N° d'ordre : 3316

THÈSE

présentée devant

l'Université de Rennes I

pour obtenir le grade de

Docteur de l'université de Rennes I
mention Informatique

par

Guillaume Feuillade

Équipe d'accueil : S4 – IRISA
École Doctorale : MATISSE
Composante unversitaire : IFSIC

Titre de la thèse :

Spécification logique de réseaux de Petri

Soutenue le 8 décembre 2005 devant la commission d'examen

Composition du jury

M.	Claude Jard	Président
M.	Ahmed Bouajjani	Rapporteur
M.	Jean-Michel Couvreur	Rapporteur
M^{me}	Sophie Pinchinat	Encadrant
M.	Serge Haddad	Examinateur
M.	Wieslaw Zielonka	Examinateur

Table des matières

Introduction

La conception d'un système informatique fiable est une tâche difficile. La réaliser impose la considération d'un grand nombre d'éventualités ; lorsque le système à concevoir doit impérativement être sûr (parce que son utilisation le requiert), et qu'il est de taille conséquente, la conception ou la validation du système par un humain n'est généralement pas envisageable. Cet état de fait justifie le développement des *méthodes formelles* dont l'objectif est de garantir le bon fonctionnement d'un système en s'appuyant sur des formalismes et des méthodes de preuve formelles. La fiabilité est assurée par une validation mathématique du système. Pour des systèmes de taille abordable cette validation peut s'effectuer via un opérateur humain, en utilisant par exemple un assistant de preuve ; en revanche, lorsque le système est complexe, il devient nécessaire de considérer des méthodes pour valider le système de manière *automatique*.
Par opposition aux systèmes dit *transformationnels*, qui produisent un résultat à partir de données initiales, nous considérons dans ce document des systèmes *réactifs*. Ces derniers sont des systèmes maintenant une interaction permanente avec les utilisateurs ou l'environnement ; ils sont généralement complexes et de taille conséquente.

La taille et la complexité d'un système réel ne permettent pas sa validation directe. Il est nécessaire de travailler sur une *abstraction* du système : il s'agit d'une représentation réduite du système adaptée à la manipulation par des méthodes formelles. L'abstraction d'un système doit se construire de telle manière que la validation de cette abstraction entraîne la validation du système lui-même. Les propriétés attendues d'un système réactif sont relatives à sa structure de contrôle (l'enchaînement de ses actions et leurs branchements dans le temps). L'abstraction d'un système réactif est généralement centrée sur cette structure de contrôle présentée sous la forme d'*événements* alors que les données internes du programme sont abstraites à travers des *configurations*.
Parmi les abstractions considérées dans la littérature, on distingue deux types principaux :

- les systèmes séquentiels : l'abstraction du système est un graphe dont les sommets sont les états représentant les configurations du système et les arcs sont les événements. Les systèmes séquentiels sont des modèles opérationnels faciles à manipuler et sont adaptés à la spécification de propriétés, puisque leur nature les rend aisément descriptibles [Mil90]. Leurs inconvénients sont que lorsqu'un système séquentiel est donné sous forme de composantes, la recombinaison de ces composante entraîne une explosion de la taille du système ; et lorsqu'un système séquentiel est sous une forme monolithique, il devient difficile de le décomposer en sous-systèmes afin d'étudier ces sous-

systèmes indépendamment.

– les systèmes concurrents : ils décrivent la structure de contrôle de manière plus générale puisque contrairement aux systèmes séquentiels, ils n'imposent pas strictement l'enchaînement des événements, on parle de *vrai parallélisme* [Die95]. Ce sont des systèmes où les états ne sont pas donnés explicitement. Les système concurrents sont plus compacts que les systèmes séquentiels et se décomposent naturellement en sous-systèmes reliés par un mécanisme de communication ; ils sont cependant moins intuitifs, et la spécification de leurs propriétés est difficile.

Afin de spécifier les propriétés souhaitées du système, il est nécessaire d'utiliser un formalisme non ambigu. Ce formalisme de description doit être adapté au type de propriétés à spécifier. Dans le cas des systèmes réactifs, la littérature met en avant un certain nombre de propriétés (sûreté, vivacité...). De nombreux formalismes de spécification ont été proposés, parmi ceux-ci les *logiques temporelles* sont largement admises par la communauté comme étant adaptées à la description de propriétés des systèmes réactifs [Eme90, MP92a]. L'utilisation de ces logiques peut se situer à deux niveaux de la conception d'un système :

– *a posteriori* : le système est donné et il est nécessaire de déterminer s'il vérifie un ensemble de propriétés requises (le système et la spécification constituent la donnée du problème). On utilise alors des techniques de *Model-Checking* pour déterminer si le système vérifie ces propriétés ou non [BBF+99, CGP99]. Le Model-Checking consiste à automatiser la vérification. Le Model-Checking des systèmes séquentiels est un domaine largement exploré, pour lequel le défi actuel repose essentiellement sur la recherche de méthodes efficaces pour traiter de gros systèmes ou l'extension de ces techniques à certains types de systèmes infinis.

– *a priori* : il s'agit de construire un système qui vérifie un ensemble de propriétés requises (la spécification constitue la donnée du problème). On parle de *synthèse* de système [EC82, AM94] ; le problème de décider de l'existence d'un système vérifiant les propriétés données dans une logique est nommé *problème de satisfiabilité* d'une formule logique. La problématique est ici celle de l'automatisation de la création d'un tel système ; c'est un sujet difficile pour lequel beaucoup de problèmes sont indécidables, en particulier dans le cas des modèles concurrents. L'existence d'une solution n'est pas toujours satisfaisante : les systèmes synthétisés sont souvent séquentiels et les méthodes pour les synthétiser ont une complexité élevée. Le défi actuel de la synthèse

de système repose sur le développement de techniques de synthèse efficaces produisant des systèmes exploitables.

Le cadre de cette thèse est le problème de synthèse de systèmes concurrents. En l'absence de spécifications adaptées à la concurrence, les spécifications que nous considérons dans ce document sont les logiques temporelles qui sont couramment utilisées pour décrire les propriétés de systèmes séquentiels.
Il existe une alternative qui consiste à utiliser des systèmes concurrents en tant que spécifications. Dans cette alternative le Model-Checking consiste à vérifier l'équivalence entre le système à étudier et le système servant de spécifiant ; or nous savons que la plupart des équivalences intéressantes sont indécidable sur des systèmes concurrents. Toujours dans cette alternative, le problème de synthèse devient quant-à lui trivial, le système donné étant la solution.
Les systèmes que nous considérons sont les *réseaux de Petri* [Pet62, Mur89] : ces systèmes sont admis comme étant les plus puissants pour représenter la concurrence [SNW96] ; de plus ils sont intrinsèquement distribuables. Les résultats concernant la génération automatique de réseaux de Petri encouragent le choix de cette classe de modèles : ces résultats permettent de construire des systèmes concurrents à partir de données séquentielles. Ce dernier point constitue un argument supplémentaire justifiant le choix des logiques temporelles.

Sans intention de proposer un nouveau formalisme de spécification, nous investiguons la synthèse de réseaux de Petri à partir de spécifications en logique temporelle, et plus généralement celui de la satisfiabilité d'une formule sur la classe des réseaux de Petri. Nous prenons comme point de départ la plus expressive des logiques du temps arborescent : le Mu-Calcul [Koz83, AN01]. Le fil conducteur est la recherche de la frontière de décidabilité pour le problème de satifiabilité d'une formule sur la classe des réseaux de Petri ; nous proposons des restrictions successives de la logique pour approcher cette frontière. Cette étude s'accompagne de celle de l'expressivité de la logique quand elle est considérée uniquement sur la classe de système concurrent que sont les réseaux de Petri.

Nous montrons dans ce document qu'il est nécessaire de restreindre très fortement la logique pour rendre la synthèse décidable, et que des logiques pourtant faibles permettent d'obtenir l'expressivité des machines à compteurs lorsque ces logiques sont interprétées sur la classe des réseaux de Petri. Nous nous attachons également à dissocier le rapport entre la logique et la structure d'un réseau, du rapport entre la logique et l'initialisation du réseau.

Chapitre 1

État de l'art, réseaux de Petri et synthèse

Sommaire

1.1 Contexte

1.1.1 Logique et systèmes séquentiels

Les logiques temporelles, qu'elles soient propositionnelles ou modales, sont connues pour être un formalisme adéquat pour la spécification de propriétés comportementales des systèmes réactifs [Pnu77, MP92a, BBF+99]. Parmi les logiques du temps arborescent, le Mu-Calcul constitue la logique la plus expressive, dans le sens ou tout énoncé dans une logique du temps arborescent peut être traduit en un énoncé du Mu-Calcul. Le Mu-Calcul est une logique reposant sur la composition de fonctions monotones et l'utilisation de points fixes introduite par Kozen [Koz83]. Interprété sur des systèmes réactifs le Mu-Calcul permet la spécification de tout type de propriétés de sûreté (*safety* en anglais) et de vivacité (*liveness* en anglais) [Eme90]. De plus, le Mu-Calcul est directement lié à la notion de bisimulation : deux systèmes bisimilaires ne peuvent être distingués par une formule du Mu-Calcul et pour toute paire de systèmes non-bisimilaires[1] il existe une formule qui les distingue ; la bisimulation constitue la notion adéquate pour capturer le comportement d'un système en considérant uniquement ses exécutions et ses branchements de choix [JW96].

Le Mu-Calcul possède son reconnaisseur : les automates alternants [AN01]. Les exécutions d'un système forment un arbre (infini dans le cas des systèmes réactifs) appelé *arbre des exécutions* ; une formule du Mu-Calcul ne différencie pas le système de son arbre d'exécutions (ils sont bisimilaires). Les automates alternants acceptent des arbres infinis suivant certains critères de gain. Le critère généralement associé au Mu-Calcul est appelé *critère de parité*. Pour chaque formule ϕ du Mu-Calcul, il est possible de construire un automate alternant avec critère de parité $\mathcal{A}_\phi$ qui reconnaît exactement les arbres d'exécution des modèles de ϕ [SE89, EJ91], la réciproque est également vraie [Niw88].

Les automates alternants avec le critère de parité, et par conséquent le Mu-Calcul, sont proches des jeux à deux joueurs avec information parfaite, en particulier des *jeux de parité* [EJ91]. Cette proximité confère au Mu-Calcul sa principale propriété : *la propriété du modèle fini* (*Small Model Property* en anglais). Elle provient directement de l'existence d'une *stratégie positionnelle* dans tout jeu de parité pour le joueur gagnant[2] [EJ91]. La propriété du modèle fini est l'existence, pour tout formule ϕ du Mu-Calcul qui possède au moins un modèle, d'un système

[1] Il est nécessaire que ces systèmes soient à branchement fini, ce qui est le cas des systèmes considérés dans ce document.

[2] Les jeux de parité sont déterminés, c.à-d qu'un des deux joueurs a toujours une stratégie gagnante [Dav64]

$\mathcal{P}$ fini modèle de ϕ ($\mathcal{P}$ possède un nombre fini d'états et de transitions). Le calcul d'une stratégie gagnante dans le jeu de parité associé à une formule ϕ du Mu-Calcul permet la synthèse effective d'un tel modèle fini de ϕ lorsqu'il existe.

La synthèse de système à partir du Mu-Calcul, du fait de la nature intrinsèquement séquentielle des automates alternants, produit un modèle fini lui aussi séquentiel. Les méthode actuelles pour obtenir un modèle distribué d'une formule ϕ reposent généralement sur la spécification *a priori* d'une architecture distribuée. Une telle architecture peut être par exemple un ensemble de *sites* avec des transition propres et un ensemble de transitions de synchronisation ; une solution est alors un ensemble de systèmes dotés chacun d'un alphabet partitionné en un alphabet propre et un ou des alphabets communs[3]. Le produit de tous ces systèmes, synchronisés sur les actions communes, constitue le modèle de la formule ϕ. La synthèse de modèles avec de telles contraintes d'architecture est généralement indécidable [PR90] ; on notera toutefois l'existence de cas particuliers pour lesquels la synthèse est décidable, comme par exemple les architectures en *pipe-line*.

Ne pas fixer l'architecture revient à considérer la synthèse de modèles *concurrents*. Ces modèles se "distribuent" en sous-modèles communicants selon un mécanisme résultant du fruit de la synthèse. Le principal obstacle à la synthèse de modèles concurrents est le manque de reconnaisseurs adaptés, ou plus généralement l'inadéquation des méthodes connues avec les systèmes concurrents. Bien entendu, la synthèse de modèles concurrents constitue une problématique difficile, pour laquelle on s'attend à établir que de nombreuses situations n'ont pas de solution.

1.1.2 Réseaux de Petri et synthèse

Les réseaux de Petri (non étiquetés) forment une classe puissante de modèles concurrents et distribuables. Le problème de synthèse de réseaux de Petri, tel que présenté originellement, consiste à trouver, étant donné un modèle (un langage ou un graphe), un réseau de Petri équivalent à ce modèle (la relation d'équivalence étant l'égalité de langages dans le cas où le modèle est un langage, ou l'isomorphisme de graphes dans le cas où le modèle est un graphe). La synthèse de réseaux constitue un problème important : synthétiser un réseau à partir d'un système séquentiel constitue une méthode pour "extraire" la concurrence du modèle séquentiel et apporte une solution algorithmique pour distribuer un tel modèle.

[3] Il peut y avoir autant d'alphabets partagés que de couples de sites.

Le problème de synthèse à été initialement étudié par Erhenfeucht et Rozenberg pour les graphes finis et les réseaux élémentaires [ER90a, ER90b]. Le nombre de réseaux élémentaires sur un ensemble d'actions données étant fini, ce problème possède une solution triviale. Les auteurs cités avaient pour but proposer une théorie axiomatique sur les graphes pour la synthèse de réseaux. Le résultat est l'invention du concept de *régions* d'un graphe : une région est un sous ensemble de sommets du graphe en bijection avec les marquages d'un réseau *atomique* constitué d'une unique place. Un arc étiqueté par l'action a *entre* ou *sort* d'une région suivant le signe du résultat de l'action a sur l'unique place du réseau atomique ; lorsque cette action est nulle (le marquage de la place n'est pas modifié par a), les sommets aux extrémités de l'arc sont tous deux à l'intérieur ou tous deux à l'extérieur de la région. Le problème de synthèse pour les graphes finis et les réseaux élémentaires a, par la suite, été démontré NP-complet [BBD97].

La concept de régions a été étendu pour permettre la synthèse de réseaux plus généraux. Le problème de synthèse pour des réseaux purs bornés à partir de graphes finis ou de langages rationnels est décidable [BBD95] (l'algorithme de synthèse est polynomial pour le graphe et exponentiel en la taille de l'expression régulière). L'ensemble des résultats concernant la synthèse de réseaux bornés est décrit dans [BD99]. Concernant les réseaux non bornés, la synthèse est décidable dans le cas des langages ou des graphes *context-free* réguliers ou déterministes et elle est indécidable pour les langages *context-free* et les *HMSC* [Dar04].

En montrant la décidabilité du problème de synthèse pour des graphes *automatique* ainsi que pour des *spécifications automatiques* [BD04] proposent la synthèse à partir de spécifications : le réseau synthétisé ne l'est plus à partir d'un unique modèle mais à partir d'une spécification (une famille de modèles). C'est dans cette optique que s'inscrit ce document, la spécification étant ici une formule logique.

1.2 Problématique de cette thèse

Lorsqu'on se donne une classe de spécifications $\mathbf{S}$ et une classe de modèles $\mathbf{C}$, on distingue trois problématiques : le model-checking, la satisfiabilité et le contrôle, qui s'énoncent comme suit :

Problème 1.2.1 (Model-checking : MC($\mathbf{S}$,$\mathbf{C}$)). Étant donnés une spécification $\Phi \in \mathbf{S}$, et un modèle $Z \in \mathbf{C}$, a-t-on Z satisfait Φ ?

Problème 1.2.2 (Satisfiabilité : Sat($\mathbf{S}$,$\mathbf{C}$)). Étant donnée une spécification $\Phi \in \mathbf{S}$, existe-t-il un modèle $Z \in \mathbf{C}$ telle que Z satisfasse Φ ?

le dernier problème requiert l'introduction d'une sous-classe $\mathbf{C}'$ de $\mathbf{C}$, il s'énonce comme suit :

Problème 1.2.3 (Contrôle : Cont($\mathbf{S}$,$\mathbf{C}$,$\mathbf{C}'$)). Étant donnés une spécification $\Phi \in \mathbf{S}$, un modèle $Z \in \mathbf{C}$, et une relation de composition de modèles '$\times$', existe-t-il un contrôleur $C \in \mathbf{C}'$ tel que $C \times Z$ satisfasse Φ ?

Les deux derniers problèmes sont proches. Le problème **Sat**($\mathbf{S}$,$\mathbf{C}$) est un problème de contrôle particulier (les classes $\mathbf{C}$ et $\mathbf{C}'$ sont identiques et Z est le modèle universel). Sous certaines conditions, **Cont**($\mathbf{S}$,$\mathbf{C}$,$\mathbf{C}'$) est équivalent à un problème de satisfiabilité ; en particulier, lorsque $\mathbf{S}$ est le Mu-calcul, le problème **Cont**($\mathbf{S}$,$\mathbf{C}$,$\mathbf{C}$) (sous observation totale) peut se ramener au problème **Sat**($\mathbf{S}$,$\mathbf{C}$) en utilisant le principe du quotient d'une formule par un modèle fini, présenté dans [AVW03].

L'objet de cette thèse est l'étude du problème **Sat**($\mathbf{S}$,$\mathbf{C}$) lorsque $\mathbf{S}$ est le Mu-Calcul et $\mathbf{C}$ est la classe des réseaux de Petri non étiquetés.

1.2.1 Mu-Calcul et réseaux de Petri

Lorsque la classe $\mathbf{S}$ de spécifications est le Mu-calcul ou un fragment de celui-ci, et que la classe $\mathbf{C}$ des modèles est une classe de réseaux de Petri, le problème **MC**($\mathbf{S}$,$\mathbf{C}$) à été largement étudié par J.Esparza [Esp98, Esp97] : pour les classes de réseaux possédant un langage régulier (réseaux bornés, réseaux 1-saufs ...), le problème se ramène à celui du model-checking de processus fini, qui est décidable [AN01] ; en revanche, pour les réseaux de Petri généraux, étiquetés ou non étiquetés, le model-checking est indécidable. Dans [Esp97], Esparza montre que ce problème est indécidable même pour des classes $\mathbf{S}$ de logiques du temps arborescent particulièrement faibles.
Toujours pour $\mathbf{C}$ une classe de réseaux de Petri, le problème **Sat**($\mathbf{S}$,$\mathbf{C}$) est peu abordé dans la littérature. Lorsque la classe $\mathbf{S}$ de modèles est la classe des réseaux de Petri étiquetés, le problème est décidable. En effet, trois propriétés permettent d'arriver à cette conclusion : 1) le Mu-calcul a la propriété du modèle fini ; 2) à partir d'un modèle fini, il est possible de construire un réseau de Petri étiqueté qui possède les même comportements ; 3) le problème de satisfiabilité d'une formule du Mu-calcul sur la classe des modèles séquentiels est décidable.

En revanche, il n'existe pas, à notre connaissance, de résultats concernant la synthèse de réseaux de Petri non étiquetés. Même dans les cas favorables où la synthèse d'un modèle séquentiel, lorsqu'il existe, est effective, son utilisation pour construire un réseau non étiqueté possédant les mêmes comportements ne fournit qu'une méthode incomplète : un modèle séquentiel pris au hasard n'a en général

aucune raison de se "distribuer" en un réseau de Petri non étiqueté. De plus, on peut montrer que la propriété du modèle fini du Mu-Calcul n'est plus valable sur la classe des réseaux de Petri non-étiquetés, au sens où il existe des formules dont tout réseau solution possède un graphe de marquages qui n'est pas bisimilaire à un processus fini.

1.2.2 La synthèse de réseaux de Petri dans cette thèse

Dans cette thèse, nous étudions le problème **Sat**(**S**,**C**) lorsque **S** est le Mu-Calcul ou un fragment du Mu-Calcul et **C** est la classe des réseaux de Petri non étiquetés. Comme nous l'avons mentionné précédemment, le Mu-Calcul est lié à la notion de bisimulation. En conséquence, les méthodes de synthèse de réseau à partir de graphes ne sont pas appropriées : d'une part ces méthodes reposent sur l'isomorphisme, qui est une équivalence fortement axée vers les états d'un système contrairement à la bisimulation, et d'autre part les seules spécifications de graphes pour lesquelles la synthèse est effective à ce jour sont les spécifications automatiques, qui sont particulièrement éloignées des spécifications logiques. En effet, une spécification automatique donne l'ensemble des états du graphe sous la forme d'un langage rationnel, ce qui constitue un élément primordial pour le procédé de synthèse, alors qu'une spécification logique n'impose pas en général une similitude structurelle entre ses différents modèles.

Par opposition, les méthodes de synthèse à partir d'un langage rationnel offrent un point de départ solide au développement de méthodes de synthèse à partir de spécifications logiques pour deux raisons : premièrement, l'arbre des exécutions d'un système déterministe peut être vu comme un langage ; dans ce cadre, deux systèmes possédant le même langage sont bisimilaires. Par conséquent la notion de langage "capte" la notion de bisimulation inhérente à la logique. Deuxièmement, les méthodes de synthèse à partir d'un langage offrent des solutions pour la synthèse à partir d'un intervalle de langage rationnels : la synthèse consiste à produire un réseau de Petri dont le langage est compris, au sens de l'inclusion, entre deux langages rationnels L et L' donnés. La synthèse pour les intervalles de langages est possible grâce au résultat suivant : étant donné un langage rationnel L, il est possible de construire un réseau de Petri dont le langage $L_{\mathcal{N}}$ est le plus petit langage de réseau tel que $L \subseteq L_{\mathcal{N}}$ [Dar04]. Il suffit alors de vérifier que $\mathrm{L}_{\mathcal{N}} \subseteq L'$, ce qui est décidable [May84].

Les intervalles de langages rationnels seront vus dans ce document comme une spécification logique simple : les modèles d'un intervalle de langages sont les systèmes dont le langage appartient à l'intervalle. Nous nous attachons à

déterminer la frontière de décidabilité de la synthèse entre le Mu-Calcul et les intervalles de langages rationnels. Nous considérons également le problème du marquage qui s'énonce comme suit :

Problème 1.2.4 (Marquage : Marq(S,C)). Étant donnés $\phi \in \mathbf{S}$ une formule logique, et $\mathcal{N} \in \mathbf{C}$ un réseau de Petri non initialisé, existe-t-il un marquage initial m_0 de $\mathcal{N}$ tel que le graphe des marquages de $\mathcal{N}$ accessibles à partir de m_0 satisfasse ϕ.

1.2.3 Plan du document

Dans la suite de ce chapitre, nous introduisons les réseaux de Petri et leurs différentes sous-classes, puis nous présentons la méthode de synthèse à partir de langages rationnels et à partir d'intervalles de langages rationnels.

Dans le Chapitre 2, nous introduisons le Mu-Calcul, noté $\mathbf{L}_\mu$; nous étudions le problème $\mathbf{Sat}(\mathbf{L}_\mu,\mathbf{C})$ lorsque $\mathbf{C}$ est la classe $\mathbf{P_N}$ des réseaux de Petri non étiquetés. Nous montrons que le problème $\mathbf{Sat}(S,\mathbf{P_N})$ est indécidable lorsque $\mathbf{S}$ est un fragment de $\mathbf{L}_\mu$ n'autorisant pas les imbrications de points fixes (on parle du fragment *alternation-free* de $\mathbf{L}_\mu$). Nous montrons également que le problème pour $\mathbf{Marq}(\mathbf{S},\mathbf{C})$ est indécidable lorsque $\mathbf{S}$ est le fragment *alternation-free* de $\mathbf{L}_\mu$ et $\mathbf{C}$ est une sous-classe particulièrement restreinte de $\mathbf{P_N}$; en particulier, les réseaux de cette classe ont tous un graphe des marquages fini. Nous proposons enfin une sémantique de $\mathbf{L}_\mu$ basée sur les langages pour laquelle nous établissons l'équivalence avec la sémantique usuelle de $\mathbf{L}_\mu$; cette sémantique basée sur les langages, plus adaptée aux réseaux dans notre cadre, sera utilisée dans les chapitres suivants pour en simplifier les preuves.

Dans le Chapitre 3, nous proposons un fragment syntaxique particulièrement restreint de $\mathbf{L}_\mu$: le *Nu-Calcul Conjonctif*, noté $\mathbf{L}_\nu$, dans lequel seules les conjonctions et les plus grands points fixes sont autorisés. Nous associons à $\mathbf{L}_\nu$ un ensemble de reconnaisseurs de langages : les *spécifications modales*. Ces spécifications sont une adaptation aux langages de la sous-classe des automates alternants qui correspond au sentences de $\mathbf{L}_\nu$. Nous montrons l'équivalence d'expressivité entre $\mathbf{L}_\nu$ et les spécifications modales et nous donnons les algorithmes de traduction entre les spécifications de ces deux formalismes. Nous montrons la décidabilité du problème $\mathbf{Sat}(\mathbf{S},\mathbf{P_N})$ lorsque $\mathbf{S}$ est une famille de spécifications obtenue par une restriction structurelle sur les spécifications modales.

Dans le Chapitre 4, nous proposons le concept de *théorie d'un réseau* : la théorie d'un réseau $\mathcal{N}$ est une formule de $\mathbf{L}_\nu$ dont les modèles dans $\mathbf{P_N}$ sont

les réseaux qui contiennent au moins les places de $\mathcal{N}$. Nous montrons que la théorie d'un réseau $\mathcal{N}$ peut s'exprimer sous la forme d'un intervalle de langages rationnels. Nous utilisons enfin la théorie d'un réseau pour montrer l'indécidabilité du problème $\mathbf{Sat}(\mathbf{L}_\nu,\mathbf{P_N})$.

1.3 Préliminaires

1.3.1 Langages et images commutatives

On se fixe un alphabet fini $\Sigma = \{a_1, \dots a_n\}$. On considère les langages sur Σ, d'éléments typiques $L, R, ...$ avec les notations habituelles : L^*, $L.a$ (pour $a \in \Sigma$), etc. On note $1 \in \Sigma^*$ pour le mot vide ; lorsque u et v sont deux éléments de Σ^*, $u.v$ désigne la concaténation de u et v, et $u^* = \{u^k \mid k \in \mathbb{N}\}$, où u^k est la concaténation de k fois le mot u. Nous noterons également Σ^ω pour l'ensemble des mots infinis sur Σ et Σ^∞ pour $\Sigma^* \cup \Sigma^\omega$.

Définition 1.3.1 (Clôture par préfixe). Soit L un langage, on dit que L est clos par préfixe si et seulement si $1 \in L$ et tout mot $a_1 \dots a_n \in L$ vérifie $a_1 \dots a_{n-1} \in L$. La *clôture par préfixe* d'un langage L est le plus petit langage clos par préfixe contenant L, on la note $\overline{L}$. On notera également $L_{w\backslash}$ l'ensemble $\{v \in \Sigma^* \mid w.v \in L\}$ des suffixes de w dans L.

Remarquons que le langage vide n'est par définition pas clos par préfixe, nous le traiterons donc séparément lorsque c'est nécessaire ; en particulier, pour un langage clos par préfixe L, pour tout mot w, le langage $L_{w\backslash}$ est soit clos par préfixe (si $w \in L$) soit vide. Par la suite L dénotera toujours un langage clos par préfixe sur l'alphabet Σ.

Lorsque u est un mot sur Σ, on notera $|u|$ la *longueur* de u. On notera également $|u|_a$ pour le nombre d'occurrences de la lettre $a \in \Sigma$ dans le mot u.

Définition 1.3.2 (Image commutative). L'application de Parikh $[.] : \Sigma^* \to \mathbb{N}^n$ est l'application définie par

$$[u] = \langle |u|_{a_1}, \dots, |u|_{a_n} \rangle$$

On dit alors que $[u]$ est l'*image commutative* (ou *image de Parikh*) de u.

Définition 1.3.3 (Semi-linéaire). Soit $\mathcal{M} = (\mathcal{M}, \cdot, 1)$ un monoïde. Un sous ensemble de $\mathcal{M}$ est *linéaire* s'il s'exprime sous la forme $m \cdot \mathcal{F}^*$ où $m \in \mathcal{M}$, $\mathcal{F}$ est un sous ensemble fini de $\mathcal{M}$ et $\mathcal{F}^*$ est le plus petit sous-monoïde de $\mathcal{M}$ contenant

$\mathcal{F}$. Une union finie de sous-ensembles linéaires de $\mathcal{M}$ est un sous-ensemble semi-linéaire de $\mathcal{M}$.

On dira qu'un langage L est *semi-linéaire* lorsque son image commutative $[L]$ est un sous-ensemble semi-linéaire du monoïde commutatif $\mathbb{N}^n$ dont le produit $\cdot$ est l'addition de n-vecteurs et l'élément neutre 1 est le vecteur identiquement nul.

Définition 1.3.4 (Intervalles de langages). Soit L_1 et L_2 deux langages clos par préfixe sur Σ. Nous définissons un système de spécifications simple sur les éléments de Σ^* : les *intervalles de langages*. Un *intervalle de langages* est un élément $[L_1, L_2]$; on dit qu'un langage L sur Σ *appartient* à $[L_1, L_2]$, noté $L \in [L_1, L_2]$, si $L_1 \subseteq L \subseteq L_2$. L'ensemble des intervalles de langages sur Σ sera noté $\textbf{INT}_{\boldsymbol{\Sigma}}$.

1.3.2 Processus

Nous définissons les systèmes de transition déterministes, appelés *processus* dans [AVW03].

Définition 1.3.5. Un *processus* est un tuple $\mathcal{P} = \langle S, s_0, t\rangle$, où :
- S est l'ensemble des *états*,
- $s_0 \in S$ est l'*état initial*,
- $t : S \times \Sigma \to S$ est une fonction de transition partielle.

Pour tout processus $\mathcal{P} = \langle S, s_0, t\rangle$, il y a une *a-transition* de s vers s' dans $\mathcal{P}$ lorsque $t(s, a) = s'$; on note alors $s \xrightarrow{a} s'$. L'état s' est appelé *successeur* de s par a.
Un *chemin* dans $\mathcal{P}$ est une séquence $v \in \Sigma^*$ telle que l'extension $t : S \times \Sigma^* \to S$ de la fonction de transition aux mots soit définie à l'état s_0. On dit que le chemin v *atteint* l'état $s = t(s_0, v)$. Par extension, un chemin infini dans $\mathcal{P}$ est une séquence $v \in \Sigma^\omega$ telle que tous ses préfixes soient des chemins dans $\mathcal{P}$.

Les processus peuvent être composés entre eux de manière synchrone en utilisant le *produit synchrone*.

Définition 1.3.6. Soient deux processus $\mathcal{P}_1 = \langle S_1, s_0^1, t_1\rangle$ et $\mathcal{P}_2 = \langle S_2, s_0^2, t_2\rangle$, le *produit synchrone* de $\mathcal{P}_1$ et $\mathcal{P}_2$ est le processus $\mathcal{P}_1 \times \mathcal{P}_2 = \langle S_1 \times S_2, (s_0^1, s_0^2), t\rangle$ où la fonction de transition t est définie de la manière suivante :

– $t((s_1, s_2), a) = (t_1(s_1, a), t_2(s_2, a))$ lorsque $t_1(s_1, a)$ et $t_2(s_2, a)$ sont définis,
– sinon $t((s_1, s_2), a)$ n'est pas défini.

Définition 1.3.7 (Langage d'un processus). Le *langage* du processus $\mathcal{P} = \langle S, s_0, t\rangle$ est l'ensemble de ses chemins, et se note $\mathcal{L}(\mathcal{P})$.

Proposition 1.3.8. *Si* $\mathcal{P} = \mathcal{P}_1 \times \mathcal{P}_2$, *alors* $\mathcal{L}(\mathcal{P}) = \mathcal{L}(\mathcal{P}_1) \cap \mathcal{L}(\mathcal{P}_2)$.

Démonstration. La preuve est immédiate puisque le langage d'un processus est l'ensemble de ses chemins et que la fonction de transition étendue t de $\mathcal{P}_1 \times \mathcal{P}_2$ vérifie, si t_1 et t_2 sont les fonctions de transition étendues de $\mathcal{P}_1$ et $\mathcal{P}_2$, la propriété : pour tout $u \in \Sigma^*$, $t((s_0^1, s_0^2), u)$ est défini si est seulement $t_1(s_0^1, u)$ et $t_2(s_0^2, u)$ sont définis. □

1.4 Les réseaux de Petri

Dans cette section, nous présentons les réseaux de Petri. Les réseaux de Petri ont été introduits par Carl Adam Petri [Pet62]. Notre thèse concernant principalement les réseau de Petri non étiquetés, nous axons la présentation sur ce type de réseaux ; nous donnons toutefois les définitions relatives aux réseaux étiquetés. Les concepts de *graphe des marquages* et de *langage* d'un réseau présentés dans cette section sont centraux dans cette thèse car ils correspondent aux deux équivalences principales considérées pour la synthèse de réseau : l'isomorphisme et l'égalité des langages. Pour plus de détails sur les réseaux de Petri, le lecteur pourra se référer à [Mur89].

Définition 1.4.1 (Réseau de Petri). Un *réseau de Petri* est un tuple $\mathcal{N} = \langle P, T, F, W\rangle$ où
– $P = \{p_1, \ldots, p_n\}$ est un ensemble fini de *places*,
– $T = \{t_1, \ldots, t_m\}$ est un ensemble fini de *transitions*,
– $F \subseteq (P \times T) \cup (T \times P)$ est un ensemble fini d'arcs définissant la *relation de flot*,
– $W : F \to \mathbb{N}$ est une pondération des arcs.

Un réseau de Petri est un graphe bipartite orienté. Ce graphe possède deux sortes de sommets, les *places* et les *transitions*. Les arcs sont pondérés et relient une place à une transition ou une transition à une place. On convient que $W(p, t) = 0$ lorsque $(p, t) \notin F$ et que $W(t, p) = 0$ lorsque $(t, p) \notin F$.
On représente graphiquement un réseau de Petri comme sur la Figure 1.1, les ronds représentent les places et les rectangles les transitions. On omettra le poids des arcs lorsque celui-ci est égal à 1.

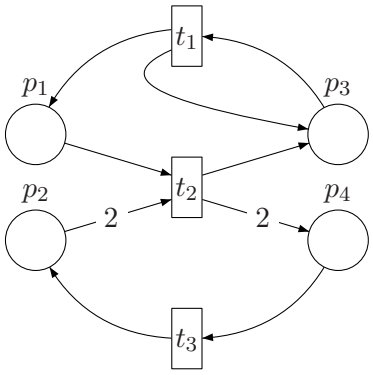

FIG. 1.1 – Un exemple de réseau de Petri.

On dit d'une place p qu'elle est *en entrée* de la transition t lorsque $(p,t) \in F$; similairement, on dit d'une place p qu'elle *en sortie* de t lorsque $(t,p) \in F$. On note

- ${}^\bullet p$ pour les transitions dont p est en entrée, ${}^\bullet p = \{t \in T \mid (t,p) \in F\}$,
- $p^\bullet$ pour les transitions dont p est en sortie, $p^\bullet = \{t \in T \mid (p,t) \in F\}$,
- ${}^\bullet t$ pour les places en entrée de t, ${}^\bullet t = \{p \in P \mid (p,t) \in F\}$,
- $t^\bullet$ pour les places en sortie de t, $t^\bullet = \{p \in P \mid (t,p) \in F\}$.

On dit qu'un réseau de Petri est *pur* lorsqu'aucune de ses places n'est en entrée et en sortie d'une même transition, c.à-d. que pour toute transition t, ${}^\bullet t \cap t^\bullet = \emptyset$.

Les places d'un réseau contiennent des *jetons*. Un *marquage* m est une fonction $m : P \to \mathbb{N}$ qui décrit la répartition des jetons. Ainsi, lorsque $m(p) = 2$, on dit que la place p contient 2 jetons. Un *réseau initialisé* est un réseau de Petri $\mathcal{N}$ assorti d'un marquage m_0 dit *marquage initial*; on le note $(\mathcal{N}, m_0)$.

Exemple 1.4.2. *La Figure 1.2 représente le réseau de la Figure 1.1 initialisé avec 1 jeton dans la place p_1, 2 jetons dans la place p_2 et aucun jeton dans les places p_3 et p_4.*

Définition 1.4.3 (Tir de transitions). Une transition $t \in T$ est dite *active* (resp. *inactive*) au marquage m si pour toute place $p \in {}^\bullet t$, $W(p,t) \geq m(p)$ (resp. il existe une place $p \in {}^\bullet t$ telle que $W(t,p) < m(p)$). Lorsque la transition t est active au marquage m, on définit le *tir* de la transition t par l'action qui modifie le marquage m en un marquage m' défini pour toute place $p \in P$ par $m'(p) = m(p) - W(p,t) + W(t,p)$. On note $m[t\rangle$ lorsque t est active au marquage m et $m[t\rangle m'$ lorsque m' est le résultat du tir de t depuis m, avec pour tout place

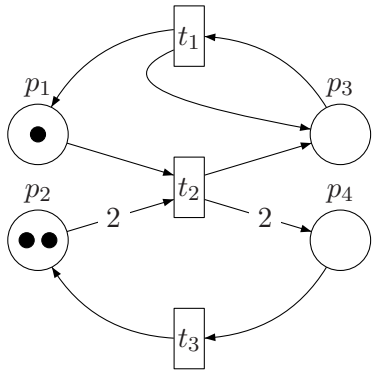

FIG. 1.2 – Un exemple de réseau de Petri initialisé.

p, $m'(p) = m(p) - W(p,t) + W(t,p)$. On étend cette notation à *une séquence de tir* $u \in T^*$ avec $u = t_1.t_2 \dots t_n$ en posant $m[u\rangle m'$ lorsqu'il existe une suite de marquages $m_1, \dots, m_{n-1}$ telle que $m[t_1\rangle m_1[t_2\rangle \dots m_{n-1}[t_n\rangle m'$. La relation $[.\rangle$ entre les marquages de $\mathcal{N}$ est nommée *relation de tir* de $\mathcal{N}$.

On dit d'un marquage m de $\mathcal{N}$ qu'il est *accessible* depuis le marquage initial m_0 lorsqu'il existe une séquence de tir u telle que $m_0[u\rangle m$.

Exemple 1.4.4. *Considérons le réseau initialisé* $(\mathcal{N}, m_0)$ *de la Figure 1.2, pour lequel* $m_0(p_1) = 1$, $m_0(p_2) = 2$, $m_0(p_3) = 0$ *et* $m_0(p_4) = 0$ *; nous avons les relations de tir suivantes :*

- $m_0[t_2\rangle m'$, *avec* $m'(p_1) = 0$, $m'(p_2) = 0$, $m'(p_3) = 1$ *et* $m'(p_4) = 2$,
- $m'[t_1\rangle m''$ *(ou* $m_0[t_2.t_1\rangle m''$*), avec* $m''(p_1) = 1$, $m''(p_2) = 0$, $m''(p_3) = 1$, $m''(p_4) = 2$,
- t_2 *est inactive au marquage* m''.
- t_3 *est active au marquage* m'' *(ou* $m_0[t_2.t_1.t_3\rangle$*).*

1.4.1 Comportements d'un réseau de Petri

Les comportements d'un réseau de Petri sont définis par ses séquences de tir. Il existe différentes structures pour représenter ces comportements, parmi lesquelles le *graphe des marquages* où le *langage* du réseau que nous définissons ici.

Définition 1.4.5 (Graphe des marquages). Le *graphe des marquages* du réseau $\mathcal{N}$ est le graphe orienté et étiqueté noté $\mathcal{G}(\mathcal{N})$ dont les sommets sont les marquages de $\mathcal{N}$ accessibles depuis m_0 et dont les arcs sont définis par la relation de tir de

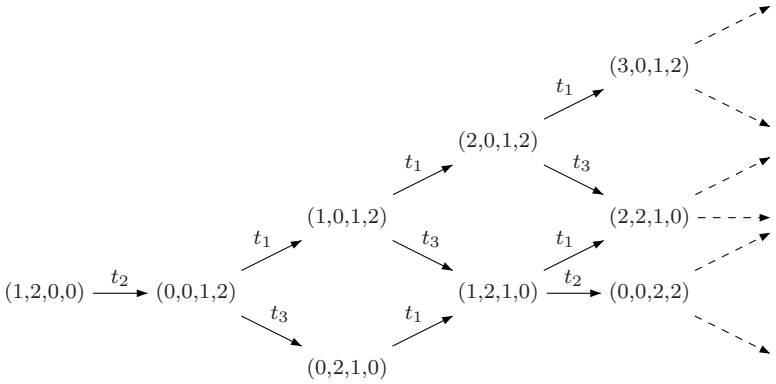

FIG. 1.3 – Graphe des marquages de $(\mathcal{N}, m_0)$.

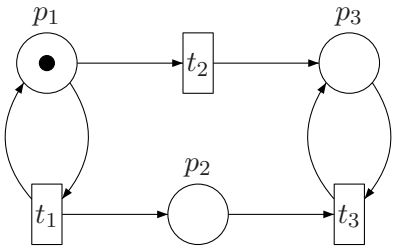

FIG. 1.4 – Le réseau de Petri initialisé $(\mathcal{N}^1, m_0^1)$.

$\mathcal{N}$: il existe un arc de m à m' étiqueté par t si et seulement si $m[t\rangle m'$.
Le *graphe des marquages* du réseau initialisé $(\mathcal{N}, m_0)$ est la restriction notée $\mathcal{G}(\mathcal{N}, m_0)$ de $\mathcal{G}(\mathcal{N})$ aux marquages accessibles de $\mathcal{N}$ depuis m_0.

Exemple 1.4.6. *La Figure 1.3 représente une partie du graphe des marquages du réseau initialisé $(\mathcal{N}, m_0)$ de la Figure 1.2. Ce graphe est infini ; en effet, il est possible de tirer indéfiniment la séquence de transitions t_2, t_1, t_3 pour obtenir une suite de marquages qui associe successivement à la place p_3 un nombre arbitrairement grand.*

Définition 1.4.7 (Langage d'un réseau). Le langage d'un réseau initialisé $(\mathcal{N}, m_0)$ est le langage $\mathcal{L}(\mathcal{N}, m_0) \in T^*$ défini par $\mathcal{L}(\mathcal{N}, m_0) = \mathcal{L}(\mathcal{G}(\mathcal{N}, m_0))$.

Exemple 1.4.8. *Le langage du réseau initialisé* $(\mathcal{N}^1, m_0^1)$ *de la Figure 1.4 est le langage* $\mathcal{L}(\mathcal{N}^1, m_0^1) = \{t_1^n \mid n \in \mathbb{N}\} \cup \{t_1^n.t_2.t_3^m \mid n \geq m\}$.

Le graphe des marquages d'un réseau initialisé $(\mathcal{N}, m_0)$ peut être vu comme le processus $\mathcal{P} = \langle S, s_0, \delta \rangle$ avec $S = \{m \mid \exists u \in \Sigma^*, m_0[u\rangle m\}$, $s_0 = m_0$ et $t(m, a) = m'$ tel que $m[a\rangle m'$. Il est alors possible de redéfinir le langage du réseau par $\mathcal{L}(\mathcal{N}, m_0) = \mathcal{L}(\mathcal{P})$.

1.4.2 Réseaux de Petri étiquetés

Tels que nous avons défini les comportements d'un réseau de Petri, les actions (étiquettes des transitions pour le graphe, éléments de l'alphabet pour le langage) sont les transitions du réseau. Cependant, une définition plus générale utilise une fonction d'étiquetage sur un alphabet d'actions fini Σ. On parle alors de *réseau de Petri étiqueté*.

Définition 1.4.9 (Réseau de Petri étiqueté). Un *réseau de Petri étiqueté* sur un alphabet Σ est un couple $(\mathcal{N}, l)$ où $\mathcal{N} = (P, T, F, W)$ est un réseau de Petri et $l : T \to \Sigma$ est une *fonction d'étiquetage* des transitions de $\mathcal{N}$.

La fonction l sera étendue à $l : T^* \to \Sigma^*$ par $l(a_1 \ldots . a_k) = l(a_1) \ldots . l(a_k)$.
Nous redéfinissons maintenant les notions de graphe des marquages et de langage en prenant en compte l'étiquetage.

Définition 1.4.10 (Graphe des marquages d'un réseau étiqueté). Le *graphe des marquages* du réseau étiqueté $(\mathcal{N}, l)$ est le graphe orienté et étiqueté noté $\mathcal{G}(\mathcal{N}, l)$ dont les sommets sont les marquages de $\mathcal{N}$ accessibles depuis m_0 et dont les arcs sont définis par la relation de tir de $\mathcal{N}$: il existe un arc de m à m' étiqueté par $l(t)$ si et seulement si $m[t\rangle m'$.
Le *graphe des marquages* du réseau étiqueté initialisé $(\mathcal{N}, l, m_0)$ est la restriction notée $\mathcal{G}(\mathcal{N}, l, m_0)$ de $\mathcal{G}(\mathcal{N}, l)$ aux marquages accessibles de $\mathcal{N}$ depuis m_0.

Remarque 1.4.11. Lorsque l'étiquetage n'est pas injectif, le graphe des marquages d'un réseau étiqueté n'est pas nécessairement déterministe (il peut exister deux arcs issus d'un même sommet et portant la même étiquette).

Définition 1.4.12 (Langage d'un réseau étiqueté). Le langage d'un réseau étiqueté initialisé $(\mathcal{N}, l, m_0)$ est le langage $\mathcal{L}(\mathcal{N}, l, m_0) \in \Sigma^*$ défini par $\mathcal{L}(\mathcal{N}, l, m_0) = \mathcal{L}(\mathcal{G}(\mathcal{N}, l, m_0))$.

Les réseaux étiquetés sont plus expressifs (en terme de langage comme de graphe des marquages) que les réseaux non étiquetés. En effet, même si l'on se

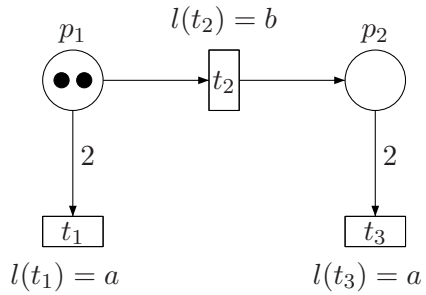

FIG. 1.5 – Un réseau étiqueté initialisé dont le langage n'est pas réalisable par un réseau non étiqueté.

restreint au réseaux étiquetés dont le graphe des marquages est déterministe, il existe des graphes de marquages (resp. des langages) de réseau étiqueté qui ne sont pas réalisables par un réseau non étiqueté, comme le montre l'exemple suivant.

Exemple 1.4.13. *Dans le réseau étiqueté de la Figure 1.5 : la transition t_1 étiquetée par a est active au marquage initial ; après le tir de la transition étiquetée par b, aucune transition étiquetée par a n'est active ; après un deuxième tir de b, la transition t_3 étiquetée par a est tirable. Ce comportement est impossible dans le cas d'un réseau non étiqueté ; en effet, si a est active au marquage initial et qu'elle ne l'est plus après le tir de b, alors cela signifie que le tir de b supprime des jetons utiles pour celui de a. Par conséquent, un second tir de la transition b ne peut rendre a active de nouveau.*

L'expressivité des réseaux étiquetés dépasse celle des processus comme en témoigne la proposition suivante :

Proposition 1.4.14. *Soit $\mathcal{P}$ un processus fini, il existe un réseau étiqueté dont le graphe des marquages est isomorphe à $\mathcal{P}$.*

Démonstration. Il suffit de construire le réseau $(\mathcal{N}, l)$ où les places de $\mathcal{N}$ sont les états de $\mathcal{P}$. L'ensemble des transitions de $\mathcal{N}$ est défini comme suit : pour toute transition $s \xrightarrow{a} s'$ de $\mathcal{P}$ il existe une transition de $\mathcal{N}$ étiquetée par a, en entrée de s' et en sortie de s. Les arcs de $\mathcal{N}$ sont tous de poids 1 et le marquage initial de $\mathcal{N}$ est le marquage qui associe un unique jeton à la place correspondant à l'état initial de $\mathcal{P}$. □

Exemple 1.4.15 (Illustration de la Proposition 1.4.14). *Le processus de la Figure 1.6 est isomorphe au graphe des marquages du réseau de la Figure 1.7.*

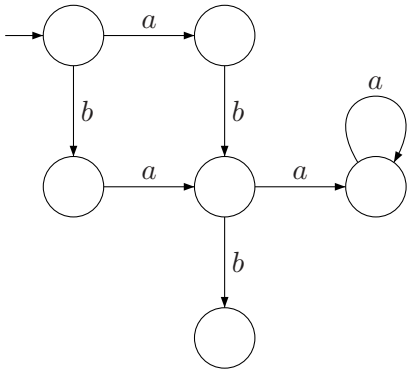

FIG. 1.6 – Un processus.

