procède de la façon suivante : pour chaque non-terminal A, dans le membre droit de la règle de A, on ajoute un nouveau sommet, noté $v_{\mathcal{A}}^{UnS}$ (il possède la couleur *UnS*), et pour chaque autre état, on ajoute les nouvelles transitions qui y conduisent pour toutes les sorties non-spécifiées :

$$\left\{ v \xrightarrow{\mu} v^{UnS} \mid v \in Q_A \wedge \mu \in \Sigma^{!\delta} \wedge \mu \notin \Gamma_{(\Delta(\mathcal{G}))^\omega}(v) \right\}.$$

On rappelle que la valeur $\Gamma_{(\Delta(\mathcal{G}))^\omega}(v)$ re présente l'ensemble des sorties possible dans un état, sa valeur peut être déterminé, et rendu uniforme (au prix éventuel qu'une recopie de règle).

Par construction les équations suivantes sont valides :

$$\mathrm{Traces}_{\mathcal{C}(UnS)}((CS(\mathcal{G}))^\omega) \subseteq \mathsf{STraces}(\mathcal{G}^\omega).\Sigma_{\mathcal{G}}^{!\delta} \tag{4.2}$$

$$\mathrm{Traces}_{\overline{\mathcal{C}}(UnS)}((CS(\mathcal{G}))^\omega) = \mathsf{STraces}(\mathcal{G}^\omega) \tag{4.3}$$

$$\mathrm{Traces}((CS(\mathcal{G}))^\omega) = \mathsf{STraces}(\mathcal{G}^\omega).\Sigma_{\mathcal{G}}^{!\delta} \cup \mathsf{STraces}(\mathcal{G}^\omega) \tag{4.4}$$

Si on détaille, l'inégalité (4.2) exprime que les traces reconnues en *UnS* sont des traces de suspension de $\mathcal{G}$ prolongés par des sorties. L'égalité (4.3) est vérifiée car on a $\mathrm{Traces}_{\overline{\mathcal{C}}(UnS)}((CS(\mathcal{G}))^\omega)$, qui sont les traces d'exécutions aboutissant en dehors de la couleur *UnS*, sont des traces suspendues de $\Delta(\mathcal{G})$, et

donc $\mathsf{STraces}((\mathcal{G})^\omega)$. Enfin, l'égalité (4.4) résulte de l'union de (4.2) et (4.3). On peut cependant relever que cette union n'est pas disjointe, il y a des traces qui sont à la fois dans $\mathrm{Traces}_{\overline{\mathcal{C}}(\mathit{UnS})}((CS(\mathcal{G}))^\omega)$ et $\mathrm{Traces}_{\mathcal{C}(\mathit{UnS})}((CS(\mathcal{G}))^\omega)$, ce peut être l'observation d'une exécution dans $\Delta(S)$ et d'une autre conduisant à *UnS*.

En développant la définition de $\mathsf{MinFTraces}(\mathcal{G}^\omega)$, on obtient

$$\mathsf{MinFTraces}(\mathcal{G}^\omega) = \mathrm{Traces}_{\mathcal{C}(\mathit{UnS})}((CS(\mathcal{G}))^\omega) \setminus \mathrm{Traces}_{\overline{\mathcal{C}}(\mathit{UnS})}((CS(\mathcal{G}))^\omega) \tag{4.5}$$

Remarque 9. On peut noter que dans le processus classique pour les IOLTS finis, la déterminisation précède la complétion en sortie. Ce choix provient du fait que la déterminisation n'est pas toujours possible pour les systèmes engendrés par des HR-grammaires. Ainsi nous poussons la construction valable dans le cas général aussi loin que possible. Notre définition de la couleur *UnS* est donc légèrement différente de celle usuelle. Mais les verdicts seront également adaptés pour tenir compte de cette modification. À la fois pour le traitement en-ligne et hors-ligne (dans le cas des systèmes pondérés).

Σ^τ-clôture. En utilisant la proposition 77, à partir du système $CS(\mathcal{G})$ on peut construire une HR-grammaire, notée $Clo(CS(\mathcal{G}))$, dont le graphe engendré ne possède plus aucune action interne, et possède les mêmes ensembles de traces de couleur à couleur que $(CS(\mathcal{G}))^\omega$. On en déduit immédiatement que $Clo(CS(\mathcal{G}))$ satisfait les même égalités et inégalités que (4.2), (4.3) et (4.4) :

$$\mathrm{Traces}_{\mathcal{C}(\mathit{UnS})}((Clo(CS(\mathcal{G})))^\omega) \subseteq \mathsf{STraces}(\mathcal{G}^\omega).\Sigma_{\mathcal{G}}^{!\delta} \tag{4.6}$$

$$\mathrm{Traces}_{\overline{\mathcal{C}}(\mathit{UnS})}((Clo(CS(\mathcal{G})))^\omega) = \mathsf{STraces}(\mathcal{G}^\omega) \tag{4.7}$$

$$\mathrm{Traces}((Clo(CS(\mathcal{G})))^\omega) = \mathsf{STraces}(\mathcal{G}^\omega).\Sigma_{\mathcal{G}}^{!\delta} \cup \mathsf{STraces}(\mathcal{G}^\omega) \tag{4.8}$$

De surcroit, l'égalité (4.5) se transpose de façon immédiate à $Clo(CS(\mathcal{G}))$:

$$\mathsf{MinFTraces}(\mathcal{G}^\omega) = \mathrm{Traces}_{\mathcal{C}(\mathit{UnS})}((Clo(CS(\mathcal{G})))^\omega) \setminus \mathrm{Traces}_{\overline{\mathcal{C}}(\mathit{UnS})}((Clo(CS(\mathcal{G})))^\omega) \tag{4.9}$$

Le testeur canonique. Lorsque la HR-grammaire $Clo(CS(\mathcal{G}))$ est pondérée, la proposition 78 permet de construire $\mathcal{D}(Clo(CS(\mathcal{G})))$ qui engendre un IOLTS déterministe. À partir de $\mathcal{D}(Clo(CS(\mathcal{G})))$ la grammaire notée $Can(\mathcal{G})$ est appelée par abus de langage le *testeur canonique* de $\mathcal{G}$. Elle est obtenue par les opérations suivantes :

— la nouvelle couleur, *Fail* est ajoutée et les états de $\mathcal{D}(Clo(CS(\mathcal{G})))$ sont colorés par *Fail* s'ils sont formés d'états tous colorés par *UnS* dans $Clo(CS(\mathcal{G}))$, qu'ils reconnaissent donc les traces d'exécutions aboutissant toutes à *UnS*.

— les entrées et sorties sont inversées dans $Can(\mathcal{G})$ (vis-à-vis de $\mathcal{G}$).

À partir de la construction et de l'égalité (4.9) on obtient :

$$\mathrm{Traces}_{\mathcal{C}(\mathit{Fail})}((Can(\mathcal{G}))^\omega) = \mathsf{MinFTraces}(\mathcal{G}^\omega) \tag{4.10}$$

$$\mathrm{Traces}_{\overline{\mathcal{C}}(\mathit{Fail})}((Can(\mathcal{G}))^\omega) = \mathsf{STraces}(\mathcal{G}^\omega) \tag{4.11}$$

puis

$$\mathrm{Traces}((Can(\mathcal{G}))^\omega) = \mathsf{STraces}(\mathcal{G}^\omega) \cup \mathsf{MinFTraces}(\mathcal{G}^\omega) \tag{4.12}$$

Ici l'union est disjointe.
De plus

$\mathrm{Traces}_{\overline{\mathcal{C}}(\mathit{Fail})}((Can(\mathcal{G}))^\omega) = \mathrm{Traces}_{\overline{\mathcal{C}}(\mathit{UnS})}((Clo(CS(\mathcal{G})))^\omega) = \mathsf{STraces}(\mathcal{G}^\omega)$ d'après l'égalité (4.7)

et

$$\begin{aligned}\mathrm{Traces}_{\mathcal{C}(Fail)}((Can(\mathcal{G}))^\omega) &= \mathrm{Traces}_{\mathcal{C}(UnS)}((Clo(CS(\mathcal{G})))^\omega) \setminus \mathrm{Traces}_{\overline{\mathcal{C}}(UnS)}((CS(\mathcal{G}))^\omega) \\ &= \mathsf{MinFTraces}(\mathcal{G}^\omega) \text{ d'après l'égalité (4.9)}\end{aligned}$$

On déduit immédiatement de l'égalité (4.10) que la suite de tests $\mathcal{TS}$ réduite au testeur canonique, $\mathcal{TS} = \{Can(\mathcal{G})\}$, est correcte et exhaustive (voir § I.2). La suite $\mathcal{TS}$ est également stricte, ce qui provient de l'égalité qui suit :

$$\mathrm{Traces}((Can(\mathcal{G}))^\omega) \cap \mathsf{MinFTraces}(\mathcal{G}^\omega) = \mathrm{Traces}_{\mathcal{C}(Fail)}((Can(\mathcal{G}))^\omega)$$

en notant que l'égalité (4.12) est une union disjointe et en utilisant l'égalité (4.11).

Exemple 14. La figure 4.9 illustre les règles du testeur canonique obtenu pour l'exemple 13. On peut souligné qu'il a été suspendu, et que les actions internes {`try`, `throw`, `catch`, `intern`}, ont été supprimées. L'état portant la couleur F sont ceux qui ont la couleur *Fail*.

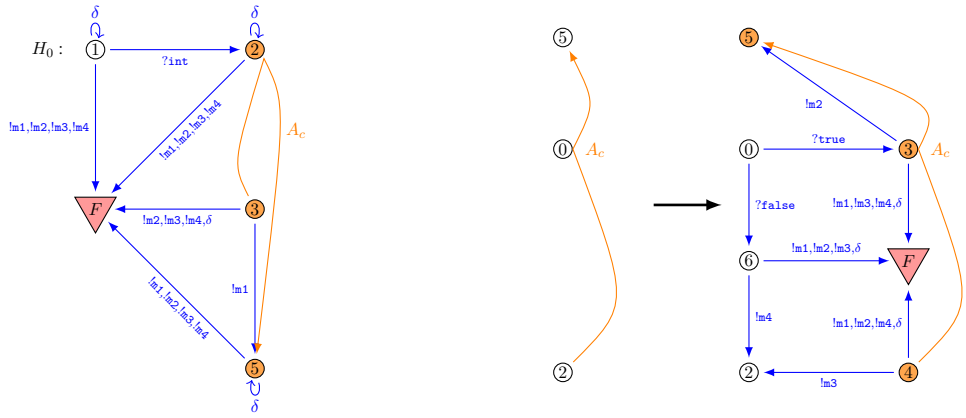


FIGURE 4.9 – Exemple de testeur canonique.

Sélection de cas de test par objectifs. Les propriétés satisfaites par le testeur canonique sont très importantes. Cependant, il est parfois intéressant de focaliser le test sur un sous-ensemble de comportements. Les *objectifs de test* sont conçu avec cette idée. Dans le cas présent, un objectif de test sera présenté comme un IOLTS déterministe et fini. Comme le produit d'un IOLTS fini et de l'IOLTS engendré par une HR-grammaire est lui-même engendré par une HR-grammaire, cela fonctionnera.

Définition 22 (Objectif de test)**.** Un *objectif de test* est un IOLTS déterministe et fini, noté $\mathcal{TP}$, sur l'alphabet Σ_o^δ, avec une couleur particulière, *Accept*, de sorte que les états la possédant n'aient pas de successeurs.

Comme nous l'avons déjà vu, le produit $\mathcal{P}$ entre $Can(\mathcal{G})$ et $\mathcal{TP}$ est une HR-grammaire. Sur ce produit on définit les couleurs de la façon suivante :

— $\mathcal{C}_\mathcal{P}(Fail) = \mathcal{C}_{Can(\mathcal{G})}(Fail) \times Q_{\mathcal{TP}}$
— $\mathcal{C}_\mathcal{P}(Pass) = \overline{\mathcal{C}}_\mathcal{P}(Fail) \times \mathcal{C}_{\mathcal{TP}}(Accept)$
— $\mathcal{C}_\mathcal{P}(None) = Coreach(\Sigma_o^\delta, \mathcal{C}_\mathcal{P}(Pass)) \setminus \mathcal{C}_\mathcal{P}(Pass)$
— $\mathcal{C}_\mathcal{P}(Inc) = Q_P \setminus (\mathcal{C}_\mathcal{P}(Fail) \cup \mathcal{C}_\mathcal{P}(Pass) \cup \mathcal{C}_\mathcal{P}(None))$

Par construction, chacun des sommets ne possédera qu'une seule couleur parmi $\{Fail, Pass, None, Inc\}$. Les sommets de couleur *Fail* ou *Pass* n'ont pas de successeurs, et ceux colorés par *Inc* ne possèdent que des successeurs de couleur *Fail* ou *Inc*.

Pour éviter les sommets de couleur *Inc* où l'objectif ne peut plus être satisfait, les transitions étiquetés par une sortie (une entrée de $\mathcal{G}$, fournie par l'environnement) et qui aboutissent à un état coloré par *Inc* sont élagués. De même que celle quittant les états *Inc*. Ainsi, les exécutions qui aboutissent à un état coloré par *Inc* se termine nécessairement par une entrée.

Ainsi, le cas de test $\mathcal{TC}$ obtenu à partir de $\mathcal{G}$ et $\mathcal{TP}$ est le produit $\mathcal{P}$, avec les nouvelles couleurs *Fail*, *Pass*, *None*, *Inc* et élagué comme indiqué ci-dessus.

Complexité du calcul du testeur canonique

La génération formelle de tests de conformité repose principalement sur le calcul du testeur canonique. Nous en avons décrit les étapes dans le paragraphe précédent.

