ÉCOLE NORMALE SUPÉRIEURE DE CACHAN

THÈSE

présentée
pour obtenir

LE GRADE DE DOCTEUR EN SCIENCES
DE L'ÉCOLE NORMALE SUPÉRIEURE DE CACHAN
Spécialité INFORMATIQUE

PAR

Catherine DUFOURD

Réseaux de Petri avec Reset/Transfert : Décidabilité et Indécidabilité

Soutenue le 23 octobre 1998 devant le jury composé de :

MM.	Paul GASTIN	Président
	Michel LATTEUX	Rapporteur
	Pierre MCKENZIE	Rapporteur
	Alain FINKEL	Directeur de thèse
	Javier ESPARZA	Examinateur
	Philippe SCHNOEBELEN	Examinateur

Laboratoire **S**pécification et **V**érification – URA 2236 CNRS;
ENS de Cachan; 61, Avenue du Président Wilson; 94235 CACHAN Cedex

Remerciements

Je remercie les membres du jury d'avoir dit oui, pour mon plus grand plaisir.

Je remercie Paul GASTIN qui m'a fait l'honneur de présider ce jury.

Michel LATTEUX et Pierre MCKENZIE ont accepté de rapporter ma thèse et je les en remercie fortement. Leurs lectures attentives m'ont permis de corriger, d'amélirorer et d'enrichir grandement le document. Je remercie aussi Pierre pour sa confiance, ses conseils déterminants et ses encouragements tout au long de ces années.

Je remercie Javier ESPARZA pour ses critiques constructives sur l'ensemble du travail.

Je remercie vivement Philippe SCHNOEBELEN pour nos nombreuses conversations, pour sa disponibilité et pour sa vision toujours très originale des problèmes et très juste des solutions.

Je remercie infiniment Alain FINKEL, mon directeur de thèse, pour avoir orchestré cette soutenance et tous les moments en amont. Je suis très reconnaissante à Alain de m'avoir donné la chance d'intégrer son laboratoire et par conséquent de faire une thèse; reconnaissante pour sa disponibilité (il faut lire patience), sa confiance et son attention; pour ses idées claires, directes et magiques; pour ses éclairages et son aiguillage.

I wish to thank Petr JANČAR with whom I really enjoyed to talk about counter-machines !

Mes amis m'ont fait vivre des moments LIFACiens, LSViens et ORSAYens inoubliables. Merci à Râmak, Olivier, Moufida, Béatrice, Gérard, Serge, Zakariæ, François, Sophie, Yan, Grégoire, Emmanuel, Patricia, Nicky, Christophe, Anne-Cécile, Fabrice, Laurent, Rémi et tous les autres qui ont fait office de régulateur de moral. J'ajoute un clin d'œil à Faraz, Tarek et Alexandre. Merci aussi à mon ami Alain TAPP, toujours partant pour discuter, dont le tour s'en vient bientôt.

Et je ne veux oublier personne, alors je remercie tous les chercheurs du LSV, de l'Université de Montréal, de l'Université d'Orsay et d'ailleurs, dont l'enseignement, le travail scientifique ou la passion de la recherche ont orienté mes choix. Et la liste est belle !

Un gros merci à ma famille et en particulier à mes parents, mes deux grand-pères et Jean-Claude, ainsi qu'à Danièle, Joséphine et Gérard qui ont aidé à l'organisation du plus important (puisqu'il implique qu'on est arrivé au bout) : le pot !

Et enfin, je te remercie et je t'embrasse très fort François Lo pour ta patience, ton support logistique, tes lectures, tes re-lectures, tes sur-lectures, tes over-lectures, sans lesquelles j'aurais terminé bien avant ma rédaction, et j'aurais eu bien tort !

À Frédérique, Charlotte, Marianne, Morgane, Rosalie et Alexis

Table des matières

Chapitre 1

Introduction

Cette thèse s'inscrit dans le cadre des méthodes formelles de vérification de systèmes parallèles et distribués.

La sûreté de fonctionnement des logiciels équivaut aujourd'hui à des enjeux énormes, tant au plan humain (sécurité) qu'au plan financier. Un système de contrôle-commande d'une centrale nucléaire, un système de surveillance du reseau électrique d'un pays, un système de commande de vol d'un avion ou encore un système opérant des transactions d'argent électroniques ne sont que quelques exemples de la vie courante qui nous concernent directement.

Les logiciels orchestrant de tels systèmes doivent réagir *correctement* et en particulier face aux situations critiques. Les méthodes formelles de spécification et de validation de systèmes ont pour objectif d'assurer la fiabilité de ces logiciels, c'est-à-dire leur bonne spécification et l'absence d'erreur.

Idéalement, pour concevoir le logiciel d'un système donné, il faudrait spécifier formellement ce système à l'aide d'un modèle mathématique à partir duquel on pourrait raisonner, vérifier les propriétés attendues, puis engendrer automatiquement le code correspondant. En réalité, ce processus se heurte à plusieurs problèmes d'ordres pratique et théorique.

Un premier obstacle apparaît au niveau de la définition du modèle. Si celui-ci est trop expressif, alors on ne peut pas, mathématiquement, l'analyser de façon automatique. Un second obstacle est l'explosion combinatoire des états possibles des systèmes. Malgré les avancées spectaculaires de la technologie des ordinateurs, il arrive que l'on soit incapable d'analyser intégralement des systèmes, même de petite taille, par manque d'espace ou de temps.

Le défi (antagoniste) actuel est de proposer des modèles qui soient suffisamment expressifs pour décrire la réalité des systèmes, tout en restant raisonnablement analysables.

Pour représenter les systèmes dits *réactifs*, plusieurs modèles mathématiques existent et sont à la base de nombreux outils de vérification utilisés dans des grands sites industriels et universitaires. Voici une liste (non exhaustive) de références d'outils avec leur modèle sous-jacent :

- Design-CPN [DC], PEP [GRT+95], PAPETRI [BJP90] sont fondés sur le modèle des *réseaux de Petri* [Pet62];
- SPIN [Spi] est fondé sur le modèle des *automates communicants* [Boc78];
- Uppaal [Upp] est fondé sur le modèle des *automates temporisés* [AD90];
- HyTech [HyT] est fondé sur le modèle des *systèmes hybrides* [Hen96];
- Etc.