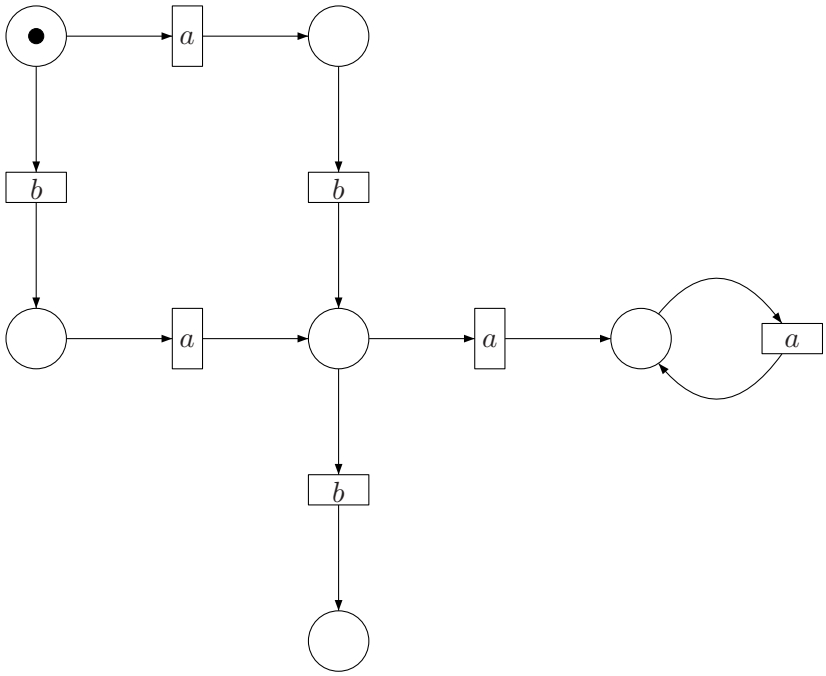

FIG. 1.7 – Exemple de réseau de Petri étiqueté dont le graphe des marquages est isomorphe à celui du processus de la Figure 1.6.

La preuve de la Proposition 1.4.14 donne une méthode de synthèse possible de réseau étiqueté à partir d'un processus. Cette synthèse présente peu d'intérêt ; en effet, le résultat est un réseau aussi séquentiel que le processus d'origine : la synthèse n'exploite pas la concurrence inhérente aux réseaux de Petri.

Les réseaux de Petri non étiquetés constituent le sous-ensemble des réseaux étiquetés dont la fonction d'étiquetage est injective. Par souci de clarté, nous considérerons désormais qu'un réseau non étiqueté est un réseau où $\Sigma = T$ et où l est l'identité. Nous utiliserons alors Σ pour désigner les transitions d'un réseau non étiqueté. Nous considérerons désormais, sauf mention contraire, uniquement des réseaux non étiquetés.

1.4.3 Représentation vectorielle de réseaux de Petri

L'objet de cette section est de proposer un présentation alternative des réseaux de Petri où $\mathcal{N}$ est un ensemble de vecteurs, chaque vecteur représentant une place. Cette présentation est adaptée à la synthèse de réseaux telle qu'elle est présentée dans [Dar04] ainsi que dans la Section 1.5 de ce document.

Soit $\Sigma = a_1, \ldots, a_n$ un alphabet fini. Soit $\mathcal{N} = (P, \Sigma, F, W)$ un réseau de Petri[4] et m_0 un marquage initial de $\mathcal{N}$. Une place $p \in P$ peut être identifiée au vecteur de dimension $2n + 1$ suivant :

$$p = \langle m_0(p), W(p, a_1), \ldots, W(p, a_n), W(a_1, p), \ldots W(a_n, p)\rangle.$$

Sous cette représentation, tout vecteur de $\mathbb{N}^{2n+1}$ représente alors une place de réseau. Nous adopterons les notations suivantes :

- $<p, {}^\bullet a>$ pour $W(p, a)$,
- $<p, a^\bullet>$ pour $W(a, p)$,
- $<p, a>$ pour $W(a, p) - W(p, a)$.

Nous étendons cette notation à toute séquence de transitions $u = a^1, \ldots a^m$ avec $<p, u> = \Sigma_{k\in[1,m]}<p, a^k>$. Avec cette notation, une place p est identifiée au vecteur

$$p = \langle m_0(p), <p, {}^\bullet a_1>, \ldots, <p, {}^\bullet a_n>, <p, a_1^\bullet>, \ldots, <p, a_n^\bullet>\rangle$$

La relation de tir s'exprime alors en fonction de $<p, .>$:

$$m[a\rangle \quad \text{ssi} \quad \forall p \in P, m(p) - <p, {}^\bullet a> \geq 0 \tag{1.1}$$

$$m[a\rangle m' \quad \text{ssi} \quad m[a\rangle \text{ et } \forall p \in P, m'(p) = m(p) + <p, a> \tag{1.2}$$

[4] Rappelons que les réseaux que nous considérons sont non-étiquetés.

et dans le cas d'une séquence de tir $u \in \Sigma^*$, nous avons :

$$m[u\rangle m' \quad \text{ssi} \quad m[u\rangle \text{ et } m'(p) = m(p) + \sum_{a\in\Sigma} |u|_a.<p,a> \tag{1.3}$$

Lorsqu'on utilise la représentation des places sous forme de vecteur, les places suffisent alors à définir le réseau. Nous donnons une nouvelle définition de réseau de Petri utilisant cette représentation.

Définition 1.4.16 (Réseau de Petri sous forme vectorielle). Un réseau de Petri initialisé sous forme vectorielle sur l'alphabet Σ est un ensemble de places $\{p_1, \ldots, p_k\}$ où chaque place est un vecteur de dimension $2.n + 1$.

En utilisant cette représentation, il nous est possible de définir facilement l'ensemble d'opérations suivantes :
- la *suppression* d'une place p dans un réseau $\mathcal{N}$ est la construction du réseau $\mathcal{N} \setminus \{p\}$,
- l'*ajout* d'une place p dans un réseau $\mathcal{N}$ est la construction du réseau $\mathcal{N} \cup \{p\}$,
- la *substitution* d'une place p par une place p' dans un réseau $\mathcal{N}$ est la construction du réseau $\mathcal{N} \cup \{p'\} \setminus \{p\}$.

Exemple 1.4.17. *Les réseaux initialisés $(\mathcal{N}, m_0)$ et $(\mathcal{N}^1, m_0^1)$ des Figures 1.2 et 1.4 sur l'alphabet $\{t_1, t_2, t_3\}$ se représentent de manière vectorielle comme suit :*

$$(\mathcal{N}, m_0) = \left\{ \begin{array}{l} \langle 1,0,1,0,1,0,0\rangle, \\ \langle 2,0,2,0,0,0,2\rangle, \\ \langle 0,1,0,0,1,1,0\rangle, \\ \langle 0,0,0,1,0,2,0\rangle \end{array} \right\}$$

$$(\mathcal{N}^1, m_0^1) = \left\{ \begin{array}{l} \langle 1,1,1,0,1,0,0\rangle, \\ \langle 0,0,0,1,1,0,0\rangle, \\ \langle 1,0,0,1,0,1,1\rangle \end{array} \right\}$$

Lorsque $\mathcal{N} = \mathcal{N}_1 \biguplus \mathcal{N}_2$, et que m est un marquage de $\mathcal{N}$, on note $m|_{\mathcal{N}_1}$ la restriction de m aux places de $\mathcal{N}_1$.

Proposition 1.4.18. *Si $(\mathcal{N}, m_0)$ est un réseau initialisé tel que $\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2$, alors $\mathcal{G}(\mathcal{N}, m_0)$ est isomorphe au produit synchrone $\mathcal{G}(N_1, m_0|_{\mathcal{N}_1}) \times \mathcal{G}(N_2, m_0|_{\mathcal{N}_2})$.*

Démonstration. On montre que η tel que $\eta(m) = (m|_{\mathcal{N}_1}, m|_{\mathcal{N}_2})$ est un isomorphisme. Remarquons que η est injectif, et que par conséquent, η^{-1} est bien défini. Lorsque $m \xrightarrow{a} m'$, par définition $m[a\rangle m'$, nous avons alors pour tout $p \in \mathcal{N}$,

$m(p) - <p, {}^\bullet a> \geq 0$ et $m'(p) = p + <p, a>$; ceci est vrai pour $p \in \mathcal{N}_1$ comme pour $p \in \mathcal{N}_2$. Par conséquent $m|_{\mathcal{N}_1}[a\rangle m'|_{\mathcal{N}_1}$ et $m|_{\mathcal{N}_2}[a\rangle m'|_{\mathcal{N}_2}$, ce qui entraîne $\eta(m) \xrightarrow{a} \eta(m')$.
Réciproquement, lorsque $(m|_{\mathcal{N}_1}, m|_{\mathcal{N}_2}) \xrightarrow{a} (m'|_{\mathcal{N}_1}, m'|_{\mathcal{N}_2})$, par définition du produit synchrone et du graphe des marquages (Définitions 1.3.6 et 1.4.5), nous avons $m|_{\mathcal{N}_1}[a\rangle m'|_{\mathcal{N}_1}$ et $m|_{\mathcal{N}_2}[a\rangle m'|_{\mathcal{N}_2}$. Puisque $\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2$, il en résulte $m[a\rangle m'$, donc

$$\eta^{-1}(m|_{\mathcal{N}_1}, m|_{\mathcal{N}_2}) \xrightarrow{a} \eta^{-1}(m'|_{\mathcal{N}_1}, m'|_{\mathcal{N}_2}).$$

Puisque $\eta(m_0) = (m_0|_{\mathcal{N}_1}, m_0|_{\mathcal{N}_1})$, les marquages accessibles de $\mathcal{G}(\mathcal{N})$ depuis m_0 sont exactement les sommets de $\mathcal{G}(N_1) \times \mathcal{G}(N_2)$ accessibles depuis $\eta(m_0)$, η est alors un isomorphisme. □

Proposition 1.4.19. *Si $(\mathcal{N}, m_0)$ est un réseau initialisé tel que $\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2$, alors $\mathcal{L}(\mathcal{N}, m_0) = \mathcal{L}(\mathcal{N}_1, m_0|_{N_1}) \cap \mathcal{L}(\mathcal{N}_2, m_0|_{N_2})$.*

Démonstration. La démonstration est immédiate par les Propositions 1.3.8 et 1.4.18. □

Étant donné un réseau initialisé $(\mathcal{N}, m_0)$, la Proposition 1.4.19 nous permet de décomposer le langage de $(\mathcal{N}, m_0)$ de la manière suivante :

$$\mathcal{L}(\mathcal{N}, m_0) = \bigcap_{p \in \mathcal{N}} \mathcal{L}(\{p\}, m_0|_{\{p\}}).$$

De même, la Proposition 1.4.18 nous permet de décomposer le graphe des marquages de $(\mathcal{N}, m_0)$ où $\mathcal{N} = \{p_1, \ldots, p_k\}$ de la manière suivante :

$$\mathcal{G}(\mathcal{N}, m_0) \text{ est isomorphe à } \mathcal{G}(\{p_1\}, m_0|_{\{p_1\}}) \times \ldots \times \mathcal{G}(\{p_k\}, m_0|_{\{p_k\}}).$$

1.4.4 Quelques sous-classes des réseaux de Petri

De très nombreuses sous-classes de réseaux de Petri ont été étudiées, nous ne présenterons ici que les sous classes utilisées dans cette thèse. Pour un inventaire plus complet, le lecteur pourra se référer à [Mur89]. Nous noterons **PN** pour la classe des réseaux de Petri (non-étiquetés) ainsi que **LPN** pour celle des réseaux de Petri étiquetés.

1.4.4.1 Sous-classes dépendant du marquage initial

Il existe quantité de sous-classes de réseaux de Petri dépendant du marquage initial. Nous ne présentons ici que celles qui concernent l'existence d'une borne pour les marquages durant toute séquence de tir.

Définition 1.4.20 (Sous-classes de réseaux initialisés). Soit $(\mathcal{N}, m_0)$ un réseau de Petri initialisé. Nous définissons les sous-classes suivantes :

- **PNB** : la classe des *réseaux bornés* ; $(\mathcal{N}, m_0)$ est un réseau borné si et seulement si il existe une borne b telle que pour tout marquage accessible m depuis m_0 et pour toute place p, $m(p) \leq b$.
- **PNkB** : la classe des *réseaux k-bornés* (pour $k \in \mathbb{N}$ donné) ; $(\mathcal{N}, m_0)$ est un réseau k-borné si et seulement si pour tout marquage accessible m depuis m_0 et pour toute place p, $m(p) \leq k$.
- **PNSF** : la classe des *réseaux saufs* ; **PNSF** = **PN1B**.

Lorsqu'un réseau $(\mathcal{N}, m_0)$ est borné ($(\mathcal{N}, m_0) \in$ **PNB**), alors il existe k tel que $(\mathcal{N}, m_0) \in$ **PNkB**. Plus généralement, nous pouvons définir **PNB** par :

$$\mathbf{PNB} = \bigcup_{k \in \mathbb{N}} \mathbf{PNkB}$$

Remarque 1.4.21. La classe **PNkB** pour un k donné est finie : toute place p vérifie nécessairement $m_0(p) \leq k$ et pour toute transition a, $<p, {}^\bullet a> \leq k$ et $<p, a^\bullet> \leq k$. Par conséquent, il existe au plus $(2.|\Sigma| + 1)^{k+1}$ places de réseaux, ce qui entraîne :

$$|\mathbf{PNkB}| \leq 2^{(2.|\Sigma|+1)^{k+1}} \text{ et en particulier } |\mathbf{PNSF}| \leq 2^{(2.|\Sigma|+1)^2}.$$

1.4.4.2 Sous-classes à caractérisation structurelle

Une restriction **C** de la classe **PN** est dite *structurelle* si elle ne concerne pas le marquage initial du réseau. On dira alors indifféremment que $\mathcal{N}$ appartient à la classe **C** ou que $(\mathcal{N}, m_0)$ appartient à **C**.

Définition 1.4.22 (Sous-classes structurelles de PN). Soit $\mathcal{N}$ un réseau de Petri. Nous définissons quelques sous classes de réseaux de Petri indépendantes du marquages initial.

- **PNSB** : la classe des *réseaux de Petri structurellement bornés* ; $\mathcal{N}$ est structurellement borné si et seulement si pour tout marquage initial m_0, il existe une borne b telle que pour tout marquage accessible m depuis m_0 et pour toute place p, $m(p) \leq b$.
- **PNO** : la classe des *réseaux de Petri ordinaires* ; $\mathcal{N}$ est ordinaire si et seulement si tous ses arcs ont un poids égal à 1.
- **PNGM** : la classe des *graphes d'événements* (*marked graphs* en anglais) ; $\mathcal{N}$ est un graphe marqué si et seulement si $\mathcal{N}$ est un réseau ordinaire et chaque place a exactement une transition en entrée et une transition en sortie : $|{}^\bullet p| = |p^\bullet| = 1$.

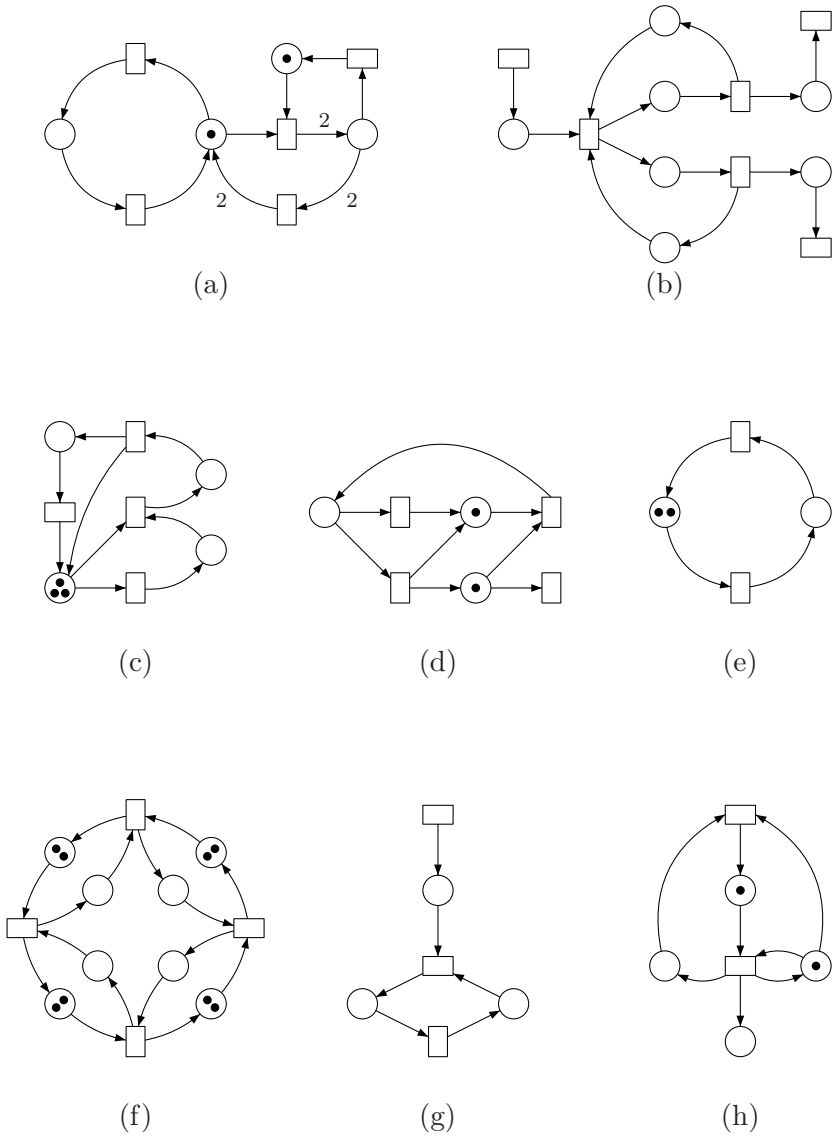

FIG. 1.8 – Exemples de réseaux de Petri de différentes sous-classes.

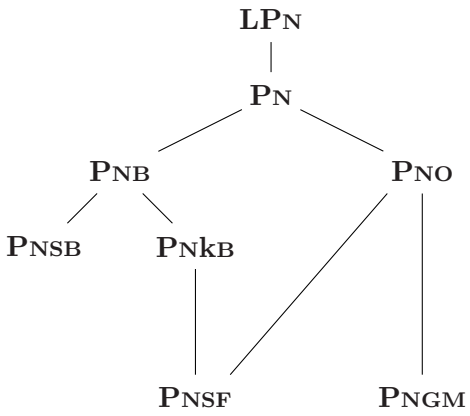

FIG. 1.9 – Hiérarchie d'expressivité des classes de réseaux de Petri.

1.4.4.3 Hiérarchie des sous-classes

Exemple 1.4.23. *Parmi les réseaux présentés dans la Figure 1.8 :*
- *le réseau* (a) *est dans* **PNB** *et plus précisément dans* **PN2B**, *mais pas dans* **PNSB**,
- *le réseau* (b) *est dans* **PNO**, *mais ni dans* **PNB**, *ni dans* **PNGM**,
- *le réseau* (c) *est dans* **PNO** *et* **PNSB** *mais pas dans* **PNGM** *ni* **PNSF**,
- *le réseau* (d) *est dans* **PNSF** *et* **PNSB** *mais pas dans* **PNGM**,
- *le réseau* (e) *est dans* **PNSF**, **PNSB** *et* **PNGM**,
- *le réseau* (f) *est dans* **PNGM** *et* **PNSB** *mais pas dans* **PNSF**,
- *le réseau* (g) *est dans* **PNGM** *mais pas dans* **PNSB**
- *le réseau* (h) *est dans* **PNSF** *mais pas dans* **PNSB**

Nous avons déjà vu que $\mathbf{PN} \subsetneq \mathbf{LPN}$. Par définition, nous avons également les inclusions de classes suivantes :

$$\mathbf{PNSF} \subset \mathbf{PNkB} \subset \mathbf{PNB}, \mathbf{PNSB} \subset \mathbf{PNB}, \mathbf{PNSF} \subset \mathbf{PNO}, \mathbf{PNGM} \subset \mathbf{PNO}.$$

Les réseaux de l'Exemple 1.4.23 nous permettent d'affirmer que ces inclusions sont strictes. Nous obtenons alors la hiérarchie présentée sur la Figure 1.9.

1.5 La synthèse de réseaux de Petri, méthode des régions

Nous présentons ici de manière succincte la méthode de synthèse de réseaux de Petri décrite dans [Dar04]. Nous détaillons les méthodes et les résultats concer-

nant la synthèse de réseaux *à partir d'un langage*. En effet, les logiques considérées dans ce document ne font pas de distinction entre deux modèles qui possèdent le même langage (voir la notion de bisimulation, Section 2.1.2, page 34) ; les méthodes de synthèse à isomorphisme près sont alors inadaptées à la synthèse à partir de spécifications logiques.

Le problème de synthèse est alors ici, pour une classe $\mathbf{C}$ de langages donnée, de décider uniformément, à partir d'un langage $L \in \mathbf{C}$ clos par préfixe, s'il existe un réseau initialisé $(\mathcal{N}, m_0) \in \mathbf{PN}$ tel que $\mathcal{L}(\mathcal{N}, m_0) = L$ et, dans l'affirmative, de construire un tel réseau.

1.5.1 Procédure de synthèse

La méthode présentée ici utilise la représentation vectorielle des réseaux de Petri et la décomposition de la Proposition 1.4.19. Fixons l'alphabet $\Sigma = \{a_1, \ldots, a_n\}$.

Définition 1.5.1 (Réseau atomique). Un réseau est *atomique* lorsqu'il est composé d'une unique place.

Définition 1.5.2 (Régions). Soit un mot $u \in \Sigma^*$, un réseau atomique $(\mathcal{N}, m_0) = \{p\}$ est *une région* de u si $u \in \mathcal{L}(\mathcal{N}, m_0)$. Soit $L \subseteq \Sigma^*$, un réseau atomique $(\mathcal{N}, m_0) = \{p\}$ est *une région* de L si $(\mathcal{N}, m_0)$ est une région de chaque mot $u \in L$.

Proposition 1.5.3 ([Dar04]). *Le langage L est le langage d'un réseau de Petri si et seulement si l'ensemble des régions de L contient un sous-ensemble fini $\{p_1, \ldots, p_m\}$ tel que pour tout $a \in \Sigma$ et pour tout $u \in L$, si $u.a \notin L$, alors il existe i tel que $\{p_i\}$ n'est pas une région de $u.a$. Lorsque cette condition est satisfaite, $L = \mathcal{L}(\mathcal{N}, m_0)$ avec $(\mathcal{N}, m_0) = \{p_1, \ldots, p_m\}$.*

À partir de la Proposition 1.5.3, il est possible de définir une la procédure de synthèse suivante :

Procédure 1.5.4 (Procédure de synthèse). La procédure de synthèse pour une classe $\mathbf{C}$ de langages se décompose en deux étapes :

(i) calculer une représentation effective de l'ensemble des régions d'un langage L, sachant que cet ensemble est toujours infini[5].

[5]Tous les vecteurs de places du type $\langle n, 0, \ldots, 0\rangle$ pour $n \in \mathbb{N}$ sont des régions de tout langage.

(ii) décider si un sous-ensemble fini de ces régions suffit à rejeter tous les mots minimaux (pour l'ordre préfixe) de $\Sigma^* \setminus L$ sachant que ces mots peuvent former un ensemble infini.

Afin de répondre à l'étape (i), il est nécessaire de restreindre la classe **C**. La restriction proposée dans [Dar04] consiste à imposer la semi-linéarité, pour tout langage $L \in \mathbf{C}$, de l'image commutative $[L]$. Une deuxième restriction plus forte est ajoutée : les *dérivations à droite* $L_{/a_i}$, avec $a_i \in \Sigma$ et $L_{/a_i} = \{u \in \Sigma^* \mid u.a_i \in L\}$, sont telles que pour tout $a_i \in \Sigma$, $[L_{/a_i}]$ est semi-linéraire.

Soit $u.a$ un mot de L, avec $u \in \Sigma^*$ et $a \in \Sigma$. D'après les relations de tir (1.1) et (1.2) page 21, pour un réseau atomique $(\mathcal{N}, m_0) = \{p\}$, sous la condition $m_0[u\rangle m$, la relation $m_0[u.a\rangle$ s'écrit :

$$m_0(p) + \sum_{b \in \Sigma} |u|_b \times <p, b> \quad \geq \quad <p, {}^\bullet a> \tag{1.4}$$

Rappelons qu'une place p est un vecteur :

$$p = \langle m_0(p), <p, {}^\bullet a_1>, \ldots, <p, {}^\bullet a_n>, <p, a_1^\bullet>, \ldots, <p, a_n^\bullet> \rangle$$

Toute place p peut être identifiée à un vecteur $\mathbf{x}$ de dimension $2.n + 1$ sur $\mathbb{N}$. Notons

$$\mathbf{x} = \langle x_0, \ldots, x_n, x_{n+1}, \ldots x_{2n+1} \rangle$$

Le vecteur $\mathbf{x}$ est une région de L si et seulement si pour tout mot $u.a_j \in L$

$$x_0 + \sum_{i \in [1,n]} |u|_{a_i} \times (x_{n+i} - x_i) \quad \geq \quad x_j \tag{1.5}$$

Ce résultat provient de l'Inéquation (1.4) et du fait que L est clos par préfixe. En effet, si l'Inégalité (1.5) est vérifiée pour tout $u.a_j \in L$, alors pour tout $v \in L$, nous avons $m_0[v'\rangle$ pour tout préfixe v' de v (puisque $v' \in L$) et par conséquent $m_0[v\rangle$.

En utilisant l'hypothèse que toutes les dérivations à droite $L_{/a_j}$ ont une image commutative semi-linéaire, nous pouvons exprimer :

$$[L_{/a_j}] = \bigcup_{d=1}^{D} (\mathbf{e}_d + (\mathbf{F}_d)^*),$$

avec $D \in \mathbb{N}$, $\mathbf{e}_d$ un vecteur de $\mathbb{Z}^n$, $\mathbf{F}_d$ un sous ensemble fini de vecteurs de $\mathbb{Z}^n$ et $(\mathbf{F}_d)^*$ est le plus petit sous monoïde de $\mathbb{N}^n$ contenant $\mathbf{F}_d$ (voir Définition 1.3.3). Sous cette notation, l'ensemble *fini* d'inéquations

$$\sum_{i\in[1,n]} \mathbf{e}_d[i] \times (x_{n+i} - x_i) \quad \geq \quad x_j - x_0 \tag{1.6}$$

$$\sum_{i\in[1,n]} \mathbf{f}[i] \times (x_{n+i} - x_i) \quad \geq \quad 0 \tag{1.7}$$

avec $\mathbf{f}$ parcourant $\mathbf{F}_d$, suffit à exprimer que $\mathbf{x}$ est une région de $L_{/a_j}$. Les Inéquations (1.6) et (1.7) sont en fait la reformulation de l'Inéquation (1.5) exprimée pour tout $u.a_j$ avec $u \in L_{/a_j}$.
Puisque l'ensemble des Inéquations (1.6) et (1.7) est fini, et que $L = \bigcup_{j\in[1,n]} L_{/a_j}.a_j$, alors nous avons un système fini d'inéquations.

Soit $\mathcal{H}$ le système composé de l'ensemble des Inéquations (1.6) et (1.7) pour tout $[L_{/a_j}]$, augmenté des contraintes $x_k \geq 0$ pour tout $k \in [0, 2n]$. En plongeant ce système dans l'ensemble $\mathbb{Q}$ des rationnels, les solution de $\mathcal{H}$ dans $\mathbb{Q}^{2n+1}$ forment un cône rationnel muni d'un ensemble fini de générateurs $\mathbf{x}_1, \ldots, \mathbf{x}_m$ (voir [Sch86]). Un vecteur $\mathbf{x}$ est alors solution de $\mathcal{H}$ si et seulement si $\mathbf{x} = \Sigma_{i\in[1,m]} q_l \mathbf{x}_l$ où les coefficients q_l sont des rationnels positifs. De plus, il est possible de calculer un ensemble minimal de générateurs $\mathbf{x}_1, \ldots, \mathbf{x}_m$ en utilisant l'algorithme de Chernikova [Che65] ; ces vecteurs forment un ensemble de *rayons extrémaux* du cône de solutions de $\mathcal{H}$. Nous supposerons pour la suite que les $\mathbf{x}_l$ sont des vecteurs de $\mathbb{N}^{2n+1}$ (leurs composantes sont positives, et il est possible, sans perdre de généralité, de les multiplier par un rationnel positif q_l pour les ramener dans $\mathbb{N}^{2n+1}$). Nous avons alors une représentation finie de l'ensemble des régions de L :

$$\mathbf{x} \in \mathbb{N}^{2n+1} \text{ est une région de } L \text{ ssi } \mathbf{x} = \Sigma_{i\in[1,m]} q_l \mathbf{x}_l$$

où les q_l désignent des rationnels positifs.

Considérons le réseau $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ composé des rayons extrémaux de $\mathcal{H}$, nous le nommerons $(\mathcal{N}^L, m_0^L)$ dans la suite.

Proposition 1.5.5 (Darondeau [Dar04]). *Le langage $\mathcal{L}(\mathcal{N}^L, m_0^L)$ est le plus petit langage de réseau contenant L.*

L'étape (ii) de la Procédure 1.5.4 consiste à décider de l'égalité $\mathcal{L}(\mathcal{N}^L, m_0^L) = L$. La décision est donnée par la proposition suivante :

Proposition 1.5.6 (Darondeau [Dar04]). *Lorsque* $[L \setminus L_{/a_i}]$ *est semi-linéaire pour tout* $i \in [1, n]$*, l'égalité* $\mathcal{L}(\mathcal{N}^L, m_0^L) = L$ *est décidable.*

Les hypothèses qui sont la semi-linéarité de $[L_{/a_i}]$ et de $[L \setminus L_{/a_i}]$ sont vérifiées pour deux classes particulières de langages : les langages rationnels et les langages context-free déterministes. D'où

Théorème 1.5.7 (Darondeau [Dar04]). *La synthèse de réseaux de Petri est décidable pour les langages rationnels et les langages context-free déterministes.*

1.5.2 Intervalles de langages

Nous donnons enfin le résultat qui constitue le point de départ de cette thèse : la satisfiabilité des intervalles de langages sur la classe des réseaux de Petri.

Théorème 1.5.8. *Le problème* $\mathbf{Sat}(\textsc{Int}_{\Sigma},\textsc{Pn})$ *est décidable.*

Ce dernier théorème est une conséquence de la partie consacrée au contrôle dans [Dar04]. Il repose sur la Proposition 1.5.5 et sur le fait qu'il est possible de décider $\mathcal{L}(\mathcal{N}, m_0) \subseteq L'$ pour un langage rationnel clos par préfixe L' ; cette décidabilité est une conséquence directe du problème d'atteignabilité dans les réseaux de Petri, lui même décidable [May84].

Les intervalles de langages constituent un système de spécifications simple pour lequel la satisfiabilité est décidable. La méthode de synthèse repose sur la donnée d'un sous-langage L du réseau à synthétiser, sur la possibilité de construire le réseau possédant le plus petit langage de réseau contenant L et sur la possibilité de vérifier $\mathcal{L}(\mathcal{N}, m_0) \subseteq L'$ pour le réseau $(\mathcal{N}, m_0)$ ainsi construit.
Nous montrons dans le Chapitre 3 qu'il est possible d'étendre ce résultat à certaines spécifications qui correspondent à la donnée d'un ensemble infini d'intervalles de langages.

Chapitre 2

Mu-calcul et réseaux de Petri

Sommaire

L'objet de ce chapitre est l'étude des problèmes **Sat**(**S**,**C**) et **Marq**(**S**,**C**) lorsque **S** est le Mu-Calcul (noté $\mathbf{L}_\mu$) et $\mathbf{C} = \mathbf{PN}$. Dans un premier temps, nous introduisons le Mu-Calcul doté d'une sémantique sur les processus. Nous montrons ensuite que le problème **Sat**($\mathbf{L}_\mu$,**PN**) est indécidable en réduisant à celui-ci un problème concernant les machines à compteurs. Nous montrons, avec une méthode similaire, que le problème **Marq**($\mathbf{L}_\mu$,**C**) est indécidable lorsque **C** est l'une des classe suivante : **LPN**, **PN**, **PNB**, **PNSB**, **PNO**, ou **PNGM**. Nous proposons enfin une sémantique pour $\mathbf{L}_\mu$ orientée vers les langages.

2.1 Le Mu-Calcul

Les logiques temporelles ont été introduites pour la spécification, l'analyse et la vérification de systèmes dits *réactifs* [MP92b]. Parmi les logiques dites *du temps arborescent* (qui énoncent des propriétés sur l'ensemble des exécutions possibles d'un système), et par opposition aux logiques *du temps linéaire* (qui énoncent des propriétés sur une exécution possible d'un système), nous nous intéressons au Mu-Calcul qui est la plus expressive ; en particulier, $\mathbf{L}_\mu$ est strictement plus expressive que $\mathbf{CTL}^*$. L'origine du Mu-Calcul dans la littérature est multiple, une référence principale étant l'introduction du Mu-Calcul modal par Kozen dans [Koz83].

Une logique temporelle peut être *propositionnelle* où *modale* ; dans le premier cas, le système à étudier est "décoré" par des propositions sur ses *états*, sur lesquelles s'appuient les énoncés logiques ; dans le second cas, les énoncés logiques s'appuient sur les étiquettes des *transitions* du système[1]. L'introduction de propositions sur les réseaux de Petri est malaisée ; en effet, il est difficile de poser des propositions sur les états du processus associé à un réseau de Petri. Dans la littérature, les propositions le plus souvent utilisée sont alors des propositions en rapport avec les comportements possibles du réseau (telle action est active), ou avec le contenu des places (telle place est vide). Ces propositions n'apportent en général aucune expressivité : il est possible de les simuler par des énoncés en logique modale. Nous choisissons donc de ne présenter ici que le Mu-Calcul modal, dépourvu de propositions.

2.1.1 Syntaxe et sémantique du Mu-Calcul

Dans cette partie, $\Sigma = \{a, b, \ldots\}$ est un alphabet fini et $Var = \{X, X_1, X_2, \ldots\}$ est un ensemble fini de variables.

[1] Une logique peut également être modale et propositionnelle, bien que cela n'apporte pas de gain d'expressivité.

2.1.1.1 Syntaxe du Mu-Calcul

Définition 2.1.1 (Syntaxe du Mu-Calcul). L'ensemble des formules du Mu-Calcul est noté $\mathbf{L}_\mu$ et est défini par la grammaire suivante :

$$(\mathbf{L}_\mu \ni) \quad \beta_1, \beta_2 ::= \texttt{true} \mid X \mid \langle a \rangle \beta_1 \mid \neg \beta_1 \mid \beta_1 \vee \beta_2 \mid \mu X.\beta_1(X)$$

où $a \in \Sigma$ et, pour assurer l'existence de points fixes, on requiert que la variable X soit sous la portée d'un nombre pair de symboles de négations $\neg$ dans $\beta_1(X)$ pour toute formule $\mu X.\beta_1(X)$. On requiert également que toute occurrence d'une variable X soit *gardée*, c'est à dire sous la portée d'un opérateur $\langle a \rangle$.

Nous utilisons les notations usuelles suivantes :

$$\begin{array}{rcl@{\qquad}rcl}
\texttt{false} & \stackrel{\text{def}}{=} & \neg\texttt{true} & \rightarrow^a & \stackrel{\text{def}}{=} & \langle a \rangle \texttt{true} \\
[a]\beta_1 & \stackrel{\text{def}}{=} & \neg\langle a \rangle(\neg\beta_1) & \not\rightarrow^a & \stackrel{\text{def}}{=} & [a]\texttt{false} \\
\beta_1 \wedge \beta_2 & \stackrel{\text{def}}{=} & \neg(\neg\beta_1 \vee \neg\beta_2) & \nu X.\beta_1(X) & \stackrel{\text{def}}{=} & \neg\mu X.\neg\beta_1(\neg X) \\
\beta_1 \Rightarrow \beta_2 & \stackrel{\text{def}}{=} & \neg\beta_1 \vee \beta_2 & & &
\end{array}$$

Dans une sentence $\mu X.\beta(X)$ (resp. $\nu X.\beta(X)$), l'expression $\mu.X$ (resp. $\nu.X$.) est nommée *opérateur liant* (*binding operator* en anglais) de la variable X ; on le note θX et on dit qu'il est de *type* μ (resp. ν).

Une *sentence* est une formule du Mu-Calcul sans variable libre : chaque variable X est sous la portée d'un opérateur $\mu.X$ ou $\nu.X$.

2.1.1.2 Sémantique sur les processus

L'interprétation d'une formule du Mu-Calcul sur un processus est le sous-ensemble des états du processus qui satisfont la formule relativement à une valuation donnée *val* des variables libres de la formule.

Définition 2.1.2 (Sémantique du Mu-Calcul). Une formule $\beta \in \mathbf{L}_\mu$ est interprétée sur un processus $\mathcal{P} =< S, s_0, t >$ relativement à une valuation $val : Var \rightarrow 2^S$ par l'ensemble $[\![\alpha]\!]^{val}_{\mathcal{P}} \subseteq S$ défini inductivement par :

$$\begin{array}{l}
[\![\texttt{true}]\!]^{[val]}_{\mathcal{P}} = S \\
[\![X]\!]^{[val]}_{\mathcal{P}} = val(X) \\
[\![\neg\beta]\!]^{[val]}_{\mathcal{P}} = S \setminus [\![\beta]\!]^{[val]}_{\mathcal{P}} \\
[\![\beta_1 \vee \beta_2]\!]^{[val]}_{\mathcal{P}} = [\![\beta_1]\!]^{val}_{\mathcal{P}} \cup [\![\beta_2]\!]^{val}_{\mathcal{P}} \\
[\![\langle a \rangle \beta]\!]^{val}_{\mathcal{P}} = \{ s \in S \mid \exists s' : t(s,a) = s' \text{ et } s' \in [\![\beta]\!]^{val}_{\mathcal{P}} \} \\
[\![\mu X.\beta(X)]\!]^{val}_{\mathcal{P}} = \bigcap \{ V \subseteq S \mid [\![\beta]\!]^{val(V/X)}_{\mathcal{P}} \subseteq V \}
\end{array}$$

où $val(V/X) : Var \to 2^S$ est la valuation $val(V/X)(X') = V(X')$ pour tout variable $X' \in Var$ telle que $X' \neq X$ et $val(V/X)(X) = V$.

L'interprétation $[\![\mu X.\beta(X)]\!]_{\mathcal{P}}^{val}$ (respectivement $[\![\nu X.\beta(X)]\!]_{\mathcal{P}}^{val}$) est alors le plus petit (resp. plus grand) point fixe de la fonction de 2^S dans 2^S qui associe à U l'ensemble $[\![\beta(X)]\!]_{\mathcal{P}}^{val(U/X)}$.

Remarquons que la sémantique des sentences de $\mathbf{L}_\mu$ est indépendante de la valuation val. On note alors $[\![\beta]\!]_{\mathcal{P}}$ pour l'interprétation d'une sentence β relativement à n'importe quelle valuation. On dit que le processus $\mathcal{P}$ *satisfait* la sentence β (notation $\mathcal{P} \models \beta$) si l'état initial s_0 de $\mathcal{P}$ appartient à $[\![\beta]\!]_{\mathcal{P}}$.

2.1.2 Mu-Calcul et Bisimulation

Définition 2.1.3 (Bisimulation [Par81]). Soit $\mathcal{P}_1 = \langle S_1, s_0^1, t_1 \rangle$ et $\mathcal{P}_2 = \langle S_2, s_0^2, t_2 \rangle$ deux processus sur l'alphabet Σ, une *bisimulation* entre $\mathcal{P}_1$ et $\mathcal{P}_2$ est une relation binaire $\rho \subseteq S_1 \times S_2$ telle que $(s_1, s_2) \in \rho$ et

- pour tout $a \in \Sigma$ et pour toute transition $s_1 \xrightarrow{a} s_1'$, il existe un état $s_2' \in P_2$ tel que $s_2 \xrightarrow{a} s_2'$ et $(s_1', s_2') \in \rho$,
- et vice-versa, pour tout $a \in \Sigma$ et pour toute transition $s_2 \xrightarrow{a} s_2'$, il existe un état $s_1' \in P_1$ tel que $s_1 \xrightarrow{a} s_1'$ et $(s_1', s_2') \in \rho$.

On dit que les états $s_1 \in S_1$ et $s_2 \in S_2$ sont *bisimilaires* lorsqu'il existe une bisimulation ρ entre $\mathcal{P}_1$ et $\mathcal{P}_2$ telle que $(s_1, s_2) \in \rho$. On dit que les processus $\mathcal{P}_1$ et $\mathcal{P}_2$ sont bisimilaires lorsque s_0^1 et s_0^2 sont bisimilaires.
La bisimulation est une relation d'équivalence sur les processus.

Proposition 2.1.4 (Mu-Calcul et processus bisimilaires [JW96]). *Soient $\mathcal{P}_1$ et $\mathcal{P}_2$ deux processus bisimilaires. Pour toute sentence β de $\mathbf{L}_\mu$, $\mathcal{P}_1 \models \beta$ si et seulement si $\mathcal{P}_2 \models \beta$.*

2.1.3 Expressivité du Mu-Calcul

Le Mu-Calcul est une logique temporelle particulièrement expressive ; il est par exemple plus expressif que **CTL** ou **CTL***. En particulier [JW96] montre que les propriétés que l'on peut décrire en Mu-Calcul sont exactement les propriétés qui sont exprimables en logique Monadique du Second Ordre (logique **MSO**) et qui ne distinguent pas deux modèles bisimilaires.

Exemple 2.1.5. *On montre quelques propriétés exprimables dans* $\mathbf{L}_\mu$ *:*

- *un processus satisfait la sentence* $\mu X. \bigvee_{a\in\Sigma}\langle a\rangle X \vee \not\rightarrow^b$ *si il existe un chemin fini du processus atteignant un état ne possédant pas de transition sortante étiquetée par b ;*
- *un processus satisfait la sentence* $\mu X. \bigwedge_{a\in\Sigma}[a]X \vee \langle b\rangle$ *si tout chemin du processus passe par un état possédant une transition sortante étiquetée par b, ou peut être prolongé pour atteindre un état possédant une transition sortante étiquetée par b ;*
- *un processus satisfait la sentence* $\nu X. \bigvee_{a\in\Sigma}[a]X \wedge \langle b\rangle$ *si tout chemin du processus ne passe que par des états possédant une transition sortante étiquetée par b ;*
- *un processus satisfait la sentence* $\nu X.\mu Y. \bigvee_{a\in\Sigma}(\langle a\rangle(X \wedge (Y \vee \langle b\rangle)$ *s'il existe un chemin infini du processus passant infiniment souvent par des états possédant une transition sortante étiquetée par b ;*

Nous définissons un fragment particulier de $\mathbf{L}_\mu$: le *Mu-Calcul sans alternation* (*alternation-free Mu-Calculus* en anglais), noté $\mathbf{afL}_\mu$: une sentence β est sans alternation lorsque pour toute variable X de β, l'opérateur liant θX a le même type que tout opérateur liant sur le chemin syntaxique de β entre X et θX.

Le fragment $\mathbf{afL}_\mu$ est strictement moins expressif que $\mathbf{L}_\mu$; ce fragment reste cependant toujours strictement plus expressif que $\mathbf{CTL}^*$. Le fragment $\mathbf{afL}_\mu$ présente des avantages de complexité puisque son problème de Satisfiabilité est polynomial.

2.1.4 Systèmes d'équations du Mu-Calcul

Nous donnons une version restreinte du concept de systèmes d'équations de $\mathbf{L}_\mu$ qui nous permet d'exprimer de manière concise certaines sentences particulières de $\mathbf{L}_\mu$. Pour plus de détails le lecteur pourra se référer à [AN01]. Soit n un entier et $\phi_1, \ldots \phi_n$ un ensemble de formules de $\mathbf{L}_\mu$ avec $X_1, \ldots X_n$ comme variables libres. Les systèmes que l'on considère sont de la forme :

$$\begin{array}{rcl} X_1 & = & \phi_1(X_1, \ldots, X_n) \\ \vdots & & \vdots \\ X_n & = & \phi_n(X_1, \ldots, X_n) \end{array}$$

Nous interprétons ici un tel système, pour un processus $\mathcal{P} =< S, s_0, t >$ comme un sous ensemble de $(2^S)^n$; une solution est alors un vecteur $< V_1, \ldots, V_n >$ de $(2^S)^n$. Parmi l'ensemble des solutions, nous considérons uniquement la plus petite

(au sens de l'inclusion) ou la plus grande ; le système peut alors être exprimé comme un point fixe, avec $\theta \in \{\nu, \mu\}$, de la manière suivante :

$$< V_1, \ldots, V_n >= [\![\, \theta < X_1, \ldots, X_n > . < \phi_1, \ldots, \phi_n > (< X_1, \ldots, X_n >) \,]\!]_{\mathcal{P}}$$

Ce système peut être résolu de manière symbolique, indépendemment d'une quelconque interprétation, en utilisant le principe d'élimination de Gauss (qui est une conséquence du principe généralisé de Bekič). La résolution symbolique du système est la construction de n sentences de $\mathbf{L}_\mu$, $\Psi_1, \ldots, \Psi_n$, telles que pour tout processus $\mathcal{P} =< S, s_0, t >$ et pour tout $1 \leq i \leq n$, $[\![\, \Psi_i \,]\!]_{\mathcal{P}}$ est la i$^{\text{e}}$ composante de la solution du système interprété sur $\mathcal{P}$, c.-à-d. $[\![\, \Psi_i \,]\!]_{\mathcal{P}} = V_i$. Nous notons :

$$< \Psi_1, \ldots, \Psi_n >= \theta < X_1, \ldots X_n > . < \phi_1, \ldots, \phi_n > (< X_1, \ldots, X_n >)$$

Proposition 2.1.6 (Principe d'élimination de Gauss). *Soit ϕ_1 et ϕ_2 deux formules de $\mathbf{L}_\mu$, avec X_1 et X_2 comme variables libres ; soit $\psi_1(X_2) = \theta X_1.\phi_1(X_1, X_2)$, avec $\theta \in \{\nu, \mu\}$ et soit*

$$< \Psi_1, \Psi_2 >= \theta < X_1, X_2 > . < \phi_1, \phi_2 > (< X_1, X_2 >).$$

Alors $\Psi_2 = \theta X_2.\phi_2(\psi_1(X_2), X_2)$ et $\Psi_1 = \psi_1(\Psi_2)$.

Cette proposition permet de calculer récursivement

$$< \Psi_1, \ldots, \Psi_n >= \theta < X_1, \ldots, X_n > . < \phi_1, \ldots, \phi_n > (< X_1, \ldots, X_n >)$$

en utilisant l'algorithme suivant :

Algorithme 2.1.7 (Calcul des solutions symboliques). ˜

1. *calculer $\psi_1(X_2, \ldots X_n) = \theta X_1.\phi_1(X_1, \ldots, X_n)$,*
2. *calculer*

   $$< \Psi_2, \ldots, \Psi_n >= \theta < X_2, \ldots X_n > . < \phi_2, \ldots \phi_n > (< \psi_1(X_2, \ldots X_n), X_2 \ldots, X_n >)$$

   suivant la même méthode,
3. *calculer $\Psi_1 = \psi_1(\Psi_2, \ldots, \Psi_n)$.*

TAB. 2.1 – Résultats principaux du Mu-Calcul sur les réseaux de Petri

	Model-Checking	**Satisfiabilité**
LPN	Indécidable [Esp97]	Décidable [Rab72, Eme85]
PN	Indécidable [Esp97]	**?**

2.2 Mu-Calcul et réseaux de Petri

2.2.1 Notations

Soit Σ un alphabet. Nous interprétons les formules de $\mathbf{L}_\mu$ sur les graphes de marquages de réseaux de Petri, qui sont considérés ici comme des processus. Soit $(\mathcal{N}, m_0)$ un réseau de Petri initialisé, on dit que $(\mathcal{N}, m_0)$ *satisfait* une formule ϕ de $\mathbf{L}_\mu$ lorsque $\mathcal{G}(\mathcal{N}, m_0) \models \phi$; on adoptera pour la suite la notation $(\mathcal{N}, m_0) \models \phi$ pour $\mathcal{G}(\mathcal{N}, m_0) \models \phi$.