Il y a deux étapes qui engendre une explosion exponentielle dans la construction du testeur canonique : la déterminisation, qui n'est pas évitable (et est déjà présente dans le cas des systèmes finis), et une autre lors du calcule de la Σ^τ-clôture (le calcul de $Clo(CS(\mathcal{G}))$). Comme nous en avons discuté pour le diagnostic et l'opacité, cette explosion résulte de la possibilité d'avoir des chemins étiquetés par Σ^τ, qui traverse les membres droits des règles. En pratique pour atteindre la limite exponentielle, il faudrait des arbres complets de tels chemins non-observables. En pratique dans le cas de la clôture, cette borne est rarement atteinte.

Voici un résumé de la complexité de chacune des étapes impliquées dans la construction de $Can(\mathcal{G})$:

1) Calcul de $\Delta(\mathcal{G})$: **polynomial en temps**
2) Calcul de $CS(\mathcal{G})$: **polynomial en temps**
3) Calcul de $Clo(CS(\mathcal{G}))$: **exponentiel en espace**
4) Vérifier si $Clo(CS(\mathcal{G}))$ is weighted : **polynomial en temps**
5) Determiniser $Clo(CS(\mathcal{G}))$: **exponential en espace**
6) Produit synchrone $Can(\mathcal{G}) \times \mathcal{TP}$ et sélection : **polynomial en temps**

Propriétés des tests engendrés

Le présent paragraphe résume les propriétés des cas de test tels qu'on les définit. Compte tenu de l'existence d'objectifs de test, nous considérerons également une propriété de *précision* qui associe la survenue du verdict *Pass* à la satisfaction de l'objectif de test.

Correction et sévérité. De part la construction d'un cas de test, $\mathcal{P} = Can(\mathcal{G}) \times \mathcal{TP}$, et la définition de $\mathcal{C}_\mathcal{P}(Fail)$, ainsi que l'élagage la sélection par $\mathcal{TP}$ n'ajoute aucun coloriage *Fail* par rapport au testeur canonique, $Can(\mathcal{G})$, donc on a $\mathrm{Traces}_{\mathcal{C}(Fail)}(\mathcal{TC}^\omega) = \mathrm{Traces}(\mathcal{TC}^\omega) \cap \mathrm{Traces}_{\mathcal{C}(Fail)}((Can(\mathcal{G}))^\omega)$. L'équation suivante dérive de l'équation (4.10) : $\mathrm{Traces}_{\mathcal{C}(Fail)}(\mathcal{TC}^\omega) = \mathrm{Traces}(\mathcal{TC}^\omega) \cap \mathsf{MinFTraces}(\mathcal{G}^\omega) \subseteq \mathsf{MinFTraces}(\mathcal{G}^\omega)$. Ceci établit simultanément la correction (l'inclution) et la sévérité (l'égalité).

Exhaustivité. En remarquant que pour toute trace individuelle, on peut construire un objectif de test qui la reconnaît, on obtient l'exhaustivité en parcourant l'ensemble des objectifs de test. On a bien : $\bigcup_{\mathcal{TC} \in \mathcal{TS}} \mathrm{Traces}_{\mathcal{C}(Fail)}(\mathcal{TC}^\omega) \supseteq \mathsf{MinFTraces}(\mathcal{G}^\omega)$.

Précision. On complète les propriétés déjà vues par la *précision* qui valide un cas de test vis-à-vis d'un objectif de test. Elle exprime que le verdict *Pass* est fournit dès que possible lorsque l'objectif est vérifié. Ce qui peut être formalisé dans la définition suivante.

Définition 23 (Précision)**.** Un cas de test $\mathcal{TC}$ est *précis* vis-à-vis d'une spécification $\mathcal{M}$ et d'un objectif de test $\mathcal{TP}$ si on a $\text{Traces}_{\mathcal{C}(Pass)}(\mathcal{TC}^\omega) = \text{Traces}_{\mathcal{C}(Accept)}(\mathcal{TP}) \cap \mathsf{STraces}(\mathcal{M}) \cap \text{Traces}(\mathcal{TC}^\omega)$.

Il est simple de démontrer qu'un cas de test engendré par le testeur canonique $Can\mathcal{G}$ et un objectif de test $\mathcal{TP}$ est précis. En effet, par construction, les états colorés par le verdict *Pass* sont ceux qui sont colorés par *Accept* dans $\mathcal{TP}$ et qui ne sont pas colorés par *Fail* dans $Can(\mathcal{G})$. Ainsi, on a $\text{Traces}_{\mathcal{C}(Pass)}(\mathcal{TC}^\omega) = \text{Traces}_{\mathcal{C}(Accept)}(\mathcal{TP}) \cap \mathsf{STraces}(\mathcal{G}^\omega)$, ce qui implique la précision (car on a aussi $\text{Traces}_{\mathcal{C}(Pass)}(\mathcal{TC}^\omega) \subseteq \text{Traces}(\mathcal{TC}^\omega)$).

II.6 Génération de tests *en-ligne* pour les HR-grammaires

Nous avons déjà exprimé que comme tous les modèles caractérisant les langages algébriques, les graphes réguliers ne sont pas déterminisables (ou plus précisément leur déterminisation n'est pas un graphe régulier). Cependant, cette caractéristique n'interdit pas de les utiliser pour la génération formelle de tests. Les travaux de Tretmans [143] esquissent une technique de génération formelle de tests. Ce processus consiste à produire un cas de test sans construire de testeur canonique déterministe. Une telle technique que ce soit en-ligne ou hors-ligne est applicable aux systèmes engendrés par des HR-grammaires. Nous allons à présent introduire cette technique, et démontrer les propriétés satisfaites par les tests ainsi engendrés.

Construction des cas de test

Seule la déterminisation n'est pas certaine d'aboutir dans le processus de création de test hors-ligne. Ainsi les premières étapes sont exactement les mêmes. À partir de la spécification, on réalise la suspension, la complétion en sortie et la Σ^τ-clôture. Le processus va ainsi démarrer à partir d'une spécification complétée en sortie, et close vis-à-vis des actions internes : $Clo(CS(\mathcal{G}))$. À présent, on ne souhaite (ne peut) pas construire le testeur canonique. La HR-grammaire $Clo(CS(\mathcal{G}))$ ainsi construite satisfait les équations (4.6), (4.7), (4.8) et (4.9) :

$$\begin{aligned}
\text{Traces}_{\mathcal{C}(UnS)}((Clo(CS(\mathcal{G})))^\omega) &\subseteq \mathsf{STraces}(\mathcal{G}^\omega).\Sigma^{!\delta}_{\mathcal{G}} \quad (4.6)\\
\text{Traces}_{\overline{\mathcal{C}}(UnS)}((Clo(CS(\mathcal{G})))^\omega) &= \mathsf{STraces}(\mathcal{G}^\omega) \quad (4.7)\\
\text{Traces}((Clo(CS(\mathcal{G})))^\omega) &= \mathsf{STraces}(\mathcal{G}^\omega).\Sigma^{!\delta}_{\mathcal{G}} \cup \mathsf{STraces}(\mathcal{G}^\omega) \quad (4.8)
\end{aligned}$$

et

$$\mathsf{MinFTraces}(\mathcal{G}^\omega) = \text{Traces}_{\mathcal{C}(UnS)}((Clo(CS(\mathcal{G})))^\omega) \setminus \text{Traces}_{\overline{\mathcal{C}}(UnS)}((Clo(CS(\mathcal{G})))^\omega) \quad (4.9)$$

Produit et coloriage

L'étape suivante consiste simplement à réaliser le calcul du produit de $Clo(CS(\mathcal{G}))$ avec un objectif de test $\mathcal{TP}$ (comme précédemment, un IOLTS fini, déterministe et complet). Posons $\mathcal{P} = Clo(CS(\mathcal{G})) \times \mathcal{TP}$ pour ce produit, on peut définir les couleurs suivantes dans ce produit, en effectuant un calcul de co-accessibilité :

— $\mathcal{C}_\mathcal{P}(UnS) = \mathcal{C}_{Clo(CS(\mathcal{G}))}(UnS) \times Q_{\mathcal{TP}}$
— $\mathcal{C}_\mathcal{P}(Pass) = \overline{\mathcal{C}}_{Clo(CS(\mathcal{G}))}(UnS) \times \mathcal{C}_{\mathcal{TP}}(Accept)$
— $\mathcal{C}_\mathcal{P}(None) = Coreach(\Sigma^\delta_o, \mathcal{C}_\mathcal{P}(Pass)) \setminus \mathcal{C}_\mathcal{P}(Pass)$
— $\mathcal{C}_\mathcal{P}(Inc) = Q_\mathcal{P} \setminus (\mathcal{C}_\mathcal{P}(Fail) \cup \mathcal{C}_\mathcal{P}(Pass) \cup \mathcal{C}_\mathcal{P}(None))$

Calcul des cas de test

La dernière étape consiste à construire un cas de test sous la forme d'un IOLTS. Ce processus repose sur un parcours du graphe engendré par $Clo(CS(\mathcal{G}))$, en appliquant une technique similaire à celle de Tretmans [143]. Ces cas de test seront représentés par des arbres finis alternant des choix pour les entrées du systèmes, et des arbres pour les réponses possibles du système. Chacun des nœuds del'arbre est porteur d'un verdict.

Plus précisément, un tel arbre fini est représenté par un ensemble clos par préfixes de mots dans $(\Sigma_o^\delta)^*.(\{\mathit{Fail}, \mathit{Pass}, \mathit{None}, \mathit{Inc}\} \cup \{\varepsilon\})$. Étant donné un arbre θ, et un symbole a, l'arbre formé par a ayant pour fils l'arbre θ est noté $a;\theta$, et est définit par $a;\theta \stackrel{\text{def}}{=} \{au \mid u \in \theta\}$. De plus, étant données deux arbres θ, θ', l'arbre formé par l'union de ces deux arbres est noté $\theta + \theta'$.

Un cas de test $\mathcal{TC}$ est un arbre construit à partir de $\mathcal{P}$ en utilisant comme argument un ensemble d'états PS de $(cloCS(\mathcal{G}))^\omega$. Le cas de test est défini en appliquant récursivement l'algorithme suivant, en commençant avec l'état initial $\mathcal{C}_\mathcal{P}(\texttt{init})$.

Algorithme de construction de tests en-ligne.
Choisir de façon non-déterministe parmi une des actions suivantes :

1. (* Conclure le cas de test *)
   $\theta := \{\mathit{None}\}$
2. (* Fournir une nouvelle entrée pour l'implémentation *)
   Choisir une action $a \in out(PS)$ telle que
   $(PS \texttt{ after } a) \cap (\mathcal{C}_\mathcal{P}(\mathit{Pass}) \cup \mathcal{C}_\mathcal{P}(\mathit{None})) \neq \emptyset$
   $\theta := a;\theta'$
   où θ' est obtenu en appelant récursivement l'algorithme avec $PS' = (PS \texttt{ after } a)$
3. (* Vérifier la sortie suivante de l'implémentation *)
   $$\theta := \sum_{a \in X_1} a;\mathit{Fail} + \sum_{a \in X_2} a;\mathit{Inc} + \sum_{a \in X_3} a;\mathit{Pass} + \sum_{a \in X_4} a;\theta'$$
   avec :
   — $X_1 = \{a \mid PS \texttt{ after } a \subseteq \mathcal{C}_\mathcal{P}(\mathit{UnS})\}$
   — $X_2 = \{a \mid (PS \texttt{ after } a \subseteq (\mathcal{C}_\mathcal{P}(\mathit{Inc}) \cup \mathcal{C}_\mathcal{P}(\mathit{UnS}))) \wedge (PS \texttt{ after } a \cap \mathcal{C}_\mathcal{P}(\mathit{Inc}) \neq \emptyset)\}$
   — $X_3 = \{a \mid PS \texttt{ after } a \cap \mathcal{C}_\mathcal{P}(\mathit{Pass}) \neq \emptyset\}$
   — $X_4 = \{a \mid (PS \texttt{ after } a \cap \mathcal{C}_\mathcal{P}(\mathit{Pass}) = \emptyset) \wedge (PS \texttt{ after } a \cap \mathcal{C}_\mathcal{P}(\mathit{None}) \neq \emptyset)\}$
   — θ' est obtenu en appelant récursivement l'algorithme avec $PS' = (PS \texttt{ after } a)$

   Ensuite, un tel arbre est transformé en un IOLTS cas de test, $\mathcal{TC}$.

A chaque étape, l'algorithme fait un choix non-déterministe : poursuivre ou s'arrêter. On peut orienter ce choix suivant un ensemble de paramètres, par exemple en interdisant de produire des cas de test ne contenant ni le verdict *Fail* ni *Pass*.

Propriétés des cas de test engendrés *en-ligne*

L'un des intérêts majeurs à la génération formelle de cas de test à partir de spécification provient de la possibilité de démontrer certaines propriétés satisfaites par les suites de tests produites. Cet intérêt subsiste dans la cas des suites de tests engendrés en-ligne. Les arguments sont essentiellement les même que pour le cas hors-ligne, mais nous les résumons dans les lignes qui suivent.

Correction et sévérité Par définition de l'ensemble X_1, les traces de $\mathcal{TC}$ qui aboutissent à un état soloré par *Fail* sont dans $\mathrm{Traces}((Clo(CS(\mathcal{G})))^\omega) \setminus \mathrm{Traces}_{\overline{\mathcal{C}(\mathit{UnS})}}((Clo(CS(\mathcal{G})))^\omega) = \mathsf{MinFTraces}(\mathcal{G}^\omega)$. Ainsi $\mathrm{Traces}_{\mathcal{C}(\mathit{Fail})}(\mathcal{TC}) = \mathsf{MinFTraces}(\mathcal{G}^\omega) \cap \mathrm{Traces}(\mathcal{TC})$ ce qui démontre à la fois correction et sévérité dans le cas en-ligne.