Dans cette thèse, nous allons étudier et comparer, en particulier sous l'angle de la *décidabilité* de propriétés comportementales, plusieurs extensions du modèle des réseaux de Petri.

Réseaux de Petri

Les réseaux de Petri [Pet62] constituent un des modèles les plus utilisés du parallélisme et des systèmes distribués (depuis plus de trente-cinq ans). Ils permettent de donner une représentation compacte et finie de systèmes de transitions infinis, tout en restant partiellement analysables.

Les propriétés les plus étudiées sont le *caractère borné du réseau* (l'ensemble des états accessibles est-il fini?), la *couverture* (le réseau peut-il atteindre un état qui contienne un état donné?) ou encore l'*accessibilité* (un état donné est-il accessible par le réseau?). Tous ces problèmes sont décidables bien que très coûteux (les bornes inférieures de complexité sont exponentielles en espace).

Extensions des réseaux de Petri

Malgré ses qualités, le modèle des réseaux de Petri peut faire l'objet d'une critique. Les opérations élémentaires autorisées dans les réseaux de Petri, *retirer* ou *ajouter* des constantes aux *places*, ne sont pas toujours adaptées aux besoins des modélisateurs. Voyons quelques exemples :

- Pour modéliser le *Cambridge Fast Ring Service* [Bil91], Billington utilise des *mises à zéro* ou *Reset* des places.
- Pour modéliser des systèmes à *flux de tâches*, Badouel et Olivier définissent les *réseaux reconfigurables* [BO98] qui comportent des *transferts* de jetons. Dans ce modèle, une place peut être éliminée, sans toutefois que ses jetons soient perdus car ils sont répartis au sein des autres places.
- Dans [Cia94], Ciardo donne une bibliographie de plusieurs exemples de modélisation qui utilisent les *Transferts* (ou qui auraient eu avantage à les utiliser).

Ni les Reset, ni les Transfert ne peuvent être modélisés par les réseaux de Petri. Ces exemples montrent donc l'intérêt et la nécessité d'incorporer au modèle de nouvelles primitives.

On aimerait disposer d'un modèle plus riche, mais en minimisant la perte au niveau de la décidabilité ou, mieux encore, au niveau de la complexité. Ce compromis est délicat, car il existe des exemples d'extensions apparemment simples, qui peuvent simuler les machines de Turing (l'extension la plus connue est sans doute la classe des réseaux de Petri dotés de *deux arcs inhibiteurs* [Hac75]).

VALK [Val78a, Val78b] a proposé deux extensions strictes des réseaux de Petri. L'extension la plus faible, les *Post Self-Modifying nets*, autorise à ajouter aux places des *combinaisons linéraires* du marquage courant du réseau (généralisant les constantes). Pour ce modèle, la couverture et le caractère borné du réseau et d'une place restent décidables, mais l'accessibilité devient indécidable. La seconde extension, les *Strong Self-Modifying nets*, est plus puissante. Elle autorise une transition à *soustraire* en plus d'additionner aux places, des combinaisons linéaires du marquage courant du réseau. Cette fois on obtient un modèle aussi puissant qu'une machine à compteurs et toutes les propriétés non triviales sont indécidables.

En 1977, ARAKI et KASAMI [AK77] ont regardé ce qu'il advient lorsque l'on ajoute la possibilité de mettre à zéro (par des arcs de *Reset*) des places en franchissant les transitions. Pour cette extension, l'accessibilité est indécidable.

L'accessibilité devient en fait très vite indécidable dès que l'on ajoute un peu de puissance au modèle d'origine; par exemple, *deux* arcs de type *Reset*, *Transfert* ou encore *Self-Modifying nets* suffisent [DFS98].

Voici un résumé de la situation telle qu'elle nous apparaissait au début de cette thèse, au mieux de notre connaissance :

	Terminaison	Couverture	Réseau borné	Place bornée	Réseau structurellement borné	Accessibilité	Blocage Vivacité t-Vivacité
Strong Self-Modifying	**I**	**I**	**I**	**I**	**I**	**I**	**I**
Reset	?	?	?	?	?	**I**	?
Transfert	?	?	?	?	?	**I**	?
Post Self-Modifying	**D**	**D**	**D**	**D**	?	**I**	?
Petri	**D**	**D**	**D**	**D**	**D**	**D**	**D**

(I=Indécidable et D=Décidable).

[Ajouté en novembre 1998] En fait, lorsque la thèse a été rapportée, nous avons pris connaissance d'un résultat supplémentaire : ARNOLD et LATTEUX [AL76, AL78] ont prouvé en 1976 que la Couverture est décidable pour le modèle des VASE (*Vector Addition Systems with Erasing*) qui s'apparente en fait aux réseaux de Petri avec Reset.

Motivations

Nous avons abordé l'étude des extensions des réseaux de Petri avec les motivations principales suivantes :

- "Unifier" les résultats concernant les modèles basés sur les réseaux de Petri et comportant des arcs étendus (*Reset*, *Transfert*, *Post Self-Modifying*, ...) et mettre en évidence les points communs de ces extensions.
- Étudier une liste de problèmes sur ces extensions de réseaux de Petri et répondre aux questions ouvertes de décidabilité.
- Mettre en évidence des (grandes) classes de réseaux partiellement analysables qui étendent les réseaux de Petri et donner des complexités sur l'analyse de ces classes.
- Séparer les problèmes en montrant qu'ils ne sont pas décidables pour les mêmes classes de réseaux; ou, au contraire, relier des problèmes en montrant qu'ils sont équivalents (on peut alors ne considérer qu'un problème au lieu de plusieurs).
- Séparer les classes de réseaux en montrant que l'on ne peut pas décider des mêmes problèmes dans des classes distinctes; ou, au contraire, montrer comment deux classes sont équivalentes.
- Affiner les frontières entre ce qui est décidable et ce qui ne l'est pas, et pointer à quel moment un modèle "bascule d'un camp à l'autre" (en donnant par exemple une borne sur le nombre d'arcs étendus suffisant à rendre un problème indécidable).