On rappelle que lorsque $(\mathcal{N}, l, m_0)$ est un réseau de Petri étiqueté initialisé de **LPN**, les transitions de $\mathcal{G}(\mathcal{N}, l, m_0)$ sont étiquetés sur Σ par l. Lorsque $(\mathcal{N}, m_0)$ est un réseau de Petri non étiqueté de **PN**, les transitions de $\mathcal{G}(\mathcal{N}, m_0)$ sont les transitions de $\mathcal{N}$ qui sont confondues avec les éléments de Σ.

2.2.2 Model-Checking et Satisfiabilité

Nous rappelons les principaux résultats connus concernant les problèmes **MC**($\mathbf{L}_\mu$,**C**) et **Sat**($\mathbf{L}_\mu$,**C**) lorsque $\mathbf{C} \in \{\mathbf{LPN}, \mathbf{PN}\}$; le Tableau 2.1 résume ces résultats.

1. **MC**($\mathbf{L}_\mu$,**LPN**) : ce problème a été montré indécidable par J. Esparza [Esp97]. Ce résultat repose sur une réduction du problème de l'arrêt d'une machine de Minsky.

2. **Sat**($\mathbf{L}_\mu$,**LPN**), la décidabilité de ce problème est immédiate au regard des

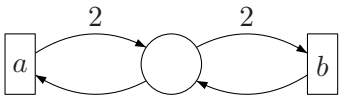

FIG. 2.1 – Une place du réseau $\mathcal{N}$.

deux faits suivants : Fait1) $\mathbf{L}_\mu$ a la propriété du modèle fini[2]. Fait2) La classe des processus finis est strictement incluse dans la classe des graphes des marquages de réseaux de Petri étiquetés (voir Proposition 1.4.14 page 19). Par conséquent le problème de satisfiabilité sur la classe des réseaux de Petri étiquetés n'est pas plus difficile que le problème de satisfiabilité sur la classe des processus finis qui est décidable [Rab72, Eme85].

3. **MC**($\mathbf{L}_\mu$,**PN**) : ce problème est très similaire à celui du model checking des réseaux de Petri étiquetés. Il a été montré indécidable par J. Esparza [Esp97].

4. **Sat**($\mathbf{L}_\mu$,**PN**) : à notre connaissance, ce sujet n'a pas encore été considéré. Nous montrons dans ce chapitre que ce problème est indécidable. La preuve complète est présentée dans la suite.

Le Mu-Calcul, lorsqu'il est interprété sur la classe des réseaux de Petri (non étiquetés) ne vérifie pas la propriété du modèle fini dans le sens où il existe des sentence de $\mathbf{L}_\mu$ dont toutes les solutions sont des réseaux dont le graphe des marquages est infini. L'exemple suivant illustre ce point.

Exemple 2.2.1. *Soit ϕ la sentence de $\mathbf{L}_\mu$ sur l'alphabet $\Sigma = \{a, b\}$ définie comme suit :*

$$\phi = \nu X.\langle a\rangle X \wedge \nu X.\langle a\rangle\langle b\rangle X \wedge \not\to^b$$

Un processus $\mathcal{P}$ satisfait la sentence ϕ si et seulement si :

(1) il existe un chemin infini étiqueté par a^ω dans $\mathcal{P}$,

(2) il existe un chemin infini étiqueté par $(a.b)^\omega$ dans $\mathcal{P}$,

(3) l'état initial de $\mathcal{P}$ ne possède pas de transition sortante étiquetée par b.

Cette sentence se "traduit" sur la classe des réseaux non-étiquetés par : un réseau $(\mathcal{N}, m_0)$ satisfait ϕ si et seulement si

[2]On rappelle que cette propriété énonce l'existence d'un modèle fini (processus fini) de toute formule satisfiable

(i) le langage de $\mathcal{N}$ contient le langage a^*

(ii) le langage de $\mathcal{N}$ contient le langage $(a.b)^*$

(iii) il existe une place p de $\mathcal{N}$ qui inactive la transition b au marquage m_0.

où les conditions (i), (ii) et (iii) se déduisent respectivement des conditions (1), (2) et (3).
De (i) nous déduisons pour toute place p de $\mathcal{N}$:

$$<p, a^\bullet> - <p, {}^\bullet a> \geq 0 \quad (2.1)$$
$$m_0(p) - <p, a^\bullet> \geq 0 \quad (2.2)$$

De (ii) nous déduisons pour tout place p de $\mathcal{N}$:

$$<p, a^\bullet> - <p, {}^\bullet a> + <p, b^\bullet> - <p, {}^\bullet b> \geq 0 \quad (2.3)$$
$$m_0(p) + <p, a^\bullet> - <p, {}^\bullet a> - <p, {}^\bullet b> \geq 0 \quad (2.4)$$

De (iii) nous déduisons l'existence d'une place p' telle que

$$m_0(p') - <p', {}^\bullet b> < 0 \quad (2.5)$$

cette place p' vérifiant les Inéquations (2.4) et (2.5), nous en déduisons

$$<p', a^\bullet> - <p', {}^\bullet a> > 0$$

Par conséquent $\mathcal{N}$ possède une place p' similaire à la place de la Figure 2.1 ; puisque a^* appartient au langage de $\mathcal{N}$, et que la transition a incrémente strictement le marquage de la place p', il en résulte que le graphe des marquages de $\mathcal{N}$ est infini (les marquages de la place p' peuvent être arbitrairement grands).
Le réseau composé de l'unique place qui est celle de la Figure 2.1 est un exemple de modèle de ϕ.

2.3 Machines à compteurs

Nous présentons dans cette section les *machines de Minsky* [Min67] également nommées *machines à compteurs*. Nous énonçons un problème indécidable concernant les machines à compteurs que nous utiliserons par la suite pour démontrer l'indédicabilité du problème $\mathbf{Sat}(\mathbf{L}_\mu, \mathbf{PN})$.

Définition 2.3.1. Une *machine à compteurs* est un tuple

$$(\{q_0, \ldots, q_{n+1}\}, \{c_0, \ldots, c_m\}, \{\delta_0, \ldots, \delta_n\})$$

où les c_i sont les *compteurs*, les q_k sont les *états* ; q_0 est *l'état initial* et q_{n+1} est l'unique *état final*. Pour tout $0 \leq k < n+1$, δ_k est la *règle de transition* associée à l'état q_k ; puisque q_{n+1} est final, il ne possède pas de règle de transition. Les états sont typés : chaque état est soit de type I soit de type II.

Un état est de type I si sa règle de transition s'exprime comme suit (pour $i \in [0, m]$ et $k \in [1, n]$) :

$$c_i := c_i + 1;\ \text{goto}\ q_k$$

ou il est de type II, auquel cas sa règle de transition s'exprime comme suit (pour $i \in [0, m]$ et $k, k' \in [1, n]$) :

$$\text{if } c_i = 0 \text{ then } (c_i := c_i + 1;\ \text{goto}\ q_k) \text{ else } (c_i := c_i - 1;\ \text{goto}\ q_{k'})$$

Une *configuration* J de M est un tuple $(q_k, j_0, \ldots, j_m)$ où q_k est un état et $j_0, \ldots, j_m$ sont des entiers positifs qui désignent les valeurs des compteurs. Une configuration initiale est une configuration de la forme $(q_0, j_0, \ldots, j_m)$. Une *exécution* de M pour une configuration J est une séquence de configurations dont le premier élément est J, et telle que tout couple de configurations consécutives $(q_k, j_0, \ldots j_m)$ et $(q'_k, j'_0, \ldots j'_m)$ dans le séquence vérifie : les valeurs $q'_k, j'_0, \ldots j'_m$ sont le résultat de l'application de la règle de transition δ_k lorsque les valeurs des compteurs $c_0, \ldots, c_m$ sont respectivement $j_0, \ldots j_m$. Remarquons qu'une machine M a une unique exécution pour une configuration donnée J puisqu'à chaque état est associé une unique règle de transition déterministe.

On dit que M *termine* pour la configuration J si et seulement si son exécution contient une configuration de la forme $(q_{n+1}, j_0, \ldots, j_m)$ (rappelons que q_{n+1} est final) ; de manière équivalente, puisque q_{n+1} est l'unique état ne possédant pas de règle de transition, M termine si l'exécution de M pour J est finie.
Soit M une machine à compteurs, nous définissons le *langage* de M comme l'ensemble des configurations J telles que l'exécution de M termine pour J.

Remarque 2.3.2. La Définition 2.3.1 diffère légèrement de celle de [Min67] où la définition de la règle de transition d'un état q_l de type II est :

$$\text{if } c_i = 0 \text{ then goto } q_k \text{ else } (c_i := c_i - 1;\ \text{goto}\ q_{k'}) \tag{2.6}$$

Toutefois, les machines que nous présentons ici contiennent les machines de Minsky (celles de [Min67]) du point de vue du problème de terminaison : nous expliquons

ci-dessous comment transformer une machine de Minsky en une machine à compteurs au sens de la Définition 2.3.1 de telle manière que l'ensemble des configurations pour lesquelles les deux machines s'arrêtent coïncide. Nous transformons une machine de Minsky M_0 donnée en une machine à compteurs M_1 comme suit : M_1 est identique à M_0 à l'exception des transitions (2.6) de type II qui sont remplacées par les deux transitions suivantes :

$$\delta_l = \text{ if } c_i = 0 \text{ then } c_i := c_i + 1;\ \text{goto } q_{l'} \text{ else } (c_i := c_i - 1;\ \text{goto } q_{k'})$$

$$\delta_{l'} = \text{ if } c_i = 0 \text{ then } c_i := c_i + 1;\ \text{goto } q_{l'} \text{ else } (c_i := c_i - 1;\ \text{goto } q_k)$$

où l'état $q_{l'}$ est un nouvel état "intermédiaire" de type II.

Puisque l'état $q_{l'}$ est uniquement accessible depuis l'état q_l en incrémentant c_i, la première alternative du "if" de $\delta_{l'}$ n'est jamais exécutée. Les exécutions de M_1 et M_0 sont presque identiques : intuitivement, une exécution de M_0 se transforme en une exécution de M_1 en exécutant consécutivement les règles δ_l et $\delta_{l'}$ à la place de la règle originale (2.6). Réciproquement, par construction lors de chaque exécution de M_1, toute application de δ_l est immédiatement suivie d'une application de $\delta_{l'}$ avec $c_i \neq 0$; le remplacement de cette séquence de règles par la Règle (2.6) permet de retrouver l'exécution correspondante dans M_0. Finalement M_0 termine pour J si et seulement si M_1 termine pour J.

On considère le *problème de la vacuité du langage d'une machine à deux compteurs* qui s'énonce comme suit :

Problème 2.3.3 (Problème de la vacuité du langage d'une machine à deux compteurs). Soit M une machine à deux compteurs, existe-t-il une configuration initiale J^0 de M telle que M termine pour J^0 ?

Puisque les machines à deux compteurs permettent de simuler les machines de Turing [Min67] et que le problème de la vacuité du langage d'une machine de Turing est indécidable [HU90], le Problème 2.3.3 est indécidable. Par la Remarque 2.3.2, nous obtenons :

Théorème 2.3.4. *La vacuité d'une machine à deux compteurs (au sens de la Définition 2.3.1) est indécidable*

2.4 Indécidabilité du problème $\mathbf{Sat}(\mathbf{L}_\mu,\mathbf{P_N})$

Dans un souci de clarté, nous présentons la preuve dans le cas de réseaux de Petri purs. Nous donnons ensuite la méthode pour obtenir le cas des réseaux impurs.

2.4.1 Places de réseaux de Petri pour simuler des compteurs

Dans la suite nous utilisons des places de réseaux de Petri, en conjonction avec un alphabet spécialisé, pour simuler le comportement de compteurs. Similairement à un compteur, une place de réseaux de Petri permet de stocker un entier : la valeur $m(p)$. Nous donnons les définitions qui permettent d'utiliser une place de réseau de Petri comme un compteur et de tester si sa valeur est égale à zéro.

2.4.1.1 Des places en tant que compteurs

On suppose que l'alphabet Σ contient deux éléments particuliers c^+ et c^-. Ces transitions sont choisies pour représenter les mise-à jour du compteur c : la transition c^+ correspond à l'instruction $c := c+1$ et la transition c^- correspond à l'instruction $c := c-1$. L'opération qui permet de tester si la valeur du compteur c est égale à zéro est considérée dans la sous-Section 2.4.1.3.

On note Σ_0 pour l'ensemble $\Sigma \setminus \{c^+, c^-\}$, d'élément typique a.

Définition 2.4.1 (Place simulant un compteur). Une place p d'un réseau $\mathcal{N}$ *simule* le compteur c si et seulement si :

$$<p, c^+> > 0, \quad <p, c^-> = -<p, c^+> \quad \text{et} \quad <p, a> = 0, \forall a \in \Sigma_0$$

D'après cette définition, lorsque la place p simule le compteur c, la valeur $m(p)$ du compteur c est incrémentée (resp. décrémentée) d'un entier $\lambda = <p, c^+> = -<p, c^->$ fixé à chaque fois que la transition c^+ (resp. c^-) est tirée ; de plus la valeur $m(p)$ reste inchangée lors du tir de tout autre transition de Σ_0. Afin de raffiner la simulation du compteur c, nous donnons quelques techniques pour ramener ce facteur λ à 1.

Remarque 2.4.2. La Définition 2.4.1 concerne uniquement la structure du réseau, c.à-d. qu'aucun marquage particulier n'est imposé.

Définition 2.4.3 (Place normalisée simulant un compteur). Une place p qui simule le compteur c est dite *normalisée* si et seulement si $<p, c^+> = 1$.

Par définition, une place p normalisée qui simule c vérifie :

$$<p, c^+> = 1, \quad <p, c^-> = -1 \quad \text{et} \quad <p, a> = 0, \forall a \in \Sigma_0$$

Remarque 2.4.4. Un réseau peut posséder plusieurs copies de la place normalisée qui simule c, et ces places peuvent avoir des marquages différents le long des séquences de tir. Lorsque l'on s'abstient de considérer les marquages, toutes ces places sont structurellement identiques, nous notons p_c toute place normalisée qui simule c. Toutefois, comme nous le montrons plus loin, parmi toutes ces places, seule celle possédant le plus petit marquage initial peut inactiver une transition le long de n'importe quelle séquence de tir.

Le lemme suivant montre qu'il est toujours possible de remplacer une place qui simule le compteur c par sa "version normalisée" p_c sans modifier le comportement du réseau.

Lemme 2.4.5. *Soit $(\mathcal{N}, m_0)$ un réseau de Petri initialisé ; soit p une place de $\mathcal{N}$ qui simule le compteur c, et soit $\mathcal{N}'$ le réseau obtenu en remplaçant p par p_c. Il existe un marquage initial m'_0 pour $\mathcal{N}'$ tel que m'_0 et m_0 soient identiques sur l'ensemble des places héritées de $\mathcal{N}$ et tel que $\mathcal{G}(\mathcal{N}', m'_0)$ et $\mathcal{G}(\mathcal{N}, m_0)$ soient bisimilaires.*

Démonstration. Soit P l'ensemble des places de $\mathcal{N}$ et soit $P' = P \setminus \{p\}$. Définissons m'_0 par :

$$\begin{array}{rrcl} & m'_0|_{P'} & = & m_0|_{P'} \\ \text{et} & m'_0(p_c) & = & \lfloor m_0(p)/<p, c^+> \rfloor \end{array}$$

Soit ρ la relation binaire

$$\rho = \{(m, m') \mid m|_{P'} = m'|_{P'} \text{ et } m'(p_c) = \lfloor m(p)/<p, c^+> \rfloor\}$$

On montre que ρ est une bisimulation.
Puisque p simule c, pour toute séquence de tir $u \in \Sigma^*$, avec $u = a_1 \ldots . a_k$, telle que $m_0[u\rangle m$ dans $\mathcal{N}$, nous avons $m(p) = m_0(p) + \Sigma_{j\in[1,k]} <p, a_j>$. Puisque $<p_c, a> = 0$, pour tout $a \in \Sigma_0$, et puisque $<p, c^+> = -<p, c^->$ nous avons $m(p) = m_0(p) + \Delta_u.<p, c^+>$, avec la convention $\Delta_u = |u|_{c^+} - |u|_{c^-}$. Par conséquent $m'_0(p_c) + \Delta_u = \lfloor m(p)/<p, c^+> \rfloor$. Ce résultat est suffisant pour montrer que ρ est une bisimulation. □

2.4.1.2 Réseaux de compteurs

Dans cette partie nous considérons deux compteurs c_0 et c_1, puisque seuls deux compteurs sont nécessaires pour la preuve du résultat principal. Les définitions suivantes peuvent toutefois se généraliser à un nombre arbitraire de compteurs.

En conséquence, nous supposons que l'alphabet Σ est $\{c_0^+, c_0^-, c_1^+, c_1^-\}$.

Définition 2.4.6 (Réseau de compteurs). Un *réseau de compteurs* (sur Σ) est un réseau de Petri pur étiqueté où chaque place simule c_0 ou c_1. Un *réseau de compteur normalisé* est un réseau de compteurs où chaque place est normalisée et possédant au plus une place qui simule chaque compteur. Nous utilisons $\mathcal{N}'$ comme élément typique de la classe des réseaux de compteurs.

Définition 2.4.7. Soit $\mathcal{N}'$ un réseau de compteurs de marquage initial m'_0 et dont l'ensemble des places est $P = \{p_1, \ldots, p_k\}$. La *forme normale* de $(\mathcal{N}', m'_0)$ (à isomorphisme près) est le réseau initialisé $(\mathcal{C}, n_0)$, où $\mathcal{C}$ est un réseau de compteurs normalisé particulier et n_0 est son marquage initial, obtenus par la construction en deux étapes suivante :

Procédure de Normalisation

Étape 1 Pour tout $j \in [1, k]$, remplacer la place p_j par sa version normalisée $p''_j = p_{c_i}$ (où $i \in \{0, 1\}$ dépend du compteur que simule p_j) ; nous obtenons un réseau de compteurs $\mathcal{N}''$ dont l'ensemble des places est $P'' = \{p''_j \mid j \in [1, k]\}$. Pour tout $p''_j \in P''$, poser $m''_0(p''_j)$ comme étant $\lfloor m_0(p)/<p, c_i^+>\rfloor$.

Étape 2 Partitionner P'' en deux ensembles, éventuellement vides, P_0 et P_1, composés respectivement des places qui simulent c_0 et des places qui simulent c_1 ; définir le réseau $\mathcal{C}$ et son marquage initial n_0 par restriction de $(\mathcal{N}'', m''_0)$ à chaque place dont le marquage initial est minimal dans son ensemble d'appartenance (qui est P_0 ou P_1). Si cette place n'existe pas, $P_0 = \emptyset$ par exemple, alors $\mathcal{C}$ contient uniquement une place qui simule c_1 ; si les deux places n'existent pas alors $\mathcal{C}$ est le réseau vide.

Après normalisation, l'ensemble des places d'un réseau de compteurs en forme normale est un sous-ensemble de $\{p_{c_0}, p_{c_1}\}$. C'est un sous-ensemble strict lorsque l'une des places p_{c_i} est manquante parce que le réseau de compteurs en entrée de la Procédure de Normalisation ne possède pas de place simulant le compteur c_i.

Lemme 2.4.8. *Soit $(\mathcal{N}', m'_0)$ un réseau de compteurs initialisé ; soit $(\mathcal{C}, n_0)$ sa forme normale obtenue par la Procédure de Normalisation. Les graphes des marquages $\mathcal{G}(\mathcal{N}', m'_0)$ et $\mathcal{G}(\mathcal{C}, n_0)$ sont bisimilaires.*

Démonstration. Soit$(\mathcal{N}'', m''_0)$ le réseau initialisé obtenu par l'Étape 1 de la Procédure de Normalisation appliquée à $(\mathcal{N}', m'_0)$. Par application du Lemme 2.4.5 pour chacune des places de $\mathcal{N}'$, $\mathcal{G}(\mathcal{N}'', m''_0)$ et $\mathcal{G}(\mathcal{N}', m'_0)$ sont bisimilaires.

On note P'' l'ensemble des places de $\mathcal{N}''$ et P l'ensemble des places de $\mathcal{C}$. Nous montrons que la relation binaire ρ entre les marquages accessibles de $\mathcal{N}''$

et les marquages accessibles de $\mathcal{C}$, définie par $\rho = \{(m'', n) \mid n = m''|_P\}$ est une bisimulation : pour tout paire $(m'', n) \in \rho$, on suppose que la transition c_i^- (pour un certain $i \in \{0, 1\}$) est inactive au marquage m'' de $\mathcal{N}''$. Puisque, m'' est accessible, il existe $u \in \Sigma^*$ tel que $m_0''[u\rangle m''$. Il existe alors une place p'' dans P'' telle que $m_0''(p'') + |u|_{c_i^+} - |u|_{c_i^-} = 0$. Par construction de $\mathcal{C}$, il existe une place $p' \in P'$ qui simule le compteur c_i et telle que $m_0'(p') \leq m_0''(p'')$; ceci implique que la transition c_i^- est également inactive au marquage n de $\mathcal{C}$. Par contraposée, si $n[c_i^-\rangle n'$ alors $m''[c_i^-\rangle n''$, et de plus $n' = n''|_P$, ce qui implique $(n'', n') \in \rho$. La réciproque est triviale puisque $\mathcal{C}$ est une restriction de $\mathcal{N}''$. Les transitions autres que les c_i^- sont des cas triviaux. Finalement, ρ est une bisimulation. □

Une conséquence immédiate du Lemme 2.4.8 est qu'un réseau de compteurs et sa forme normale satisfont les même sentences de $\mathbf{L}_\mu$.

2.4.1.3 Test à zéro

Soit $\mathcal{N}$ un réseau de Petri et m un marquage de $\mathcal{N}$, nous définissons ici une sentence de $\mathbf{L}_\mu$ qui énonce, pour un certain $i \in \{0, 1\}$: "il existe une place p qui simule le compteur c_i dans $\mathcal{N}$ et cette place indique, au marquage m, que la valuation du compteur c_i est égale à zéro". Lorsque $\mathcal{N}$ possède une place p qui simule c_i, et telle que $m(p) - <p, c_i^-> < 0$ (la valuation de c_i est zéro), alors c_i^- est inactive au marquage m et la sentence $\not\rightarrow^{c_i^-}$ est satisfaite. Toutefois, cette sentence n'est pas suffisante puisque la transition c_i^- peut être inactivée par une autre place de $\mathcal{N}$ qui ne simule pas c_i.

Définition 2.4.9. La *sentence de test à zéro du compteur* c_i est notée Θ_i et est la sentence :

$$\Theta_i \quad \stackrel{\text{def}}{=} \quad \nu X. \stackrel{c_i^-}{\not\rightarrow} \wedge(\langle c_i^+\rangle\langle c_i^-\rangle X) \wedge (\langle c_{1-i}^+\rangle X) \wedge (\langle c_{1-i}^+\rangle\langle c_{1-i}^-\rangle X)$$

Nous montrons que le test à zéro est réellement effectué lorsqu'une place simulant le compteur c_i ne permet pas la transition c_i^-. Dans la suite, avec i valant 0 ou 1, nous notons U_i pour l'ensemble de mots $U_i = \{c_i^+.c_i^-,\ c_{1-i}^+\ ,\ c_{1-i}^+.c_{1-i}^-\}$. Remarquons que, par définition, lorsqu'une place p simule le compteur c_i, elle vérifie pour tout $u \in U_i$, $<p, u> = 0$.

Lemme 2.4.10. *Si* $(\mathcal{N}, m) \models \Theta_i$ *alors pour tout* $u \in U_i^*$, *il existe* m' *tel que* $m[u\rangle m'$ *et* $(\mathcal{N}, m') \models \Theta_i$.

Démonstration. Soit $\mathcal{M}$ l'ensemble des marquages accessibles de $(\mathcal{N}, m)$ (les sommets de $\mathcal{G}(\mathcal{N}, m)$) et soit

$$\beta \quad \stackrel{\text{def}}{=} \quad \stackrel{c_i^-}{\not\longrightarrow} \wedge(\langle c_i^+\rangle\langle c_i^-\rangle X) \wedge (\langle c_{1-i}^+\rangle X) \wedge (\langle c_{1-i}^+\rangle\langle c_{1-i}^-\rangle X)$$

Nous avons $\Theta_i = \nu X.\beta$. L'interprétation de β sur $\mathcal{G}(\mathcal{N}, m)$ relativement à la valuation *val* est :

$$\begin{aligned}\llbracket \beta \rrbracket^{val}_{\mathcal{N},m} = \quad & \{m'' \in \mathcal{M} \mid c_i^- \text{ est inactive au marquage } m''\} \\ & \cap \\ & \textstyle\bigcap_{u\in U_i}\{m'' \in \mathcal{M} \mid m''[u\rangle m' \text{ et } m' \in val(X)\}\end{aligned}$$

L'ensemble de marquages $V = \llbracket \Theta_i \rrbracket_{\mathcal{N},m}$ est un (plus grand) point-fixe, ce qui implique $V = \llbracket \beta \rrbracket^{val(V/X)}_{(\mathcal{N},m)}$. En particulier, si $m_1 \in V$ alors :

$$m_1 \in \bigcap_{u\in U_i} \{m'' \in \mathcal{M} \mid m''[u\rangle m' \text{ et } m' \in V\}$$

Par conséquent $m_1 \in V$ implique pour tout $u \in U_i$ l'existence d'un marquage m_2^u tel que $m_1[u\rangle m_2^u$ avec $m_2^u \in V$. Par induction sur les mots de U_i^*, pour tout $u \in U_i^*$, il existe m' tel que $m[u\rangle m'$ et $(\mathcal{N}, m') \models \Theta_i$. □

Lemme 2.4.11. *Si $(\mathcal{N}, m) \models \Theta_i$ alors pour toute place p de $\mathcal{N}$ et pour tout $u \in U_i$, $<p, u> \geq 0$*

Démonstration. Supposons qu'il existe une place p dans $\mathcal{N}$ et un mot $u \in U$ tels que $<p, u> < 0$. Il existe alors un entier n tel que $m(p) + <p, u^n> < 0$; par conséquent il n'existe pas de marquage m' tel que $m[u^n\rangle m'$, ce qui contredit le Lemme 2.4.10. □

Lemme 2.4.12. *Si $(\mathcal{N}, m) \models \Theta_i$ alors $\mathcal{N}$ possède une place p qui simule le compteur c_i et telle que :*

$$m(p) + <p, c_i^-> < 0$$

Démonstration. Soit $\{p_1, \ldots p_k\}$ l'ensemble des places de $\mathcal{N}$ qui ne simulent pas le compteur c_i. Nous montrons dans un premier temps qu'il existe $v \in U_i^*$ tel que pour tout $l \in [1, k]$, $m(p_l) + <p_l, v> + <p_l, c_i^-> \geq 0$.

Soit $l \in [1, k]$. Puisque p_l ne simule pas le compteur c_i alors soit $<p_l, c_i^+> > 0$ et l'une des égalités de ($<p, c_i^-> = -<p, c_i^+>$ et $<p, c_{1-i}^+> = <p, c_{1-i}^-> = 0$) est fausse, soit $<p_l, c_i^+> \leq 0$. Lorsque $<p_l, c_i^+> \leq 0$, puisque le Lemme 2.4.11 implique $<p_l, c_i^+.c_i^-> \geq 0$, nous obtenons $<p_l, c_i^-> \geq 0$.

Soit $\{A, B\}$ la partition de $\{p_1, \dots p_k\}$ telle que A est composé des places p_l telle que $<p_l, c_i^-> \geq 0$ et B est composé des places p_l telles que $<p_l, c_i^-> < 0$. Pour les places de A, quelque soit le mot $v \in U_i^*$ choisi, par le Lemme 2.4.11, nous avons déjà $m(p_l) + <p_l, v> + <p_l, c_i^-> \geq 0$. Nous nous préoccupons maintenant uniquement des places $p_l \in B$.

Pour toute place $p_l \in B$, puisque p_l ne simule pas c_i, trois cas sont possibles :

– soit $<p_l, c_i^+> \neq -<p_l, c_i^->$, alors par le Lemme 2.4.11 nous avons nécessairement $<p_l, c_i^+> > -<p_l, c_i^->$ ce qui implique

  (i) $<p_l, c_i^+.c_i^-> > 0$,

– soit $<p_l, c_i^+> = -<p_l, c_i^->$, alors

  (ii) soit $<p_l, c_{1-i}^+> > 0$,
  (iii) soit $<p_l, c_{1-i}^+> = 0$ (puisque par le Lemme 2.4.11 $<p_l, c_{1-i}^+> \geq 0$) et $<p_l, c_{1-i}^-> \neq 0$, alors $<p_l, c_{1-i}^+.c_{1-i}^-> > 0$.

Par conséquent, pour toute place $p_l \in B$, quelque soit le cas correspondant à p_l, il existe $u_l \in U_i$ tel que $<p_l, u_l> > 0$ (dans le cas (i), $u_l = c_i^+.c_i^-$, dans le cas (ii), $u_l = c_{1-i}^+$, et dans le cas (iii), $u_l = c_{1-i}^+.c_{1-i}^-$). Il est alors possible de choisir un entier n_l pour imposer $m(p_l) + <p_l, u_l^{n_l}> + <p_l, c_i^-> \geq 0$. Posons maintenant $v = u_1^{n_1} u_2^{n_2} \dots u_k^{n_k}$. Par le Lemme 2.4.11 nous avons $<p_l, v> \geq <p_l, u_l^{n_l}>$ pour toute place $p_l \in B$. Nous en déduisons, $m(p_l) + <p_l, v> + <p_l, c_i^-> \geq 0$ pour toute place $p_l \in B$ (et comme expliqué précédemment, c'est également vrai pour les places $p_l \in A$ restantes).

L'hypothèse $(\mathcal{N}, m) \models \Theta_i$ et le Lemme 2.4.10 impliquent pour tout $w \in U_i^*$ l'existence d'un marquage m'' vérifiant $m[w\rangle m''$ et tel que c_i^- est inactive au marquage m''. Par conséquent, il doit nécessairement exister une place p qui inactive le tir de la transition c_i^-, c.-à-d. tel que $m(p) + <p, v> + <p, c_i^-> < 0$. Puisque les places $\{p_1, \dots, p_k\}$ ne peuvent vérifier cette propriété, p fait partie des places qui simulent c_i, ce qui entraîne par la Définition 2.4.1, $<p, v> = 0$. Finalement, $m(p) + <p, c_i^-> < 0$. □

Remarquons que si une place p qui simule le compteur c_i dans le Lemme 2.4.12 est normalisée, alors $m(p) + <p, c_i^-> = -1$ et donc $m(p) = 0$. Ce résultat montre que la sentence Θ_i exprime effectivement le test à zéro pour une place simulant le compteur c_i ; dans les réseaux de compteurs normalisés la réciproque est également vraie, comme énoncé par le lemme suivant :

Lemme 2.4.13. *Soit $\mathcal{C}$ un réseau de compteurs normalisé tel que p_{c_i} est une place de $\mathcal{C}$,*

$$(\mathcal{C}, n_0) \models \Theta_i \text{ si et seulement si } n_0(p_{c_i}) = 0$$

2.4.2 Sentence du Mu-Calcul associée à une machine à compteurs

Nous montrons maintenant le résultat principal de cette section qui est :

Théorème 2.4.14. *Le problème* $\mathbf{Sat}(\mathbf{afL}_\mu, \mathbf{PN})$ *est indécidable*

Afin de prouver ce théorème, nous établissons une réduction du problème de la vacuité du langage d'une machine à deux compteurs. La réduction consiste à construire, pour une machine à deux compteurs M, une sentence Φ_M de $\mathbf{afL}_\mu$ telle qu'il existe un réseau de Petri satisfaisant Φ_M si et seulement si le langage de M est non vide.
Pour obtenir ce résultat, nous construisons la sentence Φ_M de telle manière qu'il existe un réseau de Petri satisfaisant Φ_M si et seulement s'il existe un réseau de compteurs normalisé satisfaisant Φ_M. Nous montrons ensuite que si un réseau de compteurs normalisé satisfait Φ_M alors il simule une exécution finie de M, similairement à la preuve de [Esp97] pour l'indécidabilité du Model-Checking. Réciproquement, nous montrons que toute exécution finie de M, s'il en existe, est simulée par un réseau de compteurs satisfaisant Φ_M.

2.4.2.1 Spécification de machine à compteur en Mu-Calcul

Nous utilisons les résultats de la sous-Section 2.4.1 pour définir une sentence associée à une machine à deux compteurs M donnée. Lorsqu'elle est interprétée sur les réseaux de compteurs, cette sentence exprime l'existence d'une configuration initiale J^0 telle que l'exécution de M pour J^0 termine.

Dans cette partie, nous considérons une machine à deux compteurs $M = (\{q_0, \ldots, q_{n+1}\}, \{c_0, c_1\}, \{\delta_0, \ldots, \delta_n\})$ donnée.

Définition 2.4.15 (Sentence associée à une machine à deux compteurs M).
Soit $\{X_0, \ldots, X_{n+1}\}$ un ensemble de variables et $\phi_0, \ldots, \phi_{n+1}$ les formules de $\mathbf{L}_\mu$ définies de la manière suivante, où $0 \leq l < n+1$ et où Θ_i est la sentence de test à zéro du compteur c_i de la Définition 2.4.9 :

si $\delta_l = c_i := c_i + 1;$ goto q_k, alors

$$\phi_l(X_0,\ldots,X_{n+1}) \stackrel{\text{def}}{=} \langle c_i^+\rangle X_k$$

sinon ($\delta_l =$ if $c_i = 0$ then $(c_i := c_i + 1;$ goto $q_k)$ else $(c_i := c_i - 1;$ goto $q_{k'})$)

$$\phi_l(X_0,\ldots,X_{n+1}) \stackrel{\text{def}}{=} (\langle c_i^+\rangle X_k \wedge \Theta_i) \vee \langle c_i^-\rangle X_{k'}$$

si $l = n+1$

$$\phi_{n+1}(X_0,\ldots,X_{n+1}) \stackrel{\text{def}}{=} \texttt{true}$$

Soient $\Psi_0,\ldots,\Psi_{n+1}$ les sentences définies par :

$$< \Psi_0,\ldots,\Psi_{n+1} >= \mu < X_0,\ldots,X_{n+1} > . < \phi_0,\ldots,\phi_{n+1} > (< X_0,\ldots X_{n+1} >)$$

La sentence *associée à la machine à compteurs* M, notée Φ_M, est la sentence Ψ_0.

Remarque 2.4.16. Dans la formule $(\langle c_i^+\rangle X_k \wedge \Theta_i) \vee \langle c_i^-\rangle X_{k'}$, les sous-formules $(\langle c_i^+\rangle X_k \wedge \Theta_i)$ et $\langle c_i^-\rangle X_{k'}$ sont mutuellement exclusives. En effet, $\langle c_i^-\rangle X_{k'}$ est vérifiée pour une certaine valuation de X_k si et seulement si Θ_i ne l'est pas.

Nous prouvons maintenant que s'il existe un réseau de Petri initialisé $(\mathcal{N}, m_0)$ tel que $(\mathcal{N}, m_0) \models \Phi_M$ alors il existe un réseau de compteurs normalisé $(\mathcal{C}, n_0)$ tel que $(\mathcal{C}, n_0) \models \Phi_M$.

Lemme 2.4.17. *Soit $(\mathcal{N}, m_0)$ un réseau de Petri initialisé, et soit P_c l'ensemble de ses places qui simulent soit le compteur c_0 soit le compteur c_1. Nous avons :*

$$(\mathcal{N}, m_0) \models \Phi_M \textit{ implique } (\mathcal{N}|_{P_c}, m_0|_{P_c}) \models \Phi_M$$

Démonstration. Soit P l'ensemble des places de $\mathcal{N}$; soit P_c le sous-ensemble de P composé des places qui simulent soit le compteur c_0 soit le compteur c_1. *A priori*, P_c peut être vide ou composé de places simulant toutes un même compteur. La preuve donnée ici reste valide dans ces cas particuliers. Dans la suite, nous notons $\mathcal{N}'$ pour $\mathcal{N}|_{P_c}$.

Soit V l'ensemble des marquages accessibles de $\mathcal{N}$ depuis m_0 et soit W l'ensemble des marquages accessibles de $\mathcal{N}'$ depuis $m_0|_{P_c}$. Considérons le projecteur de V dans W :

$$\begin{array}{rccc} \pi : & V & \to & W \\ & m & \mapsto & m|_{P_c} \end{array}$$

Pour toute valuation *val* de $\{X_0,\ldots X_{n+1}\}$ vers 2^V, et pour tout $l \in [0, n+1]$, nous montrons le résultat suivant :

$$\pi(\llbracket\, \phi_l(< X_0,\ldots X_{n+1} >)\, \rrbracket^{val}_{\mathcal{N},m_0}) \quad \subseteq \quad \llbracket\, \phi_l(< X_0,\ldots X_{n+1} >)\, \rrbracket^{\pi\circ val}_{\mathcal{N}',m_0|_{P_c}} \quad (2.7)$$

Remarquons que si val est une valuation sur V ($val : Var \to V$) alors $\pi \circ val$ est la projection de val dans W ; donc $\pi \circ val$ est une valuation sur W. Nous considérons les différents cas pour ϕ_l :

- si $l = n+1$, nous avons $\phi_{n+1} = \texttt{true}$; puisque $\pi(V) \subseteq W$, nous avons immédiatement $\pi([\![\texttt{true}]\!]^{val}_{\mathcal{N},m_0}) \subseteq [\![\texttt{true}]\!]^{\pi\circ val}_{\mathcal{N}',m_0|_{P_c}}$.

- si $\phi_l(< X_0, \ldots X_{n+1} >) = \langle c_i^+ \rangle X_k$, alors pour tout $m \in [\![\langle c_i^+ \rangle X_k]\!]^{val}_{\mathcal{N},m_0}$ il existe $m' \in val(X_k)$ tel que $m[c_i^+\rangle m'$; puisque $\mathcal{N}'$ est une restriction de $\mathcal{N}$, nous avons $\pi(m)[c_i^+\rangle\pi(m')$ (la restriction est plus permissive) ; nous en déduisons $\pi(m') \in \pi \circ val(X_k)$ et donc $\pi(m) \in [\![\langle c_i^+ \rangle X_k]\!]^{\pi\circ val}_{\mathcal{N}',m_0|_{P_c}}$.

- si $\phi_l(< X_0, \ldots X_{n+1} >) = (\langle c_i^+ \rangle X_k \wedge \Theta_i) \vee \langle c_i^- \rangle X_{k'}$, alors pour tout $m \in [\![\phi_l(< X_0, \ldots X_{n+1} >)]\!]^{val}_{\mathcal{N},m_0}$, deux cas peuvent survenir suivant la sous-formule satisfaite par $(\mathcal{N}, m)$ (voir la Remarque 2.4.16) ; dans le premier cas, il existe $m' \in val(X_{k'})$ tel que $m[c_i^-\rangle m'$, ce qui entraîne $\pi(m)[c_i^-\rangle\pi(m')$, comme dans l'item précédent ; dans le second cas, $(\mathcal{N}, m) \models \Theta_i$ et il existe $m' \in val(X_k)$ tel que $m[c_i^+\rangle m'$. Le Lemme 2.4.12 nous assure l'existence d'une place p de $\mathcal{N}$ qui simule le compteur c_i telle que $m(p) + <p, c_i^-> < 0$; par définition de P_c, nous avons $p \in P_c$, ce qui assure $(\mathcal{N}', \pi(m)) \models \Theta_i$; de plus nous avons $\pi(m)[c_i^+\rangle\pi(m')$ comme dans l'item précédent. Dans les deux cas, nous obtenons :

$$\pi([\![\phi_l(< X_0, \ldots X_{n+1} >)]\!]^{val}_{\mathcal{N},m_0}) \subseteq [\![\phi_l(< X_0, \ldots X_{n+1} >)]\!]^{\pi\circ val}_{\mathcal{N}',m_0|_{P_c}}$$

Par monotonie de l'opérateur de point fixe, l'Inégalité (2.7) est préservée et nous avons, en exprimant uniquement la première composante :

$$\pi([\![\Psi_0]\!]_{\mathcal{N},m_0}) \subseteq [\![\Psi_0]\!]_{\mathcal{N}',m_0|_{P_c}}$$

Finalement $(\mathcal{N}, m_0) \models \Phi_M$ implique $(\mathcal{N}', m_0|_{P_c}) \models \Phi_M$. □

D'après le Lemme 2.4.17, pour tout réseau de Petri $\mathcal{N}$, si $(\mathcal{N}, m_0) \models \Phi_M$, alors $(\mathcal{N}|_{P_c}, m_0|_{P_c}) \models \Phi_M$, où $\mathcal{N}|_{P_c}$ est un réseau de compteurs par construction. L'étape suivante est la Procédure de Normalisation ; nous obtenons ainsi un réseau de compteurs normalisé $(\mathcal{C}, n_0)$; le Lemme 2.4.8, assure que les graphes $\mathcal{G}(\mathcal{N}', m'_0)$ et $\mathcal{G}(\mathcal{C}, n_0)$ sont bisimilaires. Par conséquent, $(\mathcal{C}, n_0) \models \Phi_M$.

L'existence d'un réseau de Petri initialisé $(\mathcal{N}, m_0)$ satisfaisant Φ_M garantit l'existence d'un réseau de compteur normalisé $(\mathcal{C}, n_0)$ satisfaisant Φ_M ; dans la suite, nous utilisons ce dernier réseau pour exhiber une configuration initiale de M pour laquelle M termine de la manière suivante : si la place p_{c_i} existe dans $\mathcal{C}$, alors

la valeur initiale du compteur c_i est la valeur $n_0(p_{c_i})$, dans le cas ou p_{c_i} n'existe pas dans $\mathcal{C}$, il est nécessaire de rajouter une étape pour exhiber une valeur initiale pour le compteur c_i. Lors de cette étape, nous considérons temporairement des machines à compteurs "étendues", qui permettent aux compteurs de prendre des valeurs infinies ; à partir d'un réseau de compteurs normalisé satisfaisant Φ_M, nous construisons une exécution finie pour une machine étendue M^∞ ; cette machine se comporte comme M mais peut avoir des valuation infinies pour certains de ses compteurs. Formellement, les configurations de M^∞ sont :

Définition 2.4.18 (Configuration avec valuation infinie). Une configuration avec valuations infinies est une configuration $J^\infty = (q_l, j_0^\infty, j_1^\infty)$ avec $j_0^\infty, j_1^\infty \in \mathbb{N} \cup \{\infty\}$. L'exécution de M^∞ est identique à l'exécution de M avec la convention que : $\infty \neq 0, \infty + 1 = \infty$, et $\infty - 1 = \infty$.

Le Lemme 2.4.19 présenté dans la suite montre qu'il est possible de relier une exécution finie de la machine étendue M^∞ à une exécution finie de M.

Lemme 2.4.19. *Si M^∞ termine pour $J^{\infty,0}$ alors il existe une configuration initiale J^0 de M telle que M termine pour J^0.*

Démonstration. Soit $J^{\infty,0} \dots J^{\infty,k} \dots J^{\infty,f}$ l'exécution finie de M^∞, de longueur f où $J^{\infty,k} = (q_k^\infty, j_0^{\infty,k}, j_1^{\infty,k})$ et $q_f^\infty = q_{n+1}$.

Définissons $J^0 = (q_0, j_0^0, j_1^0)$ en posant $j_i^0 = j_i$ si $j_i \in \mathbb{N}$ et $j_i^0 = f + 1$ sinon ($j_i = \infty$). Soit $J^0 \dots J^h \dots$ la séquence de configurations produite par l'exécution de M pour J^0. Nous montrons par induction sur $h \in [0, f]$ que $q_h^\infty = q_h$, et que si $j_i^{\infty,h} = \infty$ alors $j_i^h \geq f - h > 0$, sinon $j_i^h = j_i^{\infty,h}$. Puisque la valeur d'un compteur ne décroit jamais de plus d'une unité à chaque étape de l'exécution, le fait de poser $f + 1$ comme valuation initiale j_i^0 du compteur c_i de M lorsque $j_i^{\infty,0} = \infty$ dans M^∞ assure qu'à chaque étape h, j_i^h reste supérieur à $f - h$ et n'atteint par conséquent jamais zéro. À chaque fois qu'un compteur c_i avec une valuation infinie est testé à zéro le long de l'exécution de M^∞ (il est donc "décrémenté"), ce test à zéro produit la même mise-à-jour de l'état courant dans M. Par conséquent, $q_h = q_h^\infty$ pour tout $h \in [0, f]$ et en particulier pour $q_f = q_{n+1}$, ce qui montre que l'exécution de M termine pour J^0.

□

Nous montrons que lorsque $(\mathcal{C}, n_0) \models \Phi_M$, M^∞ termine pour une certaine configuration initiale.

Soit V l'ensemble des marquages de $\mathcal{C}$, on définit la fonction F par :

$$\begin{array}{rccc} F & : & (2^V)^{n+2} & \to & (2^V)^{n+2} \\ & & < X_0, \dots X_{n+1} > & \mapsto & [\![< \phi_0, \dots, \phi_{n+1} > (< X_0, \dots, X_{n+1} >)]\!]_{\mathcal{C},n_0} \end{array}$$

Soient $\pi_0, \ldots, \pi_{n+1}$ les projecteurs canoniques de $(2^V)^{n+2}$ dans 2^V ; nous notons $\perp$ pour le vecteur de $(2^V)^{n+2}$ tel que $\pi_l(\perp) = \emptyset$ pour tout $0 \leq l \leq n+1$.
Dans la suite, on dit que le marquage n de $\mathcal{C}$ est *associé* à la configuration $J^\infty = (q, j_0^\infty, j_1^\infty)$ de M^∞ lorsque pour tout $i \in \{0,1\}$, soit $j_i^\infty = \infty$ et p_{c_i} n'est pas une place de $\mathcal{C}$, soit $j_i^\infty \neq \infty$ et $m(p_{c_i}) = j_i^\infty$.

Lemme 2.4.20. *Si $(\mathcal{C}, n_0) \models \Phi_M$ alors le langage de M^∞ est non vide.*

Démonstration. Soit la configuration $J^{\infty,0} = (q_0, j_0^\infty, j_1^\infty)$ définie par $j_i^\infty = n_0(p_{c_i})$ si p_{c_i} existe dans $\mathcal{C}$, et par $j_i^\infty = \infty$ sinon.
Supposons que l'exécution de M^∞ pour $J^{\infty,0}$ produise la séquence infinie de configurations $J^{\infty,0} \ldots J^{\infty,h} \ldots$. Puisque $(\mathcal{C}, n_0) \models \Phi_M$, nous avons

$$n_0 \in \pi_0(\llbracket\, \mu < X_0, \ldots, X_{n+1} > . < \phi_0, \ldots, \phi_{n+1} > (< X_0, \ldots X_{n+1} >) \,\rrbracket_{\mathcal{C},n_0})$$

D'après [Koz83], il existe un ordinal $\gamma_0 \neq 0$ tel que $n_0 \in \pi_0(F^{\gamma_0}(\perp))$.

Nous prouvons maintenant que pour toute configuration $J^{\infty,h} = (q_l, j_0^{\infty,h}, j_1^{\infty,h})$ de l'exécution de M^∞, si m_h est le marquage associé à $J^{\infty,h}$ et si $m_h \in \pi_l(F^{\gamma_h}(\perp))$ pour un certain ordinal γ_h, alors si $J_{h+1} = (q_{l'}, j_0^{\infty,h+1}, j_1^{\infty,h+1})$, son marquage associé, noté m_{h+1}, vérifie $m_{h+1} \in \pi_{l'}(F^{\gamma_{h+1}}(\perp))$ avec $\gamma_{h+1} < \gamma_h$. La preuve est par induction sur h.