Exhaustivité La preuve d'exhaustivité est similaire à celle donnée en § II.5, on construit un objectif de test $\mathcal{TP}$ pour chacune des exécutions non-conformes, et en montrant qu'un test produit aboutirait au verdict *Fail* sur cette exécution.

Précision D'après la construction de $\mathcal{TC}$, et en particulier, l'ensemble X_3, on a $\text{Traces}_{\mathcal{C}(Pass)}(\mathcal{TC}) = \text{Traces}_{\mathcal{C}(Pass)}(Clo(CS(\mathcal{G}))\times\mathcal{TP})\cap\text{Traces}(\mathcal{TC})$. Donc, de part la définition des couleurs : $\text{Traces}_{\mathcal{C}(Pass)}(\mathcal{TC}) = \text{Traces}_{\overline{\mathcal{C}}_{UnS}}(Clo(CS(\mathcal{G}))) \cap \text{Traces}_{\mathcal{C}(Accept)}(\mathcal{TP})) \cap \text{Traces}(\mathcal{TC})$. Ceci démontre, en définitive, la précision : $\text{Traces}_{\mathcal{C}(Pass)}(\mathcal{TC}) = \mathsf{STraces}(\mathcal{G}) \cap \text{Traces}_{\mathcal{C}(Accept)}(\mathcal{TP}) \cap \text{Traces}(\mathcal{TC})$.

Application de la génération de tests en-ligne

Dans les paragraphes précédents, nous avons vu comment engendrer des cas de test en-ligne, sans avoir à produire le testeur canonique (déterministe). En fait, en s'appuyant sur la méthode proposée en § II.4, il est possible de réaliser un tel calcul sans avoir à construire le graphe modélisant le système (ni même la partie visité par une exécution de longueur k). En utilisant l'algorithme décrit en § II.6, on peut calculer un ensemble de chemins symboliques compatibles : chacun de ces chemins étant mémorisé comme un simple couple composé d'un état, et d'un mots de non-terminaux.

Dans l'exemple qui suit, nous illustrons l'application de ces techniques sur le graphe régulier décrit dans l'exemple 13 (ce dernier est déterministe, mais sera satisfaisant pour les besoins de l'exemple).

Exemple 15. Considérons que la suite d'entrées déjà calculée soit formée de $8, true, true, true$. On peut vérifier que l'extrémité du chemin symbolique est le couple suivant : $(1, A_c A_c A_c)$. Ainsi, si l'implémentation fournit une sortie quelconque, ce test produira le verdict **fail**. Autrement, par exemple, le choix aléatoire de l'algorithme pourrait être de produire $false$, pour atteindre $(6, A_c A_c A_c A_c)$, alors, la sortie `"Vous avez arrêté"` (le message `m4`) est attendue, aboutissant à la configuration $(2, A_c A_c A_c A_c)$. Alors, le testeur attend simplement le message `"Un texte"` (le message `m3`), a chaque étape, on supprime un A_c. Ultimement, on atteint l'état $(5, \varepsilon)$ qui sera atteint après la réception de `"Fin,"` (message `m1`). Ceci marquerait la fin du test.

III Systèmes algébriques quantitatifs

III.1 Systèmes quantitatifs

Jusqu'à présent nous nous sommes intéressés aux systèmes à ensemble infini d'états. Ces structures possèdent un pouvoir de modélisation très important. En revanche il ne peuvent rendre compte d'éléments de nature quantitative.

Les chaînes de Markov ou les processus de décision markoviens sont des modèles qui intègrent des probabilités, et permettent donc de pouvoir quantifier, de mesurer l'importance de certains événements. Une part importante de la théorie de ces modèles est consacrée aux modèles finis. Au cours des dix dernières années un ensemble de travaux a examiné des extensions à des systèmes à états infinis. Ainsi les automates à piles probabilistes ou les systèmes de canaux avec perte probabilistes ont été considérés [59, 23, 8]. Ces chaînes de Markov infinies sont décrites par des représentations finies sur lesquelles il est possible de calculer des probabilités de façon effective, par exemple en utilisant la logique PCTL qui est une extension de CTL [74]. Ainsi, cette logique permet d'exprimer, entre autre choses, la probabilité de satisfaire une formule CTL de chemins. On peut voir PCTL comme une variante de CTL où les quantifications usuelles sont remplacées par la comparaison d'une mesure probabiliste à une valeur de seuil (une garde) : la formule est satisfaite si la probabilité de l'ensemble des exécutions qui la satisfont correspond à une garde donnée. Le fragment *qualitatif* de PCTL est une restriction à des gardes dont les valeurs possibles sont exclusivement 0 et 1. Le fragment global où toutes les valeurs sont considérés est

appelé PCTL *quantitative*. Le problème de *model-checking* pour les logiques probabilistes sur les chaînes de Markov infini a été étudié de façon approfondie. En particulier, lorsque ces chaînes sont définies par des automates à pile probabilistes, en particulier dans un ensemble de travaux menés par Kucera et d'autres co-auteurs [24, 59, 99, 23], les résultats les plus fondamentaux étant la décidabilité de la variante qualitative et l'indécidabilité de celle quantitative.

Dans ce paragraphe nous considérons une extension probabiliste des graphes réguliers (voir le chapitre 2). On va donc définir les grammaires de graphes déterministes, probabilisées, où les arcs terminaux sont étiquetés par des probabilités. Ces grammaires définissent donc des chaînes de Markov à états infinis, et constituent une généralisation naturelle des automates à pile probabilistes. On présentera une extension des résultats Etessamy *et al.* [59] relatifs au problème du *model-checking* de PCTL. Plus précisément, nous démontrons la décidabilité du fragment qualitatif du *model-checking* de PCTL pour les graphes réguliers probabilistes, nous présentons en détail une méthode de calcul de valeur approchée pour la probabilité d'une formule de chemins ; enfin, nous présentons une réduction établissant l'indécidabilité du *model-checking* exact des formules de PCTL quantitatif.

Chaînes de Markov et PCTL

Une *chaîne de Markov* (en temps discret) est un triplet $\mathcal{M} = (S, s_0, p)$ constitué d'un ensemble S d'états (potentiellement infini), un état initial s_0, et une fonction de transition probabilisée $p : S \times S \to [0, 1]$ telle que pour tout état s, on ait $\sum_{s' \in S} p(s, s') = 1$. Pour simplifier, on considère que le système de transition sous-jacent est de degré sortant fini. Étant donnée un ensemble de propositions atomiques AP, une *chaîne de Markov étiquetée* $\mathcal{M} = (S, s_0, p, \ell)$ est une chaîne de Markov (S, s_0, p) possédant une fonction d'étiquetage $\ell : S \to \mathsf{AP}$. Introduite en 1994, [74], PCTL est une extension de CTL avec des probabilités. Cette logique peut exprimer des propriétés quantitatives sur les exécutions des chaînes de Markov, par exemple, avec une propriété 0.9 les messages envoyés seront acquittés un moment ultérieur. On peut présenter la syntaxe de PCTL de la façon suivante :

$$\varphi ::= \mathtt{tt} \mid a \mid \neg\varphi \mid \varphi \wedge \psi \mid \mathbf{X}^{\sim\rho}\varphi \mid \varphi\, \mathbf{U}^{\sim\rho}\psi$$

avec $a \in \mathsf{AP}$ une proposition atomique, $\rho \in [0, 1]$ et $\sim \in \{\leq, <, >, \geq\}$. Les opérateurs $\mathbf{X}^{\sim\rho}$ et $\mathbf{U}^{\sim\rho}$ sont respectivement les généralisations probabilistes des opérateurs *suivant* et *jusqu'à*. On rappels les raccourcis classiques pour CTL : ultimement ($\mathbf{F}$) et universellement ($\mathbf{G}$) : $\mathbf{F}\,\varphi \equiv \mathtt{tt}\,\mathbf{U}\,\varphi$ et $\mathbf{G}\,\varphi \equiv \neg\mathbf{F}\,\neg\varphi$. Leurs extensions probabilistes $\mathbf{F}^{\sim\rho}$ et $\mathbf{G}^{\sim\rho}$ seront également utilisés dans la suite.

Soit $\mathcal{M} = (S, s_0, p, \ell)$ une chaîne de Markov étiquetée, et $s \in S$ un état. Pour toute formule (non-probabiliste) ϕ de CTL, on note $\mathbb{P}(s \models \phi)$ la mesure de l'ensemble des chemins dans $\mathcal{M}$ issus de s et qui satisfont ϕ. On peut observer que pour deux ensembles d'états V_1 et V_2, l'ensemble des chemins satisfaisant $\mathbf{X}\,V_1$ ou $V_1\,\mathbf{U}\,V_2$ est clairement mesurable. On définit par récurrence la sémantique d'une formule de PCTL φ surr $\mathcal{M}$:

$$\begin{array}{ll}
\mathtt{tt}^\omega = S & \varphi \wedge \psi^\omega = \varphi^\omega \cap \psi^\omega \\
a^\omega = \{s \in S \mid a \in \ell(s)\} & \mathbf{X}^{\sim\rho}\varphi^\omega = \{s \in S \mid \mathbb{P}(s \models \mathbf{X}\,\varphi^\omega) \sim \rho\} \\
\neg\varphi^\omega = S \setminus \varphi^\omega & \varphi\,\mathbf{U}^{\sim\rho}\psi^\omega = \{s \in S \mid \mathbb{P}(s \models \varphi^\omega\,\mathbf{U}\,\psi^\omega) \sim \rho\} \\
\mathbf{F}^{\sim\rho}\varphi^\omega = \{s \in S \mid \mathbb{P}(s \models \mathbf{F}\,\varphi^\omega) \sim \rho\} & \mathbf{G}^{\sim\rho}\varphi^\omega = \{s \in S \mid \mathbb{P}(s \models \mathbf{G}\,\varphi^\omega) \sim \rho\}
\end{array}$$

On note $s \models \varphi$ lorsque on a $s \in \varphi^\omega$.

Dans la suite on va s'intéresser à interpréter les formules de PCTL sur des chaînes de Markov définies par des graphes réguliers probabilistes.

III.2 Graphe réguliers probabilistes

Pour illustrer certaines des constructions présentées dans ce paragraphe, nous nous appuierons sur l'exemple qui suit.

Exemple 16. La figure 4.10 propose un exemple de HR-grammaire. Formellement, cette grammaire est définie comme suit, $\mathcal{G} = (\{Z\}_0 \cup \{A\}_2, \{V_1, V_2\}_1 \cup \{a, d\}_2, \{(Z, H_Z), (A12, H_A)\}, \{Z\})$. Le symbole non-terminal Z (*resp.* A) est le seul d'arité 0 (*resp.* 2) ; l'ensemble $\{V_1, V_2\}$ (*resp.* $\{a, d\}$) contient les deux couleurs (*resp.* étiquettes d'arc) ; les hypergraphes H_Z, H_A représentent les membres droit des règles (représentés dans la figure). Pour des question de simplicité $\overline{V_1}$ représente l'absence de la couleur V_1. La figure 4.11 présente la première étape de récriture ainsi qu'une partie du graphe limite $\mathcal{G}^\omega$.

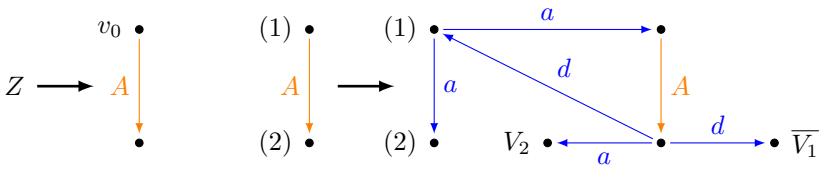


FIGURE 4.10 – Règles de la grammaire $\mathcal{G}$.

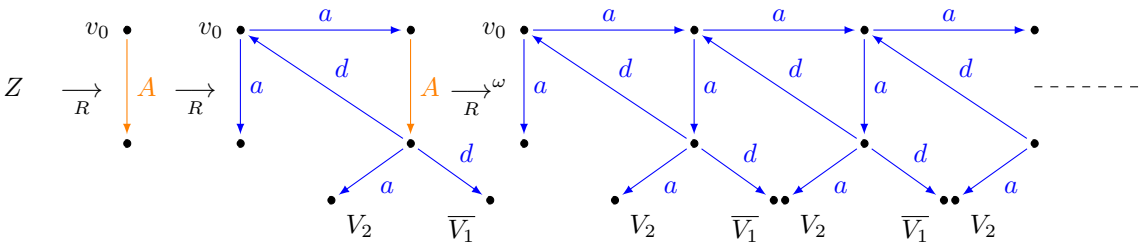


FIGURE 4.11 – Application successive de la récriture complète, itérée ainsi que le graphe engendré.

Pour définir une chaîne de Markov à partir d'un graphe régulier, on définit, pour toute HR-grammaire $\mathcal{G}$, et et chaque graphe H dans $\mathcal{G}^\omega$, la fonction de comptage $\# : V_H \times T_2 \to \mathbb{N}$, avec $\#(v, a) = |\{v' \mid v \xrightarrow{a} v'\}|$, qui associe à tout couple (v, a) le nombre de a-arcs ayant v pour origine. On peut observer que deux sommets distincts, v et v', de H ont la même valeur pour $\#$ dès que $\mathsf{Can}(v) = \mathsf{Can}(v')$.

Définition 24 (Grammaire d'hypergraphe probabilisée)**.** Une *grammaire d'hypergraphe probabilisée* (abrégée PHR-grammaire) $\mathcal{P}$, est un couple $(\mathcal{G}, \mu)$ où $\mathcal{G} = (N, T, R, H_0)$ est une HR-grammaire, $\mu : T_2 \to [0, 1]$ est une application, et pour tout sommet $v \in R^\omega$ la somme des μ-valeurs de tous les arcs issus de v vaut 1 : $\sum_{a \in T_2} \mu(a)\#(v, a) = 1$.