Résultats

Dans cette thèse, nous définirons les G-réseaux ("G" pour *Généralisés*) qui étendent les *Strong Self-Modifying nets*. Dans ce nouveau modèle, une transition peut ajouter ou retrancher aux places l'évaluation de polynômes (à coefficients entiers positifs) fonction du marquage courant du réseau. Nous appellerons arcs *G-Post*, les arcs étiquetés par des polynômes qui *ajoutent* des jetons aux places (en sortie des transitions).

Comme pour les *Strong Self-Modifying nets*, toutes les propriétés non triviales sont indécidables pour les G-réseaux. Cependant, nous définirons et nous étudierons une variété de sous-classes de G-réseaux, toutes analysables à des degrés divers.

Voici le tableau récapitulatif des résultats de décidabilité à la fin de notre étude (nous avons regroupé les classes pour lesquelles on peut décider des mêmes problèmes) :

	Terminaison	Couverture	Réseau borné	Place bornée	Réseau structurellement borné	Accessibilité Blocage	Vivacité t-Vivacité
G-Réseaux *Strong Self Modifying*	I	I	I	I	I	I	I (6)
G-Post + Reset Reset	D	D (2)	(4) I	I	I (5)	I	I
G-Post + Transfert Transfert	D	D	D (1)	I	I	I	I
G-Post *Post Self-Modifying*	D	D	D	D	D (3)	I	I
Petri	D	D	D	D	D	D	D

Note : un réseau est *structurellement borné* s'il est borné quel que soit le marquage initial.

Commentaires relatifs aux nouveaux résultats

(1) L'algorithme de construction d'un arbre de couverture de Karp et Miller [KM69] s'étend directement ou partiellement à plusieurs classes de réseaux étendus, permettant de donner de nombreux résultats de décidabilité (entourés d'une ligne pointillée numérotée par (1)).

Cette méthode est de type *chaînage avant*, c'est-à-dire que l'on explore toutes les exécutions possibles à partir du marquage initial. L'algorithme s'adapte parfaitement aux *G-Post* qui sont, comme les réseaux de Petri, *fortement structurés*. L'algorithme s'étend partiellement aux G-Post + Transfert, permettant de décider du caractère borné du réseau. L'algorithme s'étend aussi partiellement aux G-Post + Reset, permettant de décider de la terminaison. Par contre, cet algorithme ne permet pas de résoudre la couverture pour les réseaux de Petri avec arcs de Reset ou avec arcs de Transfert (et en fait, il n'existe pas d'algorithme pour construire un arbre de couverture pour les réseaux avec Reset).

(2) La couverture est décidable pour la large classe des réseaux de Petri autorisant les arcs de Reset ET les arcs G-Post (qui inclut les Transfert). Nous obtiendrons ce résultat en deux temps :

- Nous montrerons comment *déplier* les réseaux G-Post + Reset dans la classe des réseaux de Petri avec arcs de Reset, tout en préservant la couverture (ainsi que le caractère borné du réseau et d'une place).
- Nous montrerons que la couverture est décidable pour les réseaux de Petri avec arcs de Reset par l'application d'un algorithme récent de type *chaînage arrière* proposé dans [AČJY96] et dans [FS97]. Pour décider si un marquage m est couvert, on remonte dans toutes les exécutions possibles permettant d'accéder à m. On calcule ainsi l'ensemble de tous les prédécesseurs possibles de tout marquage couvrant m et il suffit ensuite de vérifier si le marquage initial fait partie de cet ensemble.

  [Ajouté en novembre 1998] Cette stratégie est la même que celle apparaissant en [AL76, AL78].

(3) Le caractère structurellement borné est décidable pour les réseaux de Petri avec arcs G-Post (qui incluent les *Post Self-Modifying nets*). Nous ramènerons ce problème à un problème d'accessibilité d'un réseau de Petri (qui est décidable). Pour cela, nous utiliserons le dépliage des G-Post vers les réseaux de Petri, que nous venons d'évoquer.

(4) Le caractère borné du réseau est indécidable pour les réseaux de Petri avec *trois* arcs de Reset. Ce résultat est surprenant et va à l'encontre de l'intuition; il a été difficile à obtenir et contredit un énoncé de décidabilité dans [KCK+97]. La preuve consiste en une réduction du problème de caractère borné d'une machine à compteurs. Ce résultat implique directement l'indécidabilité du caractère borné d'une place pour les réseaux avec arcs de Reset ou avec arcs de Transfert.

(5) Le caractère structurellement borné est indécidable pour les réseaux avec arcs de Reset ou avec arcs de Transfert. Ce résultat provient d'une adaptation de la preuve précédente.

(6) La vivacité et le blocage sont indécidables dans les classes pour lesquelles l'accessibilité est indécidable, car l'accessibilité se réduit (polynomialement) à ces deux propriétés.

Séparation des problèmes de caractère borné

Bien que cela n'apparaisse pas directement à la lecture du tableau, nous avons séparé les trois problèmes de type caractère borné (réseau, place et structurellement borné).

- Pour les réseaux de Petri avec arcs de Transfert, le *caractère borné du réseau* est décidable, alors que le *caractère borné d'une place* est indécidable. Ce résultat est intéressant, car ces deux propriétés sont souvent traitées comme une seule dans la littérature.

- Pour les réseaux de Petri avec Reset, le *caractère borné du réseau* est semi-décidable, alors que *caractère structurellement borné* n'est ni dans Σ_1, ni dans Π_1. En effet, nous montrerons que pour cette classe, le problème de caractère borné d'une machine à compteurs, ainsi que le problème de NON-terminaison, se réduisent tous les deux au problème du caractère structurellement borné. Rappelons que pour les réseaux de Petri, c'est l'inverse, car le caractère structurellement borné se résout en temps polynomial [MR80], alors que le caractère borné du réseau nécessite un espace exponentiel [Lip76].

- Pour les réseaux de Petri avec arcs de Reset, le *caractère borné d'une place* est semi-décidable et le *caractère structurellement borné* est donc strictement plus difficile.