Soit m_h le marquage associé à $J^{\infty,h}$. Si $m_h \in \pi_l(F^{\gamma_h}(\perp))$, nous savons que

$$m_h \in \pi_l(\llbracket < \phi_0, \ldots, \phi_{n+1} > (< X_0, \ldots X_{n+1} >) \,\rrbracket^{val}_{\mathcal{C},n_0})$$

pour une certaine valuation val, ce qui correspond à

$$m_h \in \llbracket\, \phi_l(< X_0, \ldots X_{n+1} >) \,\rrbracket^{val}_{\mathcal{C},n_0}$$

Considérons la transition de $J^{\infty,h} = (q_l, j_0^{\infty,h}, j_1^{\infty,h})$ vers $J^{\infty,h+1} = (q_{l'}, j_0^{\infty,h+1}, j_1^{\infty,h+1})$:

- soit q_l est de type I, avec $\phi_l(< X_0, \ldots X_{n+1} >) = \langle c_i^+ \rangle X_k$, où $k = l'$; il existe $m' \in val(X_{l'})$ tel que $m_h[c_i^+\rangle m'$. Puisque $j_i^{\infty,h+1} = j_i^{\infty,h} + 1$ et $j_{1-i}^{\infty,h+1} = j_{1-i}^{\infty,h}$, nous obtenons $m' = m_{h+1}$;

- soit q_l est de type II, avec $\phi_l(< X_0, \ldots X_{n+1} >) = (\langle c_i^+ \rangle X_k \wedge \Theta_i) \vee \langle c_i^- \rangle X_{k'}$. Deux cas sont possibles :
  - si $j_i^{\infty,h} = 0$, alors puisque $m_h(p_{c_i}) = 0$, le Lemme 2.4.13 assure $m_h \in \llbracket\, \langle c_i^+ \rangle X_k \wedge \Theta_i \,\rrbracket^{val}_{\mathcal{C},n_0}$, avec $k = l'$; il existe alors $m' \in val(X_{l'})$ tel que $m_h[c_i^+\rangle m'$, de nouveau puisque $j_i^{\infty,h+1} = 1$ et $j_{1-i}^{\infty,h+1} = j_{1-i}^{\infty,h}$, nous obtenons $m' = m_{h+1}$.

– sinon $j_i^{\infty,h} \neq 0$, c.-à-d. soit p_{c_i} existe dans $\mathcal{C}$ et $m_h(p_{c_i}) \neq 0$, soit p_{c_i} n'existe pas dans $\mathcal{C}$; dans les deux cas, le Lemme 2.4.13 assure $m_h \notin [\![\, \Theta_i \,]\!]^{val}_{\mathcal{C},n_0}$; nous sommes dans le cas où $m_h [\![\, \langle c_i^- \rangle X_{k'} \,]\!]^{val}_{\mathcal{C},n_0}$, avec $k' = l'$. Il existe alors $m' \in val(X_{l'})$ tel que $m_h[c_i^-\rangle m'$, de nouveau puisque $j_i^{\infty,h+1} = j_i^{\infty,h} - 1$ et $j_{1-i}^{\infty,h+1} = j_{1-i}^h$, nous obtenons $m' = m_{h+1}$.

Dans tous les cas, $m_{h+1} \in val(X_{l'})$.

Puisque $m_h \in \pi_l(F^{\gamma_h}(\bot))$, nous avons nécessairement $\gamma_h \neq 0$; donc γ_h est soit un ordinal successeur $\gamma_h = \gamma_{h+1} + 1$, soit un ordinal limite $\gamma_h = \bigcup_{\gamma<\gamma_h} \gamma$. Si $\gamma_h = \gamma_{h+1}+1$, alors $m_{h+1} \in \pi_{l'}(F^{\gamma_{h+1}}(\bot))$ avec $\gamma_{h+1} < \gamma_h$. Si, $\gamma_h = \bigcup_{\gamma<\gamma_h} \gamma$, alors $m_{h+1} \in \pi_{l'}(F^{\gamma_{h+1}}(\bot))$ pour un certain $\gamma_{h+1} < \gamma_h$.

En partant de $m_0 \in \pi_0(F^{\gamma_0}(\bot))$, avec $\gamma_0 \neq 0$, nous construisons en utilisant le résultat précédent, une séquence infinie décroissante d'ordinaux

$$\gamma_0 > \gamma_1 > \ldots > \gamma_h > \ldots$$

ce qui contredit l'ordre bien fondé des ordinaux.

Par conséquent, l'exécution de M^∞ s'arrête nécessairement pour la configuration initiale $J^{\infty,0}$, et le langage de M^∞ n'est pas vide. □

Nous considérons maintenant le réseau de compteurs normalisé "complet" $\mathcal{K}$ qui possède les deux places p_{c_0} et p_{c_1}, et nous montrons comment l'initialiser avec un marquage n_0 de telle manière que lorsque M termine pour une configuration initiale donnée, nous ayons $(\mathcal{K}, n_0) \models \Phi_M$.

Nous montrons dans un premier temps un lemme intermédiaire qui permet d'effectuer un pas en arrière dans la simulation de M par le réseau $\mathcal{K}$:

Lemme 2.4.21. *Soient $J = (q_l, j_0, j_1)$ et $J' = (q_{l'}, j'_0, j'_1)$ deux configurations consécutives le long d'une exécution de M, et soient m et m' deux marquages de $(\mathcal{K}, n_0)$ respectivement associés à J et J'. Si $m' \in [\![\, \Psi_{l'} \,]\!]_{\mathcal{K},n_0}$, alors $m \in [\![\, \Psi_l \,]\!]_{\mathcal{K},n_0}$.*

Démonstration. Supposons $m' \in [\![\, \Psi_{l'} \,]\!]_{\mathcal{K},n_0}$; considérons la règle de transition qui permet de passer de J à J' :

– si q_l est de type I, il est évident que $m[c_i^+\rangle m'$. Par définition de Ψ_l, nous obtenons $m \in [\![\, \Psi_l \,]\!]_{\mathcal{K},n_0}$.
– sinon q_l est de II et concerne le compteur c_i.
  – si $j_i = 0$, alors d'après le Lemme 2.4.13 nous avons $m \in [\![\, \Theta_i \,]\!]_{\mathcal{K},n_0}$, et de plus $m[c_i^+\rangle m'$. Par définition de Ψ_l, nous obtenons $m \in [\![\, \Psi_l \,]\!]_{\mathcal{K},n_0}$.
  – sinon, $j_i \neq 0$, ce qui entraîne $m[c_i^-\rangle m'$. Par définition de Ψ_l, nous obtenons $m \in [\![\, \Psi_l \,]\!]_{\mathcal{K},n_0}$.

□

Lemme 2.4.22. *Si M termine pour $J^0 = (q_0, j_0, j_1)$, alors $(\mathcal{K}, n_0) \models \Phi_M$, où n_0 est le marquage associé à J^0.*

Démonstration. Soit $J^0 \dots J^f$ l'exécution finie de M pour J^0 et soit n le marquage de $\mathcal{K}$ associé à J^f. Puisque $\Psi_{n+1} = \texttt{true}$, nous avons trivialement $n \in [\![\Psi_{n+1}]\!]_{\mathcal{K},n_0}$. Par application itérée du Lemme 2.4.21, nous obtenons $n_0 \in [\![\ \Psi_0\ ]\!]_{\mathcal{K},n_0}$, c.-à-d. $(\mathcal{K}, n_0) \models \Phi_M$. □

2.4.2.2 Preuve du théorème principal

Nous sommes maintenant à même de prouver le Théorème 2.4.14. Soit M une machine à deux compteurs, soit Φ_M sa sentence associée.

Nous montrons que le problème de la vacuité du langage de M est équivalent au problème de satisfiabilité de Φ_M sur la classe des réseaux de Petri.

S'il existe un réseau initialisé $(\mathcal{N}, m_0)$ tel que $(\mathcal{N}, m_0) \models \Phi_M$ alors, par les Lemmes 2.4.17 et 2.4.8, nous obtenons un réseau de compteurs normalisé $(\mathcal{C}, n_0)$ tel que $(\mathcal{C}, n_0) \models \Phi_M$; le Lemme 2.4.20 nous assure alors que le langage M^∞ n'est pas vide, et le Lemme 2.4.19 nous permet d'affirmer que celui de M n'est pas vide.

Réciproquement, supposons que le langage de M ne soit pas vide, alors l'exécution de M termine pour une certaine configuration initiale $J^0 = (q_0, j_0, j_1)$. D'après le Lemme 2.4.22, le réseau de compteurs normalisé complet $(\mathcal{K}, n_0)$ avec $n_0(p_{c_0}) = j_0$ et $n_0(p_{c_1}) = j_1$ vérifie $(\mathcal{K}, n_0) \models \Phi_M$.

Nous avons réduit le problème de la vacuité du langage d'une machine à deux compteurs au problème de satisfiabilité de $\mathbf{L}_\mu$ sur la classe des réseaux purs de **Pn**. Par le Théorème 2.3.4, le problème de Satisfiabilité est indécidable.

2.4.3 Le cas des réseaux impurs

Nous considérons maintenant le cas des réseaux impurs. Nous ne ferons pas ici la preuve formelle du résultat mais nous donnons la méthode qui permet de l'obtenir.

La Définition 2.4.1 d'une place simulant un compteur reste identique dans le cas des réseaux impurs ; cependant, la Définition 2.4.3 devient plus restrictive :

Définition 2.4.23 (Place normalisée simulant un compteur). Une place p qui simule le compteur c est dite *normalisée* si et seulement si $<p, {}^\bullet c^+> = 1$, $<p, c^{+\bullet}> = 0$, $<p, {}^\bullet c^-> = 0$ et pour tout $a \in \Sigma_0$, $<p, {}^\bullet a> = 0$.

Le Lemme 2.4.5 n'est plus valide ; en effet, par exemple une place p qui simule un compteur c peut vérifier $<p, c^+> \geq 0$ et rendre inactive la transition c^+ à un certain marquage accessible m lorsque $m(p) - <p, c^{+\bullet}> < 0$. En revanche, le résultat du Lemme 2.4.17 peut être étendu, et s'énoncer de la manière suivante :

Lemme 2.4.24. *Soit $(\mathcal{N}, m_0)$ un réseau de Petri initialisé, soit P_c l'ensemble de ses places qui simulent soit le compteur c_0 soit le compteur c_1 et soit $(\mathcal{N}', m'_0)$ le réseau composé des versions normalisées des places appartenant à P_c. Nous avons :*

$$(\mathcal{N}, m_0) \models \Phi_M \text{ implique } (\mathcal{N}', m'_0) \models \Phi_M$$

Démonstration (Sketch). Le principe de la preuve reste inchangé par rapport à celle du Lemme 2.4.17 ; en effet, les places qui simulent les compteurs et qui sont impures (et peuvent par conséquent restreindre les comportements du réseau) peuvent être traitées comme les places qui ne simulent pas un compteur (dont la preuve montre qu'elle ne participent pas à la satisfaction de Φ_M), à l'exception des tests à zéro. Il est donc possible de remplacer les places simulant les compteurs qui sont impures par leur version normalisée (qui est pure) sans remettre en cause la satisfaction de Φ_M. □

Le reste de la preuve d'indécidabilité de $\mathbf{Sat}(\mathbf{afL}_\mu, \mathbf{PN})$ n'est pas modifié puisque le Lemme 2.4.24 nous munit d'un réseau de compteurs pur.

2.5 Le problème du marquage

Dans cette section nous abordons le problème $\mathbf{Marq}(\mathbf{S},\mathbf{C})$. Nous montrons que ce problème est indécidable lorsque $\mathbf{S}$ est le fragment *alternation-free* de $\mathbf{L}_\mu$ et que $\mathbf{C}$ est une classe de l'ensemble $\{\mathbf{LPN}, \mathbf{PN}, \mathbf{PNB}, \mathbf{PNSB}, \mathbf{PNO}, \mathbf{PNGM}\}$. Nous rappelons également les cas pour lesquels ce problème est décidable.

2.5.1 Indécidabilité de Marq(S,C) pour plusieurs classes C

Soit $\mathcal{B}$ le réseau de la Figure 2.2. Le réseau $\mathcal{B}$ est un réseau de compteurs normalisé composé des places p_0 et p_1 simulant c_0 et c_1 auquel ont été ajoutées les places $p_{\bar{0}}$ et $p_{\bar{1}}$ "complémentaires" des deux places simulant chacune un compteur. Le réseau $\mathcal{B}$ appartient aux classes $\mathbf{LPN}, \mathbf{PN}, \mathbf{PNB}, \mathbf{PNSB}, \mathbf{PNO}$ et $\mathbf{PNGM}$.

Nous montrons le résultat suivant :

Théorème 2.5.1. *Le problème* $\mathbf{Marq}(\mathbf{afL}_\mu,\mathbf{C})$ *est indécidable pour toute classe* $\mathbf{C}$ *contenant le réseau* $\mathcal{B}$.

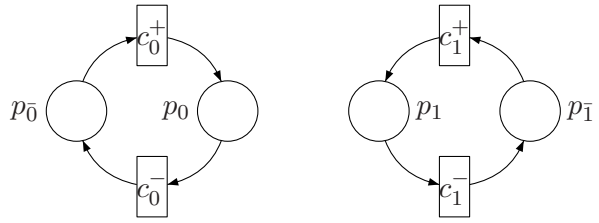

FIG. 2.2 – Le réseau $\mathcal{B}$.

La preuve de ce théorème repose sur la réduction du Problème indécidable 2.3.3 de la vacuité du langage d'une machine à deux compteurs. Cette réduction est analogue à celle que nous avons proposée pour la preuve du Théorème 2.4.14 : pour toute machine à compteurs M, nous construisons une formule Ξ_M, analogue à celle de la Définition 2.4.15 (au test à zéro près), et telle qu'il existe un marquage m_0 tel que $(\mathcal{B}, m_0) \models \Xi_M$ si et seulement si le langage de M est non vide.

Soit Σ l'alphabet $\{c_0^+, c_0^-, c_1^+, c_1^-\}$, nous donnons la définition de la sentence associée à une machine à compteurs M :

Définition 2.5.2 (Sentence associée à une machine à deux compteurs). Soit $\{X_0, \ldots, X_{n+1}\}$ un ensemble de variables et $\phi_0, \ldots, \phi_{n+1}$ les formules de $\mathbf{L}_\mu$ définies de la manière suivante, avec $0 \leq l < n+1$:

si $\delta_l = c_i := c_i + 1;$ goto q_k, alors
$$\phi_l(X_0, \ldots, X_{n+1}) \stackrel{\text{def}}{=} \langle c_i^+ \rangle X_k$$

sinon ($\delta_l =$ if $c_i = 0$ then $(c_i := c_i + 1;$ goto $q_k)$ else $(c_i := c_i - 1;$ goto $q_{k'})$)
$$\phi_l(X_0, \ldots, X_{n+1}) \stackrel{\text{def}}{=} (\langle c_i^+ \rangle X_k \wedge \not\rightarrow^{c_i^-}) \vee \langle c_i^- \rangle X_{k'}$$

si $l = n+1$
$$\phi_{n+1}(X_0, \ldots, X_{n+1}) \stackrel{\text{def}}{=} \texttt{true}$$

Soient $\Psi_0, \ldots, \Psi_{n+1}$ les sentences définies par :

$$< \Psi_0, \ldots, \Psi_{n+1} >= \mu < X_0, \ldots, X_{n+1} > . < \phi_0, \ldots, \phi_{n+1} > (< X_0, \ldots X_{n+1} >)$$

La sentence *associée à la machine à compteurs* M, notée Ξ_M, est la sentence Ψ_0.

Remarque 2.5.3. Les places complémentaires $p_{\bar{0}}$ et $p_{\bar{1}}$ de $\mathcal{B}$ n'interviennent pas dans la satisfaction de la formule Ξ_M. En effet, lorsque la machine possède une exécution finie, les compteurs de la machine restent bornés lors de cette exécution ; le marquage initial des places complémentaires doit simplement être suffisamment élevé pour permettre aux places simulant les compteurs d'atteindre cette borne.

Dans la suite on dit qu'un marquage m de $\mathcal{B}$ est *associé* à une configuration (q, j_0, j_1) de M lorsque $m(p_0) = j_0$ et $m(p_1) = j_1$. Contrairement à la notion de marquage associé à une configuration définie pour les réseaux de compteurs page 52, la notion définie ici n'associe pas un unique marquage à une configuration ; en effet, le nombre de jetons dans les places complémentaires peut varier.

Nous énonçons trois lemmes utiles à la preuve du théorème, leur preuve sera donnée plus loin.

Lemme 2.5.4. *Soit m_0 un marquage de $\mathcal{B}$, si $(\mathcal{B}, m_0) \models \Xi_M$ alors le langage de M n'est pas vide.*

Lemme 2.5.5. *Soient $J = (q_l, j_0, j_1)$ et $J' = (q_{l'}, j'_0, j'_1)$ deux configurations consécutives le long d'une exécution de M, et soient m et m' deux marquages accessibles de $(\mathcal{B}, m_0)$ respectivement associés à J et J' avec $m'(p_{\bar{0}}) \geq 1$ et $m'(p_{\bar{1}}) \geq 1$. Si $m' \in [\![\Psi_{l'}]\!]_{(\mathcal{B},m_0)}$, alors $m \in [\![\Psi_l]\!]_{(\mathcal{B},m_0)}$.*

Lemme 2.5.6. *Si l'exécution de M pour $J^0 = (q_0, j_0, j_1)$ est finie, alors il existe m_0 associé à J^0 tel que $(\mathcal{B}, m_0) \models \Xi_M$.*

Nous sommes maintenant en mesure de prouver le Théorème 2.5.1 d'indécidabilité du problème du marquage :

Démonstration du théorème 2.5.1. Pour toute machine M à deux compteurs, le Lemme 2.5.4 montre que s'il existe un marquage initial m_0 pour le réseau $\mathcal{B}$ tel que $(\mathcal{B}, m_0) \models \Xi_M$, alors le langage de M est non vide ; le Lemme 2.5.6 montre qui si le langage de M est non vide, alors il existe un marquage initial m_0 pour le réseau $\mathcal{B}$ tel que $(\mathcal{B}, m_0) \models \Xi_M$. Ces deux résultats démontrent l'équivalence entre le problème de la vacuité du langage d'une machine à deux compteurs et le problème de l'existence d'un marquage initial m_0 pour le réseau $\mathcal{B}$ tel que $(\mathcal{B}, m_0) \models \Xi_M$. Ce dernier problème est une instance particulière du problème $\mathbf{Marq}(\mathbf{afL}_\mu, \mathbf{C})$ où $\mathbf{C}$ est un classe contenant le réseau $\mathcal{B}$. Le problème de la vacuité du langage d'une machine à deux compteurs étant indécidable, le problème $\mathbf{Marq}(\mathbf{afL}_\mu, \mathbf{C})$ est indécidable pour toute classe $\mathbf{C}$ contenant le réseau $\mathcal{B}$. □

2.5.2 Annexe : preuves des lemmes

Nous donnons ici les preuves des Lemmes 2.5.4, 2.5.5, et 2.5.6. Ces preuves sont très similaires à celles des Lemmes 2.4.20, 2.4.21, 2.4.22 et ne présentent par conséquent qu'un intérêt limité.

Soit V l'ensemble des marquages de $\mathcal{B}$. Soient $\pi_0, \ldots, \pi_{n+1}$ les projecteurs canoniques de $(2^V)^{n+2}$ dans 2^V ; on note $\perp$ pour le vecteur vérifiant $\pi_l(\perp) = \emptyset$ pour tout $0 \leq l \leq n+1$.

Démonstration du Lemme 2.5.4. Soit la configuration $J^0 = (q_0, j_0, j_1)$ définie par $j_i = m_0(p_i)$.
Supposons que l'exécution de M pour J^0 produise la séquence infinie de configurations $J^0 \ldots J^h \ldots$. Puisque $(\mathcal{B}, m_0) \models \Xi_M$, nous avons

$$m_0 \in \pi_0(\llbracket\, \mu < X_0, \ldots, X_{n+1} > . < \phi_0, \ldots, \phi_{n+1} > (< X_0, \ldots X_{n+1} >) \,\rrbracket_{(\mathcal{B},m_0)})$$

D'après [Koz83], il existe un ordinal $\gamma_0 \neq 0$ tel que $m_0 \in \pi_0(F^{\gamma_0}(\perp))$.

On montre pour chaque configuration $J^h = (q_l, j_0^h, j_1^h)$ de l'exécution de M, si m_h est le marquage associé à J^h et si $m_h \in \pi_l(F^{\gamma_h}(\perp))$ pour un ordinal γ_h, alors si $J_{h+1} = (q_{l'}, j_0^{h+1}, j_1^{h+1})$, son marquage associé, noté m_{h+1}, vérifie $m_{h+1} \in \pi_{l'}(F^{\gamma_{h+1}}(\perp))$ avec $\gamma_{h+1} < \gamma_h$. La preuve est par induction sur h.

Soit m_h le marquage associé à J^h. Si $m_h \in \pi_l(F^{\gamma_h}(\perp))$, nous avons

$$m_h \in \pi_l(\llbracket < \phi_0, \ldots, \phi_{n+1} > (< X_0, \ldots X_{n+1} >) \,\rrbracket^{val}_{(\mathcal{B},m_0)})$$

pour une certaine valuation val, ce qui nous donne

$$m_h \in \llbracket\, \phi_l(< X_0, \ldots X_{n+1} >) \,\rrbracket^{val}_{(\mathcal{B},m_0)}$$

Considérons la transition de $J^h = (q_l, j_0^h, j_1^h)$ à $J^{h+1} = (q_{l'}, j_0^{h+1}, j_1^{h+1})$:

- soit q_l est de type I, alors $\phi_l(< X_0, \ldots X_{n+1} >) = \langle c_i^+ \rangle X_k$, avec $k = l'$; dans ce cas il existe $m' \in val(X_{l'})$ tel que $m_h[c_i^+\rangle m'$. Puisque $j_i^{h+1} = j_i^h + 1$ et $j_{1-i}^{h+1} = j_{1-i}^h$, nous avons nécessairement $m' = m_{h+1}$;
- soit q_l est de type II, alors $\phi_l(< X_0, \ldots X_{n+1} >) = (\langle c_i^+ \rangle X_k \wedge \not\rightarrow^{c_i^-}) \vee \langle c_i^- \rangle X_{k'}$. Nous considérons deux cas suivant la valeur de j_i^h :
  - si $j_i^h = 0$, alors puisque $m_h(p_i) = 0$, nous avons $m_h \in \llbracket\, \langle c_i^+ \rangle X_k \wedge \not\rightarrow^{c_i^-} \rrbracket^{val}_{(\mathcal{B},m_0)}$ avec $k = l'$; il existe alors $m' \in val(X_{l'})$ tel que $m_h[c_i^+\rangle m'$. Puisque $j_i^{h+1} = 1$ et $j_{1-i}^{h+1} = j_{1-i}^h$, alors $m' = m_{h+1}$.

– sinon $j_i^h \neq 0$, dans ce cas $m_h[c_i^-\rangle$, ce qui implique $m_h \in [\![\langle c_i^-\rangle X_{k'}]\!]^{val}_{(\mathcal{B},m_0)}$, avec $k' = l'$; il existe alors $m' \in val(X_{l'})$ tel que $m_h[c_i^-\rangle m'$. De nouveau, puisque $j_i^{h+1} = j_i^h - 1$ et $j_{1-i}^{h+1} = j_{1-i}^h$, alors $m' = m_{h+1}$.

Dans tous les cas, $m_{h+1} \in val(X_{l'})$.

Puisque $m_h \in \pi_l(F^{\gamma_h}(\bot))$, nous avons nécessairement $\gamma_h \neq 0$; soit γ_h est un ordinal successeur $\gamma_h = \gamma_{h+1} + 1$, soit γ_h est un ordinal limite $\gamma_h = \bigcup_{\gamma<\gamma_h} \gamma$. Si $\gamma_h = \gamma_{h+1} + 1$, alors $m_{h+1} \in \pi_{l'}(F^{\gamma_{h+1}}(\bot))$ avec $\gamma_{h+1} < \gamma_h$. Sinon, $\gamma_h = \bigcup_{\gamma<\gamma_h} \gamma$, alors $m_{h+1} \in \pi_{l'}(F^{\gamma_{h+1}}(\bot))$ pour un certain $\gamma_{h+1} < \gamma_h$.

En partant de $m_0 \in \pi_0(F^{\gamma_0}(\bot))$, avec $\gamma_0 \neq 0$, nous construisons une séquence décroissante infinie d'ordinaux

$$\gamma_0 > \gamma_1 > \ldots > \gamma_h > \ldots$$

ce qui contredit la propriété d'ordre bien fondé des ordinaux.

Par conséquent, l'exécution de M pour la configuration initiale J^0 est nécessairement finie, et le langage de M est non vide. □

Démonstration du Lemme 2.5.5. Supposons $m' \in [\![\Psi_{l'}]\!]_{(\mathcal{B},m_0)}$; on considère la règle de transition entre J et J' en rappelant que par hypothèse $m'(p_{\bar{0}}) \geq 1$ et $m'(p_{\bar{1}}) \geq 1$, ce qui justifie que les places $p_{\bar{0}}$ et $p_{\bar{1}}$ n'interviennent pas dans la considérations suivantes :

– Si q_l est de type I, nous avons $m[c_i^+\rangle m'$. Par définition de Ψ_l, nous obtenons $m \in [\![\Psi_l]\!]_{(\mathcal{B},m_0)}$.
– Sinon q_l est de type II et concerne c_i.
  – si $j_i = 0$, alors c_i^- est inactive au marquage m, et $m[c_i^+\rangle m'$. Par définition de Ψ_l, nous obtenons $m \in [\![\Psi_l]\!]_{(\mathcal{B},m_0)}$.
  – sinon $j_i \neq 0$, alors $m[c_i^-\rangle m'$. Par définition de Ψ_l, nous obtenons $m \in [\![\Psi_l]\!]_{(\mathcal{B},m_0)}$.

□

Démonstration du Lemme 2.5.6. Soit $J^0 \ldots J^f$ l'exécution finie de M pour J^0 ; soit m_f le marquage de $\mathcal{B}$ associé à J^f et tel que $m_f(p_{\bar{0}}) = f$ et $m_f(p_{\bar{1}}) = f$. Puisque $\Psi_{n+1} = \texttt{true}$, nous avons trivialement $m_f \in [\![\Psi_{n+1}]\!]_{(\mathcal{B},m_0)}$. Par application inductive du Lemme 2.5.5 en tenant compte du fait qu'à chaque étape, le marquage m_h associée à une configuration J^h vérifie $m_h(p_{\bar{0}}) \geq h$ et $m_h(p_{\bar{1}}) \geq h$, nous obtenons $m_0 \in [\![\Psi_0]\!]_{(\mathcal{B},m_0)}$, ce qui implique $(\mathcal{B}, m_0) \models \Xi_M$.

□

2.6 Mu-Calcul pour les langages

Cette section est dédiée à la présentation d'une sémantique sur les langages pour $\mathbf{L}_\mu$. L'objet de cette nouvelle sémantique est de simplifier les preuves dans les chapitres suivants. Nous avons présenté dans le Chapitre 1 deux représentations pour les réseaux de Petri : le langage d'un réseau et son graphe des marquages ; la notion de langage est plus adaptée à la notion de bisimulation (elle prend en compte les séquences d'actions et leurs branchements et fait abstraction des états) ; de plus, elle se déduit plus directement des relations de tir d'un réseau.
La sémantique proposée dans cette section associe à une formule de $\mathbf{L}_\mu$ et à un langage (de processus) un ensemble de mots ; ces mots représentent les états (du processus) avec leur passé, ou encore les nœuds de l'arbre des exécutions.

Nous montrons que la sémantique classique présentée au début de ce chapitre (Définition 2.1.2) est équivalente, dans le cas des systèmes déterministes à branchements finis, à la sémantique sur les langages que nous présentons dans la suite.

2.6.1 Sémantique sur les langages clos par préfixe pour $\mathbf{L}_\mu$

Nous définissons une interprétation des formules du Mu-Calcul sur les langages clos par préfixe sur l'alphabet Σ. L'interprétation d'une formule de $\mathbf{L}_\mu$ sur un langage clos par préfixe L est l'ensemble des mots de L satisfaisant la formule, relativement à une interprétation val. Une valuation val est une fonction des variables libres de la formule dans les parties de L. L'interprétation d'une formule sur un langage n'est pas nécessairement close par préfixe.

Définition 2.6.1 (Sémantique de $\mathbf{L}_\mu$ sur les langages clos par préfixe). L'interprétation sur un langage clos par préfixe $L \subseteq \Sigma^*$ d'une sentence $\beta \in \mathbf{L}_\mu$ relativement à une valuation $val : Var \to L$ est l'ensemble $[\![\beta]\!]^{val}_L \subseteq L$ inductivement défini par :

$$
\begin{array}{l}
[\![\, \mathtt{true} \,]\!]^{val}_L = L \\
[\![\, X \,]\!]^{val}_L = val(X) \\
[\![\, \neg\beta \,]\!]^{val}_L = L \setminus [\![\, \beta \,]\!]^{val}_L \\
[\![\, \beta_1 \vee \beta_2 \,]\!]^{val}_L = [\![\, \beta_1 \,]\!]^{val}_L \cup [\![\, \beta_2 \,]\!]^{val}_L \\
[\![\, \langle a \rangle \beta \,]\!]^{val}_L = \{w \in L \mid w.a \in [\![\, \beta \,]\!]^{val}_L\} \\
[\![\, \mu X.\beta(X) \,]\!]^{val}_L = \bigcap \{V \subseteq L \mid [\![\, \beta \,]\!]^{val(V/X)}_L \subseteq V\}
\end{array}
$$

où la valuation $val(V/X) : Var \to \mathcal{P}(L)$ est donnée par $val(V/X)(X') = V(X')$ pour toute variable $X' \in Var$ telle que $X' \neq X$ et $val(V/X)(X) = V$.

L'interprétation $[\![\, \mu X.\beta(X) \,]\!]^{val}_L$ (resp. $[\![\, \nu X.\beta(X) \,]\!]^{val}_L$) est le plus petit point-fixe (resp. plus grand point-fixe) de la fonction de 2^L dans 2^L qui à V associe

$[\![\beta]\!]_L^{val(V/X)}$. La sémantique des sentences de $\mathbf{L}_\mu$ ne dépend pas de la valuation ; on note $[\![\beta]\!]_L$ l'interprétation d'une sentence β relativement à toute valuation.

2.6.2 Équivalence des deux sémantiques de $\mathbf{L}_\mu$

Proposition 2.6.2 (Équivalence des sémantiques). *Soit $\mathcal{P}$ un processus de langage L ($\mathcal{L}(\mathcal{P}) = L$) et d'état initial s_0 ; soit ϕ une sentence de $\mathbf{L}_\mu$, $s_0 \in [\![\phi]\!]_\mathcal{P}$ si et seulement si $1 \in [\![\phi]\!]_L$.*

Démonstration. Pour les besoins de cette preuve nous manipulons deux valuations : nous désignerons par $val_\mathcal{P}$ la valuation sur les états du processus et par val_L la valuation sur les mots du langage du processus $\mathcal{P}$.
Avec $\mathcal{P} = \langle S, s_0, t \rangle$, on supposera sans perdre de généralité que tous les états de $\mathcal{P}$ sont accessibles. On montre pour toute formule β de $\mathbf{L}_\mu$ pour tout couple $(val_\mathcal{P}, val_L)$ de valuations tels que, pour tout $X \in Var(\beta)$,

$$val_L = \{w \in \Sigma^* \mid t(s_0, w) \in val_\mathcal{P}\} \tag{2.8}$$

que l'on a le résultat suivant :

$$[\![\beta]\!]_L^{val_L} = \{w \in \Sigma^* \mid t(s_0, w) \in [\![\beta]\!]_\mathcal{P}^{val_\mathcal{P}}\}$$

La preuve se fait par induction sur la formule β ; nous rappelons les deux sémantiques lorsqu'elles sont utiles.

– `true` : ce cas est trivial
– X : le résultat est immédiat par l'Égalité (2.8).

– $\neg\beta$, Les sémantiques sont :

$$\begin{array}{lcl} [\![\neg\beta]\!]_L^{val_L} & = & L \setminus [\![\beta]\!]_L^{val_L} \\ [\![\neg\beta]\!]_\mathcal{P}^{val_\mathcal{P}} & = & S \setminus [\![\beta]\!]_\mathcal{P}^{val_\mathcal{P}} \end{array}$$

le processus $\mathcal{P}$ étant déterministe,

$$L = \biguplus_{s \in S} \{w \in \Sigma^* \mid t(s_0, w) = s\}$$

par conséquent, par hypothèse d'induction,

$$[\![\neg\beta]\!]_L^{val_L} = \bigcup_{s \in T} \{w \in \Sigma^* \mid t(s_0, w) = s\}$$

avec $T = S \setminus [\![\beta]\!]_\mathcal{P}^{[val_\mathcal{P}]}$, ce qui nous permet de conclure

$$[\![\neg\beta]\!]_L^{val_L} = \{w \in \Sigma^* \mid t(s_0, w) \in [\![\neg\beta]\!]_\mathcal{P}^{val_\mathcal{P}}\}$$

– $\beta_1 \vee \beta_2$, ce cas est immédiat par l'induction.

– $\langle a \rangle \beta$, les sémantiques sont :

$$\begin{array}{lcl} [\![\langle a \rangle \beta]\!]_L^{val_L} & = & \{v \in L \mid v.a \in [\![\beta]\!]_L^{val_L}\} \\ [\![\![\langle a \rangle \beta]\!]\!]_{\mathcal{P}}^{val_{\mathcal{P}}} & = & \{s \in S \mid \exists s' : t(s,a) = s' \text{ et } s' \in [\![\![\beta]\!]\!]_{\mathcal{P}}^{val_{\mathcal{P}}}\} \end{array}$$

par hypothèse d'induction, en posant $w = v.a$, nous obtenons

$$[\![\langle a \rangle \beta]\!]_L^{val_L} = \{v \in L \mid t(s_0, v.a) \in [\![\![\beta]\!]\!]_{\mathcal{P}}^{val_{\mathcal{P}}}\}$$

ce qui implique :

$$[\![\langle a \rangle \beta]\!]_L^{val_L} = \{v \in L \mid t(s_0, v) \in [\![\![\langle a \rangle \beta]\!]\!]_{\mathcal{P}}^{val_{\mathcal{P}}}\}$$

– $\mu X.\beta(X)$, les sémantiques sont :

$$\begin{array}{lcl} [\![\mu X.\beta(X)]\!]_L^{val_L} & = & \bigcap\{V_1 \subseteq L \mid [\![\beta]\!]_L^{val_L(V_1/X)} \subseteq V_1\} \\ [\![\![\mu X.\beta(X)]\!]\!]_{\mathcal{P}}^{val_{\mathcal{P}}} & = & \bigcap\{V_2 \subseteq S \mid [\![\![\beta]\!]\!]_{\mathcal{P}}^{val_{\mathcal{P}}(V_2/X)} \subseteq V_2\} \end{array}$$

soit $V_2 \subseteq S$ tel que $[\![\![\beta]\!]\!]_{\mathcal{P}}^{val_{\mathcal{P}}(V_2/X)} \subseteq V_2$, on pose $V_1 = \{w \in \Sigma^* \mid t(s_0, w) \in V_2\}$; les valuations $val_L(V_1/X)$ et $val_{\mathcal{P}}(V_2/X)$ vérifient l'Égalité (2.8), nous pouvons par conséquent appliquer l'hypothèse d'induction pour obtenir

$$[\![\beta]\!]_L^{val_L(V_1/X)} = \{w \in \Sigma^* \mid t(s_0, w) \in [\![\![\beta]\!]\!]_{\mathcal{P}}^{val_{\mathcal{P}}(V_2/X)}\}$$

nous en déduisons $[\![\beta]\!]_L^{val_L(V_1/X)} \subseteq V_1$. Ce résultat nous permet d'obtenir :

$$[\![\mu X.\beta(X)]\!]_L^{val_L} \subseteq \{w \in \Sigma^* \mid t(s_0, w) \in [\![\![\mu X.\beta(X)]\!]\!]_{\mathcal{P}}^{val_{\mathcal{P}}}\}$$

L'autre inclusion s'obtient de la même manière, en posant $V_1 \subseteq L$ tel que $[\![\![\beta]\!]\!]_L^{val_L(V_1/X)} \subseteq V_1$.

Finalement, puisque $1 \in \{w \in \Sigma^* \mid t(s_0, w) = s_0\}$ et que $t(s_0, 1) = s_0$, nous obtenons $s_0 \in [\![\![\phi]\!]\!]_{\mathcal{P}}$ si et seulement si $1 \in [\![\phi]\!]_L$ □

D'après la Proposition 2.6.2, $\mathcal{P} \models \beta$ si et seulement si $1 \in [\![\beta]\!]_L$. On dira dans ce cas que le langage L *satisfait* la sentence β, noté $L \models \beta$.

2.7 Conclusion

Nous avons montré dans ce chapitre que tous les problèmes reliant le Mu-Calcul aux réseaux de Petri non étiquetés sont indécidables : le cas du Model-Checking est un résultat antérieur [Esp97], mais ceux de Satisfiabilité et de Marquage sont ici nouveaux. Pour ces deux derniers problèmes, nous avons proposé une réduction du problème de la vacuité du langage d'une machine à compteurs. Cette réduction repose en grande partie sur la simulation du test à zéro et aux branchements qui en découlent. Considérant ce point, nous proposons, dans le chapitre suivant, un fragment syntaxique du Mu-Calcul dépourvu de l'opérateur $\vee$ qui sera l'objet de notre étude dans la suite de ce document.

Pour imposer l'existence d'une place qui simule un compteur c_i, nous avons proposé la formule Θ_i qui énonce l'existence de cette place. Dans le Chapitre 4, nous proposons une méthode générale pour construire une formule qui exprime l'existence d'une place quelconque donnée.

Chapitre 3

Spécifications modales pour la synthèse

Sommaire

Nous présentons dans ce chapitre un fragment syntaxique de $\mathbf{L}_\mu$, *le Nu-Calcul conjonctif*, noté $\mathbf{L}_\nu$. Nous introduisons un reconnaisseur de langages associé au Nu-Calcul conjonctif : *les spécifications modales*. Nous montrons que les modèles d'une spécification modale adoptent une structure particulière de treillis ; puis nous établissons l'équivalence d'expressivité entre ces spécifications et $\mathbf{L}_\nu$ en donnant des algorithmes constructifs de traduction.

La dernière section de ce chapitre est consacrée à une extension des résultats de la synthèse de réseaux de Petri : nous définissons une restriction structurelle sur les spécifications modales et nous identifions des classes de spécifications pour lesquelles le problème de satisfiabilité est décidable. Ces classes de spécifications modales sont strictement plus expressives que les intervalles langages (elles correspondent à des unions infinitaires d'intervalles de langages). Les preuves de décidabilité sont constructives et nous donnons les algorithmes de synthèse.

3.1 Nu-Calcul conjonctif

Le Nu-Calcul conjonctif est un fragment syntaxique du Mu-Calcul dont l'introduction est motivée par les résultats d'indécidabilité du Chapitre 2 ; il est défini de la manière suivante :

Définition 3.1.1 (Nu-Calcul conjonctif). L'ensemble des formules du Nu-Calcul conjonctif, noté $\mathbf{L}_\nu$, est le fragment syntaxique de $\mathbf{L}_\mu$ obtenu par restriction de la grammaire de $\mathbf{L}_\mu$ à la grammaire suivante, avec $a \in \Sigma$:

$$(\mathbf{L}_\nu \ni) \quad \beta_1, \beta_2 ::= \mathtt{true} \mid X \mid \ \rightarrow^a \ \mid [a]\beta_1 \mid \ \not\rightarrow^a \ \mid \beta_1 \wedge \beta_2 \mid \nu X.\beta_1(X)$$

L'interprétation sur un langages clos par préfixe $L \subseteq \Sigma^*$ d'une formule $\beta \in \mathbf{L}_\nu$, relativement à une valuation $val : Var \to 2^L$ est identique à la sémantique de la même formule plongée dans $\mathbf{L}_\mu$; nous rappelons sa définition inductive :

$$\begin{array}{l}
[\![\, \mathtt{true} \,]\!]_L^{val} = L \\
[\![\, X \,]\!]_L^{val} = val(X) \\
[\![\, \rightarrow^a \,]\!]_L^{val} = \{w \in L \mid w.a \in L\} \\
[\![\, \not\rightarrow^a \,]\!]_L^{val} = \{w \in L \mid w.a \notin L\} \\
[\![\, [a]\beta \,]\!]_L^{val} = \{w \in L \mid w.a \in [\![\, \beta \,]\!]_L^{val}\} \cup \{w \in L \mid w.a \notin L\} \\
[\![\, \beta_1 \wedge \beta_2 \,]\!]_L^{val} = [\![\, \beta_1 \,]\!]_L^{val} \cap [\![\, \beta_2 \,]\!]_L^{val} \\
[\![\, \nu X.\beta(X) \,]\!]_L^{val} = \bigcup\{V \subseteq L \ \mid \ [\![\, \beta \,]\!]_L^{val(V/X)} \supseteq V\}
\end{array}$$

L'opérateur $\langle a \rangle \beta$ de $\mathbf{L}_\mu$ s'exprime $[a]\beta \wedge \rightarrow^a$ dans $\mathbf{L}_\nu$. Toutefois, les opérateurs suivants ne peuvent être exprimés dans $\mathbf{L}_\nu$: $\beta_1 \vee \beta_2$, $\mu X.\beta(X)$, $\neg\beta$, $\mathtt{false}$. L'opérateur disjonctif $\vee$ n'est présent qu'implicitement à travers l'opérateur $[a]\beta$ qui s'exprime

dans $\mathbf{L}_\mu$ par $\langle a \rangle \beta \vee \not\rightarrow^a$.

3.2 Les spécifications modales et leurs modèles

Nous proposons dans cette section un reconnaisseur de langages : les spécifications modales. Nous montrons dans la section suivante que les spécifications modales sont équivalentes aux sentences de $\mathbf{L}_\nu$; à ce titre, les spécifications modales forment un adaptation des automates alternants associés aux sentences de $\mathbf{L}_\nu$ aux langages clos par préfixe (alors que les automates alternant opèrent sur des arbres de calcul).

La motivation de l'introduction des spécifications modales est double : premièrement, elle permet de simplifier les preuves, en particulier pour établir un lien avec les intervalles de langages (voir Chapitre 4, page 87) et les résultats antérieurs de synthèse de réseaux de Petri ; deuxièmement, elle permet l'extraction d'un fragment structurel pour lequel la synthèse de réseaux de Petri est décidable.

3.2.1 Spécifications modales

3.2.1.1 Définitions

Définition 3.2.1 (Spécifications modales). Une spécification modale est un tuple $S = \langle \{C_a\}_{a \in \Sigma}, I \rangle$ où, pour tout $a \in \Sigma$, C_a est un langage rationnel des mots qui *doivent* être prolongeables par une action a et I est le langage rationnel des mots interdits. L'*opérateur de complétion* associé à S, noté C_S est l'application $C_S : 2^{\Sigma^*} \rightarrow 2^{\Sigma^*}$ définie par : $C_S(L) = \bigcup_{a \in \Sigma} (L \cap C_a).a$.

Une spécification modale définit un ensemble de modèles qui sont des langages clos par préfixe. La sémantique d'une spécificaton modale est définie par un ensemble de modèles de la manière suivante :

$$mod(S) = \{L \subseteq \Sigma^* \mid C_S(L) \subseteq L \wedge L \cap I = \emptyset\}$$

À partir de cette définition, nous disons que S est *satisfiable* si $mod(S) \neq \emptyset$ et que L *satisfait* S si $L \in mod(S)$.

Les modèles de S sont les langages qui satisfont les deux conditions suivantes :
- pour tout mot w de L dans C_a, $w.a$ doit être un mot de L,
- aucun mot de L ne doit appartenir à I.

Remarquons que les modèles d'une spécification modale ne sont pas nécessairement des langages rationnels. Toutefois, les spécifications modales étant définies pour être équivalentes (en terme de modèles) à un fragment du Mu-Calcul, elles héritent

de la *propriété du modèle fini* comme nous le montrons dans la suite, ce qui entraîne que lorsque $mod(S)$ est non-vide, S possède un modèle rationnel.

3.2.1.2 Représentation graphique

Afin d'être en mesure de donner des exemple visuels, nous définissons une représentation graphique des spécifications modales sous la forme d'*automates modaux*. Un automate modal regroupe tous les composants d'une spécification modale. Un automate modal est un automate sans état final où chaque arc est représenté soit par un trait plein soit par un trait pointillé.

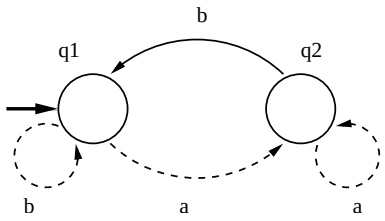

FIG. 3.1 – Un automate modal

Soit $\Sigma = \{a, b, c\}$. Nous notons $\mathcal{L}(q)$ le langage de l'automate lorsque l'état q est l'état final et où toute transition est considérée comme une transition d'automate usuel, c.-à-d. en remplaçant les transitions en pointillées par des transitions en traits pleins. Dans la Figure 3.1 $\mathcal{L}(q_1) = (a^*b^+)^*$ et $\mathcal{L}(q_2) = (a^*b^+)^*.a^+$.

Intuitivement, un automate modal se comprend relativement aux règles suivantes

1. un arc plein sortant d'un état q, étiqueté par a et entrant dans un état q' signifie que toute séquence w de transitions (w est un mot du langage à reconnaître) qui mène à l'état q *doit* être prolongeable par a et que $w.a$ mène à l'état q' ;
2. un arc pointillé sortant d'un état q, étiqueté par a et entrant dans un état q' signifie que toute séquence w *peut* être prolongée par la transition a, auquel cas $w.a$ mène à l'état q' ;
3. l'absence d'arc étiqueté par a sortant d'un état q signifie que la transition a est interdite après toute séquence menant à l'état q.

Ces trois règles informelles peuvent se reformuler en terme de spécification modale. Lorsque $S = \langle \{C_a\}_{a\in\Sigma}, I\rangle$ est la spécification associée à l'automate, les trois règles deviennent :

1. un arc plein sortant d'un état q et étiqueté par a signifie $\mathcal{L}(q) \subseteq C_a$
2. les arcs pointillés sont présents pour définir le langage $\mathcal{L}(q)$ de tout état q de l'automate
3. l'absence d'arc étiqueté par a sortant d'un état q signifie $\mathcal{L}(q).a \subseteq I$

Exemple 3.2.2. *L'automate de la figure 3.1 représente la spécification modale* $S = \langle \{C_a\}_{a\in\Sigma}, I\rangle$ *avec* $C_a = \emptyset$, $C_b = (b^*a)^*$, $C_c = \emptyset$ *et* $I = (a+b)^*.c$.

3.2.2 Cohérence et S-clôture

Nous disons qu'une spécification $S = \langle \{C_a\}_{a\in\Sigma}, I\rangle$ est *cohérente* lorsque "S est satisfiable" implique $I \cap C_S(\Sigma^*) = \emptyset$. Intuitivement, une spécification cohérente est une spécification que ne donne pas de contrainte contradictoire, c.-à-d. que pour tout mot w, aucune action a est simultanément imposée par S ($w \in C_a$) et interdite par S ($w.a \in I$).

Lemme 3.2.3. *Pour toute spécification modale satisfiable, il existe un spécification modale cohérente possédant le même ensemble de modèles.*

Démonstration. Soit $S = \langle \{C_a\}_{a\in\Sigma}, I\rangle$ une spécification modale satisfiable, nous construisons la spécification modale $S' = \langle \{C'_a\}_{a\in\Sigma}, I\rangle$ telle que, pour tout $a \in \Sigma$, $C'_a = C_a \setminus \{w \in \Sigma^* \mid w.a \in I\}$. Par construction $I \cap C_S(\Sigma^*) = \emptyset$, donc S' est cohérente. Il est évident que $mod(S) = mod(S')$. □

À partir de ce point, nous considérons uniquement des spécifications modales cohérentes. Supposons qu'un langage L vérifie $L \cap I = \emptyset$ mais pas $C_S(L) \subseteq L$, il est possible de "compléter" L afin d'obtenir un modèle de S.

Définition 3.2.4 (S-clôture). La *S-clôture* d'un langage clos par préfixe L, notée $L{\uparrow^S}$, est le plus petit langage L' tel que $L \subseteq L'$ et $L' \in mod(S)$.

Lemme 3.2.5. *La S-clôture d'un langage rationnel est rationnelle.*

Démonstration. Nous montrons cette propriété en construisant un automate fini qui reconnaît la S-clôture d'un langage clos par préfixe L ; la construction s'effectue en utilisant la méthode suivante ;

1. construire l'automate $\mathcal{A}$ qui reconnaît $L \cup \bigcup_{a\in\Sigma} C_a.a$,
2. supprimer de $\mathcal{A}$ tous les états non terminaux. Cette étape produit un nouvel automate $\mathcal{A}'$ qui reconnaît le plus grand langage clos par préfixe et inclus dans $L \cup \bigcup_{a\in\Sigma} C_a.a$,
3. retourner $\mathcal{L}(\mathcal{A}')$.