Remarque 10. Pour cette définition on peut faire les deux commentaires suivants :

1) Cette définition interdit les sommets de degré sortant infini. De fait, il est délicat de définir de façon judicieuse une notion de distribution pour un ensemble infini d'arcs issus d'un sommet. En revanche le degré entrant infini ne pose pas de problème.
2) Nous aurions pu définir les probabilités sur la chaîne de Markov engendrée en définissant des poids et une fonction de normalisation. Il nous a semblé plus simple d'avoir une valeur uniforme associée à un symbole, sachant que ces derniers ne servent qu'à définir la distribution et peuvent, donc, être redéfini pour toujours permettre d'avoir une PHR-grammaire.

Proposition 86. *Étant donnée une* HR*-grammaire* $\mathcal{G}$ *et une application* $\mu : T_2 \to [0, 1]$*, on peut décider si* $(\mathcal{G}, \mu)$ *est une* PHR*-grammaire.*

Démonstration. Le théorème 8 nous permet de supposer que $\mathcal{G}$ est complètement extérieure. Il est donc possible d'identifier les sommets de degré infini. S'il existe un tel sommet accessible, $(\mathcal{G}, \mu)$ ne peut être une PHR-grammaire pour toute valeur de $\mu(a)$. Lorsqu'il n'y a aucun sommet de ce type, on peut utiliser le résultat de Caucal [37, proposition 3.13 (b)] : il existe de façon effective un coloriage de $\mathcal{G} = (N, T, R, H_0)$ où les couleurs représentent le degré de chacun des sommet (pour chacune des étiquettes). Ainsi en utilisant ce coloriage, on peut déterminer le degré sortant de tout sommet et vérifier $\sum_{a \in T_2} \mu(a)\#(v, a) = 1$ pour chacun des sommets créé dans le graphe engendré par $\mathcal{G}$. □

Exemple 17. Si on considère le graphe présenté dans l'exemple 16. L'application de probabilité μ, définie par $\mu(a) = \frac{1}{2}$ et $\mu(d) = \frac{1}{4}$, permet d'obtenir une PHR-grammaire. On constate que la somme des probabilités des arcs sortants en tout sommet est égale à 1.

Dans leur travaux de 2006, Esparza *et al.* [59] définissent les automates à pile probabilistes, qui sont des automates à pile avec des poids sur les transitions. Les graphes des configurations de ces systèmes sont des chaînes de Markov étiquetés. Les PHR-grammaires permettent d'engendrer ces chaînes de la même façon que les HR-grammaires engendrent les graphes des configurations des automates à pile. En pratique, les travaux sur les automates à piles probabilistes font des suppositions assez fortes sur la syntaxe de leur modèle. En particulier, ils considèrent que pour tout ensemble rationnel de configurations peut se limiter à l'état de la configuration, ce qui devient complexe lorsqu'à chaque étape de la construction on déduit de nouveaux ensembles rationnels. Néanmoins cette restriction le limite pas la structure des chaînes de Markov engendrées.

III.3 Vérification pour les graphes réguliers probabilistes

Si à présent on considère à nouveau PCTL, l'exemple qui suit illustre quelque formules de cette logique sur le graphe régulier probabiliste définit dans l'exemple 17.

Exemple 18. Si on considère les couleurs V_1 et V_2 comme des prédicats satisfait par les sommets qui les portent, le graphe régulier probabiliste définit dans l'exemple 17 constitue une chaîne de Markov étiquetée sur laquelle on peut considérer les formules suivantes :

- $\varphi_1 = V_1 \wedge \mathbf{X}^{\geqslant \frac{1}{2}} V_2$: le sommets qui satisfont φ_1 appartiennent à V_1 et avec une probabilité supérieure à $\frac{1}{2}$, leur successeur après une transition sont dans V_2. En particulier, les sommets à une fourche, sur la rangée du bas, dans la figure 4.11 satisfont φ_1. Par construction ils sont les seuls.
- $\varphi_2 = v_0 \wedge V_1 \mathbf{U}^{> \frac{2}{3}} V_2$: le sommet v_0 satisfait φ_2 si la mesure de tous les chemins qui en sont issus et qui atteignent V_2 en traversant uniquement des sommets de V_1 est supérieure à $\frac{2}{3}$.

Degré infini

Comme il a été vu précédemment le degré sortant infini n'est pas compatible avec les PHR-grammaire. En revanche le degré entrant infini n'est pas incompatible avec notre définition. Cependant pour réaliser les calculs dans les paragraphes ultérieurs ce degré peut poser quelques difficultés techniques. De façon à aplanir ces difficultés, nous ferons la supposition que les graphes réguliers probabilistes que nous considérerons ne posséderont pas de sommet de degré infini. De surcroît, comme les HR-grammaires qui les engendrent sont en forme complètement extérieure, cela signifie que ces dernières ne possède pas de sommet qui soit à la fois une entrée et une sortie.

Cette supposition technique n'est pas une restriction dans le sens où il est possible de transformer toute PHR-grammaire possédant du degré infini en une grammaire équivalente qui en est dépourvue (tout en définissant les même probabilités). En effet, pour chacun des sommets de degré infini on peut identifier le graphe accessible (qui est régulier, probabiliste). Ensuite on peut remplacer chacun des retours sur

un de ces sommets par un arc vers un nouveau sommet (au même niveau que le sommet source) qui engendre le même graphe accessible.

À présent nous allons présenter des calculs de probabilité dans les graphes réguliers probabilistes. Nous aborderons ensuite le cas des problèmes de *model-checking* qualitatif et quantitatif.$R^2[A]$

Calcul des probabilités pour les graphes réguliers probabilistes

On s'intéresse ici au problème qui consiste à calculer, étant donné un sommet initial v_0 dans le membre droit associé à un axiome, la probabilité des chemins issus de v_0 qui satisfont une formule ϕ de CTL : $\mathbb{P}_{\mathcal{M}_\mathcal{P}}(v_0 \models \phi)$. Ce calcul peut être conduit par récurrence sur la structure de ϕ, et la difficulté étant de calculer $V_1 \,\mathbf{U}\, V_2$ (V_1 et V_2 étant deux couleurs du graphe). Noté : $\mathbb{P}(v_0 \models V_1 \,\mathbf{U}\, V_2)$.

Notations et préliminaires

Étant donnée une PHR-grammaire complètement extérieure $\mathcal{P} = (N, T, R, H_0, \mu)$, engendrant un graphe régulier probabiliste de degré fini. Nous allons nous appuyer sur une décomposition en niveaux de la chaîne de Markov $\mathcal{M}_\mathcal{P}$, nous présenterons comment exprimer $\mathbb{P}(v_0 \models V_1 \,\mathbf{U}\, V_2)$ en tant que solution d'un système d'équations obtenues à partir de l'ensemble des règles de $\mathcal{P}$.

Par hypothèse, la première transition d'un chemin issu d'un sommet de niveau n peut atteindre un sommet de niveau n, $n-1$ ou $n+1$. On peut donc prévoir une décomposition par niveau de la chaîne de Markov engendrée.

Pour réaliser ce calcul, on se repose sur la régularité de la chaîne de Markov. Ainsi, pour un sommet v de la chaîne $\mathcal{M}_\mathcal{P}$, avec $\mathsf{Can}(v) \in H_A$, on note $\mathcal{M}[v]$ la restriction de $\mathcal{M}_\mathcal{P}$ au graphe $R^\omega[A]$. On peut observer que pour deux sommets distincts v et v' de $\mathcal{M}_\mathcal{P}$ tels que $\mathsf{Can}(v) = \mathsf{Can}(v') \in H_A$, les deux graphes $\mathcal{M}[v]$ et $\mathcal{M}[v']$ sont isomorphes, ce qui garanti que, pour toute formule de CTL ϕ, on ait $\mathbb{P}_{\mathcal{M}[v]}(v \models \phi) = \mathbb{P}_{\mathcal{M}[v']}(v' \models \phi)$. En particulier, si ϕ est la formule $V_1 \,\mathbf{U}\, V_2$, on a : la probabilité de satisfaire $V_1 \,\mathbf{U}\, V_2$ sans passer par un niveau inférieur est la même à partir de v et de v'. La probabilité de satisfaire $(V_1 \setminus V_2)$ en passant par 1 niveau inférieur est également indépendante du niveau, dès lors que l'état initial est un un sommet fixé dans le membre droit de la règle associée à l'axiome. Ceci nous conduit à introduire des notations pour les probabilités qui sont soit paramétrées par le contexte soit indépendantes du contexte.

Soient $A, B \in N$ deux non-terminaux tels que $B \in \mathsf{Succ}(A)$. On note $R^2[A]$ le graphe définit à partir de H_A par deux récritures successives. On considère les sorties de H_A, tous les successeurs en un pas de ces sommets sont dans le graphe $R^2[A]$. Pour tout $i \leq \rho(B)$ et tout $j \leq \rho(A)$, on notera respectivement B_i et A_j les sorties associées à B et les entrées de H_A. De plus on définit les probabilités suivantes :

— $\mathbb{D}(B_i, A_j)$ probabilité depuis v (avec $\mathsf{Can}(v) = B_i$) d'atteindre v' avec $L(v') = L(v) - 1$ et $v' = h^{-1}(A_j)$ en satisfaisant $(V_1 \setminus V_2)$ tout au long du chemin ;

— $\mathbb{W}(B_i)_A$ représente la probabilité, depuis v (avec $\mathsf{Can}(v) = B_i$), de satisfaire $(V_1 \,\mathbf{U}\, V_2$. Sans passer par un sommet de niveau inférieur.

Comme nous l'avons déjà précisé, les valeurs de $\mathbb{D}(B_i, A_j)$ et $\mathbb{W}(B_i)_A$ ne dépendent pas de v et v' mais seulement de leurs images canoniques dans les règles qui les engendrent. De plus, les formules de la forme $\mathbb{D}(B_i, A_j)$ expriment la probabilité de descendre d'un niveau, alors que celles de la forme $\mathbb{W}(B_i)_A$ expriment la probabilté de gagner, *i.e.*, de satisfaire $V_1 \,\mathbf{U}\, V_2$ sans passer par un niveau inférieur. Ce qui justifie les notations.

La décomposition par niveau des chemins est attachée à des sommets qui, lorsqu'ils sont engendrés portent un non-terminal. Donc, étant donnés des non-terminaux $A, B, C, D \in N$ tels que $B, D \in \mathsf{Succ}(A)$ et $C \in \mathsf{Succ}(B)$, on introduit les notations qui suivent pour noter les probabilités dans une portion $R^2[A]$ de la chaîne de Markov.

— $p(B_i)_A$ représente la probabilité, dans $R^2[A]$ de satisfaire $V_1 \,\mathbf{U}\, V_2$ directement depuis v (avec $\mathsf{Can}(v) = B_i$) sans visiter aucun sommet v' tel que $\mathsf{Can}(v') \in \{C_k, B_\ell\} (\ell \neq i)$.

— $p(B_i, D_h)_A$ est la probabilité, dans $R^2[A]$, d'atteindre D_j depuis B_i en satisfaisant $(V_1 \setminus V_2)$ sans passer par aucun sommet v tel que $\mathsf{Can}(v) \in \{B_\ell, D_{h'}\}$ $(\ell \neq i, h' \neq h)$.
— $\overleftarrow{p}(B_i, A_j)$ est la probabilité, dans $R^2[A]$, d'atteindre A_j depuis B_i en satisfaisant $(V_1 \setminus V_2)$ sans passer par aucun sommet v tel que $\mathsf{Can}(v) \in \{C_k, B_\ell, D_h\}$ $(\ell \neq i)$.
— $\overrightarrow{p}(B_i, C_k)_A$ est la probabilité, dans $R^2[A]$, d'atteindre C_k depuis B_i en satisfaisant $(V_1 \setminus V_2)$ sans passer par aucun sommet v tel que $\mathsf{Can}(v) \in \{A_j, C_{k'}, B_\ell\}$ $(\ell \neq i, h' \neq h)$.

Intuitivement, il y a de nombreux chemins qui partent de v (avec $\mathsf{Can}(v) = B_i$, sous contexte A) et pour lesquelles la formule $V_1 \,\mathbf{U}\, V_2$ n'est pas contredite : soit ils satisfont $V_1 \,\mathbf{U}\, V_2$ sans visiter de sommets appartenant à un hyperarc étiqueté par un non-terminal, ou bien il satisfont la formule $\mathbf{G}\,(V_1 \wedge \neg V_2)$ et aboutissent à un sommet v' qui appartient à un hyperarc non-terminal Les formules ci-dessus réalise une partition entre ces différents chemins. Comme nous l'avons déjà exprimé précédemment, les valeurs de $p(B_i)_A$, $p(B_i, D_j)_A$, $\overleftarrow{p}(B_i, A_j)$, et $\overrightarrow{p}(B_i, C_k)_A$ peuvent être calculés directement dans $R^2[A]$, en utilisant les graphes H_A, H_B, et H_E pour tous les non-terminaux $E \in Succ(A) \cup Succ(B)$.

Exemple 19. On peut calculer les probabilités pour la chaîne de Markov définie dans l'exemple 16 : $p(A_2)_A = a$, $p(A_1, A_2)_A = a$, $\overleftarrow{p}(A_2, A_1) = d$ et $\overrightarrow{p}(A_1, A_2)_A = 0$.