Résultats de complexité

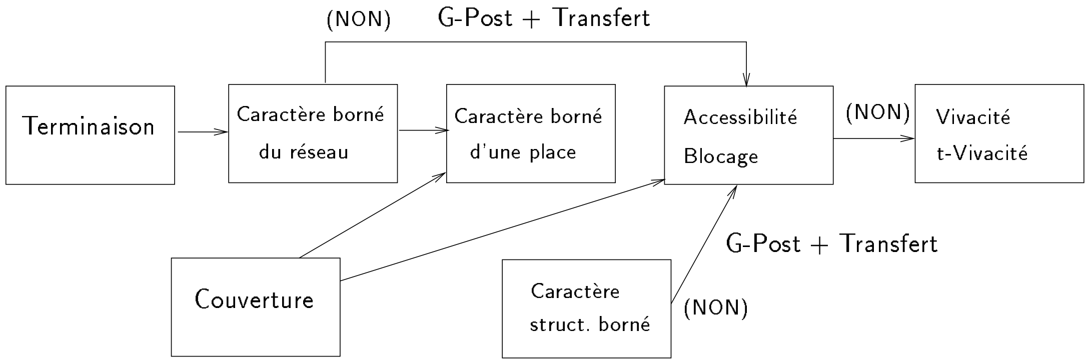
- Nous obtiendrons une borne de complexité pour les *Post Self-Modifying nets*, car pour cette classe, le dépliage vers un réseau de Petri est polynomial en temps. Connaissant la borne de complexité pour les réseaux de Petri [Rac78], nous en conclurons que pour les *Post Self-Modifying nets*, le problème du caractère borné du réseau et de la couverture se résolvent en espace exponentiel.

- Nous donnerons une normalisation des réseaux de Petri avec arcs de Reset ou avec arcs de Transfert, quadratique en temps, qui préserve la couverture, le caractère borné du réseau, d'une place, le caractère structurellement borné, l'accessibilité et la t-vivacité [DF98]. Ce résultat améliore les bornes des normalisations précédentes qui étaient exponentielles [Sif79, Pet81, KM82, Sch85, Pel92].

Relations entre les problèmes

Pour décider du caractère structurellement borné des G-Post, nous réduirons ce problème au problème d'accessibilité pour les réseaux de Petri. L'accessibilité est en fait une propriété très riche.

Au cours de notre étude, nous montrerons d'autres réductions dont voici un résumé :



Une flèche partant d'un problème A vers un problème B indique que A se réduit à B, de façon multivoque polynomiale (et l'apparition d'un (NON) indique que l'on considère le complémentaire du problème). Toutes les réductions impliquent des transformations qui préservent la classe du réseau de départ. Les flèches non étiquetées indiquent que les réductions sont valables pour les réseaux de Petri avec arcs de Reset + arcs G-Post.

Plan de la thèse

Au *chapitre 2*, nous définirons le modèle des G-réseaux et une hiérarchie de sous-classes de G-réseaux. Nous définirons les problèmes que nous allons étudier sur chacune de ces sous-classes. Nous rappellerons les définitions des *systèmes de transitions bien structurés* et nous déterminerons une caractérisation des sous-classes de G-réseaux. Nous terminerons ce chapitre avec un état de l'art sur les réseaux de Petri avec arcs étendus.

Au *chapitre 3*, nous donnerons deux constructions qui simplifient les représentations de réseaux. Tout d'abord, nous montrerons un *dépliage* des G-Post (+ Reset) vers les réseaux de Petri (+ Reset); nous en déduirons une borne de complexité pour résoudre le caractère borné des *Post Self-Modifying nets*. Ensuite, nous montrerons une *normalisation* polynomiale (en temps) des réseaux de Petri avec arcs de Reset ou avec arcs de Transfert.

Au *chapitre 4*, nous rappellerons les méthodes habituellement utilisées pour analyser les réseaux de Petri et nous les étendrons, lorsque cela sera possible, aux sous-classes de G-réseaux. Parallèlement, nous comparerons et ordonnerons les problèmes à l'aide de réductions polynomiales :

– À la *section 4.1*, nous rappellerons la définition des réductions multivoques polynomiales et à la section 4.2, nous montrerons quelles propriétés se réduisent au *caractère borné d'une place*.

– À la *section 4.3*, nous rappellerons le principe général de l'algorithme qui construit un arbre de couverture pour un réseau de Petri [KM69]. Nous montrerons comment une construction, parfois partielle, d'un arbre de couverture permet de décider de la *terminaison* pour les G-Post + Reset, du *caractère borné du réseau* pour les G-Post + Transfert ou encore du *caractère borné d'une place* pour les G-Post. Par contre, nous verrons que l'on ne peut pas étendre cet algorithme pour décider de la *couverture* pour les G-Post + Reset.

– À la *section 4.4*, nous montrerons que la *couverture* est décidable pour les G-Post + Reset en appliquant un algorithme récent [AČJY96, FS97] de type *chaînage arrière* (et en utilisant le dépliage du chapitre 3).

– À la *section 4.5*, nous montrerons que de nombreuses propriétés peuvent se ramener à l'accessibilité. Nous montrerons, par exemple, que le problème de *caractère structurellement borné* d'un G-Post est décidable, car il se ramène à un problème d'accessibilité dans les réseaux de Petri.

– À la *section 4.6*, nous montrerons que l'accessibilité se réduit à la *vivacité* et à la *t-vivacité* pour toutes les sous-classes de G-réseaux. Nous en conclurons que ces deux problèmes sont indécidables pour toutes les classes de G-réseaux qui étendent les réseaux de Petri.

Au *chapitre 5*, nous aborderons les trois problèmes de *caractère borné du réseau*, d'une *place* et *structurellement borné*, pour les classes avec arcs de Reset.

– À la *section 5.1*, nous montrerons un contre-exemple à l'énoncé de décidabilité du caractère borné pour les réseaux de Petri avec arcs de Reset dans [KCK+97].