Puisque L est clos par préfixe, alors $L \subseteq \mathcal{L}(\mathcal{A}')$ et nous avons $L\uparrow^S \subseteq \mathcal{L}(\mathcal{A}')$; en effet, si $L\uparrow^S \not\subseteq \mathcal{L}(\mathcal{A}')$ alors, puisque $L\uparrow^S$ et $\mathcal{L}(\mathcal{A}')$ sont clos par préfixe, il existe $w \in L\uparrow^S$ et $a \in \Sigma$ tels que $w.a \in \mathcal{L}(\mathcal{A}')$ et $w.a \notin L\uparrow^S$, d'où $w \in C_a$, ce qui contredit $L\uparrow^S \in mod(S)$. □

Le lemme suivant propose une nouvelle définition de à la S-clôture.

Lemme 3.2.6. *La S-clôture d'un langage clos par préfixe L est la plus petite solution de l'équation $R = L \cup C_S(R)$.*

Démonstration. Par définition $L\uparrow^S \in mod(S)$, ce qui entraîne $C_S(L\uparrow^S) \subseteq L\uparrow^S$. Puisque $L \subseteq L\uparrow^S$, nous avons $L \cup C_S(L\uparrow^S) \subseteq L\uparrow^S$. Le langage $L\uparrow^S$ étant le plus petit langage vérifiant l'inclusion, nous obtenons l'égalité $L\uparrow^S = L \cup C_S(L\uparrow^S)$. □

Exemple 3.2.7. *Soit S la spécification modale de la Figure 3.1, soit $L = (a^*)$. La S-clôture de L est $L\uparrow^S = (a^* \cup a^*.b)$.*

3.2.3 Ensemble des modèles d'une spécification modale

Les modèles d'une spécification ont une structure de treillis dont les extremums sont des langages rationnels. Nous établissons cette propriété et nous donnons la construction de ces deux modèles particuliers.

Soit $S = \langle \{C_a\}_{a\in\Sigma}, I\rangle$ une spécification modale cohérente. Soit $L^S_\bot$ le langage $\{1\}\uparrow^S$ et soit $L^S_\top$ le langage $\Sigma^* \setminus I.\Sigma^*$.

Lemme 3.2.8. *Ces quatre propositions sont équivalentes :*

1. *S est satisfiable*
2. *$L^S_\bot \in mod(S)$*
3. *$L^S_\bot \cap I = \emptyset$*
4. *$L^S_\top \in mod(S)$*

Démonstration. Les implications $2 \Rightarrow 1$, $4 \Rightarrow 1$ et $2 \Rightarrow 3$ sont triviales, nous montrons $3 \Rightarrow 2$, $1 \Rightarrow 4$ et $4 \Rightarrow 3$.

- $3 \Rightarrow 2$: le Lemme 3.2.6 nous donne $L^S_\bot = \{1\} \cup C_S(L^S_\bot)$, et par conséquent $C_S(L^S_\bot) \subseteq L^S_\bot$; par hypothèse, $L^S_\bot \cap I = \emptyset$, donc $L^S_\bot \in mod(S)$;
- $1 \Rightarrow 4$: S est cohérente, ce qui signifie $C_S(\Sigma^*) \cap I = \emptyset$, de plus $C_S(L^S_\top) \subseteq C_S(\Sigma^*)$, alors $C_S(L^S_\top) \subseteq L^S_\top$ et $L^S_\top \cap I = \emptyset$, finalement $L^S_\top \in mod(S)$;
- $4 \Rightarrow 3$: $L^S_\top \cup C_S(L^S_\top) = L^S_\top$ alors $L^S_\bot \subseteq \{1\} \cup C_S(L^S_\top) \subseteq L^S_\top$ et $L^S_\top \cap I = \emptyset$; nous obtenons $L^S_\bot \cap I = \emptyset$.

□

Ce lemme confère la *propriété du modèle fini* de $\mathbf{L}_\mu$ aux spécifications modales : si S est satisfiable, alors elle possède un modèle rationnel ($L^S_\bot$ est rationnel par le Lemme 3.2.5 et $L^S_\top$ est rationnel par définition). Ces deux modèles sont, par construction, les extrema des modèles de la spécification S ordonnée par l'inclusion : $L^S_\top$ est le plus grand modèle et $L^S_\bot$ est le plus petit.

Théorème 3.2.9. *Si S est satisfiable alors $(mod(S), \subseteq)$ est un treillis complet distributif.*

Démonstration. La preuve est immédiate par la définition de $mod(S)$. □

Exemple 3.2.10. *Soit S la spécification modale de la Figure 3.2. Quelques modèles de S sont représentés sur la Figure 3.3 ; les boites représentent les modèles (sous la forme des automates les reconnaissant) et les flèches entre les boites représentent l'inclusion des langages. Dans cette spécification, $L_\top = L7$ et $L_\bot = L1$. Il existe un nombre infini de modèles entre $L2$ et $L4$, entre $L3$ et $L5$ ou entre $L4$ et $L7$. Le modèle $L6$ montre que $L4 \cup L5 \neq L7$.*

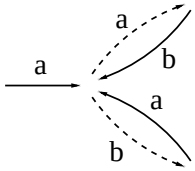

FIG. 3.2 – La spécification modale S

3.2.4 Une approche compositionnelle des spécifications modales

Nous donnons maintenant une approche compositionnelle des spécifications modales. Nous montrons que chaque spécification peut s'exprimer comme une composition de spécifications simples combinées par un ensemble d'opérateurs. Dans la section suivante, ce résultat sera utilisé pour construire une spécification associée à une formule de $\mathbf{L}_\nu$. Dans un premier temps, nous définissons les opérateurs.

3.2.4.1 Spécifications atomiques et leurs opérateurs

Définition 3.2.11. Soit $S_1 = \langle \{C^1_a\}_{a\in\Sigma}, I^1\rangle$ et $S_2 = \langle \{C^2_a\}_{a\in\Sigma}, I^2\rangle$ deux spécifications modales.

– L'*intersection* des deux spécifications est la spécification $S_1 \cap S_2 = \langle \{C^1_a \cup C^2_a\}_{a\in\Sigma}, I^1 \cup I^2\}\rangle$.

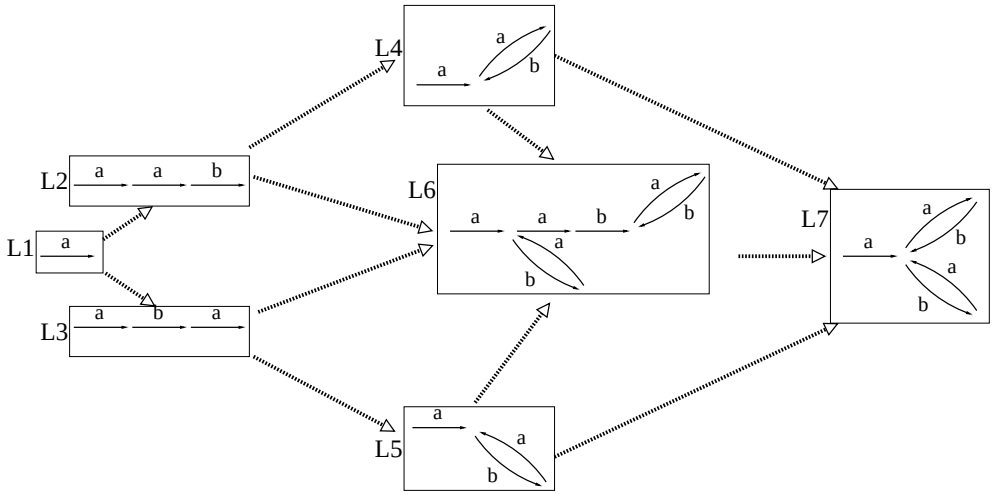

FIG. 3.3 – Quelques éléments du treillis des modèles de S

– le *préfixage* de la spécification S_1 par un langage $R \subseteq \Sigma^*$ est la spécification $R.S_1 = \langle \{R.C^1_a\}_{a\in\Sigma}, R.I^1 \rangle$.

L'intersection de deux spécifications correspond au "et" logique ; en effet, un langage est modèle de l'intersection de si et seulement si il est modèle des deux spécifications d'origine. Le lemme suivant établi ce résultat :

Lemme 3.2.12. *Soient S_1 et S_2 deux spécifications modales, $mod(S_1 \cap S_2) = mod(S_1) \cap mod(S_2)$.*

Démonstration. Pour tout $L \in mod(S_1 \cap S_2)$, nous avons $C_{S_1\cap S_2}(L) \subseteq L$ et $C_{S_1\cap S_2}(L) = C_{S_1}(L) \cup C_{S_2}(L)$, donc $C_{S_1}(L) \subseteq L$ et $C_{S_2}(L) \subseteq L$. De plus, $(I_1 \cup I_2) \cap L = \emptyset$, donc $I_1 \cap L = I_2 \cap L = \emptyset$, par conséquent $L \in mod(S_1) \cap mod(S_2)$. Réciproquement, $L \in mod(S_1) \cap mod(S_2)$ implique $C_{S_1}(L) \cup C_{S_2}(L) \subseteq L$ et $I_1 \cap L = I_2 \cap L = \emptyset$; finalement $L \in mod(S_1 \cap S_2)$. □

Le préfixage d'une spécification S par un langage R est la spécification dont les modèles sont exactement les langages L dont le suffixe $L_{w\backslash}$ de chaque mot w de R satisfait la spécification S.

Lemme 3.2.13. *Pour tout $L \subseteq \Sigma^*$,*

$$L \in mod(R.S) \Leftrightarrow \forall w \in R, L_{w\backslash} = \emptyset \text{ ou } L_{w\backslash} \in mod(S)$$

où $L_{w\backslash}$ est l'ensemble des suffixes de w dans L.

Démonstration. Pour cette preuve, les implications

$$L \cap v.\Sigma^* = v.L_{v\backslash} \quad \Rightarrow \quad L \cap v.L' = v.(L_{v\backslash} \cap L') \tag{3.1}$$
$$w.L_1 \subseteq L_2 \quad \Rightarrow \quad L_1 \subseteq L_{2w\backslash} \tag{3.2}$$

sont utilisées à plusieurs reprises sans le mentionner.

$\Rightarrow$) soit $L \in mod(R.S)$, par construction de $R.S$:

$$C_{R.S}(L) = \bigcup_{a\in\Sigma}(L \cap R.C_a).a = \bigcup_{a\in\Sigma}\bigcup_{w\in R}(L \cap w.C_a).a = \bigcup_{a\in\Sigma}\bigcup_{w\in R} w.(L_{w\backslash} \cap C_a).a$$

Puisque $C_{R.S}(L) \subseteq L$, pour tout $w \in R$ et pour tout $a \in \Sigma^*$, $w.(L_{w\backslash} \cap C_a).a \subseteq L$, alors $C_s(L_{w\backslash}) \subseteq L_{w\backslash}$. Similairement, puisque $L \cap R.I = \emptyset$, nous obtenons, pour tout $w \in R$, $L \cap w.I = \emptyset$ ce qui implique $L_{w\backslash} \cap I = \emptyset$. Finalement $L_{w\backslash} = \emptyset$ ou $L_{w\backslash} \in mod(S)$.

$\Leftarrow$) On montre d'abord que pour tout $a \in \Sigma$, $(L \cap R.C_a).a \subseteq L$. Soit $v \in L \cap R.C_a$, il existe $w \in R$ et $u \in C_a$ tels que $v = wu$. Nous avons alors $u \in L_{w\backslash} \cap C_a$ avec $L_{w\backslash} \neq \emptyset$. Par hypothèse, $L_{w\backslash} \in mod(S)$, donc $(L_{w\backslash} \cap C_a).a \subseteq L_{w\backslash}$ et en particulier $u.a \in L_{w\backslash}$. Nous en déduisons $v.a = w.u.a \in L$. Nous montrons maintenant $L \cap R.I = \emptyset$: pour tout $w \in R$, si $L_{w\backslash} = \emptyset$ alors $L \cap w.I = \emptyset$; sinon $L_{w\backslash} \in mod(S)$ alors $L \cap w.I = \emptyset$; finalement $L \cap R.I = \emptyset$. □

Définition 3.2.14 (Spécification atomique). Une spécification *atomique* est une des spécifications suivantes :
$S_{\texttt{true}} = \langle \{\emptyset\}_{a\in\Sigma}, \emptyset \rangle$,
$S_{\not\rightarrow^b} = \langle \{\emptyset\}_{a\in\Sigma}, \{b\} \rangle$ et
$S_{\rightarrow^b} = \langle \{C_a\}_{a\in\Sigma}, \emptyset \rangle$, avec $C_a = \emptyset$ pour $a \neq b$ et $C_b = \{1\}$.

L'ensemble des modèles de chaque spécification atomique s'obtient immédiatement par la définition de la spécification ; les ensembles des modèles de chaque spécification atomique sont donnés ci-dessous :

$$\begin{aligned} mod(S_{\texttt{true}}) &= \{L \in \Sigma^*\} \\ mod(S_{\not\rightarrow_b}) &= \{L \subseteq \Sigma^* \mid b \notin L\} \\ mod(S_{\rightarrow_b}) &= \{L \subseteq \Sigma^* \mid b \in L\} \end{aligned}$$

3.2.4.2 Approche compositionnelle

Théorème 3.2.15. *Chaque spécification modale peut s'exprimer comme une composition de spécifications atomiques à l'aide des opérateurs d'intersection et de préfixage.*

Démonstration. Soit $S = \langle \{C_a\}_{a\in\Sigma}, I\rangle$ une spécification modale, pour tout a dans Σ nous définissons l'ensemble $I^a = \{u \in \Sigma^* \mid u.a \in I\}$. Soit $S' = \langle \{C'_a\}_{a\in\Sigma}, I'\rangle$ la spécification définie par

$$S' = \bigcup_{a\in\Sigma} C_a.S_{\rightarrow a} \cup \bigcup_{a\in\Sigma} I^a.S_{\not\rightarrow a}$$

D'après les Définitions 3.2.11 et 3.2.14, il est évident que $S = S'$. □

3.3 Équivalence entre le Nu-Calcul conjonctif et les spécifications modales

Cette section est dédiée à la preuve du théorème suivant :

Théorème 3.3.1. *Pour tout ensemble E de langages clos par préfixe, E est l'ensemble des modèles d'une sentence β de $\mathbf{L}_\nu$ si et seulement si il existe une spécification modale S telle que $E = mod(S)$.*

Afin de prouver ce théorème, nous introduisons la notion de chemin de variable :

Définition 3.3.2 (Chemins de variables). Soit β une formule de $\mathbf{L}_\nu$, nous définissons l'application $P_\beta : var(\beta) \rightarrow \mathcal{P}(\Sigma^*)$ par induction sur la structure de β :
pour tout $X \in var(\beta)$, si
- $\beta \in \{\mathtt{true}, \rightarrow^a, \not\rightarrow^a\}$, alors $P_\beta(X) = \emptyset$,
- $\beta = Y$ et $Y \neq X$, alors $P_\beta(X) = \emptyset$,
- $\beta = X$, alors $P_\beta(X) = \{1\}$,
- $\beta = [a]\alpha$, alors $P_\beta(X) = a.P_\alpha(X)$,
- $\beta = \beta_1 \wedge \beta_2$, alors $P_\beta(X) = P_{\beta_1}(X) \cup P_{\beta_2}(X)$,
- $\beta = \nu Y.\alpha(Y)$, alors $P_\beta(X) = P_\alpha(Y)^*.P_\alpha(X)$.

Le langage $P_\beta(X)$ est l'ensemble des *chemins de la variable X* dans β.

Exemple 3.3.3. *Quelques exemples de chemins de variables :*
- *si $\beta = [a]X$, alors $P_\beta(X) = \{a\}$,*
- *si $\beta = [a][b]X \wedge [c]X$, alors $P_\beta(X) = (a.b + c)$,*
- *si $\beta = \nu Y.([a][b]Y \wedge [c]X)$, alors $P_\beta(X) = (a.b)^*.c$*

Les chemins de la variable X dans β sont les mots qui "mènent" à une occurrence de X dans la formule : lorsque $w \in [\![\, \beta \,]\!]^{val}_L$, $P_\beta(X)$ est l'ensemble des mots v tels que $w.v \in [\![\, X \,]\!]^{val}_L$ ou de manière équivalente $w.v \in val(X) \subseteq L$.

3.3.1 D'une sentence vers une spécification

Nous donnons la construction d'une spécification modale S_β à partir d'une sentence β de $\mathbf{L}_\nu$ telle que $mod(S_\beta)$ soit l'ensemble des modèles de β. Cette construction constitue la preuve de la première implication du Théorème 3.3.1 : E est l'ensemble des modèles d'une sentence de $\mathbf{L}_\nu$ implique l'existence d'une spécification S telle que $mod(S) = E$.

Cette preuve s'effectue inductivement sur la structure de la sentence β ; par conséquent, il nous est nécessaire de prouver ce résultat pour toute *formule* de $\mathbf{L}_\nu$. Puisque les modèles d'une spécifications modale ne dépendent d'aucune valuation d'un ensemble de variables, nous introduisons l'hypothèse suivante, qui dépend d'une valuation val, d'une formule β, d'un langage L et d'un mot w de L :

$$\forall X \in var(\beta),\ w.P_\beta(X) \cap L \subseteq val(X) \tag{3.3}$$

L'hypothèse (3.3) énonce que les mots de L qui coïncident avec les mots d'un chemin de variable, pour une variable X, doivent être dans $val(X)$.

Définition 3.3.4 (Spécification modale associée à une formule de $\mathbf{L}_\nu$). Nous définissons la spécification modale S_β *associée à la formule* $\beta \in \mathbf{L}_\nu$ inductivement sur la structure de β :
- $\beta \in \{\mathtt{true}, \rightarrow^a, \not\rightarrow^a\}$, S_β est donné par la définition 3.2.14,
- $\beta = X$, $S_\beta = S_{\mathtt{true}}$,
- $\beta = [a]\alpha$, $S_\beta = a.S_\alpha$,
- $\beta = \beta_1 \wedge \beta_2$, $S_\beta = S_{\beta_1} \cap S_{\beta_2}$,
- $\beta = \nu Y.\alpha(Y)$, $S_\beta = P_\alpha(Y)^*.S_\alpha$.

Exemple 3.3.5. *Soit* $\beta = [a]\nu X.([b]X \wedge \rightarrow^a \wedge \not\rightarrow^c)$*, la spécification modale associée à* β *est* $(a.b^*).(S_{\rightarrow^a} \cap S_{\not\rightarrow^c})$*, c.-à-d. la spécification* $S_\beta = \langle \{C_a\}_{a\in\Sigma}, I\rangle$ *avec :*

$$C_a = (a.b^*), C_b = \emptyset, C_c = \emptyset, I = (a.b^*.c)$$

La spécification associée à une formule est construite de telle manière que, lorsque β est une sentence, S_β et β ont le même ensemble de modèles. Comme nous l'avons mentionné, nous montrons un résultat plus large, valable pour toute formule de $\mathbf{L}_\nu$. Ce résultat est celui de la proposition suivante :

Proposition 3.3.6. *Soit* $\beta \in \mathbf{L}_\nu$*, val une valuation, L un langage clos par préfixe et w un mot de L.*

$$w \in [\![\, \beta \,]\!]^{val}_L \Leftrightarrow L_{w\backslash} \in mod(S_\beta) \text{ et l'hypothèse (3.3) est vérifiée}$$

La première implication du Théorème 3.3.1 est un corollaire de la Proposition 3.3.6 (il s'agit du cas particulier où β est une sentence et $w = 1$) :

Corollaire 3.3.7 (de la Proposition 3.3.6). *Pour tout sentence β de $\mathbf{L}_\nu$, S_β et β ont le même ensemble de modèles.*

Soit $\beta \in \mathbf{L}_\nu$, *val* une valuation, L un langage clos par préfixe et w un mot de L. La preuve de la Proposition 3.3.6 repose sur les trois lemmes suivants :

Lemme 3.3.8.
$$w \in [\![\, \beta \,]\!]_L^{val} \Rightarrow \textit{hypothèse (3.3)}$$

Lemme 3.3.9.
$$w \in [\![\, \beta \,]\!]_L^{val} \Rightarrow L_{w\backslash} \in mod(S_\beta)$$

Lemme 3.3.10.
$$L_{w\backslash} \in mod(S_\beta) \textit{ et l'hypothèse (3.3)} \Rightarrow w \in [\![\, \beta \,]\!]_L^{val}$$

Démonstration du Lemme 3.3.8. Soit $w \in [\![\beta]\!]_L^{val}$. La preuve est par induction sur la structure de β :

- $\beta \in \{\mathtt{true}, \rightarrow^a, \not\rightarrow^a\}$, $var(\beta) = \emptyset$,
- $\beta = X$, $var(\beta) = \{X\}$, alors $P_X(X) = \{1\}$ et $w \in [\![\, X \,]\!]_L^{val}$, donc $w \in val(X) \Rightarrow w.\{1\} \subseteq val(X)$
- $\beta = [a]\alpha$, $var(\beta) = var(\alpha)$ et $P_\beta(X) = a.P_\alpha(X)$, donc $w.P_\beta(X) \cap L = w.a.P_\alpha(X) \cap L$. Puisque $w.a \in [\![\alpha]\!]_L^{val}$, par hypothèse d'induction, $w.P_\beta(X) \cap L \subseteq val(X)$,
- $\beta = \beta_1 \wedge \beta_2$, alors $w.P_\beta(X) \cap L = (w.P_{\beta_1}(X) \cap L) \cup (w.P_{\beta_2}(X) \cap L)$ et par hypothèse d'induction, $(w.P_{\beta_1}(X) \cap L) \cup (w.P_{\beta_2}(X) \cap L) \subseteq val(X)$,
- $\beta = \nu Y.\alpha(Y)$, nous montrons par induction sur n que :
$$w.P_\alpha(Y)^n.P_\alpha(X) \cap L \subseteq val(X)$$
Dans la suite, nous précisons, lorsqu'une hypothèse d'induction est utilisée, s'il s'agit de l'hypothèse de l'induction sur n ou de celle sur β.
Posons $V = [\![\, \beta \,]\!]_L^{val}$; nous avons que $w \in [\![\, \beta \,]\!]_L^{val}$ est équivalent à $w \in [\![\, \alpha(Y) \,]\!]_L^{val(V/Y)}$.
  - Pour $n = 0$, par hypothèse d'induction sur β, $w.P_\alpha(X) \cap L \subseteq val(X)$
  - Pour $n + 1$, nous avons
  $$w.P_\alpha(Y)^{n+1}.P_\alpha(X) \cap L = w.P_\alpha(Y).P_\alpha(Y)^n.P_\alpha(X) \cap L$$
  Par hypothèse d'induction sur β, $w \in [\![\, \alpha(Y) \,]\!]_L^{val(V/Y)}$, ce qui implique $w.P_\alpha(Y) \cap L \subseteq V$ et donc pour tout $v \in w.P_\alpha(Y) \cap L$, $v \in V$; par hypothèse d'induction sur n, nous obtenons $v.P_\alpha(Y)^n.P_\alpha(X) \cap L \subseteq val(X)$ et finalement $w.P_\alpha(Y).P_\alpha(Y)^n.P_\alpha(X) \cap L \subseteq val(X)$

□

Démonstration du Lemme 3.3.9. La preuve est par induction sur la structure de β :

- $\beta \in \{\mathtt{true}, \rightarrow^a, \not\rightarrow^a, X\}$, par la Définition 3.2.14, $L_{w\backslash} \in mod(S_\beta)$,
- $\beta = [a]\alpha$, $S_\beta = a.S_\alpha$, si $a \in L_{w\backslash}$ alors $w.a \in [\![\, \alpha \,]\!]_L^{val}$; par hypothèse d'induction, $L_{w.a\backslash} \in mod(S_\alpha)$. Par le Lemme 3.2.13, nous obtenons $L_{w\backslash} \in mod(S_\beta)$.
- $\beta = \beta_1 \wedge \beta_2$, par le Lemme 3.2.12 nous obtenons $mod(S_\beta) = mod(S_{\beta_1}) \cap mod(S_{\beta_2})$; par hypothèse d'induction, $L_{w\backslash} \in mod(S_\beta)$.
- $\beta = \nu X.\alpha(X)$. Soit $V = [\![\, \beta \,]\!]_L^{val}$, nous avons $V = [\![\, \alpha(X) \,]\!]_L^{val(V/X)}$. Nous montrons pour tout n et pour tout $v \in (P_{\alpha(X)}(X))^n$, $w.v \in L \Rightarrow L_{w.v\backslash} \in mod(S_\alpha)$ par induction sur n.
  - Pour $n = 0$, $w \in [\![\, \alpha(X) \,]\!]_L^{val(V/X)}$ et par hypothèse d'induction sur β, $L_{w\backslash} \in mod(S_\alpha)$.
  - Pour $n+1$, $w \in V$, puisque $v = u.u'$ avec $u \in P_{\alpha(X)}(X)$, par le Lemme 3.3.8 nous avons $(u' \in L_{w.u\backslash}) \Rightarrow L_{w.u.u'\backslash} \in val(X) = V$. Par hypothèse d'induction sur n, puisque $u' \in (P_{\alpha(X)}(X))^n$, nous obtenons $L_{w.u.u'\backslash} \in mod(S_\alpha)$.

  Finalement, pour tout $v \in (P_{\alpha(X)}(X))^*$, $w.v \in L \Rightarrow L_{w.v\backslash} \in mod(S_\alpha)$. Nous appliquons le Lemme 3.2.13 pour obtenir $L_{w\backslash} \in mod(S_\beta)$.

□

Démonstration du Lemme 3.3.10. La preuve est par induction sur β :

- $\beta \in \{\mathtt{true}, \rightarrow^a, \not\rightarrow^a\ X\}$, par la Définition 3.2.14, $w \in [\![\, \beta \,]\!]_L^{var}$,
- $\beta = [a]\alpha$, $S_\beta = a.S_\alpha$, si $a \in L_{w\backslash}$ le Lemme 3.2.13 nous assure $L_{w.a\backslash} \in mod(S_\alpha)$ puis par hypothèse d'induction, $w.a \in [\![\, \alpha \,]\!]_L^{val}$. Dans les deux cas, $w \in [\![\, \beta \,]\!]_L^{val}$,
- $\beta = \beta_1 \wedge \beta_2$, $S_\beta = S_{\beta_1} \cup S_{\beta_2}$, par le Lemme 3.2.12 nous obtenons $L_{w\backslash} \in mod(S_{\beta_1}) \cap mod(S_{\beta_2})$, et par la Définition 3.3.2 ainsi que l'hypothèse (3.3), pour tout $v \in P_\beta(X)$, $v \in P_{\beta_1}(X) \cup P_{\beta_2}(X)$. Nous pouvons appliquer l'hypothèse d'induction, pour β_1 et β_2 afin d'obtenir $w \in [\![\, \beta \,]\!]_L^{val}$,
- $\beta = \nu X.\alpha(X)$, nous montrons que $(L \cap w.P_\alpha(X)^*)$ est un post-point-fixe, c.-à-d. :

$$(L \cap w.P_\alpha(X)^*) \subseteq [\![\, \alpha \,]\!]_L^{val((L \cap w.P_\alpha(X)^*)/X)}$$

  Pour tout $v \in (L_{w\backslash} \cap P_\alpha(X)^*)$:

  1) $w \in mod(S_\beta) \Rightarrow w.v \in mod(S_\alpha)$ (par le Lemme 3.2.13),
  2) pour tout $Y \in var(\beta)$ $(Y \neq X)$, $w.P_\beta(Y) \cap L \subseteq val(Y)$ et $P_\beta(Y) = (P_\alpha(X))^* P_\alpha(Y)$ implique

$$w.v.P_\alpha(Y) \cap L \subseteq val(Y)$$

3) Pour X, $v \in (L_{w\setminus} \cap P_\alpha(X)^*)$ implique

$$w.v.P_\alpha(X) \cap L \subseteq val((L \cap w.P_\alpha(X)^*)/X)$$

Les items 2) et 3) nous donnent l'hypothèse (3.3) qui, combinée avec 1) nous permettent d'appliquer l'hypothèse d'induction afin d'obtenir $w.v \in [\![\, \alpha \,]\!]_L^{var((L \cap w.P_\alpha(X)^*)/X)}$. Nous avons prouvé que $(L \cap w.P_\alpha(X)^*)$ est un post-point-fixe et $w \in (L \cap w.P_\alpha(X)^*)$; nous obtenons finalement $w \in [\![\, \beta \,]\!]_L^{val}$. □

Avec ces trois derniers Lemmes, la preuve de la Proposition 3.3.6 est immédiate :

Démonstration de la Proposition 3.3.6. $\Rightarrow$) est donné par le Lemme 3.3.10
$\Leftarrow$) est donné par les Lemmes 3.3.8 et 3.3.9. □

3.3.2 D'une spécification vers une sentence

Nous donnons la construction de la sentence β_S de $\mathbf{L}_\nu$ à partir d'une spécification modale S telle que l'ensemble des modèles de β_S soit l'ensemble $mod(S)$. Il s'agit d'une preuve constructive de la deuxième implication du Théorème 3.3.1. L'idée repose sur l'expression de S comme composition de spécifications atomiques en utilisant le Théorème 3.2.15 puis sur la construction de β_S étape par étape de telle manière que S_{β_S} et S soient structurées de manière identique, et possèdent par conséquent le même ensemble de modèles.

Lemme 3.3.11. *Pour tout langage rationnel $R \subseteq \Sigma^*$ il est possible de construire une formule $\alpha_R(X) \in \mathbf{L}_\nu$ telle que pour toute sentence β de $\mathbf{L}_\nu$, $S_{\alpha_R(\beta/X)} = R.S_\beta$.*

Démonstration. Puisque R est un langage rationnel, il peut s'exprimer par une expression régulière sur Σ. Afin de prouver la propriété inductivement sur R, nous donnons une grammaire pour les expressions régulières adaptée à cette preuve

$$\{1\} \mid a.R_1 \mid R_1 \cup R_2 \mid R_1^*$$

où $a \in \Sigma$. Nous construisons inductivement $\alpha_R(X)$ et nous montrons à chaque étape que pour tout $\beta \in \mathbf{L}_\nu$, $S_{\alpha_R(\beta/X)} = R.S_\beta$ et $P_{\alpha_R(X)}(X) = R$:

- $R = \{1\}$: soit $\alpha_R(X) = X$, nous avons trivialement $S_{\alpha_R(\beta/X)} = R.S_\beta$ et $P_{\alpha_R(X)}(X) = R$,
- $R = a.R_1$: soit $\alpha_R(X) = [a]\alpha_{R_1}(X)$, par la Définition 3.3.4 et l'hypothèse d'induction, nous avons $S_{\alpha_R(\beta/X)} = R.S_\beta$, et par la Définition 3.3.2 et l'hypothèse d'induction, nous avons $P_{\alpha_R(X)}(X) = R$,

- $R = R_1 \cup R_2$: soit $\alpha_R(X) = \alpha_{R_1}(X) \wedge \alpha_{R_2}(X)$, par la Définition 3.3.4 et l'hypothèse d'induction, nous avons $S_{\alpha_R(\beta/X)} = R.S_\beta$, et par la Définition 3.3.2 et l'hypothèse d'induction, nous avons $P_{\alpha_R(X)}(X) = R$,
- $R = R_1^*$: soit $\alpha_R(X) = \nu Y.\alpha_{R_1}(Y/X) \wedge X$. Puisque par hypothèse d'induction $P_{S_{\alpha_{R_1}(Y/X)}} = R_1.S_\beta$, par la Définition 3.3.4 nous avons $S_{\alpha(\beta/X)} = R_1^*.S_\beta = R.S_\beta$. La Définition 3.3.2 entraîne immédiatement $P_{\alpha_R(X)}(X) = R$.

□

Lemme 3.3.12. *Pour toute spécification modale S, il est possible de construire une sentence β_S de $\mathbf{L}_\nu$ telle que S et β_S aient le même ensemble de modèles.*

Démonstration. Par le Théorème 3.2.15, nous avons une décomposition de S à partir de laquelle nous construisons inductivement une formule β_S telle que $S_{\beta_S} = S$, le seul opérateur non trivial est celui de préfixage, pour lequel la preuve est donnée par le Lemme 3.3.11. □

Exemple 3.3.13. *Soit S la spécification modale de la Figure 3.2, une décomposition de S est :*

$$S = S_{\not\rightarrow^b} \cup S_{\rightarrow^a} \cup a.(a.(b.a)^*.S_{\rightarrow^b} \cap b.(a.b)^*.S_{\rightarrow^a})$$

la sentence équivalente est alors :

$$\beta_S = \not\rightarrow^b \wedge \rightarrow^a \wedge [a](\,[a]\nu X.(\,[b][a]X \wedge \rightarrow^b) \wedge [b]\nu Y.(\,[a][b]X \wedge \rightarrow^a)\,)$$

3.3.3 Preuve du Théorème 3.3.1

Nous avons établi une traduction d'une sentence de $\mathbf{L}_\nu$ vers une spécification modale, ainsi qu'une traduction inverse, ce qui suffit pour prouver le théorème principal de cette section :

Démonstration du théorème 3.3.1. Soit E l'ensemble des modèles d'une sentence β, par le Corollaire 3.3.7, $E \in mod(S_\beta)$. Réciproquement, si $E = mod(S)$ alors par le Lemme 3.3.12, il existe β_S telle que E soit l'ensemble des modèles de β_S. □

Une conséquence de cette preuve est que lorsque nous établissons des propriétés uniformes sur les spécifications modales, les sentences de $\mathbf{L}_\nu$ héritent immédiatement de ces propriétés ; c'est le cas de la structure de treillis des modèles, établie avec le Théorème 3.2.9.

3.4 Synthèse à partir de spécifications modales

Nous présentons ici une extension à la synthèse de réseaux. Nous définissons une restriction sur les spécifications modales et nous proposons un algorithme de synthèse.

3.4.1 Réseaux de Petri et spécifications modales

On dit qu'un réseau de Petri $(\mathcal{N}, m_0)$ *satisfait* une spécification modale S lorsque $\mathcal{L}(\mathcal{N}, m_0) \in mod(S)$; on dira alors que $(\mathcal{N}, m_0)$ est modèle de S.

Nous avons montré qu'une spécification modale peut représenter un ensemble infini d'intervalles de langages (voir l'Exemple 3.2.10) ; toute technique existante de synthèse de réseau ne peut donc en général pas s'appliquer au cas des spécifications modales.
De plus, l'approche usuelle de la synthèse consistant à produire le réseau possédant le plus petit langage satisfaisant certaines conditions, puis à vérifier si ce réseau est modèle de la spécification n'est pas adaptée aux spécifications modales. En effet, cette approche repose sur le fait que les résultats de synthèse actuels sont valables pour des spécifications (intervalles de langages, graphes automatiques) où un sous-ensemble des comportement de tout réseau solution est connu *a priori* et où le plus petit réseau possédant ce sous-ensemble de comportements est un modèle de la spécification si et seulement s'il existe un réseau modèle de cette spécification. Dans le cas d'une spécification modale S, le langage $L^S_\top$ représente un ensemble de comportement que tout réseau modèle de S possède nécessairement. Cependant, l'existence d'un réseau modèle de S ne garantit pas que le plus petit langage de réseau L contenant $L^S_\top$ soit modèle de S. Cela vient du fait que le langage L ne vérifie pas nécessairement $C_S(L) \subseteq L$.

3.4.2 Une famille de spécifications

Définition 3.4.1. Les langages $\mathfrak{S}_0$ et $\mathfrak{S}_1$ On définit deux sous-ensembles de Σ^* : $\mathfrak{S}_0$ désigne les langages finis de mots de Σ^*, et par conséquent les langages qui peuvent être définis par une expression régulière sans employer l'étoile ; $\mathfrak{S}_1$ désigne les langages de mots de Σ^* qui sont définis par une union finie de langages du type $(u.v^*.w)$ où u, v, w sont des mots de Σ^* avec $w \neq 1$. On dit d'une spécification $S = \langle \{C_a\}_{a\in\Sigma}, I\rangle$ qu'elle est $\mathfrak{S}_0$ (resp. $\mathfrak{S}_1$, $(\mathfrak{S}_1 \cup \mathfrak{S}_0)$) si ses composantes C_a pour tout $a \in \Sigma$ et I sont dans $\mathfrak{S}_0$ (resp. $\mathfrak{S}_1$, $(\mathfrak{S}_1 \cup \mathfrak{S}_0)$). Lorsque $L \in \mathfrak{S}_1$, on nomme *taille* de L le nombre de langages du type $(u.v^*.w)$ qu'il contient. On nomme

également *taille* d'une spécification $S = \langle \{C_a\}_{a\in\Sigma}, I\rangle$ qui est $\mathfrak{S}_1$ la somme des tailles des C_a pour $a \in \Sigma$ et de la taille de I.

On se donne $S = \langle \{C_a\}_{a\in\Sigma}, I\rangle$, avec l'hypothèse que S est satisfiable.

Définition 3.4.2. $\mathfrak{S}_1$-triplet modal Un *$\mathfrak{S}_1$-triplet modal* est un triplet (T, L, J) où T est une spécification modale $\mathfrak{S}_1$, L est un langage rationnel et J est un langage *fini* tel que $J \cap L = \emptyset$. On dit qu'un langage L' clos par préfixe satisfait le triplet (T, L, J) si et seulement si $L' \in mod(T)$, $L \subseteq L'$ et $L' \cap J = \emptyset$. On note $mod(T, L, J)$ l'ensemble des langages satisfaisant (T, L, J).

Nous utilisons maintenant la notion de $\mathfrak{S}_1$-triplet modal pour nous abstraire des composantes $\mathfrak{S}_0$ de S. Le lemme suivant permet de passer d'une spécification $\mathfrak{S}_0 \cup \mathfrak{S}_1$ à une union finie de $\mathfrak{S}_1$-triplets modaux.

Lemme 3.4.3. *Pour toute spécification modale S dans $(\mathfrak{S}_0 \cup \mathfrak{S}_1)$ il existe un ensemble fini $\{(T_k, L_k, J_k)\}_{k\in K}$ de $\mathfrak{S}_1$-triplets modaux tel que pour tout $k \in K$, J_k est un langage fini et tel que $mod(S) = \bigcup_{k\in K} mod(T_k, L_k, J_k)$.*

Démonstration. L'ensemble des triplets modaux se construit de la manière suivante :

1. on exprime la spécification S comme l'intersection de deux spécifications S^0 et S^1 respectivement $\mathfrak{S}_0$ et $\mathfrak{S}_1$,
2. On construit un ensemble fini de couples de langages $\{(L_k, J_k)\}_{k\in K}$ où les J_k sont des langages $\mathfrak{S}_0$ et tels que $mod(S^0) = \{L \subseteq \Sigma^* \text{ tel que } \exists k \in K, L_k \subseteq L \text{ et } L \cap J_k = \emptyset\}$,
3. on construit les $\mathfrak{S}_1$-triplet (T_k, L_k, J_k) en posant pour tout $k \in K$, $T_k = S^1$.

Nous prouvons uniquement le résultat de l'étape 2 c.-à-d qu'une spécification $\mathfrak{S}_0$ est équivalente à la donnée d'un ensemble fini de couples de langages $\{(L_k, J_k)\}_{k\in K}$ où les J_k sont des langages $\mathfrak{S}_0$.

Soit $S_0 = \langle \{C_a\}_{a\in\Sigma}, I\rangle$ une spécification $\mathfrak{S}_0$, puisque pour tout a, C_a est un langage fini, le langage $C_{S_0}(\Sigma^*)$ est également fini ; pour tout langage R clos par préfixe, nous avons par définition $C_{S_0}(R) \subseteq C_{S_0}(\Sigma^*)$. Lorsque R est modèle de S_0, nous avons $C_{S_0}(R) \subseteq R$ et donc $\overline{C_{S_0}(R)} \subseteq R$. Soit $\{L_k\}_{k\in K}$ le sous-ensemble fini des langages $\overline{L}$ où $L \in 2^{C_{S_0}(\Sigma^*)}$, les L_k représentent l'ensemble des valeurs possibles de $\overline{C_{S_0}(R)}$ pour tout langage R. Soit $\{I_k\}_{k\in K}$ l'ensemble fini où I_k est l'ensemble $\bigcup_{a\in\Sigma} C_a \setminus L_k$; on pose alors $J_k = I_k \cap I$, remarquons que par définition I_k est $\mathfrak{S}_0$ (puisqu'il s'agit d'un sous-langage de $\bigcup_{a\in\Sigma} C_a$) et donc que J_k est $\mathfrak{S}_0$. Par construction, un langage R est modèle de S_0, si et seulement s'il existe k tel que $\overline{C_S(R)} = L_k$ et $R \cap J_k = \emptyset$. □

Exemple 3.4.4. *Soit S la spécification sur l'alphabet $\Sigma = \{a, b\}$ définie par $C_a = \{b.b + a.(b.a)^*\}$, $C_b = \{1\}$, $I = \{b.b.b + b.(a + b).a.(a + b) + a.(b.a)^*.a\}$, la décomposition s'effectue la manière suivante :*

1. *S^1 est définie par $C_a^1 = \{a.(b.a)^*\}$, $C_b^1 = \emptyset$, $I^1 = \{a.(b.a)^*.a\}$ et S^0 est définie par $C_a^0 = \{b.b\}$, $C_b^0 = \{1\}$, $I^0 = \{b.b.b + b.(a + b).a\}$,*
2. *nous obtenons deux couples de langages (L_1, J_1) et (L_2, J_2) avec :*

$$\begin{array}{lcllcl} L_1 & = & \{1 + b\} & J_1 & = & \{b.b + b.b.b + b.a + b.(a + b).a\} \\ L_2 & = & \{1 + b + b.b + b.b.a\} & J_2 & = & \{b.b.b + b.(a + b).a\} \end{array}$$

3. *au final, nous obtenons deux $\mathfrak{S}_1$-triplets modaux : (S^1, L_1, J_1) et (S^1, L_2, J_2).*

De fait, d'après le Lemme 3.4.3, il existe un réseau $\mathcal{N}$ tel que son langage est modèle d'une spécification modale S si et seulement si il existe un $\mathfrak{S}_1$-triplet modal dont $\mathcal{L}(\mathcal{N})$ est modèle.

3.4.3 Un algorithme de synthèse

Dans un premier temps, nous définissons un ensemble de modèles particuliers d'un $\mathfrak{S}_1$-triplet modal.

Définition 3.4.5 (Modèles permissifs). Soit (T, L, J) un $\mathfrak{S}_1$-triplet modal avec $T = \langle \{C_a\}_{a\in\Sigma}, I\rangle$, un *modèle permissif* de (T, L, J) est un réseau $(\mathcal{N}, m_0)$ dont toute place p vérifie les deux conditions suivantes :

(i) $L \subseteq \mathcal{L}(\{p\})$,

(ii) pour toute composante[1], $u.w^*.v$ de T, $<p, w> \geq 0$,

(iii) pour tout $a \in \Sigma$ et pour tout $u.w^*.v$ dans C_a, $u.v.a \in \mathcal{L}(\{p\})$.

et tel que

(iv) pour tout $u.w^*.v \in I$, il existe p dans $\mathcal{N}$ tel que $u.v \notin L(\{p\})$ et $<p, w> = 0$.

(v) pour tout $u \in J$, il existe p dans $\mathcal{N}$ tel que $u \notin L(\{p\})$

On se donne un $\mathfrak{S}_1$-triplet modal (T, L, J) avec $T = \langle \{C_a\}_{a\in\Sigma}, I\rangle$.

Lemme 3.4.6. *Un modèle permissif de (T, L, J) est un modèle de (T, L, J).*

[1] Cette composante peut être dans n'importe lequel des langages constituant T, c.-à-d. dans C_a pour un certain $a \in \Sigma$ où dans I.

Démonstration. Soit $(\mathcal{N}, m_0)$ un modèle permissif de (T, L, J). Pour toute place p de $\mathcal{N}$, la condition (i) nous assure $L \subseteq \mathcal{L}(\mathcal{N}, m_0)$; la condition (v) nous assure $\mathcal{L}(\mathcal{N}, m_0) \cap J = \emptyset$; la condition (iii) combinée à la condition (ii) nous assure $C_T(\Sigma^*) \subseteq \mathcal{L}(\mathcal{N}, m_0)$ et donc $C_T(\mathcal{L}(\mathcal{N}, m_0)) \subseteq \mathcal{L}(\mathcal{N}, m_0)$; la condition (iv) nous assure enfin $\mathcal{L}(\mathcal{N}, m_0) \cap I = \emptyset$. Nous en déduisons que $(\mathcal{N}, m_0)$ est un modèle de (T, L, J). □

Lemme 3.4.7. *L'existence d'un modèle permissif de (T, L, J) est décidable.*

Démonstration. L'ensemble des places de réseau satisfaisant (i), (ii) et (iii) forme un cône convexe polyédral, pour vérifier (v), il suffit de synthétiser le réseau $(\mathcal{N}, m_0)$ composé des rayons extrémaux de ce cône et de vérifier $\mathcal{L}(\mathcal{N}, m_0) \cap J = \emptyset$. Pour vérifier (iv), pour chaque composante $u.w^*.v$ de I, on synthétise le réseau $(\mathcal{N}^{u,w,v}, m_0^{u,w,v})$ composé des rayons extrémaux du cône des places satisfaisant (i), (ii), (iii) et $<p, w> = 0$ et on vérifie $\mathcal{L}(\mathcal{N}^{u,w,v}, m_0^{u,w,v}) \cap \{u.v\} = \emptyset$.

Lorsque ces propriétés sont vérifiées, l'union des réseaux synthétisés dans le processus est un modèle permissif de S. □

Le lemme suivant est le lemme qui permet à la synthèse d'être effective ; sa preuve étant très technique et présentant peu d'intêret, elle n'est pas présentée dans la version finale de ce document. Elle est cependant disponible en contactant directement l'auteur. Ce lemme permet de traiter le cas où (T, L, J) ne possède pas de modèle permissif.

Lemme 3.4.8. *Si (T, L, J) ne possède aucun modèle permissif alors il existe un ensemble fini $\{(T_k, L_k, J_k)\}_{k \in K}$ de $\mathfrak{S}_1$-triplets modaux, tous de taille strictement inférieure à la taille de (T, L, J) et tels que tout réseau modèle de (T, L, J) soit également modèle de (T_k, L_k, J_k) pour un certain $k \in K$.*

Nous sommes maintenant en mesure d'énoncer le résultat principal de cette section.

Théorème 3.4.9. *Soit S une $(\mathfrak{S}_0 \cup \mathfrak{S}_1)$ spécification modale, l'existence d'un réseau de Petri $\mathcal{N}$ tel que $\mathcal{L}(\mathcal{N}) \in mod(S)$ est décidable.*

Démonstration. L'algorithme de synthèse est le suivant :

Étape 1 : calculer l'ensemble Triplets $= \{(T_k, L_k, J_k)\}_{k \in K}$ de $\mathfrak{S}_1$-triplets modaux en utilisant le résultat du Lemme 3.4.3 ;

Étape 2 : pour chaque $(T, L, J) \in$ Triplets, soit il existe un modèle permissif, auquel cas la synthèse est effective d'après les Lemmes 3.4.7 et 3.4.6, et l'algorithme s'achève. Soit un tel modèle n'existe pas, et nous utilisons le

résultat du Lemme 3.4.8 pour construire un ensemble fini $\{(T_h, L_h, J_h)\}_{h\in H}$ de $\mathfrak{S}_1$-triplets modaux que l'on ajoute à Triplets ;

Étape 3 : si l'ensemble Triplets est vide, la synthèse n'a pas de solutions ; sinon reprendre à l'Étape 2.