Calcul de la valeur de $\mathbb{P}(v_0 \models V_1 \,\mathbf{U}\, V_2)$

Proposition 87. *Les valeurs $\mathbb{D}(B_i, A_j)$ et $\mathbb{W}(B_i)_A$ satisfont les équations suivantes :*

$$\begin{aligned}
\mathbb{D}(B_i, A_j) = \overleftarrow{p}(B_i, A_j) \;&+\; \sum_{D_h} p(B_i, D_h)_A \cdot \mathbb{D}(D_h, A_j) \\
&+\; \sum_{C_k} \overrightarrow{p}(B_i, C_k)_A \cdot \sum_{B_\ell} \mathbb{D}(C_k, B_\ell) \cdot \mathbb{D}(B_\ell, A_j) \qquad (4.13) \\
\mathbb{W}(B_i)_A = p(B_i)_A \;&+\; \sum_{D_h} p(B_i, D_h)_A \cdot \mathbb{W}(D_h)_A \\
&+\; \sum_{C_k} \overrightarrow{p}(B_i, C_k)_A \Big(\mathbb{W}(C_k)_B + \sum_{B_j} \mathbb{D}(C_k, B_j) \cdot \mathbb{W}(B_j)_A \Big). \qquad (4.14)
\end{aligned}$$

De plus, si on ajoute les contraintes suivantes :
— *si $B_i \notin (V_1 \setminus V_2)$ alors $\mathbb{D}(B_i, A_j) = 0$ tous les A_j ;*
— *si $B_i \in V_2$ alors $\mathbb{W}(B_i)_A = 1$, et si $B_i \notin (V_1 \cup V_2)$ alors $\mathbb{W}(B_i)_A = 0$;*
Tous les $\mathbb{D}(B_i, A_j)$ et les $\mathbb{W}(B_i)_A$ sont la plus petite solution de ce système d'équations polynomiales.

Démonstration. La validité des équations 4.13 and 4.14 peut être démontrée en faisant une partition de l'ensemble des chemins issus de v avec $\mathsf{Can}(v) = B_i$ (dans le contexte A).

Plus précisément, pour l'équation 4.13, tous les chemins issus de v avec $\mathsf{Can}(v) = B_i$ vers $v' = h^{-1}(j)$ (et $L(v') = L(v) - 1$) satisfaisant $\mathbf{G}\,(V_1 \wedge \neg V_2)$ correspond exactement à l'un des cas suivants :
— il va directement de v à v' sans quitter le niveau de v ;
— il atteint un sommet v'' avec $\mathsf{Can}(v'') = (D_h)$ (sous contexte A) et $L(v'') = L(v)$, et va ensuite de v'' à v' ;
— il atteint un sommet v'' avec $\mathsf{Can}(v'') = (C_k)$ (sous contexte B) et $L(v'') = L(v) + 1$, puis retourne au niveau de v par le sommet $v^{(3)}$ avec $\mathsf{Can}(v^{(3)}) = (B, \ell)$ et ensuite atteint v'.

Ces différents cas sont représentés sur la figure 4.12 où les flèches continues représentent des chemins directes dans $R^2[A]$ les flèches en pointillé représente les probabilités récursives de descendre d'un niveau.

On procède par un raisonnement similaire pour l'équation 4.14. Tous les chemins issus de v qui satisfont $V_1 \,\mathbf{U}\, V_2$ sans passer par des sommets de niveau inférieur à $L(v)$ sont partitionnés selon les critères suivants

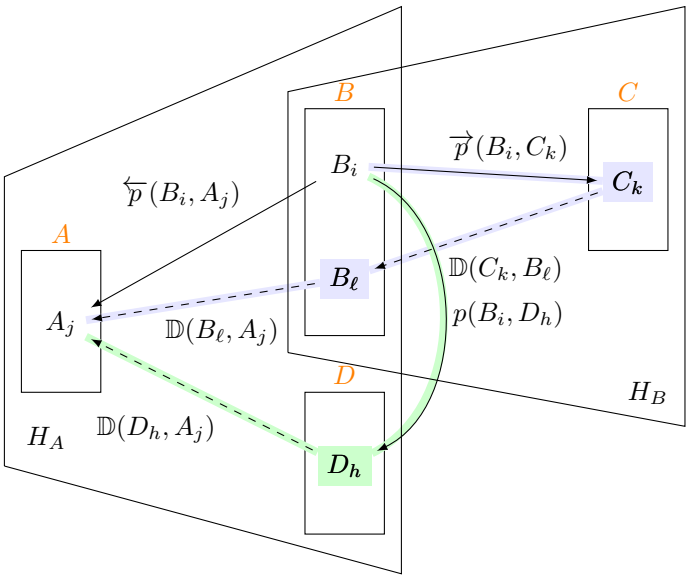


FIGURE 4.12 – Présentation symbolique de l'équation (4.13) pour $\mathbb{D}(B_i, A_j)$.

— il satisfont $V_1 \mathbf{U} V_2$ sans visiter aucun sommet appartenant à un non-terminal
— atteindre un sommet v' avec $\mathsf{Can}(v') = D_h$ (contexte A) et $L(v') = L(v)$ et ensuite, satisfaire $V_1 \mathbf{U} V_2$ sans descendre de niveau ;
— atteindre un sommet v' avec $\mathsf{Can}(v') = C_k$ (contexte B) et $L(v') = L(v) + 1$, ensuite, le chemin satisfait $V_1 \mathbf{U} V_2$ sans retour sur des sommets au niveau de v ou bien atteint un sommet v'' avec $\mathsf{Can}(v'') = B_\ell$ (sous contexte A) et $L(v'') = L(v)$ puis, à partir de v'' satisfait $V_1 \mathbf{U} V_2$ sans passer par un niveau inférieur.

Cette partition de l'ensemble des chemins issus d'un sommet v avec $\mathsf{Can}(v) = B_i$ démontre les équations 4.13 et 4.14.

Ce système d'équations définit un opérateur $\mathcal{F} : [0,1]^n \to [0,1]^n$ où n est le nombre de variables apparaissant dans le système. La valuation, $\mathcal{F}(\nu)$, des variables est obtenue en évaluant chacun des membres droits en substituant les valeurs qui y apparaissent par leur valeur dans ν. Comme cet opérateur est monotone et continu, il possède un unique point-fixe, qui est atteint en itérant $\mathcal{F}$ à partir de la valuation nulle (qui affecte 0 à chacune des variables). Notez cependant que la convergence vers ce point fixe pourrait requérir un nombre infini d'étapes.

Pour établir que $\mathbb{D}(B_i, A_j)$ et $\mathbb{W}(B_i)$ forment la *plus petite* solution du système, on considère les probabilités approchées en tronquant les chemins à la longueur k. Précisément, on définit $\mathbb{D}(B_i, A_j)^k$ la portion de probabilité de $\mathbb{D}(B_i, A_j)$ restreint aux chemins de longueur au plus k ; de façon similaire, soit $\mathbb{W}(B_i)^k_A$ la portion de probabilité de $Win(B_i)_A$ restreint aux chemins de longueur au plus k. Lorsqu'on fait tendre l'entier k vers l'infini, ces valeurs tendent respectivement vers $\mathbb{D}(B_i, A_j)$ et $\mathbb{W}(B_i)_A$. Il suffit donc d'établir que pour toute valeur de $k \in \mathbb{N}$, $\mathbb{D}(B_i, A_j)^k$ et $\mathbb{W}(B_j)^k_A$ ne sont jamais plus grand que la plus petite solution du système. Ceci peut être établi par une simple récurrence sur k. □

Notre but réel est de calculer la valeur de $\mathbb{P}(v_0 \models V_1 \mathbf{U} V_2)$. Cette valeur peut s'exprimer en utilisant celles de $\mathbb{D}(B_i, A_j)$ et $\mathbb{W}(B_i)_A$:

$$\mathbb{P}(v_0 \models V_1 \mathbf{U} V_2) = p(v_0)_{H_0} + \sum_{A_i \in \mathsf{Succ}(H_0)} \overrightarrow{p}(v_0, A_i)_{H_0} \, \mathbb{W}(A_i)_{H_0}, \tag{4.15}$$

où

— $p(v_0)_{H_0}$ est la probabilité, dans l'axiome H_0 from v_0 de satisfaire $V_1 \,\mathbf{U}\, V_2$ sans visiter aucunes des sortie ;
— $\overrightarrow{p}(v_0, A_i)_{H_0}$ est la probabilité, dans H_0, depuis v_0 d'atteindre un sommet v' avec $\mathsf{Can}(v') = A_i$ en satisfaisant $\mathbf{G}\,(V_1 \wedge V_2)$ et sans visiter de sommet v'' tel que $\mathsf{Can}(v'') = B_j$ (pour $B \in \mathsf{Succ}(H_0)$).

Exemple 20. Nous illustrons cette méthode pour le calcul de $\mathbb{P}(v_0 \models V_1 \,\mathbf{U}\, V_2)$ dans la chaîne de Markov définie dans l'exemple 17. Comme $\mathsf{Can}(v_0) = A_1$ (dans le membre droit de l'axiome) et $v_0 \notin V_2$, $p(v_0)_{H_0} = 0$ et $\overrightarrow{p}(v_0, A_1) = 1$. D'après l'équation 4.15 on déduit $\mathbb{P}(v_0 \models V_1 \,\mathbf{U}\, V_2) = \mathbb{W}(A_1)_{H_0}$. À présent on peut détailler les calculs :

$$\begin{aligned}\mathbb{W}(A_1)_{H_0} &= a\mathbb{W}(A_2)_{H_0} + a(\mathbb{W}(A_1)_A + \mathbb{D}(A_1, A_1)\mathbb{W}(A_1)_{H_0} + \mathbb{D}(A_1, A_2)\mathbb{W}(A_2)_{H_0}) \\ &= a\mathbb{W}(A_1)_A + a\mathbb{D}(A_1, A_1)\mathbb{W}(A_1)_Z,\end{aligned}$$

puisque $\mathbb{W}(A_2)_Z = 0$. La valeur de $\mathbb{W}(A_2)_A$ peut être facilement calculée : $\mathbb{W}(A_2)_A = a$. Ainsi $\mathbb{D}(A_1, A_1)$ est la plus petite solution de l'équation quadratique suivante :

$$a\mathbb{D}(A_1, A_1)^2 - \mathbb{D}(A_1, A_1) + ad = 0.$$

Si on pose $\mu(a) = \frac{1}{2}$ et $\mu(d) = \frac{1}{4}$, on obtient $\mathbb{D}(A_1, A_1) = 1 - \frac{\sqrt{3}}{2}$. Et enfin :

$$\begin{aligned}\mathbb{W}(A_1)_{H_0} &= \frac{a\mathbb{W}(A_1)_A}{1 - a\mathbb{D}(A_1, A_1)} \\ &= \frac{a^3}{(1 - a - a\mathbb{D}(A_1, A_1))(1 - a\mathbb{D}(A_1, A_1))} \\ &= \frac{2}{3}(2\sqrt{3} - 3) \approx 0.31.\end{aligned}$$

Il est important de souligner que le calcul exact des solutions du système n'est pas toujours possible. En général, les équation sont polynomiales, de degré arbitraire. En revanche, comme pour les travaux de Esparza *et al.* [59], des valeurs approchées peuvent être calculées, ce qui est résumé dans le théorème qui suit.

Théorème 88. *Soit $\mathcal{P} = (N, T, R, H_0, \mu)$ une* PHR*-grammaire, et v_0 un sommet de H_0. Pour tout rationnel $\rho \in \mathbb{Q} \cap [0, 1]$ et $\sim \in \{\leqslant, <, \geqslant, >\}$, on peut décider si $\mathbb{P}(v_0 \models V_1 \,\mathbf{U}\, V_2) \sim \rho$. De plus pour tout réel $0 < \lambda < 1$, on peut calculer deux rationnels $\rho_1, \rho_2 \in \mathbb{Q}$ tels que $\rho_1 \leq \mathbb{P}(v_0 \models V_1 \,\mathbf{U}\, V_2) \leq \rho_2$, et $\rho_2 - \rho_1 \leqslant \lambda$.*

Démonstration. D'après l'équation 4.15, décider si on a $\mathbb{P}(v_0 \models V_1 \,\mathbf{U}\, V_2) \sim \rho$ revient à décider si on a $p(v_0)_{H_0} + \sum_{A_i \in \mathsf{Succ}(H_0)} \overrightarrow{p}(v_0, A_i)_{H_0}\mathbb{W}(A_i)_{H_0} \sim \rho$. En utilisant les équations 4.13 et 4.14, la décidabilité de l'arithmétique du premier ordre des réels (due à Tarski [140]) entraîne celle de notre problème. Ensuite, en appliquant, suivant un processus dichotomique, l'algorithme de décision, on peut obtenir les approximations ρ_1 et ρ_2. □

Model-checking qualitatif pour les graphes réguliers probabilistes

Le fragment qualitatif de PCTL ne met en jeu que des valeurs de 0 et 1 pour les gardes des formules. Soit le PHR-grammaire $\mathcal{P} = (N, T, R, H_0, \mu)$. À isomorphisme près, $\mathcal{P}$ engendre une unique chaîne de Markov notée $\mathcal{M}_\mathcal{P}$ (ou $\mathcal{M}$ lorsqu'il n'y a aucune ambiguïté sur $\mathcal{P}$). On peut formuler le problème de *model-checking* qualitatif pour les PHR-grammaires noté $\mathrm{MC}^{\mathrm{PHRG}}_{qual}(PCTL)$:

Donnée : une PHR-grammaire $\mathcal{P} = (N, T, R, H_0, \mu)$ et un énoncé qualitatif $\varphi \in \mathrm{FO}$

Question : $\mathcal{M}_{\mathcal{P}}, v_0 \models \varphi$?