– À la *section 5.2*, nous donnerons l'idée générale de notre première preuve d'indécidabilité du *caractère borné pour les réseaux avec Reset*. Il s'agit d'une réduction du *dixième Problème de Hilbert*.

– À la *section 5.3*, nous donnerons une seconde preuve, détaillée cette fois, du même résultat. Cette preuve consiste en une réduction du problème de *caractère borné d'une machine à compteurs* au problème de caractère borné d'un réseau avec Reset.

– À la *section 5.4*, nous montrerons comment adapter la construction de la section 5.3 pour prouver que le *caractère structurellement borné* d'un réseau avec Reset est indécidable. De plus, nous montrerons comment réduire le problème de *NON-terminaison d'une machine à compteurs* au problème de caractère structurellement borné d'un réseau avec Reset.

– À la *section 5.5*, nous ferons la synthèse de ce chapitre et nous commenterons la séparation des trois problèmes de type caractère borné pour les réseaux avec arcs de Reset.

Nous conclurons au *chapitre 6* avec un schéma récapitulatif des propriétés décidables sur la hiérarchie des G-réseaux. Nous terminerons cette thèse sur quelques questions qui restent encore ouvertes et certaines perspectives découlant de notre travail.

Chapitre 2

Réseaux de Petri et G-réseaux

Dans ce chapitre, nous définissons les G-réseaux ("G" pour généralisés) qui constituent une extension des réseaux de Petri. Plus précisément, les G-réseaux étendent les *Self-Modifying nets* de Valk [Val78b], qui constituent eux-mêmes une extension des réseaux de Petri. Les *Self-Modifying nets*, et par conséquent les G-réseaux, sont suffisamment puissants pour simuler les machines à compteurs, ce qui rend indécidables toutes les propriétés non triviales. Cependant, au cours de notre travail, nous verrons que plusieurs sous-classes de G-réseaux, plus puissantes que les réseaux de Petri, sont (en partie) analysables. Nous définissons une hiérarchie de onze sous-classes de G-réseaux (incluant les réseaux de Petri et les *Self-Modifying nets*) dont neuf sont plus puissantes que les réseaux de Petri.

En section 2.1 nous commencerons par définir le modèle des G-réseaux dont nous donnerons un exemple; nous définirons ensuite les propriétés que nous allons étudier. En section 2.2, nous définirons la hiérarchie des sous-classes de G-réseaux. En section 2.3, nous rappellerons la définition des *systèmes de transitions bien structurés* à partir desquels nous caractériserons les sous-classes de G-réseaux selon trois catégories satisfaisant de "bons" critères d'analyse. En section 2.4, nous définirons la taille d'un G-réseau. Enfin, en section 2.5, nous donnerons un état de l'art des réseaux de Petri étendus. Nous terminerons ce chapitre avec le tableau récapitulatif des résultats de décidabilité connus pour ces réseaux étendus.

Notations générales

$\mathbb{N}$ désigne l'ensemble des entiers naturels et $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$. Pour tout $k \in \mathbb{N}$, l'ensemble $\mathbb{N}^k$ désigne l'ensemble des vecteurs de dimension k à composantes dans $\mathbb{N}$. Pour tout vecteur $v \in \mathbb{N}^k$ et pour tout entier i tel que $1 \leq i \leq k$, $v(i)$ désigne la valeur de la $i^{\text{ème}}$ composante de v. Ainsi, v s'écrit $(v(1), v(2), .., v(k))$.

Soient $k \in \mathbb{N}$ et $v_1, v_2 \in \mathbb{N}^k$:

- $\leq$ est la relation habituelle sur les vecteurs d'entiers :
  $v_1 \leq v_2$ si $\forall i, 1 \leq i \leq k, v_1(i) \leq v_2(i)$.
- $v_1 < v_2$ si $v_1 \leq v_2$ et $\exists i, 1 \leq i \leq k, v_1(i) < v_2(i)$.
- $v_1 = v_2$ si $\forall i, 1 \leq i \leq k, v_1(i) = v_2(i)$.

Soit $P = \{p_1, .., p_k\}$ un ensemble de k variables. $\mathbb{N}[P]$ ou $\mathbb{N}[p_1, .., p_k]$ désigne l'ensemble des polynômes à coefficients dans $\mathbb{N}$.

Soit Σ un alphabet fini, $\sigma \in \Sigma^*$ et $n \in \mathbb{N}$. σ^n représente le mot obtenu en concaténant n fois le mot σ. On note le mot vide par λ.

On note $|objet|$ la taille d'un $objet$. L'objet peut être un ensemble fini, un réseau ou une séquence de transitions et $|objet|$ correspond au cardinal d'un ensemble, à la taille d'un réseau ou à la longueur d'une séquence.

2.1 Définition des G-réseaux

Définition 1
Un réseau généralisé N, appelé G-réseau, est un quadruplet $N =< P, T, F, m_0 >$ où :

- *P est un ensemble fini de places*
- *T, $T \cap P = \emptyset$, est un ensemble fini de transitions*
- *$F : (P \times T) \cup (T \times P) \to \mathbb{N}[P]$ est une fonction de flot*
- *$m_0 \in \mathbb{N}^{|P|}$ est le marquage initial.*

La fonction de flot a valeur dans l'ensemble des polynômes à coefficients naturels dont les variables sont les places.

Notation : *pour $F(x, y) \in \mathbb{N}$, nous utilisons directement $F(x, y)$ comme un entier.*

Pour les *Self-Modifying nets* de VALK [Val78b], les polynômes sont de degré 1, c'est-à-dire des combinaisons linéaires.

Définition 2 *Un Self-Modifying net est un G-réseau $N =< P, T, F, m_0 >$ tel que :*

$$F : (P \times T) \cup (T \times P) \to c + \Sigma_{i=1}^{|P|} \lambda_i p_i \text{ avec } c,\ \lambda_i \in \mathbb{N} \text{ et } p_i \in P$$

Pour les réseaux de Petri [Pet62], les polynômes sont de degré 0, c'est-à-dire des constantes.