Cet algorithme termine car la taille des T de chaque triplet (T, L, J) dans Triplets décroît strictement à chaque itération de l'algorithme. □

Ce résultat peut enfin être étendu à :

Théorème 3.4.10. *Soit S une $(\mathfrak{S}_0 \cup \mathfrak{S}_1)$ spécification modale et R un langage de $\mathfrak{S}_0$, l'existence d'un réseau de Petri $\mathcal{N}$ tel que $\mathcal{L}(\mathcal{N}) \in mod(R^*.S)$ est décidable et effective.*

Démonstration. Soit $w \in R$, supposons qu'il n'existe pas de réseau $(\mathcal{N}', m'_0)$ modèle de $R^*.S$ avec $w \notin \mathcal{L}(\mathcal{N}', m'_0)$; soit $(\mathcal{N}, m_0)$ un modèle de $R^*.S$, nous avons par hypothèse $w \in \mathcal{L}(\mathcal{N}, m_0)$, ce qui entraîne, par le Lemme 3.2.13, que $(\mathcal{N}, m)$ est modèle de $R^*.S$ avec $m_0[w\rangle m$ et l'hypothèse s'applique de nouveau ; par induction, nous obtenons $w^* \in \mathcal{L}(\mathcal{N}, m_0)$. Par conséquent, pour tout mot $w \in R$, soit il existe un réseau $(\mathcal{N}', m'_0)$ modèle de $R^*.S$ avec $w \notin \mathcal{L}(\mathcal{N}', m'_0)$, soit tout réseau $(\mathcal{N}', m'_0)$ modèle de $R^*.S$ vérifie $w^* \in \mathcal{L}(\mathcal{N}', m'_0)$. Nous en déduisons que l'existence d'un réseau modèle de $R^*.S$ implique l'existence d'un réseau $(\mathcal{N}, m_0)$ modèle de $R^*.S$ et d'un sous-ensemble R' de R tels que pour tout mot w de $R \setminus R'$, $w^* \in \mathcal{L}(\mathcal{N}, m_0)$ et $u \notin \mathcal{L}(\mathcal{N}, m_0)$ pour tout mot u de R'.

Posons $S = \langle \{C_a\}_{a\in\Sigma}, I\rangle$, la procédure de synthèse consiste à rechercher, pour tout sous ensemble R' de R, un réseau modèle de $\langle \{C_a\}_{a\in\Sigma}, I \cap R'\rangle$ qui vérifie, pour tout $w \in R'$ et pour toute place p, $<p, w> = 0$; pour chaque R', la procédure de synthèse reste la procédure présentée précédement, mais la contrainte (i) des modèles permissifs devient :

(i) $L \subseteq \mathcal{L}(\{p\})$ et pour tout $w \in R'$, $<p, w> = 0$.

Cette dernière contrainte est nécessaire puisque toutes les places p qui permettent de vérifier les conditions (iv) et (v) doivent le faire uniformément suivant w. □

3.5 Conclusion

Dans ce chapitre, nous avons défini un fragment syntaxique du Mu-Calcul pour la synthèse de réseau, sur la base des résultats d'indécidabilité du chapitre précédent. Ce fragment, le Nu-Calcul conjonctif entre dans le cadre des

travaux de [Esp94] : le model checking de réseau de Petri pour des formules du Nu-Calcul conjonctif est décidable. Nous avons également introduit un reconnaisseur, les spécification modales, pour une approche basée sur les langages des formules du Nu-Cacul. Nous avons établi l'équivalence entre le Nu-Calcul et les spécifications modales. Nous avons également montré que les spécifications modale sont adaptées à l'extraction d'un sous-ensemble, les spécifications $R^*.(\mathfrak{S}_0 \cup \mathfrak{S}_1)$, pour lequel la synthèse de réseau est décidable puisque nous avons établi un nouveau résultat de synthèse. Ce résultat étend strictement les travaux de [BD99] dès qu'une spécification modale possède une composante $\mathfrak{S}_1$. En effet, dans ce cas la spécification possède une infinité de modèles, et la synthèse au cas par cas n'est pas envisageable ; ceci n'est pas vrai dans le cas des spécifications $\mathfrak{S}_0$ qui définissent une union finie d'intervalles de langages pour lesquels il est possible de décider de la synthèse en les considérant un par un. Ce résultat diffère également de celui de [BD02] (concernant les spécifications automatiques) dès qu'une spécification est $\mathfrak{S}_1$, puisque dans ce cas, d'une part l'ensemble des états du système à synthétiser n'est pas initialement connu, d'autre part il n'est pas nécessairement régulier.

Chapitre 4

Théorie structurelle d'un réseau

Sommaire

L'objet de ce chapitre est, dans un premier temps, de caractériser un réseau (non initialisé) $\mathcal{N}$ en utilisant les sentences de $\mathbf{L}_\nu$ interprétées sur la classe $\mathbf{PN}$. Nous construisons, pour tout réseau $\mathcal{N}$, une sentence $\varphi_\mathcal{N}$ qui caractérise la structure de $\mathcal{N}$. Il n'est pas possible de construire une sentence ϕ telle que $\mathcal{N}$ soit l'unique modèle de ϕ sur la classe $\mathbf{PN}$. Par conséquent, pour caractériser la structure d'un réseau, nous donnons une relation de préordre $\sqsubseteq$, purement structurelle (indépendante des marquages initiaux), sur les éléments de $\mathbf{PN}$; la sentence $\varphi_\mathcal{N}$, nommée *théorie de* $\mathcal{N}$, ainsi construite (sur un alphabet plus grand que celui sur lequel opère $\mathcal{N}$) est telle que les modèles de cette sentence sur la classe $\mathbf{PN}$ soient (après projection sur l'alphabet de $\mathcal{N}$) les réseaux $\mathcal{N}'$ qui vérifient $\mathcal{N} \sqsubseteq \mathcal{N}'$. Nous montrons ensuite que la sentence $\varphi_\mathcal{N}$ peut s'exprimer sous la forme, plus simple, d'un intervalle de langage.

Ce résultat nous permet, dans un second temps, d'établir la preuve de l'indécidabilité du problème $\mathbf{Sat}(\mathbf{L}_\nu,\mathbf{PN})$. Par opposition au réseau de compteur $\mathcal{C}$ du Chapitre 2 qui simulait le test à zéro par l'*inactivation* d'une transition, le raisonnement de ce chapitre utilise un réseau de compteurs particulier $\mathcal{N}_{k_0,k_1}$ qui permet de simuler le test à zéro par le *tir* d'une transition. La preuve utilise la théorie de $\mathcal{N}_{k_0,k_1}$ conjointement à l'énoncé, sous forme d'une sentence de $\mathbf{L}_\nu$, d'un problème indécidable sur les machines à deux compteurs.

4.1 Préliminaires

4.1.1 La logique $\mathbf{L}_\nu$

Dans ce chapitre, nous considérons la grammaire suivante pour $\mathbf{L}_\nu$:

$$(\mathbf{L}_\nu \ni) \quad \beta_1, \beta_2 ::= \mathtt{true} \mid X \mid \langle a \rangle \beta_1 \mid [a]\beta_1 \mid \not\rightarrow^a \mid \beta_1 \wedge \beta_2 \mid \nu X.\beta_1(X)$$

La sémantique associée à $\mathbf{L}_\nu$ est la sémantique sur les langages, elle est présentée dans la le Chapitre 2, Définition 2.6.1 page 60.

Remarque 4.1.1. La syntaxe diffère de celle présentée dans le Chapitre 3 (voir Définition 3.1.1 page 66) par le fait qu'elle intègre, par commodité, l'opérateur $\langle a \rangle$ et non plus l'opérateur $\rightarrow^a$. Remarquons que les deux définitions de $\mathbf{L}_\nu$ sont équivalentes puisque pour toute formule $\beta \in \mathbf{L}_\nu$, pour toute valuation val et pour tout langage L, nous avons :

$$[\![\langle a \rangle \beta]\!]_L^{[val]} = [\![[a]\beta \wedge \rightarrow^a]\!]_L^{[val]} \quad \text{et} \quad [\![\rightarrow^a]\!]_L^{[val]} = [\![\langle a \rangle \mathtt{true}]\!]_L^{[val]}$$

4.1.2 Les réseaux de Petri considérés

Dans ce chapitre, nous considérons uniquement la représentation vectorielle des réseaux de Petri.

Note 4.1.2. Nous ne ferons pas la différence entre une place p de réseau et la place $\lambda.p$ obtenue par multiplication du vecteur p par un facteur $\lambda \in \mathbb{Q}$ strictement positif[1]. On dira que p et $\lambda.p$ sont *colinéaires*.

Le lemme suivant montre qu'il est possible de remplacer p par $\lambda.p$ dans un réseau de Petri sans en changer le langage. On rappelle que le langage d'un réseau est l'intersection des langages de ses places (voir Proposition 1.4.19, page 23)

Lemme 4.1.3. *Soient p et p' deux places colinéaires, pour tout entier n_0, il existe un entier n'_0 tel que $\mathcal{L}(\{p\}, m_0) = \mathcal{L}(\{p'\}, m'_0)$ avec $m_0(p) = n_0$ et $m_0(\lambda.p) = n'_0$.*

Démonstration. Posons $p' = \lambda.p$ et $n'_0 = \lfloor \lambda.n_0 \rfloor$. On montre que pour toute séquence de tir u tirable dans $(\{p\}, m_0)$ et $(\{p'\}, m'_0)$ et pour tout $a \in \Sigma$, a est tirable apres u dans $(\{p\}, m_0)$ si et seulement si a est tirable dans $(\{p'\}, m'_0)$. Dans $(\{p\}, m_0)$, la relation $m_0[u.a\rangle$ s'écrit[2] :

$$m_0(p) + \sum_{b \in \Sigma} |u|_b \times <p, b> \geq <p, {}^\bullet a>$$

soit

$$n_0 + \sum_{b \in \Sigma} |u|_b \times <p, b> - <p, {}^\bullet a> \geq 0$$

Dans $(\{p'\}, m'_0)$ la relation $m_0[u.a\rangle$ s'écrit :

$$m_0(p') + \sum_{b \in \Sigma} |u|_b \times <p', b> \geq <p', {}^\bullet a>$$

soit

$$\lfloor \lambda.n_0 \rfloor + \sum_{b \in \Sigma} |u|_b \times \lambda.<p, b> - \lambda.<p, {}^\bullet a> \geq 0$$

Puisque $\lambda.<p, b>$ et $\lambda.<p, {}^\bullet a>$ sont des entiers, $m_0[u.a\rangle$ s'écrit :

$$n_0 + \sum_{b \in \Sigma} |u|_b \times <p, b> - <p, {}^\bullet a> \geq 0$$

Donc les relations de tir de $(\{p\}, m_0)$ et $(\{p'\}, m'_0)$ sont identiques. □

[1] Le vecteur $\lambda.p$ n'est une place de réseau que si toutes ses composantes sont dans $\mathbb{N}$.

[2] Voir l'Équation 1.4 page 28.

On notera alors $\mathcal{N} \subseteq \mathcal{N}'$ lorsque pour tout $p \in \mathcal{N}$, il existe λ strictement positif dans $\mathbb{Q}$ tel que $\lambda.p$ est dans $\mathcal{N}'$.

À partir de ce point et pour la suite de ce chapitre, nous considérons uniquement des réseaux de Petri *purs*. Autrement dit, la classe **Pn** fera référence aux réseaux de Petri purs.

Définition 4.1.4 (Réseaux réduits). On dit d'une place p qu'elle est *réduite* si elle vérifie

$$\mathsf{PGCD}(<p, a_1>, \ldots, <p, a_n>) = 1$$

Un réseau de Petri est *réduit* lorsque toutes ses places sont réduites.

Remarque 4.1.5. Soit p une place de réseau réduite. Toute place p' colinéaire à p vérifie nécessairement $p' = \lambda.p$ avec λ *entier*.

On nommera *forme réduite* d'un réseau $\{p_1, \ldots, p_l\}$ le réseau $\{p'_1, \ldots, p'_l\}$ tel que les places $p'_1, \ldots p'_l$ sont réduites et telles que pour tout $i \in [1, l]$, p'_i et p_i soient colinéaires.

4.2 La théorie d'un réseau est un intervalle de langages

Dans cette section nous nous intéressons uniquement aux structures des réseaux de Petri, c.-à-d. que nous ne considérons pas le marquage initial. Nous montrons que pour un marquage initial bien choisi, il est possible de spécifier précisément un grand nombre de propriétés structurelles d'un réseau avec une spécification qui porte uniquement sur le "régime permanent" du réseau. Les spécifications que nous donnons ici sont des sentences de $\mathbf{L}_\nu$, dont nous montrerons ensuite qu'elles peuvent se ramener à des spécifications de $\mathbf{Int}_\mathbf{\Sigma}$.

La procédure consiste à utiliser un alphabet Σ' qui est un sur-ensemble strict de l'alphabet Σ sur lequel opèrent les réseaux considérés. Nous donnons les définitions utiles pour passer d'un alphabet à l'autre et nous montrons comment il est possible de s'abstraire du marquage initial dans des sentences de $\mathbf{L}_\nu$.

Soit $\mathcal{N} = \{p_1, \ldots, p_k\}$ un réseau sur Σ', on note $\mathcal{N}|_\Sigma$ la projection de $\mathcal{N}$ sur l'alphabet $\Sigma \subseteq \Sigma'$ définie par $\mathcal{N}' = \{p_1|_\Sigma, \ldots, p_k|_\Sigma\}$ où $p|_\Sigma$ désigne la projection du vecteur p sur les composantes correspondant à Σ.

Définition 4.2.1. Soit $\mathcal{N}$ un réseau de Petri sur un alphabet Σ. On dit que $\mathcal{N}$ est *prolongé* sur un alphabet Σ' en un réseau $\mathcal{N}'$ lorsque $\Sigma \subsetneq \Sigma'$ et que $\mathcal{N}'|_\Sigma = \mathcal{N}$.

On confondra alors les marquages de $\mathcal{N}$ et $\mathcal{N}'|_\Sigma$.

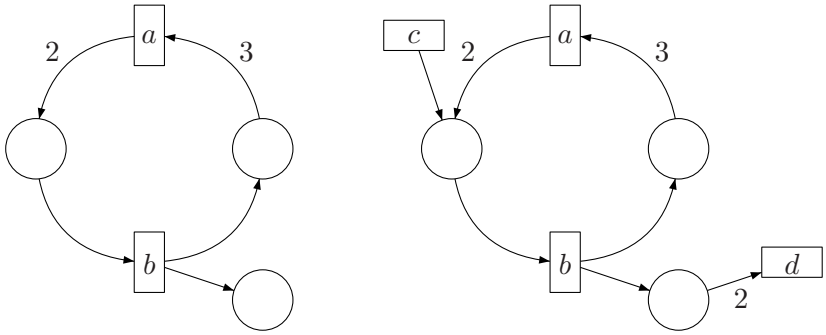

FIG. 4.1 – Un réseau et un de ses prolongements.

Exemple 4.2.2. *La Figure 4.1 représente un réseau sur* $\{a, b\}$ *et un prolongement possible de ce réseau sur* $\{a, b, c, d\}$.

Dans la suite, on notera $\mathcal{N} \models \phi$ lorsqu'il existe m_0 tel que $(\mathcal{N}, m_0) \models \phi$ ou de manière équivalente $[\![\phi]\!]_{\mathcal{N}} \neq \emptyset$. Nous considérerons également un alphabet dont les éléments sont de la forme :

$$a, b, c, a_1, \ldots, a_n, \odot, \oplus, \ominus, \boxplus, \boxminus$$

Lemme 4.2.3. *Soit* $\psi = \langle\odot\rangle\phi$ *une sentence de* $\mathbf{L}_\nu$ *avec* ϕ *une sentence sur l'alphabet* $\Sigma = \Sigma' \setminus \{\odot\}$. *Pour tout réseau* $\mathcal{N}$ *sur* Σ, $\mathcal{N} \models \phi$ *si et seulement si pour tout marquage* m_0 *de* $\mathcal{N}$, *il existe un prolongement* $\mathcal{N}'$ *de* $\mathcal{N}$ *sur* Σ' *tel que* $(\mathcal{N}', m_0) \models \psi$.

Démonstration. Nous montrons les deux implications.

$\Rightarrow$) Puisque $\mathcal{N} \models \phi$, alors il existe m'_0 tel que $(\mathcal{N}, m'_0) \models \phi$. On prolonge $\mathcal{N}$ en $\mathcal{N}'$ en posant, pour tout place p de $\mathcal{N}$

$$<p, \odot> = m'_0(p) - m_0(p)$$

Pour tout p, $m_0(p) + <p, \odot> = m'_0(p) \geq 0$ ce qui implique $m_0[\odot\rangle m'_0$. Puisque ϕ est une sentence sur Σ, $(\mathcal{N}, m'_0) \models \phi$ implique $(\mathcal{N}', m'_0) \models \phi$. Nous obtenons alors $(\mathcal{N}', m_0) \models \psi$.

$\Leftarrow$) Puisque $(\mathcal{N}', m_0) \models \langle\odot\rangle\phi$, nous avons $m_0[\odot\rangle m$, ce qui implique $(\mathcal{N}, m'_0) \models \phi$ et finalement $\mathcal{N} \models \phi$.

□

Le principe du Lemme 4.2.3 est d'*encoder* le marquage initial d'une place p dans la valeur $<p, \odot>$. Tout modèle $\mathcal{N}$ de ϕ peut alors se prolonger en un modèle $(\mathcal{N}', m_0)$ de ψ indépendamment du marquage initial de $\mathcal{N}$. La conséquence est que la satisfaction de ϕ par $\mathcal{N}$ va dépendre uniquement des comportement infinis (du *régime permanent*) de $\mathcal{N}$ puisque le choix du marquage initial est libre. Nous n'aurons alors pas à nous préoccuper de savoir si une transition est tirable au marquage initial.

4.2.1 Équations de places en Nu-calcul Conjonctif

Nous construisons une famille de sentences de $\mathbf{L}_\nu$ qui expriment des équations ou des inéquations sur les vecteurs de places d'un réseau. Cette famille est composée de sentences de deux types :

- les *inéquations universelles* sont les sentences qui expriment des inéquations qui sont vérifiées par tous les vecteurs de places,
- les *équations existentielles* sont les sentences qui expriment l'existence d'une place qui vérifie un ensemble d'équations.

Le principe utilisé (l'expression de comportements *à l'infini*) est une extension de celui de la spécification des places simulant des compteurs dans la sentence de test à zéro, vu au chapitre 2 (Définition 2.4.9, page 45).

On étend l'opérateur $\langle . \rangle$ de $\mathbf{L}_\nu$ à un mot $u \in \Sigma^*$, avec $u = a_1 \ldots . a_k$, en posant :

$$\langle u \rangle \stackrel{\text{def}}{=} \langle a_1 \rangle \ldots \langle a_k \rangle$$

Définition 4.2.4 (Inéquation universelle). Soit $u \in \Sigma^*$, on définit la sentence d'*inéquation universelle* associée à u

$$\Upsilon_\forall[u][\geq 0] \stackrel{\text{def}}{=} \nu X. \langle u \rangle X$$

Lemme 4.2.5. *Soit $u \in \Sigma^*$, $\mathcal{N} \models \Upsilon_\forall[u][\geq 0]$ si et seulement si pour toute place p de $\mathcal{N}$, $<p, u> \geq 0$.*

Démonstration. Puisque $\mathcal{N} \models \Upsilon_\forall[u][\geq 0]$, il existe m_0 tel que $(\mathcal{N}, m_0) \models \Upsilon_\forall[u][\geq 0]$, ce qui est équivalent à $m_0[u\rangle m$ et $(\mathcal{N}, m) \models \Upsilon_\forall[u][\geq 0]$. Par induction nous obtenons

$$(\mathcal{N}, m_0) \models \Upsilon_\forall[u][\geq 0] \Rightarrow \text{ pour tout } k \in \mathbb{N}, m_0[u^k\rangle$$

S'il existe une place p dans $\mathcal{N}$ tel que $<p, u> < 0$, alors il existe k tel que $m_0(p) + k.<p, u> < 0$, ce qui contredit $m_0[u^k\rangle$.

Réciproquement, si pour tout $p \in \mathcal{N}$ $<p, u> \geq 0$, alors il existe m_0 tel que $m_0[u\rangle m$ avec pour toute place p, $m(p) \geq m_0(p)$. Nous en déduisons pour tout $k \in \mathbb{N}$ $m_0[u^k\rangle$, ce qui suffit à prouver $\mathcal{N} \models \Upsilon_\forall[u][\geq 0]$. □

Définition 4.2.6. Soient k mots sur Σ, $u_1, \ldots, u_k$, et soit $a \in \Sigma$, on définit la sentence

$$\gamma_\exists[a][u_1, \ldots, u_k][=0] \stackrel{\text{def}}{=} \nu X.\ \not\rightarrow^a \wedge \langle u_1\rangle X \wedge \ldots \wedge \langle u_k\rangle X$$

Exemple 4.2.7. *Une sentence typique de celle que nous considèrerons dans la suite est la suivante :*

$$\gamma_\exists[\ominus][\ominus.\oplus, \oplus.a, b.\ominus][=0] = \nu X.\ \not\rightarrow^\ominus \wedge \langle \ominus.\oplus\rangle X \wedge \langle \oplus.a\rangle X \wedge \langle b.\ominus\rangle X$$

Lemme 4.2.8. *Soient $u_1, \ldots, u_k \in \Sigma^*$, $\mathcal{N} \models \gamma_\exists[a][u_1, \ldots, u_k][=0]$ si et seulement si*

(i) il existe une place p de $\mathcal{N}$ telle que

$$\begin{array}{rcl} <p, a> & < & 0 \\ <p, u_1> & = & 0 \\ \vdots & \vdots & \vdots \\ <p, u_k> & = & 0 \end{array}$$

et pour tout $i \in [1, k]$, pour tout préfixe w de u_i,

$$<p, w> > <p, a>$$

(ii) pour toute place p de $\mathcal{N}$, pour tout $i \in [1, k]$,

$$<p, u_i> \geq 0$$

Démonstration. Soit L_u le langage défini par l'expression régulière $L_u = (u_1 + \ldots + u_k)^*$. De $\mathcal{N} \models \gamma_\exists[a][u_1, \ldots, u_k][=0]$ nous déduisons qu'il existe m_0 tel que $(\mathcal{N}, m_0) \models \gamma_\exists[a][u_1, \ldots, u_k][=0]$, ce qui nous permet d'écrire :

$$(\mathcal{N}, m_0) \models \gamma_\exists[a][u_1, \ldots, u_k][=0] \text{ ssi } \begin{cases} \text{pour tout } i \in [1, k], m_0[u_k\rangle m \\ \quad \text{avec } (\mathcal{N}, m) \models \gamma_\exists[a][u_1, \ldots, u_k][=0] \\ \text{et } a \text{ est inactive au marquage } m_0 \end{cases}$$

Par induction sur les mots de L_u, nous obtenons :

$$(\mathcal{N}, m_0) \models \gamma_\exists[a][u_1, \ldots, u_k][=0] \text{ ssi } \begin{cases} \text{pour tout } w \in L_u, m_0[w\rangle m \\ \text{et } a \text{ est inactive au marquage } m \end{cases} \tag{4.1}$$

Nous montrons maintenant les deux implications du lemme.

$\Rightarrow$) On montre (ii) puis (i).

(ii) En supposant qu'il existe une place p dans $\mathcal{N}$ et $i \in [1,k]$ telle que $<p,u_i> < 0$, on montre suivant le même principe que pour la preuve du Lemme 4.2.5 que cela contredit $m_0[w\rangle$ pour tout $w \in L_u$.

(i) Posons $\mathcal{N} = \{p_1,\ldots,p_l\}$, supposons que pour toute place p_h de $\mathcal{N}$ il existe $v_h \in \{u_1,\ldots,u_k\}$ tel que $<p_h,v_h> \neq 0$; d'après le résultat (ii) cela implique $<p_h,v_h> > 0$. Par conséquent, il existe $n_h \in \mathbb{N}$ tel que $m_0(p_h) + <p_h,v_h^{n_h}> + <p_h,a> > 0$ (c.-à-d. que p_h n'inactive pas a au marquage m défini par $m_0[v_l^{h_l}\rangle m$).
Nous construisons alors le mot $w = v_1^{n_1}.v_2^{n_2}\ldots.v_l^{n_l}$. Par (ii), nous obtenons $<p_h,w> \geq <p,v_h^{n_h}>$ pour toute place p_h de $\mathcal{N}$. Par conséquent $m_0(p_h) + <p_h,w> + <p_h,a> > 0$ pour tout place p_h de $\mathcal{N}$, ce qui implique $m_0[w.a\rangle$. Puisque $w \in L_u$, d'après l'Équivalence 4.1, ce résultat contredit $(\mathcal{N},m_0) \models \gamma_{\exists}[a][u_1,\ldots,u_k][=0]$. Nous déduisons alors l'existence d'une place p dans $\mathcal{N}$ telle que

$$\begin{array}{rrcl} m_0+ & <p,a> & < & 0 \\ & <p,u_1> & = & 0 \\ & \vdots & \vdots & \vdots \\ & <p,u_k> & = & 0 \end{array}$$

Il nous reste à montrer que pour tout $i \in [1,k]$, pour tout préfixe w de u_i,

$$<p,w> > <p,a>$$

Remarquons que pour tout $i \in [1,k]$, $m_0[u_i\rangle m$ implique, pour tout préfixe w de u_i, $m_0(p) + <p,w> \geq 0$; or $m_0(p) + <p,a> < 0$, ce qui implique $<p,w> > <p,a>$.

$\Leftarrow$) Soit p la place vérifiant (i). Soit $\mathcal{N}'$ le réseau tel que $\mathcal{N} = \mathcal{N}' \uplus \{p\}$. Soit m le marquage qui affecte à chaque place p' de $\mathcal{N}'$ un nombre suffisant de jetons tel que $m[u_i\rangle$ pour tout $i \in [1,k]$ dans $\mathcal{N}'$. Puisque toute place de $\mathcal{N}'$ vérifie (ii), pour tout $i \in [1,k]$, le marquage m' tel que $m[u_i\rangle m'$ vérifie $m'(p') \geq m(p')$ pour toute place p' de $\mathcal{N}'$. Nous en déduisons $m[w\rangle$ pour tout $w \in L_u$ dans $\mathcal{N}'$.

Soit m_p le marquage qui associe $-(<p,a>+1)$ jetons à la place p (rappelons que $<p,a> < 0$ d'après (i)). On montre que $m_p[u_i\rangle$ dans le réseau $\{p\}$ pour tout $i \in [1,k]$. D'après (i), nous avons $<p,w> > <p,a>$ pour tout préfixe w de u_i. Nous en déduisons $m_p(p) + <p,w> > -1$ et par conséquent $m_p(p) + <p,w> \geq 0$. Nous avons alors prouvé $m_p[u_i\rangle$ pour tout $i \in [1,k]$. De (i), nous déduisons, pour tout $i \in [1,k]$, $m_p[u_i\rangle m_p$. Par induction nous obtenons

$m_p[w\rangle m_p$ pour tout $w \in L_u$. De plus puisque a est inactive au marquage m_p (par définition de m_p), alors a est inactive après toute séquence de tir $w \in L_u$.
Soit m_0 le marquage de $\mathcal{N}$ vérifiant pour toute place p'' :

$$m_0(p'') = \begin{cases} m_p(p'') \text{ si p"= p} \\ m(p'') \text{ sinon} \end{cases}$$

Le marquage m_0 ainsi construit vérifie : pour tout $w \in L_u$, $m_0[w\rangle m$ et a est inactive au marquage m. Par l'Équivalence 4.1 nous obtenons $\mathcal{N} \models \gamma_\exists[a][u_1, \ldots, u_k][=0]$.

□

Le Lemme 4.2.8 montre que la sentence $\gamma_\exists[u_1, \ldots, u_k][a][=0]$ exprime à la fois un ensemble d'équations de la forme $<p, u_i> = 0$ et un ensemble d'inéquations universelles. Nous montrons maintenant comment éliminer les inéquations universelles et obtenir une sentence qui exprime exactement l'existence d'une place p dans $\mathcal{N}$ telle que :

$$\forall i \in [1, k], <p, u_i> \quad = \quad 0 \tag{4.2}$$

Pour cela, nous avons à nouveau recours à une extension Σ_e de l'alphabet Σ : $\Sigma_e = \Sigma \cup \Sigma_\circ$ avec $\Sigma_\circ = \{\oplus, \ominus\}$. Les nouveaux événements $\oplus$ et $\ominus$ introduisent pour chaque place p deux nouvelles quantités, $<p, \oplus>$ et $<p, \ominus>$, qui disparaissent par projection sur Σ. Ces quantités sont utilisées pour différencier la place p, qui vérifie le système d'équations (4.2), des autres places de $\mathcal{N}$. Pour p, les valeurs $<p, \oplus>$ et $<p, \ominus>$ sont imposées : elles sont utilisées de manière à ce que p vérifie $<p, \ominus> < 0$ et $<p, \oplus> = 0$. Pour tout autre place p' du réseau il sera possible d'affecter librement une valeur à $<p', \oplus>$ et $<p', \ominus>$ afin de vérifier les inéquations universelles, en particulier nous imposons $<p', \oplus> \geq 0$.

Définition 4.2.9 (Équations existentielles). Soient k mots sur Σ, $u_1, \ldots, u_k$, on définit la sentence

$$\Upsilon_\exists[\oplus, \ominus][u_1, \ldots, u_k][=0] \stackrel{\text{def}}{=} \nu X.\ \not\rightarrow^{\ominus} \wedge \langle\oplus\rangle X \wedge \langle u_1.\oplus\rangle X \wedge \ldots \wedge \langle u_k.\oplus\rangle X$$

Remarque 4.2.10. Par construction, nous avons

$$\Upsilon_\exists[\oplus, \ominus][u_1, \ldots, u_k][=0] \stackrel{\text{def}}{=} \gamma_\exists[\ominus][v, v_1, \ldots, v_k][=0]$$

avec $v = \oplus$ et pour tout $i \in [1, k]$, $v_i = u_i.\oplus$.

Lemme 4.2.11. *Soient k mots sur Σ^*, $u_1, \ldots, u_k$. Soit $\mathcal{N}$ un réseau sur Σ, il existe un prolongement $\mathcal{N}'$ de $\mathcal{N}$ sur Σ_e tel que $\mathcal{N}' \models \Upsilon_\exists[\oplus, \ominus][u_1, \ldots, u_k][=0]$ si et seulement s'il existe une place p de $\mathcal{N}$ telle que*

$$\begin{array}{rcl} <p, u_1> & = & 0 \\ \vdots & \vdots & \vdots \\ <p, u_k> & = & 0 \end{array}$$

Démonstration. Soit $v = \oplus$ et pour tout $i \in [1, k]$, $v_i = u_i.\oplus$.

$\Rightarrow$) D'après la Remarque 4.2.10, $\mathcal{N}' \models \gamma_\exists[\ominus][v, v_1, \ldots, v_k][=0]$; le Lemme 4.2.8 item (i) nous assure de l'existence d'une place p vérifiant $<p, \oplus> = 0$ et pour tout $i \in [1, k]$,

$$<p, u_i. \oplus > = 0$$

Nous en déduisons pour tout $i \in [1, k]$,

$$<p, u_i> = 0$$

$\Leftarrow$) Nous exhibons un prolongement $\mathcal{N}'$ de $\mathcal{N}$ en donnant les valeurs $<p'', \oplus>$ et $<p'', \ominus>$ pour toute place p'' de $\mathcal{N}$.

– Pour p : soit $A_i = \{<p, w> \mid w \text{ préfixe de } u_i\}$, on pose

$$<p, \ominus> \quad = \quad \min(-1, \min(A_1) - 1, \ldots, \min(A_k) - 1) \tag{4.3}$$

et

$$<p, \oplus> = 0$$

remarquons que par construction, $<p, \ominus> < 0$.

– Pour tout $p' \neq p$, on pose

$$<p', \oplus> \quad = \quad \max(-<p', u_1>, \ldots, -<p', u_k>) \tag{4.4}$$

et

$$<p', \ominus> = 0$$

On montre que $\mathcal{N}' \models \gamma_\exists[\ominus][v, v_1, \ldots, v_k][=0]$. De l'Égalité (4.3), nous déduisons, pour tout $i \in [1, k]$ et pour tout préfixe w de $u_i.\oplus$:

$$<p, w> \quad > \quad <p, \ominus> \tag{4.5}$$

Pour tout $i \in [1, k]$, puisque $<p, \oplus> = 0$ et $<p, u_i> = 0$, nous déduisons

$$<p, u_i. \oplus > \quad = \quad 0 \tag{4.6}$$

Pour tout $i \in [1,k]$, pour toute place $p' \neq p$, de l'Égalité (4.4), nous déduisons

$$<p', u_i.\oplus> \;\geq\; 0 \tag{4.7}$$

Nous pouvons maintenant appliquer le Lemme 4.2.8 : les Équations (4.5) et (4.6) nous permettent d'obtenir (i) et les Inéquations (4.7) nous permettent d'obtenir (ii). Nous avons prouvé $\mathcal{N}' \models \gamma_{\exists}[\ominus][v, v_1, \ldots, v_k][=0]$, ce qui d'après la remarque 4.2.10 implique $\mathcal{N}' \models \Upsilon_{\exists}[\oplus, \ominus][u_1, \ldots, u_k][=0]$

□

4.2.2 Théorie d'un réseau exprimée dans $\mathbf{L}_\nu$

On désigne ici par *théorie d'un réseau* $\mathcal{N}$ une sentence de $\mathbf{L}_\nu$ (ou, comme nous le verrons par la suite un intervalle de $\mathbf{INT_\Sigma}$) dont les modèles dans la classe $\mathbf{PN}$ sont les réseaux qui sont liés à $\mathcal{N}$ par une certaine relation structurelle $\sqsubseteq$ (voir Définition 4.2.14 plus loin).
La théorie d'un réseau $\mathcal{N} = \{p_1, \ldots, p_l\}$ sur l'alphabet $\Sigma = \{a_1, \ldots, a_n\}$ s'exprime par une sentence $\varphi_{\mathcal{N}}$ sur l'alphabet étendu $\Sigma_e = \Sigma \cup \Sigma_\circ$ où $\Sigma_\circ = \bigcup_{i\in[1,l]}\{\odot_i, \oplus_i, \ominus_i\}$

Définition 4.2.12 (Théorie d'un réseau). Soit $\mathcal{N}$ un réseau réduit dont les places sont $\{p_1, \ldots, p_l\}$. La *théorie* du réseau *réduit* $\mathcal{N}$ est la sentence

$$\varphi_{\mathcal{N}} = \varphi_1 \wedge \ldots \wedge \varphi_l$$

avec pour tout $i \in [1,l]$:

$$\varphi_i = \langle\odot_i\rangle\gamma_{\exists}[\ominus_i][v_i, u_{i,1}, \ldots, u_{i,n}][=0]$$

où $v_i = \oplus_i.\ominus_i$ et pour tout $j \in [1,n]$, si $<p_i, a_j> > 0$, alors

$$u_{i,j} = a_j.\underbrace{\ominus_i.\ldots.\ominus_i}_{<p_i,a_j> \text{ fois}}$$

si $<p_i, a_j> < 0$, alors

$$u_{i,j} = \underbrace{\oplus_i.\ldots.\oplus_i}_{-<p_i,a_j> \text{ fois}}.a_j$$

et si $<p_i, a_j> = 0$, alors

$$u_{i,j} = a_j$$

La théorie du réseau non réduit $\mathcal{N}'$ est la sentence $\varphi_{\mathcal{N}'} = \varphi_{\mathcal{N}}$ où $\mathcal{N}$ est la forme réduite de $\mathcal{N}'$.

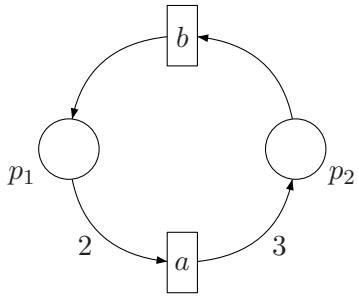

FIG. 4.2 – Le réseau $\mathcal{N}$.

Exemple 4.2.13. *La théorie du réseau $(\mathcal{N}, m_0)$ sur $\Sigma = \{a, b\}$ de la Figure 4.2 est la sentence $\varphi_{\mathcal{N}}$ définie sur le langage $\Sigma_e = \{a, b, \ominus_1, \oplus_1, \ominus_2, \oplus_2\}$ par :*

$$\begin{array}{rcccccccc} \varphi_{\mathcal{N}} = & & \langle\odot_1\rangle\nu X. & \not\rightarrow^{\ominus_1} & \langle\oplus_1.\ominus_1\rangle X & \wedge & \langle\oplus_1.\,\oplus_1\,.a\rangle X & \wedge & \langle b.\ominus_1\rangle X \\ & \wedge & \langle\odot_2\rangle\nu X. & \not\rightarrow^{\ominus_2} & \langle\oplus_2.\ominus_2\rangle X & \wedge & \langle a.\,\ominus_2\,.\,\ominus_2\,.\ominus_2\rangle X & \wedge & \langle\oplus_2.b\rangle X \end{array}$$

Suivant le principe des équations existentielles, la sentence $\varphi_{\mathcal{N}}$ induit :

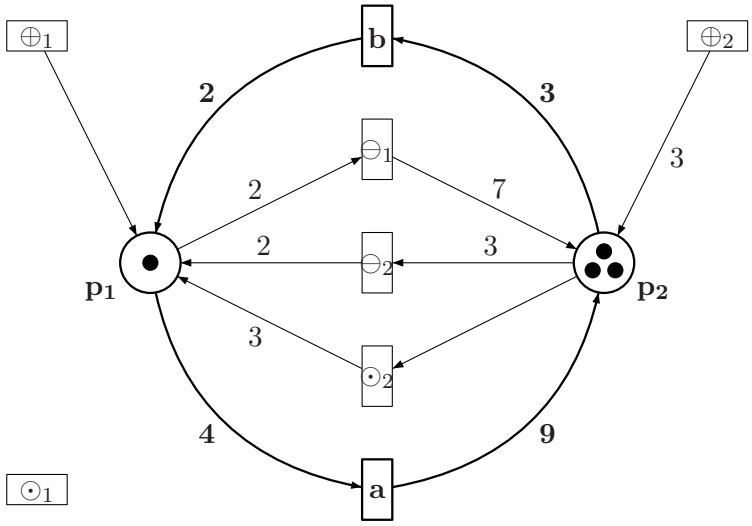

FIG. 4.3 – Un modèle de $\varphi_{\mathcal{N}}$.

– *l'existence d'une place* p'_1 *qui vérifie*

$$\begin{array}{lcllcl} <p'_1, \ominus_1> & < & 0 & <p'_1, a> & = & -2.<p'_1, \oplus_1> \\ <p'_1, \ominus_1> & = & -<p'_1, \oplus_1> & <p'_1, b> & = & -<p'_1, \ominus_1> \end{array}$$

En posant $<p'_1, \ominus_1> = \lambda_1$*, nous obtenons* $<p'_1, a> = -2.\lambda_1$ *et* $<p'_1, b> = \lambda_1$*, ce qui nous donne* $p'_1|_\Sigma = \lambda_1. < -2, 1 >$ *soit* $p'_1|_\Sigma = \lambda_1.p_1$*.*

– *l'existence d'une place* p'_1 *qui vérifie*

$$\begin{array}{lcllcl} <p'_2, \ominus_2> & < & 0 & <p'_2, a> & = & -3.<p'_2, \ominus_2> \\ <p'_2, \ominus_2> & = & -<p'_2, \oplus_2> & <p'_2, b> & = & -<p'_2, \oplus_2> \end{array}$$

En posant $<p'_2, \ominus_2> = \lambda_2$ *nous obtenons* $p'_2|_\Sigma = \lambda_2. < 3, -1 >$ *soit* $p'_2|_\Sigma = \lambda_2.p_2$*.*

La Figure 4.3 représente un exemple de réseau initialisé solution de $\varphi_\mathcal{N}$ *sur* Σ_e*. Nous montrons plus loin que tout réseau vérifiant* $p'_2|_\Sigma = \lambda_2.p_2$ *et* $p'_1|_\Sigma = \lambda_1.p_1$ *pour certains* λ_1 *et* λ_2 *entiers strictement positifs, accepte un prolongement qui satisfait* $\varphi_\mathcal{N}$*.*

On se munit d'une relation de préordre structurel :

Définition 4.2.14. On définit la relation de préordre $\sqsubseteq$ par $\mathcal{N} \sqsubseteq \mathcal{N}'$ si et seulement si $\mathcal{N} = \{p_1, \ldots, p_l\}$ et $\mathcal{N}' = \{p'_1, \ldots, p'_{l'}\}$ avec $l \leq l'$ et pour tout $i \in [1, l]$, $p_i = \lambda_i.p'_i$.

Deux réseaux $\mathcal{N}$ et $\mathcal{N}'$ vérifient $\mathcal{N} \sqsubseteq \mathcal{N}'$ lorsque $\mathcal{N}'$ contient toutes les places de $\mathcal{N}$ (à colinéarité près).

Théorème 4.2.15. *Soit* $\mathcal{N}'$ *un réseau sur* Σ*,* $\mathcal{N} \sqsubseteq \mathcal{N}'$ *si et seulement si* $\mathcal{N}'$ *peut être prolongé sur* Σ_e *en un réseau* $\mathcal{N}''$ *vérifiant* $\mathcal{N}'' \models \varphi_\mathcal{N}$*.*

Démonstration. On rappelle que $\mathcal{N} = \{p_1, \ldots, p_l\}$ et $\Sigma = \{a_1, \ldots, a_n\}$. On suppose, sans perdre de généralité que $\mathcal{N}$ est réduit.

⇒) De $\mathcal{N} \sqsubseteq \mathcal{N}'$, nous déduisons $\mathcal{N}' = \{p'_1, \ldots, p'_l\} \cup \{p'_{l+1}, \ldots, p'_{l'}\}$ avec $p'_i = \lambda_i.p_i$ pour tout $i \in [1, l]$[3]. On exhibe le prolongement $\mathcal{N}''$ de $\mathcal{N}'$ en posant, pour tout place $p'_{i'}$ de $\mathcal{N}'$, pour tout $i \in [1, l']$, avec $i \neq i'$,

$$<p'_{i'}, \ominus_i> = \max(-<p'_{i'}, a_1>, \ldots, -<p'_{i'}, a_n>) \quad (4.8)$$

$$<p'_{i'}, \oplus_i> = \max(-<p'_{i'}, a_1>, \ldots, -<p'_{i'}, a_n>) \quad (4.9)$$

[3] rappelons que puisque $\mathcal{N}$ est réduit, alors $\lambda_i \in \mathbb{N}$ et $\lambda_i \neq 0$, voir la Remarque 4.1.5

ainsi que pour tout $i \in [1, l]$,

$$<p'_i, \ominus_i> = -\lambda_i \quad (4.10)$$

$$<p'_i, \oplus_i> = +\lambda_i \quad (4.11)$$

On montre que $\mathcal{N}''$ ainsi construit vérifie pour tout $i \in [1, l]$ les conditions (i) et (ii) du Lemme 4.2.8 pour la sentence φ_i. D'après (4.10) et (4.11), la place p'_i, avec $i \in [1, l]$, vérifie $<p'_i, \ominus_i> < 0$ et $<p_i, \oplus_i. \ominus_i > = 0$. De plus, pour tout $j \in [1, n]$, soit $<p_i, a_j> > 0$, dans ce cas, par définition de $u_{i,j}$,

$$\begin{array}{rcccl} <p'_i, u_{i,j}> & = & <p'_i, a_j> & + & <p_i, a_j> \times <p'_i, \ominus_i> \\ & = & \lambda_i \times <p_i, a_j> & - & \lambda_i \times <p_i, a_j> \\ & = & 0 & & \end{array}$$

soit $<p_i, a_j> < 0$, dans ce cas, par définition de $u_{i,j}$,

$$\begin{array}{rcccl} <p'_i, u_{i,j}> & = & -<p_i, a_j> \times <p'_i, \oplus_i> & + & <p'_i, a_j> \\ & = & -\lambda_i \times <p_i, a_j> & + & \lambda_i \times <p_i, a_j> \\ & = & 0 & & \end{array}$$

soit $<p_i, a_j> = 0$,dans ce cas, par définition de $u_{i,j}$,

$$<p'_i, u_{i,j}> = 0$$

Tous les mots $u_{i,j}$ sont construits de manière à ce que les quantités négatives se trouvent à la fin du mot : lorsque $<p'_i, a_j> \geq 0$, $u_{i,j}$ commence par a_j puis suivent les occurrences de $\ominus_i$ qui vérifie $<p'_i, \ominus_i> < 0$, ce qui assure $<p'_i, w> \geq 0$ pour tout préfixe w de $u_{i,j}$; et symétriquement, lorsque $<p'_i, a_j> < 0$, $u_{i,j}$ commence par les occurrences de $\oplus_i$ qui vérifient $<p'_i, \oplus_i> > 0$, puis suit a_j ce qui assure $<p'_i, w> \geq 0$ pour tout préfixe w de $u_{i,j}$. Nous obtenons $<p'_i, w> \geq 0$ pour tout $j \in [1, n]$, pour tout w préfixe de $u_{i,j}$ et par conséquent $<p'_i, w> > <p'_i, \ominus_i>$. Nous avons prouvé l'item (i) du Lemme 4.2.8.
Concernant les autres places, pour tout $i' \in [1, l']$ avec $i' \neq i$, les Égalités (4.8) et (4.9) nous assurent pour tout $j \in [1, n]$,

$$<p'_{i'}, u^i_j> \geq 0$$

Ce dernier résultat nous assure de l'item (ii) du Lemme 4.2.8. Nous obtenons enfin $\mathcal{N}'' \models \varphi_i$ pour tout $i \in [1, l]$. Soit m_0 un marquage initial de $\mathcal{N}''$, le Lemme 4.2.3 nous assure l'existence de valuations $<p_{i'}, \odot_i>$ telles que $(\mathcal{N}', m_0) \models \langle \odot_1 \rangle \varphi_i$ pour tout $i, i' \in [1, l]$. Nous en déduisons $\mathcal{N}'' \models \varphi_{\mathcal{N}}$.

$\Leftarrow$) Pour tout $i \in [1, l]$ on montre qu'il existe $\lambda_i \in \mathbb{N}$, tel que $\lambda_i.p_i \in \mathcal{N}'$. Puisque $\mathcal{N}'' \models \varphi_i$, le Lemme 4.2.8 assure l'existence d'une place p'_i de $\mathcal{N}''$ telle que

$$<p'_i, \ominus_i> < 0 \text{ et } <p'_i, \oplus_i> + <p'_i, \ominus_i> = 0$$

ainsi que pour tout $j \in [1, n]$,

$$<p'_i, u_{i,j}> = 0$$

En posant $<p'_i, \oplus_i> = \lambda_i$, nous obtenons $<p'_i, \ominus_i> = -\lambda_i$ ainsi que les résultats suivants : $<p'_i, u_{i,j}> = 0$ implique
si $<p_i, a_j> > 0$, alors

$$\begin{aligned} <p'_i, a_j> - \lambda_i \times <p_i, a_j> &= 0 \\ <p'_i, a_j> &= \lambda_i \times <p_i, a_j> \end{aligned}$$

si $<p_i, a_j> < 0$ alors

$$\begin{aligned} \lambda_i \times (-<p_i, a_j>) + <p'_i, a_j> &= 0 \\ <p'_i, a_j> &= \lambda_i \times <p_i, a_j> \end{aligned}$$

si $<p_i, a_j> = 0$ alors

$$<p'_i, a_j> = 0$$

Nous avons prouvé $p'_i = \lambda_i.p_i$. Nous obtenons finalement $\mathcal{N}' \sqsubseteq \mathcal{N}$.

□

Nous montrons maintenant, à travers deux remarques, comment établir la théorie $\varphi_{\mathbf{C}}$ d'une *classe* $\mathbf{C}$ de réseaux, lorsque cette classe se présente d'une manière particulière.

Remarque 4.2.16. La sentence de la Définition 4.2.12 est construite de manière à imposer des valeurs relatives à tous les couples (p_i, a_j). Il est possible d'exprimer partiellement la théorie d'un réseau $\mathcal{N} = \{p_1, \ldots, p_l\}$ en laissant certaines relations de flot indéterminées ; par exemple, pour que les valeurs de tir $<p_i, a_j>$ et $<p_i, a_{j'}>$ soient égales, sans en imposer la valeur (relativement aux autres couples (p_i, a_k)). Nous définissons alors ici la théorie d'un classe $\mathbf{C}$ qui est la classe des réseaux $\mathcal{N}'$ tels que $\{p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_l\} \sqsubseteq \mathcal{N}'$ et qu'il existe λ tel que $\mathcal{N}'$ possède une place p'_i vérifiant :

$$\forall k \in [1, n] \text{ avec } k \notin \{j, j'\}, <p'_i, a_k> = \lambda \times <p_i, a_k> \text{ et } <p'_i, a_j> = <p'_i, a_{j'}>$$

Pour exprimer la théorie de **C**, il suffit de modifier la sentence $\varphi_{\mathcal{N}}$ en ajoutant un événement $\boxdot$ à Σ_e et de modifier $u_{i,j}$ et $u_{i,j'}$ de la manière suivante :

$$u_{i,j} = a_j.\boxdot, \text{ et } u_{i,j'} = a_{j'}.\boxdot$$

Suivant ce principe nous obtenons les équations suivantes pour p_i

$$<p_i, a_j> + <p_i, \boxdot> = 0 \text{ et } <p_i, a_{j'}> + <p_i, \boxdot> = 0$$

ce qui nous donne $<p_i, a_j> = <p_i, a_{j'}>$.