Pour résoudre ce problème on peut s'appuyer sur le résultat suivant :

Théorème 89. *Pour toute formule qualitative de PCTL φ, et $\mathcal{P}$ une* PHR*-grammaire. On peut construire de façon effective un coloriage, $\mathcal{P}'$, dans lequel les éléments de l'ensemble $\{v \in V_G \mid v \models \varphi\}$ sont identifiés par une nouvelle couleur.*

Démonstration. La démonstration repose sur une récurrence sur la structure de φ, en établissant que les ensembles suivants peuvent, de façon effective, être colorés dans la HR-grammaire : $\{v \in V_G \mid \mathbb{P}(v, \mathbf{X}\, V) = 1\}$, $\{v \in V_G \mid \mathbb{P}(v, \mathbf{X}\, V) = 0\}$, $\{v \in V_G \mid \mathbb{P}(v, V_1 \,\mathbf{U}\, V_2) = 1\}$ et $\{v \in V_G \mid \mathbb{P}(v, V_1 \,\mathbf{U}\, V_2) = 0\}$. Examinons dans un premier temps les deux premiers cas : $\{v \in V_G \mid \mathbb{P}(v, \mathbf{X}\, V) = 1\}$ et $\{v \in V_G \mid \mathbb{P}(v, \mathbf{X}\, V) = 0\}$. La fonction Can induit une partition parmi les sommets de la chaîne de Markov engendrée par $\mathcal{P}$. Deux sommets ont la même image par Can possèdent des ensembles de successeurs équivalent. Comme la grammaire est complètement extérieure, chacun des successeurs, de degré fini, d'un sommet engendré au niveau n sont engendrés au niveaux $n-1$ et $n+1$. Ainsi, si un sommet v est engendré par le non-terminal A, il suffit de vérifier, dans $R^2[A]$, si tous les successeurs de v appartiennent à V ou $\overline{V}$. Pour les sommets de degré infini ils sont les seuls à être à la fois des entrées et des sorties, on peut donc « propager » les couleurs qu'ils portent. Ainsi dans chacun des membres droits des règles on peut utiliser deux nouvelles couleurs pour annoter les sommets si tous leur successeur porte la couleur V ou aucun. Ces couleurs correspondent exactement aux ensembles $\{v \in V_G \mid \mathbb{P}(v, \mathbf{X}\, V) = 1\}$ et $\{v \in V_G \mid \mathbb{P}(v, \mathbf{X}\, V) = 0\}$.

Les deux autres cas $\{v \in V_G \mid \mathbb{P}(v, V_1 \,\mathbf{U}\, V_2) = 1\}$ et $\{v \in V_G \mid \mathbb{P}(v, V_1 \,\mathbf{U}\, V_2) = 0\}$, peuvent être traités de façon identique. On va détailler le coloriage de $\{v \in V_G \mid \mathbb{P}(v, V_1 \,\mathbf{U}\, V_2) = 1\}$. Pour tous les non-terminaux $B \in \mathsf{Succ}(A)$ et $i \leq \rho(B)$, on pose $R((B,i)_A) = \{A_j | \mathbb{D}(B_i, A_j) > 0\}$. Ensuite, les ensembles suivants sont définis par récurrence :

— $W_0 = (H_0 \cap V_2) \cup \{v \mid \mathsf{Can}(v) = (B,i)_A \text{ et } \mathbb{W}(B_i)_A = 1\}$, et
— $W_{n+1} = W_n \cup \{v \mid \mathsf{Can}(v) = (B,i)_A \text{ et } \mathbb{W}(B_i)_A + \sum_{A_j \in R((B,i)_A)} \mathbb{D}(B_i, A_j) = 1 \text{ et } R((B,i)_A) \subseteq W_n\}$.

Les sommets de W_0 sont directement *gagnants*, soit parce qu'il sont directement dans V_2 soit car B_i, dans le contexte A, possède une probabilité 1 de gagner sans descendre de niveau. Les sommets qui sont dans W_{n+1} sont presque sûrement gagnant (en d'autres termes, satisfont $V_1 \,\mathbf{U}\, V_2$ avec probabilité 1). Comme ces sommets atteignent l'objectif sans passer par un niveau inférieur (le facteur $\mathbb{W}(B_i)_A$) ou bien commencent par descendre de niveau et ensuite gagnent depuis A_j avec probabilité 1 (puisque on a $A_j \in W_n$).

On obtient que $\bigcup_{n=0}^{\infty} W_n = \{v \in V_G \mid \mathbb{P}(v, V_1 \,\mathbf{U}\, V_2) = 1\}$ et les W_n peuvent être calculés récursivement et annotés dans la grammaire à l'aide de couleurs. □

Indécidabilité du *model-checking* quantitatif pour les graphes réguliers probabilistes

On peut formuler le problème de *model-checking* quantitatif pour les PHR-grammaires noté $\mathrm{MC}^{\mathrm{PHRG}}_{quan}(PCTL)$:

Donnée : une PHR-grammaire $\mathcal{P} = (N, T, R, H_0, \mu)$ et un énoncé quantitatif de PCTL φ

Question : $\mathcal{M}_{\mathcal{P}}, v_0 \models \varphi$?

En premier lieu, on peut souligner que l'indécidabilité de $\mathrm{MC}^{\mathrm{PHRG}}_{quan}(PCTL)$ est une conséquence directe de son analogue pour les automates à pile probabilistes démontré par Brázdil *et al.* [24]. Néanmoins, nous pensons intéressant d'illustrer cette construction en s'appuyant sur une PHR-grammaire. On présente donc ici une réduction directe du PCP.

Théorème 90 ([24])**.** *Le problème $\mathrm{MC}^{\mathrm{PHRG}}_{quan}(PCTL)$ est indécidable.*

Démonstration. Soit $((u_i, v_i))_{i\leq n}$ une suite finie de couples de mots sur $\Sigma = \{0, 1\}$. On définit, à partir de cette instance de PCP la PHR-grammaire suivante : $\mathcal{P} = (N, T, R, H_0, \mu)$, avec :

— $N = \{\mathbf{u\text{–}v}_i \mid i \leq n\}_2$;
— $T = \{s, vert, rouge\}_1 \cup \{a, b\}_2$;
— $\mu(a) = 0.5, \mu(b) = 1$;

et l'ensemble des règles de récritures $R = (B, H_B)_{B\in N}$ représentés sur la figure 4.13

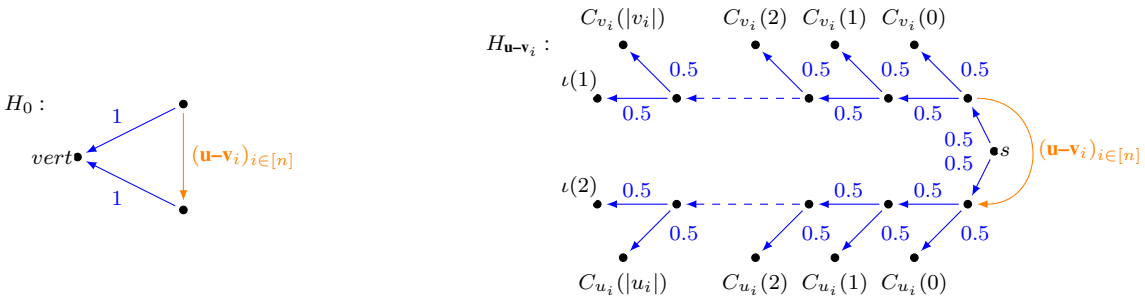


FIGURE 4.13 – HR-grammaire pour l'indécidabilité de $\mathrm{MC}^{\mathrm{PHRG}}_{quan}(PCTL)$

Les couleurs $vert$, et $rouge$ étiquette les sommets de la façon suivante. Pour tous entiers $i \leq n$, $k \leq |u_i|$, et $k' \leq |v_i|$,

$$C_{u_i}(k) = \begin{cases} vert & \text{if } u_i(k) = 1 \\ rouge & \text{if } u_i(k) = 0 \end{cases}, \quad C_{v_i}(k') = \begin{cases} vert & \text{if } v_i(k') = 0 \\ rouge & \text{if } v_i(k') = 1 \end{cases}$$

On considère la formule de PCTL suivante :

$$\varphi_0 = S \wedge (\mathtt{tt}\, \mathbf{U}^{=\frac{1}{2}} Vert)$$

avec $Vert, Rouge$ et S des propositions atomiques qui sont vérifiées respectivement par les sommets étiquetés par $vert, rouge$ et s, les terminaux d'arité 1. La formule φ_0 est valide sur la chaîne de Markov $\mathcal{M}_{\mathcal{P}}$ si et seulement si il y a une solution à l'instance de PCP formée par les couples $((u_i, v_i))_{i\leq n}$.

Dans la graphe engendré par $\mathcal{P}$, chaque sommet étiqueté par s est relié à l'origine par deux chemins. La branche supérieure, une suite de u_i et la branche inférieur une suite de v_i. Les deux suites correspondent à la même suite d'indices. Soit $I = (i_0, i_1, \cdots, i_m)$ une suite d'indices pris dans $\{1, \cdots, n\}$, et posons v_I pour représenter le s-sommet qui correspond à cette suite. La probabilité d'atteindre $rouge$ à partir de v_I est la suivante : $\mathbb{P}(v_I \models \mathtt{tt}\, \mathbf{U}\, Rouge) = \frac{1}{2}(u_{\mathsf{chem}} + v_{\mathsf{chem}})$ avec :

$$u_{\mathsf{chem}} = \sum_{j\leq m} \sum_{k\leq |u_j|} \frac{1}{2^{(\sum_{\ell<j}|u_{I_\ell}|)+k}} (u_j(k) = 0) \quad \text{et} \quad v_{\mathsf{chem}} = \sum_{j\leq m} \sum_{k'\leq |v_j|} \frac{1}{2^{(\sum_{\ell<j}|v_{I_\ell}|)+k}} (v_j(k') = 1).$$

Pour avoir $\mathbb{P}(v_I \models \mathtt{tt}\, \mathbf{U}\, Rouge) = \frac{1}{2}$ (et donc $\mathbb{P}(v_I \models \mathtt{tt}\, \mathbf{U}\, Vert) = \frac{1}{2}$) il est indispensable de suivre la même suite de lettres sur les deux branches u_{chem} et v_{chem} (de part l'unicité du développement binaire). □

III.4 Conclusion sur les graphes réguliers probabilistes

Dans ce paragraphe, nous avons défini les graphes réguliers probabilistes, ce sont des chaînes de Markov engendrées par des HR-grammaires, où les étiquettes des arcs terminaux sont des probabilités. Les résultats antérieurs sur les automates à pile probabilistes se prolongent à cette famille. À la fois le calcul approché de PCTL ainsi que le *model-checking* qualitatif de formules de PCTL sont décidables. Le *model-checking* quantitatif est en revanche indécidable.

Nous pensons que ce modèle est un bénéfice majeur par rapport aux automates à pile : il met l'accent sur les aspects structurels alors que les graphes des configurations des automates à pile mettent l'accent sur la syntaxe. En outre, nous pensons que pour identifier des classes où le *model-checking* quantitafif serait décidable l'approche structurelle pourrait se révéler utile. Une autre direction serait d'aller plus haut dans la hiérarchie de Caucal (voir les travaux de Carayol et Wöhrle [31]) et poursuivre ces travaux sur les graphes des configurations des automates à pile d'ordre supérieur.

Conclusion

Après une synthèse des questions abordées dans ce document, nous présentons quelques perspectives et problèmes ouverts.

I Synthèse

Dans ce document, nous nous sommes attachés à présenter plusieurs familles de graphes de présentation finie :

— Les graphes réguliers, définis par grammaires déterministes de graphes.
— Les graphes rationnels, définis par des transducteurs étiquetés.
— Les graphes d'accessibilité des réseaux de Petri.

Nous avons examiné un ensemble de résultats d'expressivité ou de logique pour ces familles. On peut souligner les résultats suivants :

— Décidabilité de la théorie du premier ordre des arbres rationnels.
— Indécidabilité du *model-checking* de la logique du premier ordre pour les graphes d'accessibilité des réseaux de Petri.
— L'équivalence entre les graphe rationnels et les graphes engendrés par les CHR-arbre-grammaires-séparées.

La logique du premier ordre tiens un rôle central dans cet ensemble de résultats, et nous avons cherché (en particulier pour les graphe d'accessibilité des réseaux de Petri) à examiner des restrictions pertinentes (logiques modales) ou bien des extensions (FO avec accessibilité), pour dresser une cartographie aussi précise que possible de la décidabilité pour ces modèles.

En outre, nous avons examiné plusieurs applications portant sur des systèmes sous observation partielle. D'une façon générale, ce type de contrainte induit une grande famille de problèmes d'intérêt pratique. Nous avons ici considéré trois grands problèmes :

— le diagnostic (diagnosticabilité),
— l'opacité,
— le test de conformité.

Nous avons examiné ces problèmes pour les systèmes modélisés par des graphes réguliers. Globalement les deux premiers problèmes sont indécidables pour cette famille, mais nous nous sommes appuyés sur des restrictions décidables pour assurer la décidabilité de ces problèmes (pour ces restrictions). Pour ce qui est du test de conformité, le cadre classique de la génération hors-ligne de tests, en s'appuyant sur un testeur canonique, fonctionne également pour le même type de restriction. Nous avons finalement commenté un ensemble de techniques de test en-ligne permettant la génération formelle de tests pour les systèmes dont la déterminisation n'est pas possible.

II Perspectives

Nous allons discuter de suites possibles aux travaux décrits dans ce document. Nous avons décidé de grouper ces perspectives suivant quatre thèmes :

(1) le diagnostic et l'opacité,

(2) les graphes réguliers et les automates à pile,

(3) la logique du premier ordre pour les graphes rationnels,

(4) les aspects quantitatifs.

Diagnostic et opacité. La notion d'opacité a donné lieu à un grand nombre d'extensions, telles que la k-opacité ou l'opacité infinie. La première consiste à déterminer si le système s'est trouvé dans un état secret au cours des k dernières observations, alors que l'opacité infini consiste à déterminer si le système a été *à un moment quelconque du passé* dans un état secret. Les travaux de Saboori *et al.* [130] ont introduit ces notions et les ont résolu pour les systèmes finis. Leur extension aux graphes réguliers pourrait être intéressante, d'autant plus qu'il y a plusieurs façons formelles de définir la notion de k-opacité.