Définition 3 *Un réseau de Petri est un G-réseau $N =< P, T, F, m_0 >$ tel que :*

$$F : (P \times T) \cup (T \times P) \to \mathbb{N}$$

Un marquage $m \in \mathbb{N}^k$ associe une valeur à chaque place d'un réseau à k places. Soit p la place associée à la $i^{\text{ème}}$ composante, on note $m(p)$ ou $m(i)$ la valeur de p par le marquage m. Un G-réseau peut passer d'un marquage à un autre en franchissant une transition, sous certaines conditions.

Définition 4 *Soit un G-réseau $N =< P, T, F, m_0 >$ à k places.*

Une transition $t \in T$ est franchissable à partir d'un marquage $m \in \mathbb{N}^k$, noté $m \xrightarrow{t}$, si :

$$\forall\, p \in P,\ m(p) \geq Q(m) \ \textit{avec}\ F(p,t) = Q$$

où $m = (m(p_1), m(p_2) \,..\, m(p_k))$

Intuitivement, une transition est franchissable à condition que les places contiennent "suffisamment de ressources" en fonction du marquage courant.

Définition 5 *Soit un G-réseau $N =< P, T, F, m_0 >$ et $t \in T$.*

Franchir t à partir de m mène à un marquage m', noté $m \xrightarrow{t} m'$, défini comme suit :

$$\forall\, p \in P,\ m'(p) = m(p) - Q(m) + R(m) \ \textit{avec}\ F(p,t) = Q \ \textit{et}\ F(t,p) = R$$

La définition 5 est naturellement étendue aux séquences de transitions.

Définition 6 *Soit un G-réseau $N =< P, T, F, m_0 >$ et $m \in \mathbb{N}^{|P|}$.*

- *Un marquage m' est accessible à partir de m, noté $m \xrightarrow{*} m'$, si :*

$$\exists\, \sigma \in T^* \ \textit{telle que}\ m \xrightarrow{\sigma} m'$$

- *L'ensemble des marquages accessibles par N à partir de m est noté $Acc(N, m)$:*

$$Acc(N, m) = \{m' \in \mathbb{N}^{|P|} \mid m \xrightarrow{*} m'\}$$

- *L'ensemble d'accessibilité de N est $Acc(N, m_0)$, que nous notons de façon abrégée par $Acc(N)$.*

Pour décrire un comportement du réseau à partir du marquage initial, nous parlerons d'*exécution*.

Définition 7 *Soit un G-réseau $N =< P, T, F, m_0 >$.*

Une exécution de N est une suite :

$$m_0 \xrightarrow{t_{i_1}} m_1 \xrightarrow{t_{i_2}} m_2 \,..\, m_k \xrightarrow{t_{i_k}} m_{k+1} \,..$$

avec $t_{i_j} \in T$ et $m_i \in \mathbb{N}^{|P|}$.

Nous ferons souvent référence aux places directement impliquées lors du franchissement d'une transition donnée, ou aux transitions dont les franchissements impliquent directement une place donnée. Nous utiliserons la notation suivante :

Définition 8 *Soit un G-réseau $N =< P, T, F, m_0 >$ et $x \in (P \cup T)$.*

- $^\bullet x = \{y \mid F(y, x) = Q, Q \ \textit{non nul}\ \}$.
- $x^\bullet = \{y \mid F(x, y) = Q, Q \ \textit{non nul}\ \}$.

2.1.1 G-réseau calculant la suite de Fibonacci

Voyons un exemple de G-réseau. Le G-réseau $N_{fib} =< P_{fib}, T_{fib}, F_{fib}, m_{fib} >$ qui calcule la suite de Fibonacci est illustré à la figure 2.1. Nous profitons de cet exemple pour introduire le vocabulaire associé aux G-réseaux.

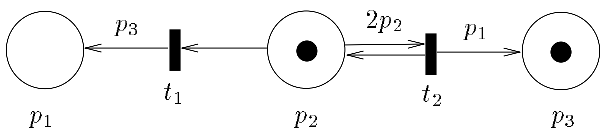


FIG. 2.1 – *G-réseau calculant les nombres de Fibonacci*

Les places sont représentées par des ronds et les transitions par des rectangles. La fonction de flot F est représentée par des arcs orientés, étiquetés par des polynômes. $F(s,d) = Q$ est représenté par l'arc orienté partant de la source $s \in (P \cup T)$ vers la destination $d \in (P \cup T)$ et étiqueté par le polynôme Q. Sur le dessin, par défaut les arcs non étiquetés le sont par la constante 1 et les arcs qui n'apparaissent pas sont étiquetés par 0.

Remarque : *étant donné un marquage m et une place p, pour parler du marquage $m(p) = n$ de p, nous dirons "p est marquée par n" ou "p contient n jetons".*

Ici, les places p_2 et p_3 contiennent toutes deux 1 jeton et la place p_1 est vide. Nous avons :

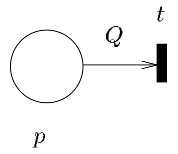
- $P_{fib} = \{p_1, p_2, p_3\}$
- $T_{fib} = \{t_1, t_2\}$
- $F_{fib}(t_1, p_1) = Q_1$ avec $Q_1(p_1, p_2, p_3) = p_3$
  $F_{fib}(p_2, t_1) = Q_2$ avec $Q_2(p_1, p_2, p_3) = 1$
  $F_{fib}(p_2, t_2) = Q_3$ avec $Q_3(p_1, p_2, p_3) = 2p_2$
  $F_{fib}(t_2, p_2) = Q_2$
  $F_{fib}(t_2, p_3) = Q_4$ avec $Q_4(p_1, p_2, p_3) = p_1$
  $F_{fib}(p_1, t_1) = F_{fib}(t_1, p_2) = F_{fib}(t_1, p_3) = F_{fib}(p_3, t_1) = F_{fib}(p_1, t_2) = F_{fib}(t_2, p_1) = F_{fib}(p_3, t_2) = Q_5$ avec $Q_5(p_1, p_2, p_3) = 0$
- $m_{fib} = (0, 1, 1)$

Remarquons que ce réseau est aussi un *Self-Modifying net*, puisque tous les polynômes sont de degré 1.