Remarque 4.2.17. Suivant le même principe que la remarque précédente pour exprimer partiellement la théorie d'un réseau $\mathcal{N} = \{p_1, \ldots, p_l\}$, avec $<p_i, a_j> = k$ et $<p_i, a_{j'}> = -k$ pour un certain $k \geq 0$ donné, c.-à-d. pour exprimer la théorie de la classe **C** qui est la classe des réseaux $\mathcal{N}'$ tels que $\{p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_l\} \sqsubseteq \mathcal{N}'$ et qu'il existe λ tel que $\mathcal{N}'$ possède une place p'_i vérifiant :

$$\forall k \in [1, n] \text{ avec } k \notin \{j, j'\}, <p'_i, a_k> = \lambda \times <p_i, a_k>$$

et, avec $k \geq 0$,

$$<p'_i, a_j> = k \text{ et } <p'_i, a_{j'}> = -k$$

il suffit de modifier la sentence $\varphi_{\mathcal{N}}$ en ajoutant les événements $\boxplus$ et $\boxminus$, en modifiant $u_{i,j}$ et $u_{i,j'}$ en

$$u_{i,j} = \boxplus.a_j \text{ et } u_{i,j'} = a_{j'}.\boxminus$$

en posant

$$u = \boxplus.\boxminus$$

et enfin en posant

$$\varphi_i \quad = \quad \gamma_\exists[\ominus_i][u, v_i, u_{i,1}, \ldots, u_{i,n}][=0]$$

Remarquons dans un premier temps que ces modifications n'influencent pas toute autre place p d'un réseau solution puisque les quantités $<p, \boxplus>$ et $<p, \boxminus>$ sont laissées libres (et peuvent alors êtres aussi grandes que nécessaire). En revanche la place p_i vérifie alors, en posant $k = <p_i, \boxplus>$:

$$<p_i, a_j> = k \text{ et } <p_i, a_{j'}> = -k$$

Il suffit maintenant de poser $\varphi_{\mathcal{N}} = \Upsilon_\forall[\boxplus][\geq 0] \wedge \varphi_1 \wedge \ldots \wedge \varphi_l$. La sentence d'équation universelle $\Upsilon_\forall[\boxplus][\geq 0]$ assure la positivité de k (en la remplaçant par $\Upsilon_\forall[\boxplus.\ominus_i][\geq 0]$ on obtiendrait $k > 0$.)

Cette dernière remarque sera utilisée plus loin pour la caractérisation dans $\mathbf{L}_\nu$ d'une classe particulière de réseaux : *les réseaux de compteurs bornés* (voir Section 4.3 page 106). La spécification de cette classe de réseaux nous permettra de montrer l'indécidabilité de $\mathbf{Sat}(\mathbf{L}_\nu,\mathbf{PN})$.

4.2.3 Nu-calcul et intervalles de langages

Nous montrons que le même résultat peut être obtenu en utilisant une spécification sous la forme d'un intervalle de langages de $\mathbf{INT}_\Sigma$ à la place d'une sentence de $\mathbf{L}_\nu$. Dans un premier temps, nous introduisons les concepts qui permettent de caractériser les sentences de $\mathbf{L}_\nu$ qui sont équivalentes à des intervalles de langages.

Définition 4.2.18 (Sentence séparées). On dit qu'une sentence ϕ de $\mathbf{L}_\nu$ est *séparée* lorsque $\phi = \phi_1 \wedge \phi_2$ où ϕ_1 s'écrit sans l'opérateur $\langle . \rangle$ et ϕ_2 s'écrit sans l'opérateur $[.]$. On note $Sep(\mathbf{L}_\nu)$ l'ensemble des sentences séparées de $\mathbf{L}_\nu$.

Exemple 4.2.19. *Considérons les sentences suivantes :*

$$\begin{array}{lcl} \phi_1 & = & [a][b] \not\rightarrow^a \\ \phi_2 & = & [a]\langle b\rangle \mathtt{true} \\ \phi_3 & = & \nu X.(\not\rightarrow^a \wedge [b]X) \wedge \nu Y.(\not\rightarrow^c \wedge \langle d\rangle Y) \\ \phi_4 & = & \nu X.\langle a\rangle \mathtt{true} \wedge [b]X \end{array}$$

Les sentences ϕ_1 et ϕ_3 sont séparées, ce qui n'est pas le cas des sentences ϕ_2 et ϕ_4.

Remarquons que les sentences $\Upsilon_\forall[u][\geq 0]$, $\gamma_\exists[a][u_1,\ldots,u_k][=0]$, ainsi que la sentence $\Upsilon_\exists[\oplus,\ominus][u_1,\ldots,u_k][=0]$ sont des sentences séparées.

Définition 4.2.20 (Spécification modale creuse). Une spécification modale $S = \langle \{C_a\}_{a\in\Sigma}, I\rangle$ est dite *creuse* lorsque $\{1\} \cup C_S(\Sigma^*)$ est un langage clos par préfixe.

Exemple 4.2.21. *L'automate modal de la figure 4.4 représente une spécification modale S creuse. Le langage $\{1\} \cup C_S(\Sigma^*)$ est le langage $\overline{(a.b.(b.b)^*.a)^*}$.*

Proposition 4.2.22. *Soit S une spécification modale creuse. Pour tout langage L clos par préfixe, $L \in mod(S)$ si et seulement si $L \in [L^S_\perp, L^S_\top]$.*

Démonstration. Pour tout langage $L \in mod(S)$, par le Théorème 3.2.9 page 71, nous avons nécessairement $L \in [L^S_\perp, L^S_\top]$.

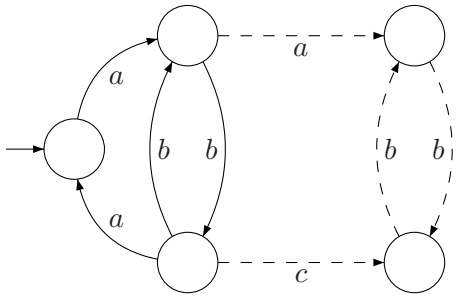

FIG. 4.4 – Exemple d'automate modal associé à une spécification modale creuse.

Montrons maintenant que si $L \in [L^S_\perp, L^S_\top]$ alors $L \in mod(S)$. Puisque S est creuse, alors $\{1\} \cup C_S(\Sigma^*)$ est clos par préfixe, ce qui entraîne que la S-clôture de $\{1\}$ est le langage $\{1\} \cup C_S(\Sigma^*)$ lui-même. Par conséquent $L^S_\perp = \{1\} \cup C_S(\Sigma^*)$, $C_S(L) \subseteq L^S_\perp$, et puisque $L^S_\perp \subseteq L$ et $C_S(L) \subseteq C_S(\Sigma^*)$, nous obtenons $C_S(L) \subseteq L$. Puisque $L \subseteq L^S_\top$, nous obtenons $L \cap I = \emptyset$. Nous avons montré $L \in mod(S)$. □

Proposition 4.2.23. *Soit ϕ une sentence de $Sep(\mathbf{L}_\nu)$, la spécification modale S_ϕ est creuse.*

Démonstration. Soit S_ϕ la spécification modale associée à ϕ. Par définition de $Sep(\mathbf{L}_\nu)$, $\phi = \phi_1 \wedge \phi_2$ où ϕ_1 s'écrit sans l'opérateur $\langle . \rangle$ et ϕ_2 s'écrit sans l'opérateur $[.]$. Nous en déduisons $S_\phi = S_{\phi_1} \cap S_{\phi_2}$. Par la Définition 3.2.11 page 71 nous obtenons $C_{S_\phi}(\Sigma^*) = C_{S_{\phi_1}}(\Sigma^*) \cup C_{S_{\phi_2}}(\Sigma^*)$. Remarquons que puisque ϕ_1 n'utilise pas l'opérateur $\langle . \rangle$, nous obtenons par définition de S_{ϕ_1} le résultat $C_{S_{\phi_1}}(\Sigma^*) = \emptyset$. Il suffit alors de montrer que $\{1\} \cup C_{S_{\phi_2}}(\Sigma^*)$ est clos par préfixe. Nous montrons pour cela que toute *formule* β de $\mathbf{L}_\nu$ qui s'écrit sans utiliser l'opérateur $[.]$ vérifie les résultats (i) et (ii) suivants :

(i) $\{1\} \cup C_{S_\beta}(\Sigma^*)$ est clos par préfixe

(ii) $\forall X \in var(\beta), P_\beta(X) \subseteq (\{1\} \cup C_{S_\beta}(\Sigma^*))$

On rappelle que $var(\beta)$ est l'ensemble des variables libres de β et que $P_\beta(X)$ désigne l'ensemble des chemins de la variable X dans β (voir Définition 3.3.2 page 74).
Le preuve est par induction sur β.

– Si $\beta \in \{\mathtt{true}, \not\rightarrow^a\}$ alors $\{1\} \cup C_{S_\beta}(\Sigma^*) = \{1\}$, ce qui entraîne (i) ; (ii) est trivial puisque β n'a pas de variable libre.

– Si $\beta = X$, (i) est vérifiée comme pour le cas précédent. La seule variable libre de β est X et $P_\beta(X) = \{1\}$, ce qui entraîne immédiatement (ii).

– Si $\beta = \langle a \rangle \beta_1$ alors $C_{S_\beta}(\Sigma^*) = \{a\} \cup a.C_{S_{\beta_1}}(\Sigma^*)$ ce qui implique (i) par hypothèse d'induction. Pour toute variable libre X de β, nous avons $P_\beta(X) = a.P_{\beta_1}(X)$; de $P_{\beta_1}(X) \subseteq (\{1\} \cup C_{S_{\beta_1}}(\Sigma^*))$ nous déduisons alors (ii).

– Si $\beta = \beta_1 \wedge \beta_2$ le résultat est immédiat.

– Si $\beta = \nu.X\beta_1(X)$ alors $C_{S_\beta}(\Sigma^*) = P_{\beta_1}(X)^*.C_{S_{\beta_1}}(\Sigma^*)$. Par hypothèse d'induction, nous avons $P_{\beta_1}(X) \subseteq (\{1\} \cup C_{S_{\beta_1}}(\Sigma^*))$, ce qui implique la clôture par préfixe de $\{1\} \cup C_{S_\beta}(\Sigma^*)$ et par conséquent (i). Pour toute variable libre Y de β, par induction, nous avons $P_{\beta_1}(Y) \subseteq (\{1\} \cup C_{S_{\beta_1}}(\Sigma^*))$, ce qui implique

$$\begin{array}{rcl} P_{\beta_1}(X)^*.P_{\beta_1}(Y) & \subseteq & P_{\beta_1}(X)^*(\{1\} \cup C_{S_{\beta_1}}(\Sigma^*)) \\ P_\beta(Y) & \subseteq & \{1\} \cup P_{\beta_1}(X)^*.(\{1\} \cup C_{S_{\beta_1}}(\Sigma^*)) \end{array}$$

D'après (i), $\{1\} \cup P_{\beta_1}(X)^*.C_{S_{\beta_1}}(\Sigma^*)$ est clos par préfixe, donc $P_\beta(Y) \subseteq \{1\} \cup P_{\beta_1}(X)^*.C_{S_{\beta_1}}(\Sigma^*)$ en enfin $P_\beta(Y) \subseteq C_{S_\beta}(\Sigma^*)$.

Dans la sentence ϕ_2, l'item (i) nous assure que S_{ϕ_2} est creuse et donc que S_ϕ est creuse.

□

Des Propositions 4.2.22 et 4.2.23, nous déduisons le corollaire suivant :

Corollaire 4.2.24. *Soit ϕ une sentence de $Sep(\mathbf{L}_\nu)$, il existe un intervalle de langage $[L_1, L_2]$ tel que pour tout langage L, $L \models \phi$ si et seulement si $L \in [L_1, L_2]$.*

Remarque 4.2.25. L'intervalle correspondant à une sentence séparée ϕ se construit en deux étapes : calculer S_ϕ puis calculer $L^S_\top$ et $L^S_\bot$. Le résultat est l'intervalle $[L^S_\top, L^S_\bot]$.

4.2.4 La théorie d'un réseau exprimée dans $\textsc{Int}_\Sigma$

Soit $\mathcal{N}$ un réseau de Petri, considérons la sentence $\varphi_\mathcal{N}$ (Définition 4.2.12). Nous montrons que $\varphi_\mathcal{N}$ est dans $Sep(\mathbf{L}_\nu)$ en étudiant ses composantes : remarquons que toute sentence d'équations existentielles ainsi que toute sentence d'inéquation universelle est dans $Sep(\mathbf{L}_\nu)$ (voir Définitions 4.2.4 et 4.2.9), nous en déduisons $\varphi_\mathcal{N} \in Sep(\mathbf{L}_\nu)$.

Soit $\mathcal{N}$ un réseau, soit $\varphi_{\mathcal{N}}$ sa théorie (Définition 4.2.12), on définit $I_{\mathcal{N}}$ l'intervalle de langages équivalent à la sentence $\varphi_{\mathcal{N}}$, obtenu par le Corollaire 4.2.24.

Théorème 4.2.26. *Soient $\mathcal{N}$ et $\mathcal{N}'$ deux réseaux, $\mathcal{N} \sqsubseteq \mathcal{N}'$ si et seulement si $\mathcal{N}'$ peut être prolongé sur Σ_e en un réseau $\mathcal{N}''$ tel que le langage $\mathcal{L}(\mathcal{N}'', m_0)$ soit dans l'intervalle $I_{\mathcal{N}}$, pour un marquage m_0.*

Démonstration. D'après le Théorème 4.2.15, $\mathcal{N} \sqsubseteq \mathcal{N}'$ si et seulement si $\mathcal{N}'$ peut être prolongé sur Σ_e en un réseau $\mathcal{N}''$ tel que $\mathcal{L}(\mathcal{N}'', m_0) \models \varphi_{\mathcal{N}}$ pour un marquage m_0. Le Corollaire 4.2.24 nous permet de conclure. □

4.3 Indécidabilité de Sat($\mathbf{L}_\nu$,$\mathbf{P_N}$)

Cette section est dédiée à la preuve du théorème suivant :

Théorème 4.3.1. *Le problème* **Sat**($\mathbf{L}_\nu$,$\mathbf{P_N}$) *de satisfiabilité d'une sentence de $\mathbf{L}_\nu$ sur la classe des réseaux de Petri purs est indécidable*

La preuve de ce théorème repose sur la réduction d'un problème indécidable concernant les machines à deux compteurs (le *problème de l'exécution bornée*, détaillé plus loin). L'idée de cette réduction, proche de celle du Chapitre 2 Section 2.4, est de définir une sentence Γ_M associée à une machine à compteur M telle que :

- s'il existe un réseau modèle de Γ_M, alors un réseau particulier $\mathcal{N}$ (réseau de compteurs bornés) est également modèle de Γ_M
- si $\mathcal{N}$ est modèle de Γ_M alors le problème de l'exécution bornée possède une solution.
- si le problème de l'exécution bornée possède une solution, alors un réseau de compteurs bornés est modèle de Γ_M.

La principale difficulté dans la logique $\mathbf{L}_\nu$ est d'effectuer le test à zéro de compteurs. En effet, lorsque δ est une transition de M de la forme (if $c_i = 0$ then ($c_i := c_i + 1$; goto q_h) else ($c_i := c_i - 1$; goto $q_{h'}$)), il convient de distinguer deux cas mutuellement exclusifs ($c_i > 0$ ou $c_i = 0$). Or, ceci était possible dans $\mathbf{L}_\mu$ grâce à l'opérateur $\vee$, alors que cet opérateur n'appartient pas à $\mathbf{L}_\nu$.

Afin d'effecteur le test à zéro d'un compteur c_i, ainsi qu'un branchement dans l'évaluation de la sentence Γ_M, le principe est d'ajouter à chaque place qui simule un compteur une place complémentaire contenant k jetons (ce qui requiert alors que les marquages de la place simulant le compteur soient bornés par k). Le test à zéro peut alors se faire sur la place complémentaire, en utilisant l'opérateur $[_{i?}]$ pour une transition $_{i?}$ bien choisie.

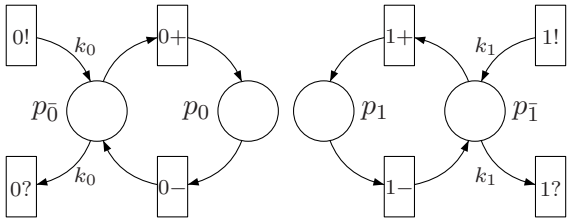

FIG. 4.5 – Le réseau $\mathcal{N}_{k_0,k_1}$.

4.3.1 Réseau de compteurs bornés

Nous définissons une sous-classe de **PN** utilisée pour la preuve du Théorème 4.3.1. Il s'agit de la classe **B** composée des réseau $\mathcal{N}_{k_0,k_1}$ de la Figure 4.5, sur un alphabet $\Sigma = \{0+, 0-, 0?, 0!, 1+, 1-, 1?, 1!\}$, où k_0, k_1 sont des entiers strictement positifs.

Définition 4.3.2 (Réseaux de compteurs bornés). La classe **B** des réseaux de compteurs bornés est définie par :

$$\mathbf{B} = \{\mathcal{N}_{k_0,k_1} \mid k_0 > 0, k_1 > 0\}$$

Le rôle des événements de Σ est le suivant pour $i \in [0,1]$:
- $i+$ représente l'opération $(c_i := c_i + 1)$,
- $i-$ représente l'opération $(c_i := c_i - 1)$,
- $i?$ sert à effectuer *le test à zéro* du compteur c_i,
- $i!$ sert à restituer les jetons utilisés lors du test à zéro du compteur c_i.

Les places de $\mathcal{N}_{k_0,k_1}$ jouent le rôle suivant, pour $i \in [0,1]$:
- p_i est la place qui simule le compteur i,
- $p_{\bar{i}}$ est la place complémentaire de p_i,

Le marquage initial de $\mathcal{N}_{k_0,k_1}$ est le marquage m_0 défini comme suit :

$$\begin{array}{rclcrcl} m_0(p_0) & = & 0 & & m_0(p_1) & = & 0 \\ m_0(p_{\bar{0}}) & = & k_0 & & m_0(p_1) & = & k_1 \end{array}$$

Suivant ce principe, le réseau $(\mathcal{N}_{k_0,k_1}, m_0)$ va permettre de simuler l'exécution de la machine M pour la configuration initiale $(q_0, 0, 0)$ avec l'hypothèse que les valuations des compteurs c_0 et c_1 restent bornées respectivement par k_0 et k_1. Pour tout $u \in \{i+, i-\}^*$ avec $m_0[u\rangle m$,
- soit le compteur c_i ne contient aucun jeton $(m(p_i) = 0)$, auquel cas $m(p_{\bar{i}}) = k_i$, la transition $i?$ est active et la transition $i-$ est inactive,

– soit le compteur c_i contient au moins un jeton ($m(p_i) > 0$), auquel cas $m(p_{\bar{i}}) < k_i$, la transition $_{i?}$ est inactive et la transition $_{i-}$ est active.

Il est alors possible de définir une formule pour le test à zéro du compteur c_i sous la forme suivante :

$$[_{i?}]\langle_{i!}\rangle\langle_{i+}\rangle\phi_{=0} \wedge [_{i-}]\phi_{\neq 0}$$

où $\phi_{=0}$ et $\phi_{\neq 0}$ représentent les contraintes de la suite de l'exécution en fonction du résultat du test dans à zéro.

4.3.2 Sentence associée à une machine à compteurs

Problème 4.3.3 (Problème de l'exécution bornée). Étant donnée une machine à deux compteurs M, déterminer si lors de l'exécution de M à partir de $(q_0, 0, 0)$ les valuations des deux compteurs restent bornées.

Proposition 4.3.4. *Le Problème 4.3.3 est indécidable.*

Démonstration. Soit M une machine à deux compteurs. Supposons que le Problème 4.3.3 soit décidable. On distingue deux cas :

– soit les compteurs de M sont non bornés durant l'exécution, dans ce cas l'exécution est nécessairement infinie
– soit les compteurs de M sont bornés durant l'exécution, dans ce cas l'ensemble des configurations parcourues durant l'exécution est fini. Il est possible de décider de l'arrêt de M en construisant la séquence de configurations : si elle repasse deux fois par la même configuration, alors l'exécution est infinie, dans le cas contraire l'exécution est finie.

Si le Problème 4.3.3 est décidable, alors le problème de l'arrêt d'une machine à compteurs l'est également, ce qui n'est pas le cas.

□

Afin de réduire le Problème 4.3.3 pour une machine M au problème $\mathbf{Sat}(\mathbf{L}_\nu,\mathbf{P_N})$, nous construisons une sentence Γ_M associée à M, avec

$$\Gamma_M = \Phi_{\mathbf{B}} \wedge \Phi_{m_0} \wedge \Phi_c \wedge \Phi_M$$

où :

– $\Phi_{\mathbf{B}}$ est la théorie de $\mathcal{N}_{k_0,k_1}$,
– Φ_{m_0} est une sentence destinée à imposer le marquage initial de manière à ce qu'il soit le marquage m_0 défini précédemment.
– Φ_c est une sentence qui contraint les comportements de ses solutions, afin qu'aucune place (qui n'est pas colinéaire à une place de $\mathcal{N}_{k_0,k_1}$) ne puisse empêcher le test à zéro tout en restant modèle de la formule qui l'exprime.

– Φ_M est une sentence satisfiable par $(\mathcal{N}_{k_0,k_1}, m_0)$, pour certains k_0, k_1, si et seulement si l'exécution de M est bornée pour $(q_0, 0, 0)$. Cette sentence est proche de celle de la Définition 2.4.15 page 48.

4.3.2.1 Théorie de la classe B

La sentence $\Phi_{\mathbf{B}}$ est la sentence $\langle\odot\rangle\varphi_{\mathbf{B}}$ où $\varphi_{\mathbf{B}}$ est théorie de la classe $\mathbf{B}$, donné par la Définition 4.2.12 modifiée suivant la Remarque 4.2.17 pour prendre en compte les paramètres strictement positifs k_0 et k_1.

On convient de l'alphabet Σ_e défini par :

$$\Sigma_e = \Sigma \cup \{\odot, \boxplus_0, \boxminus_0, \boxplus_1, \boxminus_1, \oplus_0, \ominus_0, \oplus_{\bar{0}}, \ominus_{\bar{0}}, \oplus_1, \ominus_1, \oplus_{\bar{1}}, \ominus_{\bar{1}}\}$$

Définition 4.3.5 (La sentence $\Phi_{\mathbf{B}}$). La sentence $\Phi_{\mathbf{B}}$ est la sentence $\langle\odot\rangle\varphi_{\mathbf{B}}$ où $\varphi_{\mathbf{B}}$ est la sentence :

$$\begin{array}{llllllllllllll}
 & \nu X. & \langle\boxplus_0\rangle X & & & & & & & & & & & \\
\wedge & \nu X. & \langle\boxplus_1\rangle X & & & & & & & & & & & \\
 & & & & & & & & & & & & & \\
\wedge & \nu X. & \not\rightarrow^{\ominus_{\bar{0}}} & \wedge & \langle\oplus_{\bar{0}}\rangle\langle\ominus_{\bar{0}}\rangle X & \wedge & \langle\boxplus_0\rangle\langle\boxminus_0\rangle X & & & & & & & \\
 & & & \wedge & \langle 0-\rangle\langle\ominus_{\bar{0}}\rangle X & \wedge & \langle\oplus_{\bar{0}}\rangle\langle 0+\rangle X & \wedge & \langle 0!\rangle\langle\boxminus_0\rangle X & \wedge & \langle\boxplus_0\rangle\langle 0?\rangle X & & & \\
 & & & \wedge & \langle 1-\rangle X & \wedge & \langle 1+\rangle X & \wedge & \langle 1!\rangle & \wedge & \langle 1?\rangle X & & & \\
 & & & & & & & & & & & & & \\
\wedge & \nu X. & \not\rightarrow^{\ominus_0} & \wedge & \langle\oplus_0\rangle\langle\ominus_0\rangle X & & & & & & & & & \\
 & & & \wedge & \langle 0+\rangle\langle\ominus_0\rangle X & \wedge & \langle\oplus_0\rangle\langle 0-\rangle X & \wedge & \langle 0!\rangle X & \wedge & \langle 0?\rangle X & & & \\
 & & & \wedge & \langle 1-\rangle X & \wedge & \langle 1+\rangle X & \wedge & \langle 1!\rangle & \wedge & \langle 1?\rangle X & & & \\
 & & & & & & & & & & & & & \\
\wedge & \nu X. & \not\rightarrow^{\ominus_{\bar{1}}} & \wedge & \langle\oplus_{\bar{1}}\rangle\langle\ominus_{\bar{1}}\rangle X & \wedge & \langle\boxplus_1\rangle\langle\boxminus_1\rangle X & & & & & & & \\
 & & & \wedge & \langle 1-\rangle\langle\ominus_{\bar{1}}\rangle X & \wedge & \langle\oplus_{\bar{1}}\rangle\langle 1+\rangle X & \wedge & \langle 1!\rangle\langle\boxminus_1\rangle X & \wedge & \langle\boxplus_1\rangle\langle 1?\rangle X & & & \\
 & & & \wedge & \langle 0-\rangle X & \wedge & \langle 0+\rangle X & \wedge & \langle 0!\rangle & \wedge & \langle 0?\rangle X & & & \\
 & & & & & & & & & & & & & \\
\wedge & \nu X. & \not\rightarrow^{\ominus_1} & \wedge & \langle\oplus_1\rangle\langle\ominus_1\rangle X & & & & & & & & & \\
 & & & \wedge & \langle 1+\rangle\langle\ominus_1\rangle X & \wedge & \langle\oplus_1\rangle\langle 1-\rangle X & \wedge & \langle 1!\rangle X & \wedge & \langle 1?\rangle X & & & \\
 & & & \wedge & \langle 0-\rangle X & \wedge & \langle 0+\rangle X & \wedge & \langle 0!\rangle & \wedge & \langle 0?\rangle X & & &
\end{array}$$

Lemme 4.3.6. *Soit $(\mathcal{N}, m_0)$ un réseau, $(\mathcal{N}, m_0) \models \Phi_{\mathbf{B}}$ si et seulement si il existe*

$\mathcal{N}' \sqsubseteq \mathcal{N}|_\Sigma$ *tel que* $\mathcal{N}' =$

$$\left\{\begin{array}{lcrrrrrrrrl} & & (0+) & (0-) & (0?) & (0!) & (1+) & (1-) & (1?) & (1!) & \\ (p'_0) & \langle & \lambda_0, & -\lambda_0, & 0, & 0, & 0, & 0, & 0, & 0 & \rangle, \\ (p'_{\bar{0}}) & \langle & -\lambda_{\bar{0}}, & \lambda_{\bar{0}}, & -k'_0, & k'_0, & 0, & 0, & 0, & 0 & \rangle, \\ (p'_1) & \langle & 0, & 0, & 0, & 0, & \lambda_1, & -\lambda_1, & 0, & 0 & \rangle, \\ (p'_{\bar{1}}) & \langle & 0, & 0, & 0, & 0, & -\lambda_{\bar{1}}, & \lambda_{\bar{1}}, & -k'_1, & k'_1 & \rangle, \end{array}\right\}$$

où $\lambda_0, \lambda_{\bar{0}}, \lambda_{0?}, k'_0, \lambda_1, \lambda_{\bar{1}}, \lambda_{1?}$ *et* k'_1 *sont des entiers strictement positifs.*

Démonstration. La preuve est donnée par le Lemme 4.2.15 et la Remarque 4.2.17. □

Remarque 4.3.7. Dans places décrites par le Lemme 4.3.6, pour $i \in \{0,1\}$, les relations de flot concernant les transitions $i+$ et $i-$ sont reliées par les λ et pour les places du type $p_{\bar{i}}$, les relations de flot concernant les transitions $i?$ et $i!$ sont reliées par k'_i.

Note 4.3.8. Dans la suite, lorsque $(\mathcal{N}, m_0) \models \Phi_{\mathbf{B}}$, on décompose $\mathcal{N}$ en posant $\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2$ avec $\mathcal{N}_1 = \{p'_0, p'_{\bar{0}}, p'_1, p'_{\bar{1}}\}$. On désignera par P_1 l'ensemble de places de $\mathcal{N}_1$ et par P_2 l'ensemble de places de $\mathcal{N}_2$.

Définition 4.3.9 (La sentence Φ_{m_0}). La sentence Φ_{m_0} est définie par :

$$\begin{array}{rcl} \Phi_{m_0} & = & \langle 0? \rangle \mathtt{true} \wedge \langle 0+ \rangle \not\rightarrow^{0?} \wedge \not\rightarrow^{0-} \\ & \wedge & \langle 1? \rangle \mathtt{true} \wedge \langle 1+ \rangle \not\rightarrow^{1?} \wedge \not\rightarrow^{1-} \end{array}$$

Lemme 4.3.10. *Soit* $\mathcal{N}_{k_0,k_1} \in \mathbf{B}$, $(\mathcal{N}_{k_0,k_1}, m_0) \models \Phi_{m_0}$ *si et seulement si* m_0 *vérifie*

$$\begin{array}{rclcrcl} m_0(p_0) & = & 0 & & m_0(p_1) & = & 0 \\ m_0(p_{\bar{0}}) & = & k_0 & & m_0(p_1) & = & k_1 \end{array}$$

Démonstration. Le résultat est immédiat en écrivant les inéquations de tir de $\mathcal{N}_{k_0,k_1}$ associées à la sémantique de Φ_{m_0}. □

Par la suite on nommera n_0 le marquage vérifiant les égalités du Lemme 4.3.10.

Le Lemme 4.3.10 montre que la sentence Φ_{m_0} induit le marquage initial n_0 souhaité pour le réseau $\mathcal{N}_{k_0,k_1}$. Nous donnons maintenant les propriétés qui vont nous permettre par la suite, pour tout réseau $(\mathcal{N}, m_0)$ modèle de $\Phi_{\mathbf{B}} \wedge \Phi_{m_0}$, de se ramener au réseau $(\mathcal{N}_{k_0,k_1}, n_0)$.

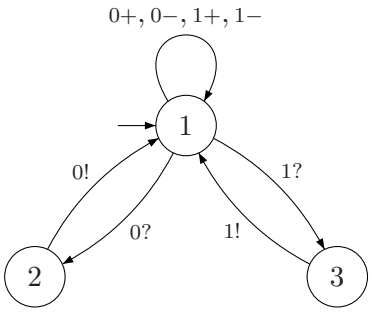

FIG. 4.6 – Le processus $\mathcal{P}_u$.

Lemme 4.3.11. *Si* $(\mathcal{N}, m_0) \models \Phi_{\mathbf{B}} \wedge \Phi_{m_0}$ *alors* m_0 *vérifie, pour les places de* P_1 *les inégalités suivantes :*

$$\begin{array}{lcl@{\qquad}lcl} m(p'_0) & < & \lambda_0 & m(p'_1) & < & \lambda_1 \\ m(p'_{\bar{0}}) & \geq & k'_0 & m(p'_{\bar{1}}) & \geq & k'_1 \\ m(p'_{\bar{0}}) & \geq & \lambda_{\bar{0}} & m(p'_{\bar{1}}) & \geq & \lambda_{\bar{1}} \\ m(p'_{\bar{0}}) & < & \lambda_{\bar{0}} + k'_0 & m(p'_{\bar{1}}) & < & \lambda_{\bar{1}} + k'_1 \end{array}$$

Démonstration. Le résultat est immédiat en interprétant la sentence Φ_{m_0} sur les places de P_1. □

Nous définissons le langage L_u qui est le langage *utile* des réseaux considérés dans cette section par

$$L_u = \overline{U^*} \text{ avec } U = \{0+, 0-, 0?.0!, 1+, 1-, 1?.1!\}$$

Nous montrons plus loin que la satisfaction de Φ_M par un réseau $(\mathcal{N}, m_0)$ est liée à sa satisfaction par $\mathcal{L}(\mathcal{N}, m_0) \cap L_u$ (voir page 121). Remarquons que le langage L_u est le langage du processus $\mathcal{P}_u$ de la Figure 4.6.

Le lemme suivant montre que si $(\mathcal{N}, m_0) \models \Phi_{\mathbf{B}} \wedge \Phi_{m_0}$ alors les langages $\mathcal{L}(\mathcal{N}_1, m_0)$ et $\mathcal{L}(\mathcal{N}_{k_0,k_1}, n_0)$ coïncident sur le langage L_u ; on rappelle que la note 4.3.8 définit le réseau $\mathcal{N}_1$.

Lemme 4.3.12. *Soit $(\mathcal{N}, m_0)$ un réseau tel que $(\mathcal{N}, m_0) \models \Phi_{\mathbf{B}} \wedge \Phi_{m_0}$, soit $\mathcal{N}_{k_0,k_1}$ le réseau de la classe* **B** *avec*

$$k_0 = \left\lfloor \frac{m_0(p_0)}{\lambda_{\bar{0}}} \right\rfloor \quad et \quad k_1 = \left\lfloor \frac{m_0(p_1)}{\lambda_{\bar{1}}} \right\rfloor$$

Les langages $\mathcal{L}(\mathcal{N}_1, m_0|_{P_1}) \cap L_u$ et $\mathcal{L}(\mathcal{N}_{k_0,k_1}, n_0) \cap L_u$ sont égaux.

Démonstration. Soit $P_1^0 = \{p'_0, p'_{\bar{0}}\}$, $P_1^1 = \{p'_1, p'_{\bar{1}}\}$ deux sous-ensembles de places de $\mathcal{N}_1$; on note $\mathcal{N}_1^0 = \mathcal{N}_1|_{P_1^0}$ et $\mathcal{N}_1^1 = \mathcal{N}_1|_{P_1^1}$. Soit $P^0 = \{p_0, p_{\bar{0}}\}$, $P^1 = \{p_0, p_{\bar{0}}\}$ une partition des places de $\mathcal{N}_{k_0,k_1}$; on note $\mathcal{N}^0 = \mathcal{N}_{k_0,k_1}|_{P^0}$ et $\mathcal{N}^1 = \mathcal{N}_{k_0,k_1}|_{P^1}$. On montre que les graphes $\mathcal{G}(\mathcal{N}_1^0, m_0|_{P_1^0}) \times \mathcal{P}_u$ et $\mathcal{G}(\mathcal{N}^0, n_0|_{P^0}) \times \mathcal{P}_u$ sont isomorphes. Ces deux graphes sont présentés dans la Figure 4.7 (les transitions $1+$, $1-$, $1?$ et $1!$ ne sont pas représentées : elles bouclent sur chacun des état) et l'isomorphisme est donné par les lignes en pointillés. L'isomorphisme est immédiat en rappelant que la forme des places de $\mathcal{N}_1^0$ est donnée par le Lemme 4.3.6, que les inéquations concernant le marquage initial sont données par le Lemme 4.3.11 et que $k_0 = \lfloor m_0(p_{\bar{0}})/\lambda_{\bar{0}} \rfloor$.

De l'isomorphisme des deux graphes, nous déduisons l'égalité de langages :

$$\mathcal{L}(\mathcal{N}_1^0, m_0|_{P_1^0}) \cap L_u = \mathcal{L}(\mathcal{N}^0, n_0|_{P^0}) \cap L_u$$

Nous pouvons montrer, en procédant de la même manière que

$$\mathcal{L}(\mathcal{N}_1^1, m_0|_{P_1^1}) \cap L_u = \mathcal{L}(\mathcal{N}^1, n_0|_{P^1}) \cap L_u$$

Ces deux égalités suffisent, grâce à la Proposition 1.4.19 page 23, qui énonce que le langage d'un réseau est l'intersection des langages de ses sous-réseaux, à démontrer

$$\mathcal{L}(\mathcal{N}_1, m_0|_{P_1}) \cap L_u = \mathcal{L}(\mathcal{N}_{k_0,k_1}, n_0) \cap L_u$$

□

4.3.2.2 Contraintes structurelles

La relation $\sqsubseteq$ induite entre les réseaux satisfaisant $\Phi_{\mathbf{B}}$ et le réseau $\mathcal{N}_{k_0,k_1}$ n'est pas suffisante pour la preuve du Théorème 4.3.1. La sentence Φ_c ajoute un ensemble de contraintes structurelles (qui ne sont pas exprimables dans $\mathbf{INT}_{\mathbf{\Sigma}}$) en assurant, en particulier que tout réseau solution de Γ_M ne possède aucune place qui interfère avec le test à zéro.

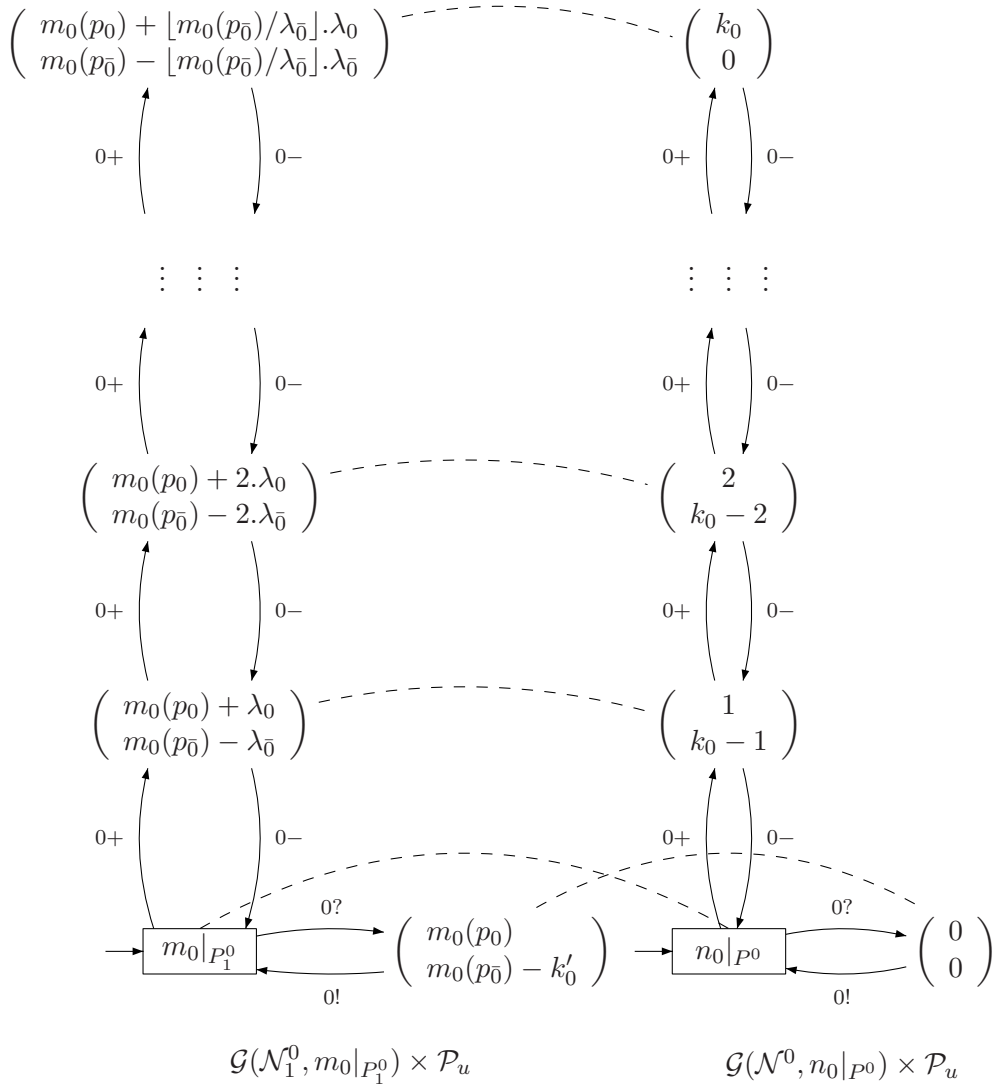

FIG. 4.7 – Les graphe $\mathcal{G}(\mathcal{N}_1^0, m_0|_{P_1^0})\times\mathcal{P}_u$ et $\mathcal{G}(\mathcal{N}^0, n_0|_{P^0})\times\mathcal{P}_u$.

Définition 4.3.13 (La sentence Φ_c). On pose

$$\begin{array}{lcllcl} U_1 & = & \{0{+}.0{-}, 1{+}.1{-}, 0?.0!, 1?.1!\} & U_1^0 & = & \{1{+}, 0{+}.0{-}, 1{+}.1{-}, 0?.0!, 1?.1!\} \\ U_2 & = & \{0{+}, 1{+}\} & U_1^1 & = & \{0{+}, 0{+}.0{-}, 1{+}.1{-}, 0?.0!, 1?.1!\} \end{array}$$

La sentence Φ_c est définie par

$$\Phi_c \stackrel{\text{def}}{=} \Phi_c^1 \wedge \Phi_c^2 \wedge \Phi_c^3$$

avec :

$$\begin{array}{rcccccc} \Phi_c^1 & \stackrel{\text{def}}{=} & & \nu X.\langle 0{+}.0{-}\rangle X & \wedge & \nu X.\langle 0?.0!\rangle \\ & & \wedge & \nu X.\langle 1{+}.1{-}\rangle X & \wedge & \nu X.\langle 1?.1!\rangle \end{array}$$

$$\Phi_c^2 \stackrel{\text{def}}{=} \nu X.\left(\bigwedge_{u\in U_1}[u]X \wedge \langle 0?\rangle \texttt{true}\right) \wedge \nu X.\left(\bigwedge_{u\in U_1}[u]X \wedge \langle 1?\rangle \texttt{true}\right)$$

et enfin

$$\begin{array}{rcccccc} \Phi_c^3 & \stackrel{\text{def}}{=} & & \nu X. & \left(\bigwedge_{u\in U_1^0}[u]X\right. & \wedge & \left.[0{+}]\ \nu Y.\left(\bigwedge_{u\in U_2}[u]Y \wedge \langle 0{-}\rangle \texttt{true}\right)\right) \\ & & \wedge & \nu X. & \left(\bigwedge_{u\in U_1^1}[u]X\right. & \wedge & \left.[1{+}]\ \nu Y.\left(\bigwedge_{u\in U_2}[u]Y \wedge \langle 1{-}\rangle \texttt{true}\right)\right) \end{array}$$

Remarquons que les sentences Φ_c^2 et Φ_c^3 peuvent s'écrire plus simplement, mais la forme proposée ici est plus adéquate aux preuves qui suivent.

Lemme 4.3.14. *Si* $(\mathcal{N}, m_0) \models \Phi_{\mathbf{B}} \wedge \Phi_{m_0} \wedge \Phi_c$ *alors pour tout* $u \in U^*$, $m_0[u\rangle m$ *implique, pour tout* $i \in [0,1]$ *:*

- *soit* $m[i{-}\rangle$ *et* $i?$ *est inactive au marquage* m*,*
- *soit* $m[i?\rangle$ *et* $i{-}$ *est inactive au marquage* m*.*

Le lemme précédent assure que pour tout réseau $(\mathcal{N}, m_0)$ tel que $(\mathcal{N}, m_0) \models \Phi_{\mathbf{B}} \wedge \Phi_{m_0} \wedge \Phi_c$ et pour toute exécution u dans U^*, il est toujours possible de statuer sur la nullité du compteur c_i. En particulier, ce lemme assure que les places de $(\mathcal{N}, m_0)$ qui ne sont pas colinéaires aux places de $\mathcal{N}_{k_0,k_1}$ ne peuvent empêcher simultanément le tir de $i{-}$ et de $i?$ et donc créer un blocage.
Afin de prouver le Lemme 4.3.14, nous prouvons 5 lemmes intermédiaires.

Lemme 4.3.15. *Si* $\mathcal{N} \models \Phi_c^1$ *toute place* p *de* $\mathcal{N}$ *vérifie :*

$$\begin{array}{llll} <p, 0{+}.0{->} & \geq 0 & <p, 1{+}.1{->} & \geq 0 \\ <p, 0?.0!> & \geq 0 & <p, 1?.1!> & \geq 0 \end{array}$$

Démonstration. Ce sont les quatre inéquations universelles de Φ_c^1 (voir la Définition 4.2.4 et le Lemme 4.2.5 page 92). □

Remarquons que les langages U_1^*, U_1^{0*}, U_1^{1*} et $U_2{}^*$ sont des sous-langages de U^* (et donc des sous-langages de L_u).

Lemme 4.3.16. *Si* $(\mathcal{N}, m_0) \models \Phi_c^2$ *alors pour tout* $i \in \{0,1\}$*, pour tout mot* $u \in U_1^*$*, nous avons :*

$$m_0[u\rangle \quad \textit{implique} \quad m_0[u.i?\rangle$$

Démonstration. La sentence Φ_c^2 est constituée de deux sentences, chacune portant sur un certain $i \in \{0,1\}$. Soit $(\mathcal{N}, m_0)$ tel que $(\mathcal{N}, m_0) \models \Phi_c^2$, soit $u \in U_1^*$ tel que $m_0[u\rangle m$, considérons la sous-formule ϕ_c^2 de Φ_c^2 avec

$$\phi_c^2 = \nu X. \left(\bigwedge_{u \in U_1} [u]X \wedge \langle i? \rangle \mathtt{true} \right)$$

Nous avons $m_0 \in [\![\phi_c^2]\!]_{(\mathcal{N},m_0)}$, ce qui implique $m \in [\![\phi_c^2]\!]_{(\mathcal{N},m_0)}$ puisque $u \in U_1^*$. Nous en déduisons $m \in [\![\langle i? \rangle \mathtt{true}]\!]_{(\mathcal{N},m_0)}$, ce qui implique $m[i?\rangle$ et donc $m_0[u.i?\rangle$. □

Lemme 4.3.17. *Si* $(\mathcal{N}, m_0) \models \Phi_c^3$ *alors pour tout* $i \in \{0,1\}$*, pour tout mot* u *tel que* $u = u_1.i+.u_2$ *avec* $u_1 \in U_1^{i*}$ *et* $u_2 \in U_2{}^*$*, nous avons :*

$$m_0[u\rangle \quad \textit{implique} \quad m_0[u.i-\rangle$$

Démonstration. La sentence Φ_c^3 est constituée de deux sentences, chacune portant sur un certain $i \in \{0,1\}$. Soit $(\mathcal{N}, m_0)$ tel que $(\mathcal{N}, m_0) \models \Phi_c^3$, soit u tel que $u = u_1.i+.u_2$ avec $u_1 \in U_1^{i*}$ et $u_2 \in U_2{}^*$, considérons les sous-formules ϕ_c^3 et ψ_c^3 de Φ_c^2 avec

$$\begin{array}{rcl} \phi_c^3 & = & \nu X. \left(\bigwedge_{u \in U_1} [u]X \wedge [i+]\ \psi_c^3 \right) \\ \psi_c^3 & = & \nu Y. \left(\bigwedge_{u \in U_2^i} [u]Y \wedge \langle i- \rangle \mathtt{true} \right) \end{array}$$

Posons $m_0[u_1\rangle m_1[i+\rangle m_2[u_2\rangle m_3$, nous avons $m_0 \in [\![\ \phi_c^3\ ]\!]_{(\mathcal{N},m_0)}$, ce qui implique $m_1 \in [\![\phi_c^3]\!]_{(\mathcal{N},m_0)}$ puisque $u_1 \in U_1^{i*}$. De $m_1[i+\rangle m_2$, nous déduisons $m_2 \in [\![\psi_c^3]\!]_{(\mathcal{N},m_0)}$. Puisque $u_2 \in U_2{}^*$, nous obtenons $m_3 \in [\![\ \psi_c^3\ ]\!]_{(\mathcal{N},m_0)}$, et en particulier $m_3 \in [\![\langle i- \rangle \mathtt{true}]\!]_{(\mathcal{N},m_0)}$. Finalement, $m_3[i-\rangle$, ce qui nous donne $m_0[u.i-\rangle$. □

Pour la suite, afin d'éviter les confusions, lorsque u est un mot sur Σ, on note $\wr u \wr$ pour l'image commutative de u, au sens de la Définition 1.3.2 page 12.