Dans cette veine, une autre question intéressante concerne le problème du diagnostic d'opacité. Il s'agit de considérer que le système n'est pas observé de la même façon par le moniteur et par l'attaquant. Ensuite, le diagnostic d'opacité consiste à être capable de déterminer avec certitude qu'il y a eu une fuite d'information (comme dans le cas du diagnostic classique). Ce problème est assez délicat pour les graphes réguliers.

Graphes réguliers et automates à pile. Nous avons étudié les graphes réguliers engendrés par des grammaires pondérées pour résoudre un ensemble de problèmes. Une ligne intéressante de recherche aujourd'hui porte sur XML, et le lien avec les automates à pile visible. Un ensemble de travaux conduits par Filiot et Gauwin avec leurs co-auteurs [61, 65] s'intéresse au traitement de flots de données structurées. Nous pensons que considérer des extensions s'appuyant sur les HR-grammaires pondérées pourrait amener de nouveaux résultats. Ces grammaires permettent, en effet, une formulation *structurelles* de propriétés syntaxiques mises en valeur par les nombreux travaux sur les automates à pile visible.

Logique du premier ordre et graphes rationnels. Nous avons présenté ici la décidabilité de la théorie du premier ordre des graphes rationnels qui sont des arbres. Si globalement la logique du premier ordre est indécidable pour les graphes rationnels il serait utile de construire un catalogue précis des propriétés du premier ordre qui sont indécidables. On peut observer, par exemple, que la formule $\exists x,\ x \to x$ est indécidable, mais que la formule $\forall x, y, z,\ (x \to y \wedge x \to z) \implies y = z$ (qui est la fonctionnalité) est décidable.

Éléments quantitatifs. Nous l'avons déjà vu, les éléments quantitatifs prennent une place de plus en plus importante en vérification. Nous proposons d'appliquer les travaux sur les graphes réguliers probabilistes pour étendre les travaux de Thorsley et Teneketzis, [142], qui définissent plusieurs notions quantitatives de diagnostic, et les étudient dans le cas fini. Les travaux de Bérard *et al.* [12] utilisent une modélisation probabiliste finie pour quantifier l'opacité d'un système. Étudier cette notion pour les systèmes récursifs permettrait d'étendre la portée de ces résultats.

On a dit plus haut que la notion de pondération était structurelle. Néanmoins, tout graphe régulier de degré sortant fini est isomorphe à un système pondéré (en faisant abstraction des étiquettes). Donc cette notion n'est pas pertinente pour définir une restriction qui aurait, par exemple, une logique PCTL quantitative décidable. En revanche, il semble que l'explicitation de la structure offerte par les HR-grammaires est prometteuse pour identifier de telles restrictions. Par exemple en forçant une linéarisation des équations définissant les probabilités à la façon des grammaires de chemin de Caucal *et al.* [39].

Bibliographie

[1] R. Alur, L. Fix, and T. A. Henzinger. Event-clock automata : a determinizable class of timed automata. *Theoretical Computer Science*, 211(1–2) :253 – 273, 1999.

[2] R. Alur and P. Madhusudan. Visibly pushdown languages. In *STOC 04*, pages 202–211. ACM, 2004.

[3] T. Araki and T. Kasami. Some decision problems related to the reachability problem for Petri nets. *Theoretical Computer Science*, 3 :85–104, 1976.

[4] T. Araki and T. Kasami. Decidability problems on the strong connectivity of Petri net reachability sets. *Theoretical Computer Science*, 4 :99–119, 1977.

[5] M. F. Atig and P. Habermehl. On Yen's path logic for Petri nets. *International Journal of Foundations of Computer Science*, 22(4) :783–799, 2011.

[6] J.-M. Autebert. *Théorie des langages et des automates*. MASSON, 1994.

[7] E. Badouel, M. Bednarczyk, A. Borzyszkowski, B. Caillaud, and P. Darondeau. Concurrent secrets. *Discrete Event Dynamic Systems*, 17 :425–446, 2007.

[8] C. Baier, N. Bertrand, and Ph. Schnoebelen. Verifying nondeterministic probabilistic channel systems against ω-regular linear-time properties. *ACM Trans. Comput. Log.*, 9(1), 2007.

[9] C. Baier and J.-P. Katoen. *Principles of model checking*. MIT Press, Cambridge, (Mass.), London, 2008.

[10] H. G. Baker. Rabin's proof of the undecidability of the reachability set inclusion problem of vector addition systems, 1973. M.I.T., Project MAC, CSGM 1979.

[11] P. Baldan, Th. Chatain, S. Haar, and B. König. Unfolding-based diagnosis of systems with an evolving topology. *Information and Computation*, 208(10) :1169–1192, October 2010.

[12] B. Bérard, J. Mullins, and M. Sassolas. Quantifying opacity. In *QEST*, pages 263–272. IEEE Computer Society, 2010.

[13] J. Berstel. *Transductions and context-free languages*. Teubner, 1979.

[14] N. Bertrand, T. Jéron, A. Stainer, and M. Krichen. Off-line test selection with test purposes for non-deterministic timed automata. *Logical Methods in Computer Science*, 8(4), 2012.

[15] N. Bertrand and C. Morvan. Probabilistic regular graphs. In *12th International Workshop on Verification of Infinite-State Systems, INFINITY'10*, volume 39 of *EPTCS*, pages 77–90, Singapore, August 2010.

[16] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. CUP, 2001.

[17] A. Blumensath and E. Grädel. Automatic Structures. In *Proceedings of 15th IEEE Symposium on Logic in Computer Science LICS 2000*, pages 51–62, 2000.

[18] B. Boigelot. *Symbolic methods for exploring infinite state spaces*. PhD thesis, Université de Liège, 1998.

[19] A. Boudet and H. Comon. Diophantine equations, presburger arithmetic and finite automata. In *CAAP*, volume 1059 of *Lecture Notes in Computer Science*, pages 30–43. Springer, 1996.

[20] P. Bouyer, F. Chevalier, and D. D'Souza. Fault diagnosis using timed automata. In *FoSSaCS'05)*, volume 3441 of *LNCS*, pages 219–233, Edinburgh, U.K., April 2005.

[21] Z. Bouziane and A. Finkel. Cyclic Petri net reachability sets are semi-linear effectively constructible. In *INFINITY'97*, volume 9 of *ENTCS*, pages 15–24. Elsevier, 1997.

[22] M. Bozga, R. Iosif, and F. Konecný. Fast acceleration of ultimately periodic relations. In *CAV'10*, volume 6174 of *Lecture Notes in Computer Science*, pages 227–242. Springer, 2010.

[23] T. Brázdil, V. Brozek, J. Holecek, and A. Kučera. Discounted properties of probabilistic pushdown automata. In *Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'08)*, volume 5330 of *Lecture Notes in Computer Science*, pages 230–242. Springer, 2008.

[24] T. Brázdil, A. Kučera, and O. Strazovský. On the decidability of temporal properties of probabilistic pushdown automata. In *Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS'05)*, volume 3404 of *Lecture Notes in Computer Science*, pages 145–157. Springer, 2005.

[25] E. Brinskma, A. Alderen, R. Langerak, J. van de Laagemat, and J. Tretmans. A formal approach to conformance testing. In *Protocol Specification, Testing and Verification (PSTV'90)*, pages 349–363, 1990.

[26] J. Bryans, M. Koutny, L. Mazaré, and P. Y. A. Ryan. Opacity generalised to transition systems. *Int. J. Inf. Sec.*, 7(6) :421–435, 2008.

[27] J. R. Büchi. On a decision method in restricted second order arithmetic. In *ICLMPS*, pages 1–11. Stanford University press, 1960.

[28] A. Carayol. Regular sets of higher-order pushdown stacks. In J. Jedrzejowicz and A. Szepietowski, editors, *MFCS*, volume 3618 of *Lecture Notes in Computer Science*, pages 168–179, 2005.

[29] A. Carayol. *Automates infinis, logiques et langages*. Thèse de doctorat, Université de Rennes 1, 2006.

[30] A. Carayol and C. Morvan. On rational trees. In Zoltán Ésik, editor, CSL 06, volume 4207 of *LNCS*, pages 225–239, 2006.

[31] A. Carayol and S. Wöhrle. The caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In P. K. Pandya and J. Radhakrishnan, editors, FSTTCS 03, volume 2914 of *LNCS*, pages 112–123, 2003.

[32] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer, 1999.

[33] F. Cassez. The Dark Side of Timed Opacity. In *Proc. of the 3rd International Conference on Information Security and Assurance (ISA'09)*, volume 5576 of *LNCS*, pages 21–30, Seoul, Korea, June 2009.

[34] F. Cassez, J. Dubreil, and H. Marchand. Synthesis of opaque systems with static and dynamic masks. *Formal Methods in System Design*, 40(1) :88–115, 2012.

[35] D. Caucal. On infinite transition graphs having a decidable monadic theory. In F. Meyer auf der Heide and B. Monien, editors, *ICALP*, volume 1099 of *Lecture Notes in Computer Science*, pages 194–205. Springer, 1996.

[36] D. Caucal. On infinite terms having a decidable monadic theory. In MFCS 02, volume 2420 of *LNCS*, pages 165–176, 2002.

[37] D. Caucal. Deterministic graph grammars. In J. Flum, E. Grädel, and T. Wilke, editors, *Logic and Automata*, volume 2 of *Texts in Logic and Games*, pages 169–250. Amsterdam University Press, 2008.

[38] D. Caucal. Synchronization of regular automata. In *34th International Symposium on Mathematical Foundations of Computer Science (MFCS'09)*, volume 5734 of *LNCS*, pages 2–23, 2009.

[39] D. Caucal and S. Hassen. Synchronization of grammars. In E. Hirsch, A. Razborov, A. Semenov, and A. Slissenko, editors, *CSR*, volume 5010 of *LNCS*, pages 110–121. Springer, 2008.

[40] D. Caucal and T. Knapik. An internal presentation of regular graphs by prefix-recognizable ones. *Theory of Computing Systems*, 34(4), 2001.

[41] N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3) :113–124, 1956.

[42] N. Chomsky. On certain formal properties of grammars. *Information and Control*, 2(2) :137–167, June 1959.

[43] S. Chédor, T Jéron, and C. Morvan. Test generation from recursive tiles systems. In *6th International Conference on Tests & Proofs*, volume 7305 of *LNCS*, pages 99–114, Prague, Czech Republic, June 2012.

[44] S. Chédor, T Jéron, and C. Morvan. Test generation from recursive tile systems. *Software Testing, Verification and Reliability*, 2014. Version étendue de [43].

[45] S. Chédor, C. Morvan, S. Pinchinat, and H. Marchand. Analysis of partially observed recursive tile systems. In *11th edition of Workshop on Discrete Event Systems*, pages 265–271, Guadalajara, Mexico, September 2012.

[46] S. Chédor, C. Morvan, S. Pinchinat, and H. Marchand. Diagnosis and opacity problems for infinite state systems modeled by recursive tile systems. *Discrete Event Dynamic Systems*, pages 1–24, 2014. Version étendue de [45].

[47] H. Comon and Y. Jurski. Multiple counter automata, safety analysis and Presburger analysis. In *CAV'98*, volume 1427 of *Lecture Notes in Computer Science*, pages 268–279. Springer, 1998.

[48] C. Constant, B. Jeannet, and T. Jéron. Automatic test generation from interprocedural specifications. In *TestCom/FATES'07*, volume 4581 of *LNCS*, pages 41–57, 2007.

[49] B. Courcelle. *Handbook of Theoretical Computer Science*, chapter Graph rewriting : an algebraic and logic approach. Elsevier, 1990.

[50] W. Damm. Languages defined by higher type program schemes. In Arto Salomaa and Magnus Steinby, editors, *ICALP 77*, volume 52 of *LNCS*, pages 164–179, 1977.

[51] Ph. Darondeau, S. Demri, R. Meyer, and C. Morvan. Petri Net Reachability Graphs : Decidability Status of FO Properties. In S. Chakraborty and A. Kumar, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2011)*, volume 13 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 140–151, 2011.

[52] Ph. Darondeau, S. Demri, R. Meyer, and C. Morvan. Petri net reachability graphs : Decidability status of first order properties. *Logical Methods in Computer Science*, 8(4) :1–28, 2012.

[53] J. Dubreil, T. Jéron, and H. Marchand. Monitoring confidentiality by diagnosis techniques. In *ECC*, pages 2584–2590, Budapest, Hungary, August 2009.

[54] H. D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, 1995.

[55] H. D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer-Verlag, 1994.

[56] S. Eilenberg. *Automata, Languages and Machines*, volume A. Academic Press, 1974.

[57] J. Esparza. Petri nets, commutative context-free grammars, and basic parallel processes. *Fundamenta Informaticae*, 31(13) :13–26, 1997.

[58] J. Esparza. Decidability and complexity of Petri net problems — an introduction. In *Advances in Petri Nets 1998*, volume 1491 of *Lecture Notes in Computer Science*, pages 374–428. Springer, 1998.

[59] J. Esparza, A. Kučera, and R. Mayr. Model checking probabilistic pushdown automata. *Logical Methods in Computer Science*, 2(1), 2006.

[60] R.J. Evey. The theory and application of pushdown store machines. Mathematical linguistic and automatic translation report NSF-10, The computation laboratory of Harvard University, 1963.

[61] E. Filiot, O. Gauwin, P.-A. Reynier, and F. Servais. Streamability of Nested Word Transductions. In Supratik Chakraborty and Amit Kumar, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2011)*, volume 13 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 312–324, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[62] A. Finkel and J. Leroux. How to compose Presburger accelerations : Applications to broadcast protocols. In *FST&TCS'02*, volume 2256 of *Lecture Notes in Computer Science*, pages 145–156. Springer, 2002.