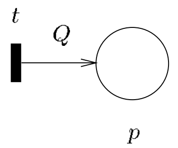
Notation : *nous notons $(s; Q; d)$ l'arc partant de s allant vers d et étiqueté par Q. Lorsque cela n'entraînera pas d'ambiguïté, nous écrirons directement l'étiquette associée au polynôme Q, au lieu de son nom. Par exemple, pour désigner l'arc partant de t_1 allant vers p_1, nous utiliserons $(t_1; Q_1; p_1)$ ou $(t_1; p_3; p_1)$.*

Définition 9

- *Un arc partant de la place p et allant vers la transition t (un arc entrant de t) de la forme $(p; Q; t)$ est un pré-arc.*



- *Un arc partant de la transition t et allant vers la place p (un arc entrant de p) de la forme $(t; Q; p)$ est un post-arc.*



Décrivons le comportement de N_{fib} dont le marquage initial est $m_{fib} = (0, 1, 1)$.

À partir de $(0, 1, 1)$ la seule transition franchissable est t_1. La transition t_1 nécessite de retirer un jeton de p_2 et a pour effet d'ajouter à p_1 le contenu courant de la place p_3, tel qu'indiqué par l'arc $(t_1; p_3; p_1)$. La transition t_2 n'est pas franchissable à partir de $(0, 1, 1)$, car elle est *inhibée* par la garde sur l'arc $(p_2; 2p_2; t_2)$. Cet arc indique que t_2 nécessite que l'on retire 2 fois le contenu courant de p_2, de la place p_2 elle-même. Ceci n'est possible que lorsque p_2 est vide, ce qui n'est pas le cas pour le marquage $(0, 1, 1)$. Un tel arc est appelé *test-à-zéro* ou encore *arc inhibiteur.*

Nous avons $m_{fib} = (0, 1, 1) \xrightarrow{t_1} (1, 0, 1)$. À partir de $(1, 0, 1)$, t_1 n'est plus franchissable mais t_2 le devient. Franchir t_2 remet un jeton dans p_2 et ajoute à p_3 le contenu courant de p_1, menant au marquage $(1, 1, 2)$ à partir duquel t_1 est à nouveau franchissable.

À partir du marquage initial $(0, 1, 1)$, il n'y a qu'une exécution possible dans N_{fib} :

$$m_{fib} = (0, 1, 1) \xrightarrow{t_1} (1, 0, 1) \xrightarrow{t_2} (1, 1, 2) \xrightarrow{t_1} (3, 0, 2) \xrightarrow{t_2} (3, 1, 5) \xrightarrow{t_1} (8, 0, 5) \cdots$$

Cette exécution est infinie, le réseau ne *termine* donc pas. L'*ensemble d'accessibilité* de N_{fib} est le suivant :

$$Acc(N_{fib}) = \{(Fib_k, 1, Fib_{k+1}) \mid k \in 2\mathbb{N}\} \cup \{(Fib_{k+1}, 0, Fib_k) \mid k \in 2\mathbb{N} + 1\}$$

où $Fib_0 = 0$, $Fib_1 = 1$ et $Fib_i = Fib_{i-2} + Fib_{i-1}$ $(i \geq 2)$. Ce réseau n'est pas *borné* puisqu'il engendre une infinité de marquages différents. Les places p_1 et p_3 ne sont pas bornées puisqu'elles peuvent contenir un nombre arbitrairement grand de jetons. Par contre, la place p_2 est bornée : elle oscille entre les valeurs 0 et 1 et l'on dira qu'elle est *1-bornée.*

Remarque : *nous utiliserons le terme "arc étendu" pour parler des arcs étiquetés par des polynômes.*

2.1.2 Propriétés des G-réseaux

Voici neuf problèmes (ou propriétés) tous décidables pour les réseaux de Petri, que nous allons étudier pour les G-réseaux et pour différentes sous-classes de G-réseaux.

• Problème de la terminaison

Entrée : $< N >$ un G-réseau

Sortie : $\begin{cases} \textit{oui} & \text{s'il existe une exécution infinie dans } N \\ \textit{non} & \text{sinon} \end{cases}$

Exemple : N_{fib} ne termine pas car il contient une exécution infinie.

• Problème de la couverture

Entrée : $< N, m >$, un G-réseau N et un marquage $m \in \mathbb{N}^{|P|}$

Sortie : $\begin{cases} \textit{oui} & \text{si } \exists\, m' \in Acc(N) \text{ tel que } m \leq m' \\ \textit{non} & \text{sinon} \end{cases}$

Exemple : le marquage $(77, 0, 77)$ est couvert dans N_{fib}, car $(144, 0, 89) \in Acc(N_{fib})$ et $(77, 0, 77) \leq (144, 0, 89)$. Par contre, $(77, 3, 77)$ n'est pas couvert dans N_{fib}, car la place p_2 est bornée par 1.

• Problème du caractère borné d'un réseau

Entrée : $< N >$ un G-réseau.

Sortie : $\begin{cases} \textit{oui} & \text{si } Acc(N) \text{ est fini} \\ \textit{non} & \text{sinon} \end{cases}$

Exemple : N_{fib} n'est pas borné.

Remarque : un réseau qui termine est forcément borné. Par contre, un réseau borné peut ne pas terminer.

• Problème du caractère borné d'une place

Entrée : $< N, p >$, un G-réseau N et une place p.

Sortie : $\begin{cases} \textit{oui} & \text{si } \exists\, k \in \mathbb{N},\ \forall\, m \in Acc(N),\ m(p) \leq k \\ \textit{non} & \text{sinon} \end{cases}$

Exemple : dans N_{fib}, p_1 et p_3 ne sont pas bornées. Par contre, p_2 est bornée, car elle peut contenir au maximum 1 jeton.

• **Problème du caractère structurellement borné**

Entrée: $< N >$ un G-réseau.

Sortie: $\begin{cases} \textit{oui} & \text{si } \forall\, m \in \mathbb{N}^{|P|},\ Acc(N, m) \text{ est fini} \\ \textit{non} & \text{sinon} \end{cases}$

Exemple: N_{fib} n'est pas structurellement borné, puisque à partir de $(0, 1, 1)$, N_{fib} engendre une infinité de marquages différents.