Lemme 4.3.18. *Si* $(\mathcal{N}, m_0) \models \Phi_{\mathbf{B}} \wedge \Phi_{m_0} \wedge \Phi_c$ *alors pour tout mot* $u \in U^*$*, il existe un mot* v *vérifiant* $\wr u \wr = \wr v \wr$ *et* $v = v_1.v_2$ *avec* $v_1 \in U_1^*$ *et* $v_2 \in U_2^*$*, et tel que :*

$$m_0[u\rangle \quad \textit{implique} \quad m_0[v\rangle$$

Démonstration. Puisque $(\mathcal{N}, m_0) \models \Phi_{\mathbf{B}} \wedge \Phi_{m_0}$, nous déduisons des Lemme 4.3.6 et 4.3.11 que $(\mathcal{N}, m_0)$ possède les places p_0' et p_1' avec $m_0(p_0') < \lambda_0$ et $m_0(p_1') < \lambda_1$. Puisque $m_0[u\rangle$, nous obtenons

$$|u|_{0+} \geq |u|_{0-} \text{ et } |u|_{1+} \geq |u|_{1-}$$

Puisque $(\mathcal{N}, m_0) \models \Phi_{m_0}$, nous avons :

$$m_0[a\rangle \text{ pour tout } a \in \{0+, 1+, 0?, 1?\}$$

Puisque $(\mathcal{N}, m_0) \models \Phi_c$, le Lemme 4.3.15 nous assure

$$\begin{array}{llll} <p, 0+.0->\ \geq\ 0 & & <p, 1+.1->\ \geq\ 0 \\ <p, 0?.0!>\ \geq\ 0 & & <p, 1?.1!>\ \geq\ 0 \end{array}$$

Nous construisons v_1 et v_2 en utilisant l'algorithme suivant :

Étape 1 Poser w égal à u restreint à l'alphabet $\{0+, 0-, 1+, 1-\}$, $w' := 1$ et

$$w'' := \underbrace{(0?.0!)\ldots(0?.0!)}_{|u|_{0?} \text{ fois}}.\underbrace{(1?.1!)\ldots(1?.1!)}_{|u|_{1?} \text{ fois}}$$

Étape 2 Décomposer w en $w = w_1.i+.w_2.i-.w_3$ ou $w = w_1.i-.w_2.i+.w_3$ pour un $i \in \{0, 1\}$ de manière à minimiser $|w_2|$. Poser $w := w_1.w_2.w_3$ et $w' := w'.i+.i-$.

Étape 3 Si $|w|_{0-} = |w|_{1-} = 0$ alors passer à l'Étape 4, sinon reprendre à l'Étape 2.

Étape 4 Poser $v_1 := w'.w''$ et $v_2 := w$.

On montre qu'à chaque étape de l'algorithme, $m_0[w'.w''.w\rangle$. À l'Étape 1, où $w' = 1$, puisque $m_0[0?\rangle$ et puisque pour toute place p de $\mathcal{N}$, $<p, 0?.0!> \geq 0$, nous avons $m_0[0?.0!\rangle m''$ avec pour tout place p de $\mathcal{N}$, $m''(p) \geq m_0(p)$. Nous pouvons alors itérer le raisonnement pour obtenir

$$m_0[\underbrace{(0?.0!)\ldots(0?.0!)}_{|u|_{0?} \text{ fois}}\rangle m'$$

avec pour tout place p de $\mathcal{N}$, $m'(p) \geq m_0(p)$. Puisque $m_0[1?\rangle$ et pour toute place p de $\mathcal{N}$, $<p, 1?.1!> \geq 0$, nous pouvons raisonner de même, pour obtenir finalement $m_0[w''\rangle m$ avec $m(p) \geq m_0(p)$.
Pour tout préfixe $v'.a$ de w avec $a \in \{0+, 0-, 1+, 1-\}$, soit $u'.a$ le préfixe de u tel que

$$\begin{array}{rclcrcl} |u'|_{0+} & = & |v'|_{0+} & \qquad & |u'|_{0-} & = & |v'w|_{0-} \\ |u'|_{1+} & = & |v'|_{1+} & & |u'|_{1-} & = & |v'|_{1-} \end{array}$$

Pour toute place p de $\mathcal{N}$, puisque $m_0[u\rangle$, nous avons $m_0[u'.a\rangle$ et donc

$$m_0(p) + \sum_{b\in\Sigma} |u'|_b <p, b> + <p, a> \geq 0$$

puisque u' et v' possèdent le même nombre d'occurrence des éléments de $\{0+, 0-, 1+, 1-\}$, que $|u'|_{i?} = |u'|_{i!}$ et que $|v'|_{i?} = |v'|_{i!} = 0$ pour $i \in \{0, 1\}$, nous pouvons décomposer l'inéquation précédente en

$$m_0(p) + <p, v'> + |u'|_{0?} \times <p, 0?.0!> + |u'|_{1?} \times <p, 1?.1!> + <p, a> \geq 0$$

Puisque $<p, 0?.0!> \geq 0$, $<p, 1?.1!> \geq 0$ et $|u'|_{i?} \leq |w''|_{i?}$, nous obtenons :

$$m_0(p) + <p, v'> + |w''|_{0?} \times <p, 0?.0!> + |w''|_{1?} \times <p, 1?.1!> + <p, a> \geq 0$$

et donc

$$m_0(p) + <p, w''.v'.a> \geq 0$$

Puisque cette dernière inéquation est vraie pour tout préfixe v' de w, nous obtenons $m_0[w''.w\rangle$, c.-à-d. $m_0[w'.w''.w\rangle$ puisque $w' = 1$.

À l'Étape 2 où w_p et w'_p désignent les valeurs calculées à la fin de l'étape précédente (Étape 1 ou Étape 3), supposons que la décomposition est $w_p = w_1.0+.w_2.0-.w_3$, les autres cas suivant le même raisonnement[4].
Nous montrons $m_0[w'_p.0+.0-.w''.w_1.w_2.w_3\rangle$ avec l'hypothèse $m_0[w'_p.w''.w_1.0+.w_2.0-.w_3\rangle$ d'après l'étape précédente. Remarquons que nécessairement, soit $w_2 = 1+^k$, soit $w_2 = 1-^k$ pour un certain k, sinon $|w_2|$ ne serait pas minimal. On suppose $w_2 = 1+^k$, l'autre cas suit le même raisonnement. Puisque $m_0[w'_p\rangle m_1$, $m_0[0+\rangle$ et $<p, 0+.0-> \geq 0$ pour toute place p de $\mathcal{N}$, nous avons $m_0[w'_p.0+.0-.w''\rangle m'_1$ avec $m'_1(p) \geq m_1(p)$. De $m_0[w'_p.w''.w_1\rangle m_2$, nous déduisons $m_0[w'_p.0+.0-.w''.w_1\rangle m'_2$.
Pour la suite, pour chaque place p de $\mathcal{N}$, deux cas sont possibles :

[4]Ces cas sont $w_p = w_1.0-.w_2.0+.w_3$, $w_p = w_1.1+.w_2.1-.w_3$ et $w_p = w_1.1-.w_2.1+.w_3$.

– soit $<p, 0-> \geq 0$, auquel cas $m_2[0-\rangle m_2''$ avec $m_2'(p) \geq m_2''(p)$ puisque $\wr w_p'.0+.0-.w''.w_1\wr = \wr w_p'.w''.w_1.0+\wr + \wr\, 0-\wr$. De $m_2''[w_2\rangle$, nous déduisons, pour tout $h \leq k$,

$$m_0(p) + <p, w_p'.0+.0-.w''.w_1> + h \times <p, 1+> \geq 0$$

– soit $<p, 0-> < 0$, auquel cas $<p, 0+> \geq 0$. Si $<p, 1+> \geq 0$, nous obtenons $m_2'[1+^k\rangle m_3'$, c.-à-d. $m_2'[w_2\rangle m_3'$; sinon $<p, 1+> < 0$. Dans ce cas, puisque $m_2[0+.w_2.0-\rangle$, nous avons

$$m_0(p) + <p, w_p'.w''.w_1> + <p, 0+> + <p, 0-> + k \times <p, 1+> \geq 0$$

puisque $<p, 1+> < 0$, nous déduisons, pour tout $h \leq k$

$$m_0(p) + <p, w_p'.0+.0-.w''> + <p, 0-> + h \times <p, 1+> \geq 0$$

Nous obtenons $m_0[w_p'.0+.0-.w''.w_1.w_2\rangle m_3$. Puisque $\wr w_p'.0+.0-.w''.w_1.w_2\wr = \wr w_p'.w''.w_1.0+.w_2.0-\wr$, nous avons également $m_0[w_p'.w''.w_1.0+.w_2.0-\rangle m_3$ avec $m_3[w_3\rangle$. Nous déduisons finalement $m_0[w_p'.0+.0-.w''.w_1.w_2.w_3\rangle$.

L'Étape 3 ne modifie rien.

À l'Étape 4, nous avons $m_0[v_1.v_2\rangle$. Par construction, $\wr u\wr = \wr v\wr$, $v_1 \in U_1^*$ et $v_2 \in U_2^*$. □

Lemme 4.3.19. *Si $(\mathcal{N}, m_0) \models \Phi_{\mathbf{B}} \wedge \Phi_{m_0} \wedge \Phi_c$, alors pour tout $i \in \{0,1\}$, pour tout $u \in U^*$ tel que $|u|_{i+} > |u|_{i-}$, nous avons :*

$$m_0[u\rangle \quad \textit{implique} \quad m_0[u.i-\rangle$$

Démonstration. D'après le Lemme 4.3.18, il existe un mot v vérifiant $\wr u\wr = \wr v\wr$ et $v = v_1.v_2$ avec $v_1 \in U_1^*$, $v_2 \in U_2^*$ et $m_0[v\rangle$. Puisque $v_1 \in U_1^*$, $|v_1|_{i+} = |v_1|_{i-}$, et puisque $|u|_{i+} > |u|_{i-}$, nous pouvons décomposer v en posant $v = v_1.v_2'.i+.v_2''$ avec $|v_2'|_{i+} = 0$. Puisque $v_1 \in {U_1}^*$ et $U_1 \subseteq U_1^i$, nous avons $v_1.v_2' \in U_1^{i*}$ et $v_2'' \in U_2$. Le Lemme 4.3.17 nous assure $m_0[v.i-\rangle$, et puisque $\wr u\wr = \wr v\wr$, nous avons également $m_0[u.i-\rangle$. □

Lemme 4.3.20. *Si $(\mathcal{N}, m_0) \models \Phi_{\mathbf{B}} \wedge \Phi_{m_0} \wedge \Phi_c$, alors pour tout $i \in \{0,1\}$, pour tout $u \in U^*$ tel que $|u|_{i+} = |u|_{i-}$, nous avons :*

$$m_0[u\rangle \quad \textit{implique} \quad m_0[u.i?\rangle$$

Démonstration. D'après le Lemme 4.3.18, il existe un mot v vérifiant $\wr u\wr = \wr v\wr$ et $v = v_1.v_2$ avec $v_1 \in U_1^*$, $v_2 \in U_2^*$ et $m_0[v\rangle$. Puisque $|u|_{i+} = |u|_{i-}$ pour $i \in \{0,1\}$ et $v_2 \in U_2$, nous avons $v_2 = 1$. Par conséquent $v \in U_1^*$; par le Lemme 4.3.16, nous obtenons $m_0[v.i?\rangle$, et puisque $\wr u\wr = \wr v\wr$, nous avons également $m_0[u.i?\rangle$. □

Nous sommes maintenant à même de prouver le Lemme 4.3.14.

Démonstration du Lemme 4.3.14. De $(\mathcal{N}, m_0) \models \Phi_{\mathbf{B}} \wedge \Phi_{m_0}$, nous déduisons du Lemme 4.3.6 que $(\mathcal{N}, m_0)$ possède les places p'_0 et p'_1 avec $m_0(p'_0) < \lambda_0$ et $m_0(p'_1) < \lambda_1$. Puisque $m_0[u\rangle$,

$$|u|_{0+} \geq |u|_{0-} \text{ et } |u|_{1+} \geq |u|_{1-}$$

Par conséquent, pour $i \in \{0,1\}$, soit $|u|_{i+} > |u|_{i-}$, auquel cas le Lemme 4.3.19 nous assure $m_0[u.i\text{–}\rangle$, soit $|u|_{i+} > |u|_{i-}$, auquel cas le Lemme 4.3.20 nous assure $m_0[u.i?\rangle$. □

Nous montrons maintenant que le réseau $\mathcal{N}_{k_0,k_1}$ est bien modèle de Φ_c.

Lemme 4.3.21. *Pour tout k_0, k_1, nous avons $(\mathcal{N}_{k_0,k_1}, n_0) \models \Phi_c$.*

Démonstration. Le réseau $(\mathcal{N}_{k_0,k_1}, n_0)$ vérifie trivialement les inéquations universelles de la sentence Φ_c^1. De plus, pour toute séquence $u \in U_1^*$, $n_0(p_{\bar{i}}) + <p_{\bar{i}}, u> = k_i$ et donc $n_0[u\rangle m$ implique $n_0[u.i?\rangle$; nous obtenons $(\mathcal{N}_{k_0,k_1}, n_0) \models \Phi_c^2$. Pour toute séquence $u = u_1.i+.u_2$ avec $u_1 \in U_1^{i*}$ et $u_2 \in U_2^*$, $<p_i, u_1> = 0$ et $<p_i, u_2> \geq 0$ donc $n_0(p_i) + <p_i, u> > 0$; par conséquent $n_0[u\rangle$ implique $n_0[u.i\text{–}\rangle$. Nous obtenons $(\mathcal{N}_{k_0,k_1}, n_0) \models \Phi_c^3$ et finalement $(\mathcal{N}_{k_0,k_1}, n_0) \models \Phi_c$. □

4.3.2.3 Sentence pour l'existence d'une borne

Soit $M = (\{q_0, \ldots, q_{n+1}\}, \{c_0, c_1\}, \{\delta_0, \ldots, \delta_n\})$ une machine à deux compteurs.

Définition 4.3.22 (Sentence Φ_M de l'exécution bornée d'une machine à deux compteurs). Soit $\{X_0, \ldots, X_{n+1}\}$ un ensemble de variables et soient $\phi_0, \ldots \phi_{n+1}$ les formules de $\mathbf{L}_\nu$ définies comme suit, avec $l \in [0, n+1]$:

– si $\delta_l = (c_i := c_i + 1;\ \text{goto } q_h)$ alors

$$\phi_l(X_0, \ldots X_{n+1}) \stackrel{\text{def}}{=} \langle i+\rangle X_h$$

– sinon, $\delta_l = (\text{if } c_i = 0 \text{ then } (c_i := c_i + 1;\ \text{goto } q_h) \text{ else } (c_i := c_i - 1;\ \text{goto } q_{h'}))$, dans ce cas

$$\phi_l(X_0, \ldots X_{n+1}) \stackrel{\text{def}}{=} [i?]\langle i!\rangle\langle i+\rangle X_h \wedge [i\text{–}]X_{h'}$$

– enfin, pour $l = n + 1$

$$\phi_{n+1} \stackrel{\text{def}}{=} \texttt{true}$$

Soient $\Psi_0, \ldots, \Psi_{n+1}$ les sentences définies par :

$$< \Psi_1, \ldots, \Psi_{n+1} > \stackrel{\text{def}}{=} \nu < X_1, \ldots, X_{n+1} > . < \phi_1, \ldots, \phi_{n+1} > (< X_1, \ldots, X_{n+1} >)$$

La sentence Φ_M est alors la sentence Ψ_0.

Remarque 4.3.23. La différence entre la sentence Φ_M définie ci-dessus et la sentence Φ_M définie dans le Chapitre 2, si l'on fait abstraction de la méthode de test à zéro, réside uniquement dans l'opérateur de point-fixe. Au Chapitre 2, un μ-point-fixe impose la terminaison alors qu'ici, l'opérateur ν permet la non-terminaison. Dans ce chapitre, l'indécidabilité réside dans la décision de l'existence d'une borne lors de l'exécution de M.

Nous montrons maintenant une série de lemmes dont l'objectif est d'établir le Lemme *de projection*. Le Lemme de projection permet, à partir d'un réseau $(\mathcal{N}, m_0)$ tel que $(\mathcal{N}, m_0) \models \Gamma_M$ d'obtenir $(\mathcal{N}_{k_0,k_1}, n_0) \models \Phi_M$. Pour montrer ce lemme, nous montrons successivement les résultats suivants : $(\mathcal{N}_1, m_0|_{P_1}) \models \Phi_M$, puis $(\mathcal{N}_1, m_0|_{P_1}) \cap L_u \models \Phi_M$, puis $\mathcal{L}(\mathcal{N}_{k_0,k_1}, n_0) \cap L_u \models \Phi_M$ et enfin $(\mathcal{N}_{k_0,k_1}, n_0) \models \Phi_M$. Nous rappelons que $\mathcal{N}_1$ désigne le sous-réseau de $\mathcal{N}$ composé de l'ensemble P_1 de places avec $P_1 = \{p'_0, p'_{\overline{0}}, p'_1, p'_{\overline{1}}\}$.

Lemme 4.3.24. *Soit* $(\mathcal{N}, m_0)$ *tel que* $(\mathcal{N}, m_0) \models \Phi_{\mathbf{B}} \wedge \Phi_{m_0} \wedge \Phi_c$.

$$(\mathcal{N}, m_0) \models \Phi_M \textit{ implique } (\mathcal{N}_1, m_0|_{P_1}) \models \Phi_M$$

Démonstration. On note L le langage $\mathcal{L}(\mathcal{N}, m_0)$ et L_1 le langage $\mathcal{L}(\mathcal{N}_1, m_0|_{P_1})$; nous avons $L \subseteq L_1$. Pour tout valuation *val* de $\{X_1, \ldots, X_{n+1}\}$ dans 2^L, pour tout $l \in [1, n+1]$, on montre que :

$$[\![\, \phi_l \,]\!]_L^{[val]} \subseteq [\![\, \phi_l \,]\!]_{L_1}^{[val]}$$

Considérons les différents cas pour ϕ_l :

- si $l = n+1$, $\phi_l = \texttt{true}$ et puisque $L \subseteq L_1$, nous avons $[\![\, \phi_l \,]\!]_L^{[val]} \subseteq [\![\, \phi_l \,]\!]_{L_1}^{[val]}$.
- si $\phi_l(X_0, \ldots X_{n+1}) = \langle i+ \rangle X_h$, alors pour tout $w \in L$ tel que $w \in [\![\, \phi_l \,]\!]_L^{[val]}$, nous avons $w.i+ \in val(X_h)$. Puisque $L \subseteq L_1$, nous avons également $w.i+ \in L_1$, ce qui entraîne $w \in [\![\, \phi \,]\!]_{L_1}^{[val]}$.
- si $\phi_l(X_0, \ldots X_{n+1}) \stackrel{\text{def}}{=} [i?]\langle i! \rangle \langle i+ \rangle X_h \wedge [i-] X_{h'}$, pour tout $w \in L$ tel que $w \in [\![\, \phi_l \,]\!]_L^{[val]}$ quatre cas se présentent, suivant l'appartenance de $w.i?$ où de $w.i-$ à L. Les cas $w.i? \notin L$ et $w.i- \notin L$ ainsi que le cas $w.i? \in L$ et $w.i- \in L$ sont tous deux écartés par le Lemme 4.3.14, il nous reste deux cas possibles :

1. soit $w.i? \notin L$ et $w.i- \in L$, nous avons $w.i- \in val(X_h)$. Puisque $L \subseteq L_1$, nous avons également $w.i- \in L_1$, ce qui entraîne $w \in [\![\phi]\!]^{[val]}_{L_1}$.
2. soit $w.i? \in L$ et $w.i- \notin L$, nous avons $w.i?.i!.i+ \in val(X_h)$. Puisque $L \subseteq L_1$, nous avons également $w.i?.i!.i+ \in L_1$, ce qui entraîne $w \in [\![\phi]\!]^{[val]}_{L_1}$.

Par monotonie de l'opérateur de point fixe, l'inégalité est préservée et nous obtenons :

$$[\![\Psi_0]\!]^{[val]}_{L} \subseteq [\![\Psi_0]\!]^{[val]}_{L_1}$$

Par conséquent $1 \in [\![\Psi_0]\!]^{[val]}_{L}$ implique $1 \in [\![\Psi_0]\!]^{[val]}_{L_1}$ et finalement $(\mathcal{N}_1, m_0|_{P_1}) \models \Phi_M$. □

Lemme 4.3.25. *Soit $(\mathcal{N}, m_0)$ un réseau initialisé, $(\mathcal{N}, m_0) \models \Phi_M$ si et seulement si $\mathcal{L}(\mathcal{N}, m_0) \cap L_u \models \Phi_M$.*

Démonstration. Remarquons que pour tout $l \in [0, n+1]$, la formule ϕ_l est définie en utilisant uniquement des éléments de U, par conséquent, en notant $L_\mathcal{N}$ pour $\mathcal{L}(\mathcal{N}, m_0)$, pour toute valuation $val : Var \to 2^{L_\mathcal{N} \cap L_u}$, nous avons :

$$[\![\phi_l]\!]^{[val]}_{L_\mathcal{N}} = [\![\phi_l]\!]^{[val]}_{L_u \cap L_\mathcal{N}}$$

Par conséquent

$$1 \in [\![\Psi_0]\!]_{L_\mathcal{N}} \text{ si et seulement si } 1 \in [\![\Psi_0]\!]_{L_u \cap L_\mathcal{N}}$$

Nous obtenons finalement : $(\mathcal{N}, m_0) \models \Phi_M$ si et seulement si $\mathcal{L}(\mathcal{N}, m_0) \cap L_u \models \Phi_M$ □

Lemme 4.3.26 (Lemme de projection). *S'il existe $(\mathcal{N}, m_0)$ tel que $(\mathcal{N}, m_0) \models \Gamma_M$, alors $(\mathcal{N}_{k_0,k_1}, n_0) \models \Phi_M$*

Démonstration. La preuve se décompose de la manière suivante :

$$\begin{array}{cc}
(\mathcal{N}, m_0) \models \Gamma_M & \\
\Downarrow & \text{(Lemme 4.3.24)} \\
(\mathcal{N}_1, m_0|_{P_1}) \models \Phi_M & \\
\Downarrow & \text{(Lemme 4.3.25)} \\
\mathcal{L}(\mathcal{N}_1, m_0|_{P_1}) \cap L_u \models \Phi_M & \\
\Downarrow & \text{(Lemme 4.3.12)} \\
\mathcal{L}(\mathcal{N}_{k_0,k_1}, n_0) \cap L_u \models \Phi_M & \\
\Downarrow & \text{(Lemme 4.3.25)} \\
(\mathcal{N}_{k_0,k_1}, n_0) \models \Phi_M &
\end{array}$$

□

4.3.2.4 Simulation de M par $\mathcal{N}_{k_0,k_1}$ dans Φ_M

Nous donnons les lemmes qui font le lien entre l'exécution de M pour $(q_0, 0, 0)$ et la satisfaction de Φ_M par $(\mathcal{N}_{k_0,k_1}, n_0)$.

Définition 4.3.27 (Marquage de $\mathcal{N}_{k_0,k_1}$ associé à une configuration de M). Le marquage m de $\mathcal{N}_{k_0,k_1}$ *associé* à la configuration $J = (q, j_0, j_1)$ de M est le marquage défini par :

$$\begin{array}{rclcrcl} m_0(p_0) & = & j_0 & & m_0(p_1) & = & j_1 \\ m_0(p_{\bar{0}}) & = & k_0 - j_0 & & m_0(p_1) & = & k_1 - j_1 \\ m_0(p_{0?}) & = & 0 & & m_0(p_{1?}) & = & 0 \end{array}$$

Lemme 4.3.28 (Premier lemme de simulation). *Soient $J = (q_l, j_0, j_1)$ et $J' = (q_{l'}, j'_0, j'_1)$ deux configurations de M telles que J est suivie par J' lors de l'exécution de M, soient m le marquage associé à J et soit m' le marquage associé à J'. Pour tout $w \in L_u$ tel que $n_0[w\rangle m$ et $w \in [\![\Psi_l]\!]_{(\mathcal{N}_{k_0,k_1},n_0)}$, il existe $w' \in L_u$ tel que w' soit un préfixe de w, $n_0[w'\rangle m'$ et $w' \in [\![\Psi_{l'}]\!]_{(\mathcal{N}_{k_0,k_1},n_0)}$.*

Démonstration. La preuve est au cas par cas suivant la formule ϕ_l :

- si $\phi_l(X_0, \ldots X_{n+1}) = \langle i+\rangle X_{l'}$, alors $j'_i = j_i + 1$, dans ce cas $w' = w.i+$, $w' \in val(X'_l)$ et par conséquent $w' \in [\![\Psi_{l'}]\!]_{(\mathcal{N}_{k_0,k_1},n_0)}$.
- si $\phi_l(X_0, \ldots X_{n+1}) \stackrel{\text{def}}{=} [i?]\langle i!\rangle\langle i+\rangle X_h \wedge [i-]X_{h'}$, deux cas se présentent :
  1. soit $j_i \neq 0$, alors $l' = h'$ et $j'_i = j_i - 1$, dans ce cas $w' = w.i+$ et par conséquent $w' \in [\![\Psi_{l'}]\!]_{(\mathcal{N}_{k_0,k_1},n_0)}$,
  2. soit $j_i = 0$, alors $l' = h$ et $j'_i = j_i + 1$, dans ce cas $w' = w.i?.i!.i+$ et par conséquent $w' \in [\![\Psi_{l'}]\!]_{(\mathcal{N}_{k_0,k_1},n_0)}$,

□

Lemme 4.3.29 (Second lemme de simulation). *Soit M une machine à deux compteurs telle que l'exécution de M pour $J_0 = (q_0, j_0, j_1)$ soit bornée. Notons k'_0 la valuation maximale du compteur 0 et k'_1 la valuation maximale du compteur 1. Soit $\mathcal{N}_{k_0,k_1}$ le réseau de $\mathbf{B}$ avec $k_0 = k'_0 + 1$ et $k_1 = k'_1 + 1$ (pour éviter d'avoir $k_i = 0$, qui contredirait $(\mathcal{N}_{k_0,k_1}, m_0) \models \Phi_{m_0}$). Nous avons :*

$$(\mathcal{N}_{k_0,k_1}, n_0) \models \Gamma_M$$

Démonstration. Nous avons déjà $(\mathcal{N}_{k_0,k_1}, n_0) \models \Phi_{\mathbf{B}} \wedge \Phi_{m_0} \wedge \Phi_c$ (voir les Lemmes 4.3.6, 4.3.10 et 4.3.21), il nous reste à montrer $(\mathcal{N}_{k_0,k_1}, n_0) \models \Phi_M$. Deux cas sont possibles suivant l'exécution de M :

(i) soit l'exécution est infinie, auquel cas il existe une séquence de configurations *initiale* $J_0, J_1, \ldots, J_h$ et une séquence *itérée* de configurations $J_{h+1}, J_{h+2}, \ldots, J_{h'}$ telles que la séquence de configurations associée à l'exécution de M pour J_0 soit :

$$J_0, J_1, \ldots, J_h, J_{h+1}, J_{h+2}, \ldots, J_{h'}, J_{h+1}, J_{h+2}, \ldots, J_{h'} \ldots$$

(ii) soit l'exécution de M est finie, auquel cas on note $J_0, J_1, \ldots, J_{h'}$ la séquence de configurations produite, avec $J_{h'} = (q_{n+1}, j_0^{h'}, j_1^{h'})$.

Définissons une séquence de mots $u_0, \ldots, u_{h'}$ de L_u itérativement par :
- $u_0 = 1$,
- Si $J_y = (q_l, j_0^y, j_1^y)$ et $\delta_l = (c_i := c_i + 1;\ \text{goto } q_{l'})$ alors $u_{y+1} = u_y.i+$
- Si $J_y = (q_l, j_0^y, j_1^y)$, $\delta_l = (\text{if } c_i = 0 \text{ then } (c_i := c_i + 1;\ \text{goto } q_{l'}) \text{ else } (c_i := c_i - 1;\ \text{goto } q_{l''}))$ et $j_i^y \neq 0$ alors $u_{y+1} = u_y.i-$
- Si $J_y = (q_l, j_0^y, j_1^y)$, $\delta_l = (\text{if } c_i = 0 \text{ then } (c_i := c_i + 1;\ \text{goto } q_{l'}) \text{ else } (c_i := c_i - 1;\ \text{goto } q_{l''}))$ et $j_i^y = 0$ alors $u_{y+1} = u_y.i?.i!.i+$

Remarquons que puisque les compteurs sont bornés par k_0' et k_1' et que $k_0 = k_0' + 1$ et $k_1 = k_1' + 1$, tous les u_y sont dans $\mathcal{L}(\mathcal{N}_{k_0,k_1}, n_0)$ pour tout $y \in [0, h']$.

Posons $V_l\{u_y \mid l \text{ est l'index associé à } u_y\}$. On définit l'*index associé* au mot u_y par l'entier $l \in [0, n+1]$ tel que $J_y = (q_l, j_0^y, j_1^y)$. Nous montrons que le vecteur d'ensembles $< V_0, \ldots, V_{n+1} >$ est un pré-point-fixe.

Soit val la valuation de $\{X_0, \ldots, X_{n+1}\}$ dans 2^{Σ^*} définie par $val(X_l) = V_l$, on montre $u_y \in [\![\, \phi_l \,]\!]^{[val]}_{(\mathcal{N}_{k_0,k_1},n_0)}$pour tout $y \in [0, h'-1]$ avec l l'index associé à u_y. Trois cas sont possibles suivant ϕ_l :
- si $\phi_l(X_0, \ldots X_{y+1}) = \langle i+\rangle X_{l'}$, alors $j_i^{y+1} = j_i^y + 1$, dans ce cas $u_{y+1} = u_y.i+$ et l'index associé à u_{y+1} est l'. Nous en déduisons $u_y \in [\![\, \phi_l \,]\!]^{[val]}_{(\mathcal{N}_{k_0,k_1},n_0)}$.
- si $\phi_l(X_0, \ldots X_{y+1}) \stackrel{\text{def}}{=} [i?]\langle i!\rangle\langle i+\rangle X_h \wedge [i-]X_{h'}$, deux cas se présentent :
  1. soit $j_i^y = 0$, alors $j_i^{y+1} = j_i^y + 1$, dans ce cas $u_{y+1} = u_y.i+$ et l'index associé à u_{y+1} est l', par conséquent $u_y \in [\![\, \phi_l \,]\!]^{[val]}_{(\mathcal{N}_{k_0,k_1},n_0)}$.
  2. soit $j_i^y \neq 0$, alors $j_i^{y+1} = j_i^y - 1$, dans ce cas $u_{y+1} = u_y.i?.i!.i-$ et l'index associé à u_{y+1} est l'', par conséquent $u_y \in [\![\, \phi_l \,]\!]^{[val]}_{(\mathcal{N}_{k_0,k_1},n_0)}$.

Dans le cas (i), avec la convention $J_{h'+1} = J_{h+1}$, on montre de la même manière le résultat $u_{h'} \in [\![\, \phi_l \,]\!]^{[val]}_{(\mathcal{N}_{k_0,k_1},n_0)}$ où l est l'index associé à $u_{h'}$. Dans le cas (ii), il suffit de remarquer que l'index associé à $u_{h'}$ est $n+1$ et que $\phi_{n+1} = \texttt{true}$, ce qui

nous donne immédiatement $u_{h'} \in [\![\phi_{n+1}]\!]^{[val]}_{(\mathcal{N}_{k_0,k_1},n_0)}$.

Nous avons montré que le vecteur d'ensembles $< val(X_1), \ldots, val(X_{n+1}) >$ est un pré-point-fixe, ce qui implique $val(X_l) \subseteq [\![\Psi_l]\!]_{(\mathcal{N}_{k_0,k_1},n_0)}$ pour tout $l \in [0, n+1]$. En particulier, dans le cas $l = 0$, puisque nous avons $u_0 \in val(X_0)$ et $u_0 = 1$, nous en déduisons $(\mathcal{N}_{k_0,k_1}, n_0) \models \Phi_M$ et finalement $(\mathcal{N}_{k_0,k_1}, n_0) \models \Gamma_M$. □

4.3.3 Preuve du Théorème 4.3.1

Soit M une machine à deux compteurs, on montre que l'exécution de M pour $J_0 = (q_0, j_0, j_1)$ est bornée si et seulement si il existe un réseau $(\mathcal{N}, m_0)$ tel que $(\mathcal{N}, m_0) \models \Gamma_M$.

Supposons l'existence de $(\mathcal{N}, m_0)$ tel que $(\mathcal{N}, m_0) \models \Gamma_M$. Par le Lemme 4.3.26, on obtient un réseau $\mathcal{N}_{k_0,k_1} \in \mathbf{B}$ tel que $(\mathcal{N}_{k_0,k_1}, n_0) \models \Phi_M$. Soit $J_0, J_1, \ldots, J_h, \ldots$ la séquence de configurations associée à l'exécution de M pour J_0 ; puisque $1 \in [\![\Phi_0]\!]_{(\mathcal{N}_{k_0,k_1},n_0)}$, par application inductive du Lemme 4.3.28 nous obtenons une séquence de mots $w_0, w_1, \ldots, w_h, \ldots$ de L_u telle que $w_0 = 1$ et toute configuration J_h est associée au marquage m_h défini par $n_0[w_h\rangle m_h$. Les places p_0 et p_1 sont trivialement bornées pour les séquences de tir de L_u, ce qui implique que lors de l'exécution de M, les valuations des compteurs sont bornées.

Supposons maintenant que l'exécution de M pour $J_0 = (q_0, j_0, j_1)$ soit bornée. D'après le Lemme 4.3.29, il existe k_0, k_1 tels que $(\mathcal{N}_{k_0,k_1}, n_0) \models \Gamma_M$.

Nous avons réduit le problème de l'exécution bornée d'une machine à deux compteurs au problème $\mathbf{Sat}(\mathbf{L}_\nu, \mathbf{PN})$. Par la Proposition 4.3.4, ce premier problème est indécidable, ce qui entraîne l'indécidabilité de $\mathbf{Sat}(\mathbf{L}_\nu, \mathbf{PN})$.

4.4 Conclusion

Dans ce chapitre, nous avons défini la théorie d'un réseau $\mathcal{N}$ comme un intervalle de langage $I_\mathcal{N}$ tel que tout réseau $(\mathcal{N}', m'_0)$ dont le langage $\mathcal{L}(\mathcal{N}', m'_0)$ est dans $I_\mathcal{N}$ possède, pour chaque place p de $\mathcal{N}$, une place p' colinéaire à p. La théorie d'un réseau $\mathcal{N}$ montre la puissance d'expressivité des intervalles de langages sur la structure des réseaux.

Nous avons vu au Chapitre 1 que la synthèse de réseau est possible à partir des intervalles de langages, puis au Chapitre 3 que la synthèse est également possible à partir d'un fragment des spécifications modales. Nous avons montré dans ce Chapitre que la synthèse de réseaux purs est indécidable à partir des formules du

Nu-Calcul conjonctif, et par conséquent à partir des spécifications modales. Ce résultat repose sur l'expression de la théorie d'un réseau structurellement borné permettant de simuler deux compteurs et leur test à zéro, de l'expression de ce test à zéro en Nu-Calcul conjonctif et de l'expression dans cette même logique du problème indécidable de l'exécution bornée d'une machine à compteurs.

L'indécidabilité de la synthèse pour les réseaux purs à partir des formules du Nu-Calcul nous permet d'annoncer que pour ces réseaux, le problème de satisfiabilité est plus difficile que le problème du model-checking (qui est décidable sur le Nu-Calcul [Esp97]). Ce dernier résultat est à comparer aux résultats antérieurs sur la classe des réseaux étiquetés qui montrent que le model-checking est plus difficile que le problème de satisfiabilité (pour lequel nous avons le *small model property*).

Conclusion

Dans ce document, nous avons étudié les limites de la décidabilité de la synthèse de réseaux de Petri à partir de spécifications qui sont exprimées dans des logiques du temps arborescent. Nous avons montré que le problème de synthèse est indécidable pour le Mu-Calcul ainsi que pour le Nu-Calcul Conjonctif dans le cas des réseaux purs ; le Nu-Calcul conjonctif étant un fragment du Mu-Calcul, suffisamment affaibli pour que le Model-Checking de réseaux de Petri soit décidable[5]. Nous avons introduit un reconnaisseur associé aux formules du Nu-Calcul conjonctif : les spécifications modales qui sont une version simplifiée des automates alternants. Nous avons montré que le Nu-Calcul Conjonctif et les spécifications modales spécifient les mêmes modèles. Nous avons ensuite proposé un fragment structurel de ces spécifications pour lequel nous avons montré que la synthèse est décidable et effective ; ce dernier résultat étend les résultats antérieurs de synthèse de réseaux. L'ensemble de ces résultats montre que la synthèse de réseaux de Petri à partir de logiques du temps arborescent n'est possible que pour des logiques dont l'expressivité est comparable à un fragment des spécifications modales.

Les deux résultats d'indécidabilité de nos travaux, respectivement celui concernant le Mu-Calcul et celui concernant le Nu-Calcul Conjonctif, reposent sur la réduction de problèmes indécidables des machines à compteurs. Nous avons montré que l'interprétation de formules logiques sur la classe des réseaux de Petri leur confère une grande expressivité, suffisante pour simuler des machines à compteurs.

La preuve d'indécidabilité de la synthèse à partir des formules du Mu-Calcul repose sur l'introduction des réseaux de compteurs. Nous construisons pour une machine à compteur M, une formule Φ_M telle que : si Φ_M est satisfiable sur la classe des réseaux de Petri, alors il existe un réseau de compteur modèle de Φ_M ; de plus, déterminer si il existe un réseau de compteurs modèle de Φ_M se réduit au problème de déterminer l'existence d'une configuration initiale pour laquelle l'exécution de M termine. La clé de cette preuve est la construction d'une formule de test à zéro d'un compteur, qui ne peut être vérifiée que par une place de réseau de compteurs, qui témoigne alors de la vacuité du compteur.

Pour prouver l'indécidabilité de la synthèse à partir de formules du Nu-Calcul Conjonctif, nous avons proposé deux familles de formules du Nu-Calcul Conjonctif qui, lorsqu'elles sont interprétées sur la classe des réseaux de Petri, expriment des propriétés uniquement structurelles sur leurs modèles. La première famille de formules, dite *formules universelles*, permet d'exprimer, par une formule un ensemble $\mathcal{I}$ d'inéquations concernant des relations de flot. Les modèles de cette formule sont alors les réseaux dont toute place vérifie $\mathcal{I}$. La deuxième famille de formules,

[5] Le Mu-Calcul Conjonctif est dépourvu des modalités suivantes : le "ou", la négation et le plus petit point-fixe

dite *formules existentielles*, permet d'exprimer, par une formule un ensemble $\mathcal{E}$ d'équations concernant des relations de flot. Les modèles de cette formule sont alors les réseaux qui possèdent une place vérifiant $\mathcal{E}$. Nous avons montré que toute formule de ces deux familles de formules peut être exprimée avec un intervalle de langages (un réseau modèle de la formule est un réseau dont le langage appartient à l'intervalle). Nous avons utilisé les formules existentielles pour dégager le concept de théorie d'un réseau. Exprimer la théorie d'un réseau $\mathcal{N}$ consiste à construire un intervalle de langages $I_{\mathcal{N}}$ tel que tout réseau dont le langage appartient à $I_{\mathcal{N}}$ contienne, dans sa structure, le réseau $\mathcal{N}$. Nous avons également donné quelques éléments pour exprimer la théorie d'une famille de réseaux, qui pourra faire l'objet de travaux futurs.

La preuve de l'indécidabilité de la synthèse de réseaux à partir du Nu-Calcul Conjonctif repose sur trois points importants : le premier point est la théorie d'une famille **B** de réseaux, nommés réseaux de compteurs bornées ; le deuxième point est l'élaboration d'une formule de test à zéro s'exprimant sur les réseaux de compteurs bornés sans utiliser la modalité "ou" ; le troisième point est l'expression en Nu-Calcul Conjonctif d'un problème indécidable sans utiliser l'opérateur de plus petit point fixe, par opposition aux problèmes liés à la terminaison. La solution apportée au troisième point est le choix du problème consistant à décider si les compteurs d'une machine restent bornés lors d'une certaine exécution de celle-ci. Les résultats d'indécidabilité présentés dans ce document ne s'appliquent pas à la synthèse de réseaux bornés dont l'étude des limites de décidabilité constitue une perspective en vue de compléter l'étude menée sur les réseaux généraux dans cette thèse.

Suite à ces résultats d'indécidabilité, nous avons proposé une méthode de synthèse, pour un fragment structurel des spécifications modales (les $\mathfrak{S}_{0,1}$-Spécifications modales[6]) dont nous avons montré qu'elles caractérisent un treillis infini d'intervalles de langages. De fait, ce résultat de décidabilité étend la synthèse à partir de langages [Dar04] ; il est incomparable à celui de la synthèse à partir de spécifications automatiques [BD04], pour lequel l'ensemble des états du réseau est connu a priori et régulier. Les $\mathfrak{S}_{0,1}$-Spécifications correspondent à un fragment du Nu-Calcul conjonctif difficilement caractérisable syntaxiquement, ce qui renforce l'intérêt porté dans ce document aux spécifications modales.

Nous avons également montré que le problème de la recherche d'un marquage initial d'un réseau donné, pour satisfaire une sentence du Mu-Calcul donnée est

[6] Notées spécifications $R^*(\mathfrak{S}_0 \cup \mathfrak{S}_1)$ dans le Chapitre 3

indécidable, y compris pour des classes de réseaux particulièrement restreintes (réseaux ordinaires, graphes marqués). De manière plus générale, il apparaît que la difficulté du problème de synthèse réside dans la recherche d'un réseau dont les comportements *initiaux* sont adéquats alors que le problème de spécifier et de synthétiser un réseau en connaissant ses comportements *permanents* (à l'infini) est plus aisé. Nous suggérons donc d'orienter les études des problèmes de synthèse de la manière suivante : rechercher des méthodes de décomposition d'une formule logique, dans le but d'exprimer celle-ci comme la conjonction d'une formule concernant les comportements initiaux et d'une formule concernant les comportements permanents ; puis de rechercher des méthodes de synthèse d'un système constitué d'un processus fini, qui réalise les comportements initiaux, et d'un réseau, qui réalise les comportements permanents.

Bibliographie

[AM94] Anuchit Anuchitanukul and Zohar Manna. Realizability and synthesis of reactive modules. In *Computer Aided Verification, 6th International Conference, CAV '94, Stanford, California, USA*, volume 818 of *Lecture Notes in Computer Science*, pages 156–168. Springer, 1994.

[AN01] A. Arnold and D. Niwinski. *Rudiments of mu-calculus*. North-Holland, 2001.

[AVW03] André Arnold, Aymeric Vincent, and Igor Walukiewicz. Games for synthesis of controllers with partial observation. *Theoretical Computer Science*, 303(1) :7–34, June 2003.

[BBD95] Eric Badouel, Luca Bernardinello, and Philippe Darondeau. Polynomial algorithms for the synthesis of bounded nets. *Lecture Notes in Computer Science ; TAPSOFT'95 : Theory and Practice of Software Development, Aarhus, Denmark, May 22-26, 1995*, 915 :364–378, 1995.

[BBD97] Eric Badouel, Luca Bernardinello, and Philippe Darondeau. The synthesis problem for elementary net systems is np-complete. *Theor. Comput. Sci.*, 186(1-2) :107–134, 1997.

[BBF+99] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and Ph. Schnoebelen. *Systems and Software Verification. Model-Checking Techniques and ToolModel Checking*. Springer-Verlag, 1999.

[BD99] E. Badouel and P. Darondeau. Theory of regions. In *Lectures on Petri Nets I : Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, pages 529–586. Springer, 1999.

[BD02] E. Badouel and P. Darondeau. The petri net synthesis problem for automatic graphs. Technical Report 4661, INRIA Rennes, December 2002.

[BD04] Eric Badouel and Philippe Darondeau. The synthesis of petri nets from path-automatic specifications. *Information and Computation*, 193 :117–135, 2004.

[CGP99] E. M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.

[Che65] N.V. Chernikova. Algorithm for finding a general formula for the non-negative solutions of a system of inequalities. *U.S.S.R. Comput. Math. Phys.*, 5 :228–233, 1965.

[Dar04] Philippe Darondeau. Unbounded petri net synthesis. In Jörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 413–438. Springer, 2004.

[Dav64] Morton Davis. Infinite games of perfect information. In *Advances in game theory*, pages 85–101. Princeton Univ. Press, Princeton, N.J., 1964.

[Die95] Volker Diekert. *The Book of Traces*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1995.

[EC82] E. A. Emerson and E. M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2 :241–266, 1982.

[EJ91] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. *Proceedings 32nd Annual IEEE Symp. on Foundations of Computer Science*, pages 368–377, 1991.

[Eme85] E. A. Emerson. Automata, tableaux and temporal logics. In *Proc. Logics of Programs Workshop, Brooklyn, LNCS 193*, pages 79–88. Springer-Verlag, June 1985.

[Eme90] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, vol. B*, chapter 16, pages 995–1072. Elsevier, 1990.

[ER90a] A. Ehrenfeucht and G. Rozenberg. Partial (set) 2-structures. part i : basic notions and the representation problem. *Acta Inf.*, 27(4) :315–342, 1990.

[ER90b] A. Ehrenfeucht and G. Rozenberg. Partial (set) 2-structures. part ii : state spaces of concurrent systems. *Acta Inf.*, 27(4) :343–368, 1990.

[Esp94] J. Esparza. On the decidabilty of model checking for several mu-calculi and petri nets. In S. Tison, editor, *Proceedings of Trees in Algebra and Programming - CAAP '94, 19th International Colloquium 1994*, number 787 in Lecture Notes in Computer Science, pages 115–129, 1994.

[Esp97] J. Esparza. Decidability of model-checking for infinite-state concurrent systems. *Acta Informatica*, 34 :85–107, 1997.

[Esp98] J. Esparza. Decidability and complexity of Petri net problems – an introduction. In G. Rozenberg and W. Reisig, editors, *Lectures on Petri Nets I : Basic Models. Advances in Petri Nets*, number 1491 in Lecture Notes in Computer Science, pages 374–428, 1998.

[HU90] John E. Hopcroft and Jeffrey D. Ullman. *Introduction To Automata Theory, Languages, And Computation*. Addison-Wesley Longman Publishing Co., Inc., 1990.

[JW96] David Janin and Igor Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In Ugo Montanari and Vladimiro Sassone, editors, *CONCUR '96 : Concurrency Theory, 7th International Conference*, volume 1119, pages 263–277, Pisa, Italy, 26–29 1996. Springer-Verlag.

[Koz83] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27 :333–354, 1983.

[May84] Ernst W. Mayr. An algorithm for the general petri net reachability problem. *SIAM J. Comput.*, 13(3) :441–460, 1984.

[Mil90] R. Milner. Operational and algebraic semantics of concurrent processes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, vol. B*, chapter 19, pages 1201–1242. 1990.

[Min67] M. Minsky. *Computation : Finite and Infinite Machines*. Prentice Hall, Englewoods Cliffs, 1967.

[MP92a] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*, volume I : Specification. Springer-Verlag, 1992.

[MP92b] Zohar Manna and Amir Pnueli. *The temporal logic of reactive and concurrent systems*. Springer-Verlag New York, Inc., New York, NY, USA, 1992.

[Mur89] T. Murata. Petri nets : Properties, analysis and applications. *Proceedings of the IEEE*, 77(4) :541–574, April 1989.

[Niw88] Damian Niwinski. Fixed points vs. infinite generation. In *LICS, Proceedings, Third Annual Symposium on Logic in Computer Science, 5-8 July 1988, Edinburgh, Scotland, UK*, pages 402–409. IEEE Computer Society, 1988.

[Par81] D. Park. Concurrency and automata on infinite sequences. In *lncs104*, pages 167–183. Springer-Verlag, March 1981.

[Pet62] Carl Adam Petri. *Kommunikation mit Automaten.* PhD thesis, Bonn : Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962. Second Edition :, New York : Griffiss Air Force Base, Technical Report RADC-TR-65–377, Vol.1, 1966, Pages : Suppl. 1, English translation.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symp. Foundations of Computer Science, Providence*, pages 46–57, November 1977.

[PR90] A. Pneuli and R. Rosner. Distributed reactive systems are hard to synthesize. In IEEE, editor, *Proceedings : 31st Annual Symposium on Foundations of Computer Science : October 22–24, 1990, St. Louis, Missouri*, volume 2, pages 746–757, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1990. IEEE Computer Society Press.

[Rab72] Michael Oser Rabin. *Automata on Infinite Objects and Church's Problem*, volume 13. American Mathematical Society, Boston, MA, USA, 1972.

[Sch86] Alexandeer Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, Inc., New York, NY, USA, 1986.

[SE89] R. S. Streett and E. A. Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Information and Computation*, 81(3) :249–264, 1989.

[SNW96] Vladimiro Sassone, Mogens Nielsen, and Glynn Winskel. Models for concurrency,. *Theoretical Computer Science*, 170(1–2) :297–348, 1996.