[63] L. Frantzen, J. Tretmans, and T.A.C. Willemse. A symbolic framework for model-based testing. In *FATES 2006 and RV 2006, Revised Selected Papers*, volume 4262 of *LNCS*, pages 40–54, 2006.

[64] D. Gabbay. Expressive functional completeness in tense logic. In *Aspects of Philosophical Logic*, pages 91–117. Reidel, 1981.

[65] O. Gauwin, J. Niehren, and S. Tison. Queries on xml streams with bounded delay and concurrency. *Information and Computation*, 209(3) :409 – 442, 2011. Special Issue : 3rd International Conference on Language and Automata Theory and Applications (LATA 2009).

[66] S. Ginsburg and S.A. Greibach. Mappings which preserve context sensitive languages. *Information and Control*, 9 :563–582, 1966.

[67] S. Ginsburg and G.F. Rose. Preservation of languages by transducers. *Information and Control*, 9 :153–176, 1966.

[68] S. Ginsburg and E.H. Spanier. Bounded ALGOL-like languages. *Transactions od the American Mathematical Society*, pages 333–368, 1964.

[69] S. Ginsburg and E.H. Spanier. Semigroups, Presburger formulas, and languages. *Pacific Journal of Mathematics*, 16 :285–296, 1966.

[70] J. Grabowski. The decidability of persistence for vector addition systems. *Information Processing Letters*, 11 :20–23, 1980.

[71] P. Habermehl. On the complexity of the linear-time mu-calculus for Petri nets. In *ICATPN'97*, volume 1248 of *Lecture Notes in Computer Science*, pages 102–116. Springer, 1997.

[72] M. Hack. *Decidability Questions for Petri nets*. PhD thesis, MIT, 1975.

[73] M. Hack. The equality problem for vector addition systems is undecidable. *Theoretical Computer Science*, 2 :77–96, 1976.

[74] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5) :512–535, 1994.

[75] D. Hauschildt. Semilinearity of the reachability set is decidable for Petri nets. Technical Report FBI-HH-B-146/90, University of Hamburg, 1990.

[76] K. Havelund and G. Rosu. Monitoring java programs with java pathexplorer. *Electr. Notes Theor. Comput. Sci.*, 55(2) :200–217, 2001.

[77] K. Havelund and G. Rosu. Efficient monitoring of safety properties. *STTT*, 6(2) :158–173, 2004.

[78] L. Hélouet, T. Gazagnaire, and B. Genest. Diagnosis from scenarios. In *proc. of the 8th Int. Workshop on Discrete Events Systems, WODES'06*, pages 307–312, 2006.

[79] M. Hennessy and R. Milner. On observing nondeterminism and concurrency. In *ICALP'80*, volume 85 of *Lecture Notes in Computer Science*, pages 299–309. Springer, 1980.

[80] J. Hopcroft and J.J. Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8 :135–159, 1979.

[81] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation, Second Edition*. Addison-Wesley, 2001.

[82] R. Howell, P. Jančar, and L. Rosier. Completeness results for single-path Petri nets. *Information & Computation*, 106(2) :253–265, 1993.

[83] L. Hélouët, H. Marchand, B. Genest, and T. Gazagnaire. Diagnosis from scenarios, and applications. *Discrete Event Dynamic Systems : Theory and Applications*, March 2013.

[84] O. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM*, 25(1) :116–133, 1978.

[85] N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on computing*, 17(5) :935–938, October 1988.

[86] P. Jančar. Undecidability of bisimilarity for Petri nets and some related problems. *Theoretical Computer Science*, 148 :281–301, 1995.

[87] C. Jard and T. Jéron. TGV : theory, principles and algorithms. *Software Tools for Technology Transfer (STTT)*, 7(4) :297–315, 2005.

[88] B. Jeannet, T. Jéron, and V. Rusu. Model-based test selection for infinite-state reactive systems. In *5th International Symposium on Formal Methods for Components and Objects (FMCO'06), Revised Lectures*, volume 4709 of *LNCS*, pages 47–69, 2006.

[89] T. Jéron, H. Marchand, S. Pinchinat, and M-O. Cordier. Supervision patterns in discrete event systems diagnosis. In *WODES'06*, pages 262–268, July 2006.

[90] Shengbing Jiang, Zhongdong Huang, Vigyan Chandra, and Ratnesh Kumar. A polynomial algorithm for testing diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 46 :1318–1321, 2000.

[91] Richard M. Karp and Raymond E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3 :147–195, 1969.

[92] S.C. Kleene. Representation of events in nerve nets and finite automata. In C.E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–40, 1956. Princeton.

[93] Koichi Kobayashi and Kunihiko Hiraishi. Verification of opacity and diagnosability for pushdown systems. *Journal of Applied Mathematics*, 2013.

[94] N. Kobayashi and C.-H. L. Ong. Complexity of model checking recursion schemes for fragments of the modal mu-calculus. *Logical Methods in Computer Science*, 7(4), 2011.

[95] S. R. Kosaraju. Decidability of reachability in vector addition systems (preliminary version). In H. R. Lewis, B. B. Simons, W. A. Burkhard, and L. H. Landweber, editors, *STOC*, pages 267–281. ACM, 1982.

[96] M. Krichen and S. Tripakis. Conformance testing for real-time systems. *Formal Methods in System Design*, 34(3) :238–304, 2009.

[97] S.-Y. Kuroda. Classes of languages and linear-bounded automata. *Information and Control*, 7(2) :207–223, June 1964.

[98] D. Kuske and M. Lohrey. Automatic structures of bounded degree revisited. In Erich Grädel and Reinhard Kahle, editors, *CSL*, volume 5771 of *Lecture Notes in Computer Science*, pages 364–378. Springer, 2009.

[99] A. Kučera. Methods for quantitative analysis of probabilistic pushdown automata. *Electronic Notes in Theoretical Computer Science*, 149(1) :3–15, 2006.

[100] P. S. Landweber. Three theorems on phrase structure grammars of type 1. *Information and Control*, 6(2) :131–136, June 1963.

[101] K.G. Larsen, M. Mikucionis, and B. Nielsen. Online testing of real-time systems using Uppaal. In *Formal Approaches to Software Testing (FATES'04)*, volume 3395 of *LNCS*, pages 79–94, 2005.

[102] J. Leroux. Vector Addition System Reachability Problem (A Short Self-Contained Proof). In *POPL'11*, pages 307–316, 2011.

[103] J. Leroux. Vector addition system reversible reachability problem. In *CONCUR'11*, volume 6901 of *Lecture Notes in Computer Science*, pages 327–341. Springer, 2011.

[104] J. Leroux and G. Sutre. On Flatness for 2-Dimensional Vector Addition Systems with States. In *CONCUR'04*, volume 3170 of *Lecture Notes in Computer Science*, pages 402–416. Springer, 2004.

[105] M. Lohrey. Automatic structures of bounded degree. In *Proceedings of LPAR 03*, volume 2850 of *LNAI*, pages 344–358, 2003.

[106] A. Maslov. Multilevel stack automata. *Problems of Information Transmission*, 12 :38–43, 1976.

[107] A. Mateescu and A. Salomaa. *Handbook of Formal Languages*, volume 1, chapter Aspects of classical language theory, pages 175–252. Springer-Verlag, 1997.

[108] E. Mayr. An algorithm for the general Petri net reachability problem. In *Proceedings of the 13th Symposium on Theory of Computing, Milwaukee, Wisconsin, USA*, pages 238–246. ACM, 1981.

[109] E. Mayr. An algorithm for the general Petri net reachability problem. *SIAM Journal of Computing*, 13(3) :441–460, 1984.

[110] E.W. Mayr and A.R. Meyer. The complexity of the finite containment problem for Petri nets. *Journal of the ACM*, 28(3) :561–576, 1981.

[111] R. Milner. *Communication and concurency*. Prentice Hall, 1989.

[112] A. Montanari and G. Puppis. Decidability of mso theories of tree structures. In K. Lodaya and M. Mahajan, editors, *FSTTCS*, volume 3328 of *Lecture Notes in Computer Science*, pages 434–446. Springer, 2004.

[113] C. Morvan. On rational graphs. In J. Tiuryn, editor, Fossacs 00, volume 1784 of *LNCS*, pages 252–266, 2000.

[114] C. Morvan. *Les graphes rationnels*. Thèse de doctorat, Université de Rennes 1, Novembre 2001.

[115] C. Morvan. Contextual graph grammars characterizing context-sensitive languages. In *4th International Workshop on Non-Classical Models of Automata*, volume 263 of *OCB*, pages 141–153, Jena, Germany, 2010.

[116] C. Morvan and S. Pinchinat. Diagnosability of pushdown systems. In *HVC2009, Haifa Verification Conference*, volume 6405 of *LNCS*, pages 21–33, Haifa, Israel, October 2009.

[117] C. Morvan and C. Rispal. Families of automata charaterizing context-sensitive languages. *Acta Informatica*, 41 :293–314, 2005.

[118] C. Morvan and C. Stirling. Rational graphs trace context-sensitive languages. In A. Pultr and J. Sgall, editors, MFCS 01, volume 2136 of *LNCS*, pages 548–559, 2001.

[119] D. Muller and P. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theoretical Computer Science*, 37 :51–75, 1985.

[120] D. Park. Concurrency and automata on infinite sequences. In 5th GIConference, volume 104 of *LNCS*, pages 167–183, 1981.

[121] E. Pelz. Closure properties of deterministic Petri nets. In *STACS'87*, volume 247 of *Lecture Notes in Computer Science*, pages 371–382. Springer, 1987.

[122] M. Penttonen. One-sided and two-sided context in formal grammars. *Information and Control*, 25(4) :371–392, 1974.

[123] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, University of Bonn, 1962.

[124] M. Praveen. Small vertex cover makes petri net coverability and boundedness easier. In V. Raman and S. Saurabh, editors, *IPEC*, volume 6478 of *Lecture Notes in Computer Science*, pages 216–227. Springer, 2010.

[125] M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Comptes Rendus du premier congrès de mathématiciens des Pays Slaves, Warszawa*, pages 92–101, 1929.

[126] M.O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. soc.*, 141 :1–35, 1969.

[127] M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal*, 3 :114–125, 1959.

[128] A. Rabinovich. Composition theorem for generalized sum. Personal communication, 2006.

[129] G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1 : Foundations*. World Scientific, 1997.

[130] A. Saboori and C. N. Hadjicostis. Verification of infinite-step opacity and complexity considerations. *IEEE Trans. Automat. Contr.*, 57(5) :1265–1269, 2012.

[131] A. Saboori and C.N. Hadjicostis. Verification of initial-state opacity in security applications of des. In *9th International Workshop on Discrete Event Systems, 2008. WODES 2008.*, pages 328–333, 2008.

[132] J. Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

[133] A. Salomaa. *Computation and Automata*. Cambridge University Press,, 1985.

[134] M. Sampath, R. Sengupta, S. Lafortune, K. Sinaamohideen, and D. Teneketzis. Diagnosability of discrete event systems. *IEEE Trans. on Automatic Control*, 40(9) :1555–1575, 1995.

[135] M. Sampath, R. Sengupta, S. Lafortune, K. Sinaamohideen, and D. Teneketzis. Failure diagnosis using discrete event models. *IEEE Transactions on Control Systems Technology*, 4(2) :105–124, March 1996.

[136] S. Schulz. First-order logic with reachability predicates on infinite systems. In *FST&TCS'10*, pages 493–504. LIPICS, 2010.

[137] A. Schürr. Programmed graph replacement systems. In Rozenberg [129], pages 479–546.

[138] S. Shelah. The monadic theory of order. *Ann. Math.*, 102 :379–419, 1975.

[139] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26(3) :279–284, November 1988.

[140] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, Berkeley, 1951.

[141] W. Thomas. A short introduction to infinite automata. In W. Kuich, G. Rozenberg, and A. Salomaa, editors, *DLT 01*, volume 2295 of *LNCS*, pages 130–144, 2002.

[142] D. Thorsley and D. Teneketzis. Diagnosability of stochastic discrete-event systems. *IEEE Trans. Automat. Contr.*, 50(4) :476–492, 2005.

[143] J. Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software - Concepts and Tools*, 17(3) :103–120, 1996.

[144] J. Tretmans and H. Brinksma. Torx : Automated model-based testing. In A. Hartman and K. Dussa-Ziegler, editors, *First European Conference on Model-Driven Software Engineering*, pages 31–43, December 2003.

[145] S. Tripakis. Fault diagnosis for timed automata. In W. Damm and E.-R. Olderog, editors, *FTRTFT*, volume 2469 of *LNCS*, pages 205–224. Springer, 2002.

[146] T. Ushio, I. Onishi, and K. Okuda. Fault detection based on petri net models with faulty behaviors. *IEEE Int. Conf. on Systems, Man, and Cybernetics.*, 1 :113–118 vol.1, Oct 1998.

[147] R. Valk and G. Vidal-Naquet. Petri nets and regular languages. *Journal of Computer and System Sciences*, 23 :299–325, 1981.

[148] P. Wolper. *Introduction à la Calculabilité*. InterEditions, 1991.

[149] Y.-C. Wu and S Lafortune. Comparative analysis of related notions of opacity in centralized and coordinated architectures. *Discrete Event Dynamical Systems*, 23 :307–339, 2013.

[150] T-S . Yoo and S. Lafortune. Polynomial-time verification of diagnosability of partially-observed discrete event systems. *IEEE Trans. on Automatic Control*, 47(3) :1491–1495, 2002.

[151] R. S. Zeitman. *The composition method*. Phd thesis, Wayne State University, Michigan, 1994.