• **Problème de l'accessibilité**

Entrée: $< N, m >$, un G-réseau N et un marquage $m \in \mathbb{N}^{|P|}$.

Sortie: $\begin{cases} \textit{oui} & \text{si } m \in Acc(N) \\ \textit{non} & \text{sinon} \end{cases}$

Exemple: dans N_{fib}, $(8, 1, 13)$ est accessible, mais $(8, 1, 15)$ ne l'est pas (15 n'est pas un nombre de Fibonacci).

Définition 10 *Soit un G-réseau* $N =< P, T, F, m_0 >$ *et* $m \in \mathbb{N}^{|P|}$.
Le marquage m *est mort pour* N *s'il n'existe pas de transition* $t \in T$ *telle que* $m \xrightarrow{t}$.

• **Problème du blocage**

Entrée: $< N >$ un G-réseau.

Sortie: $\begin{cases} \textit{oui} & \text{si } \exists\, m \in Acc(N), \text{ tel que } m \text{ est mort pour } N \\ \textit{non} & \text{sinon} \end{cases}$

Exemple: N_{fib} ne comporte aucun blocage puisqu'à tout moment, soit t_1, soit t_2 est franchissable.

• **Problème de la vivacité d'une transition**

Entrée: $< N, t >$, un G-réseau N et une transition t.

Sortie: $\begin{cases} \textit{oui} & \text{si } \forall\, m \in Acc(N), \\ & \exists\, m' \in \mathbb{N}^{|P|}, \text{ tel que } m \xrightarrow{*} m' \text{ et } m' \xrightarrow{t} \\ \textit{non} & \text{sinon} \end{cases}$

Exemple: dans le réseau de Fibonacci, t_1 et t_2 sont vivantes. En effet, pour tout marquage $m \in Acc(N_{fib})$, soit $m \xrightarrow{t_1 t_2}$, soit $m \xrightarrow{t_2 t_1}$.

• **Problème de la vivacité d'un réseau**

Entrée: $< N >$ un G-réseau.

Sortie: $\begin{cases} \textit{oui} & \text{si } \forall\, t \in T,\ t \text{ est vivante} \\ \textit{non} & \text{sinon} \end{cases}$

Exemple: N_{fib} est vivant.

2.2 Présentation des sous-classes de G-réseaux

Nous allons définir une hiérarchie des G-réseaux. Nous commençons d'abord par donner les motivations qui nous ont amenés à identifier les classes de réseaux de cette hiérarchie.

Les *Self-Modifying nets* de Valk [Val78a, Val78b], une sous-classe des G-réseaux, constituent un modèle de calcul très puissant. Les pré-arcs des *Self-Modifying nets* peuvent implémenter les arcs inhibiteurs (comme sur l'exemple de Fibonacci) qui amènent à la simulation des machines de Turing [Hac75]. Toutes les propriétés non triviales sont donc indécidables pour cette sous-classe.

Remarque : *il y a une seconde façon d'implémenter un arc inhibiteur avec les Self-Modifying nets : il suffit de conserver une place* p_{test} *vide; une transition* t *peut alors tester une place* p *à zéro à l'aide du pré-arc* $(p_{test}; p; t)$*, qui autorise le franchissement de* t *uniquement dans le cas où* p *est vide.*

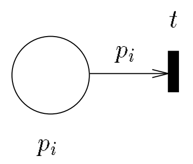
Les *Post Self-Modifying nets* [Val78b] sont une sous-classe des *Self-Modifying nets* pour lesquels seuls les post-arcs sont étendus, alors que les pré-arcs sont de type réseau de Petri. Pour les *Post Self-Modifying nets*, au contraire de la classe générale des *Self-Modifying nets*, de nombreuses propriétés sont décidables. L'indécidabilité des *Self-Modifying nets* réside donc plus dans les pré-arcs, c'est-à-dire dans les *gardes* des transitions, que dans les post-arcs.

Notre motivation est de trouver des extensions analysables de réseaux de Petri. Nous axons notre étude sur deux types d'arcs étendus : les post-arcs (qui sont robustes en terme de décidabilité) et les *arcs Reset.*

Définition 11

Étant donné un G-réseau à k *places, un arc Reset, ou arc de mise à zéro, est un pré-arc* $(p_i; Q; t)$ *où* $Q(p_1..p_k) = p_i$.

Mettre une place p_i à zéro consiste à lui retrancher son propre contenu, ce qui a pour conséquence de la vider :

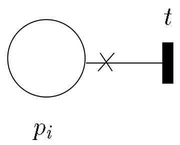


Les arcs de Reset sont des pré-arcs "faibles". Nous donnons trois arguments dans ce sens :

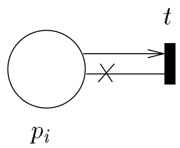
- bien que pré-arc, une mise à zéro est toujours réalisable et n'est donc jamais un obstacle au franchissement d'une transition;
- une mise à zéro est une action locale qui ne concerne qu'une seule place, sans besoin de consulter une autre place;
- une mise à zéro est implémentée à l'aide du polynôme identité.

Remarquons que l'action de l'arc Reset est très différente de celle du test-à-zéro (arc inhibiteur), même s'il est vrai qu'après le Reset, une place est à zéro. Le Reset est toujours autorisé. Par contre, le test-à-zéro constitue une garde très forte. Remarquons encore que le Reset est simulable avec le test-à-zéro, mais que le contraire n'est pas vrai. Pour simuler l'action du Reset avec le test-à-zéro, il suffit de retrancher des jetons aussi longtemps que la place n'est pas testée vide.

À des fins pratiques, nous adopterons une description graphique abrégée des arcs Reset. Un arc Reset est représenté par un arc non orienté muni d'une croix :

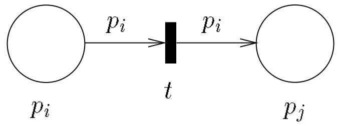


Remarque : *dans les G-réseaux, on ne peut pas décrire directement une implémentation du style :*

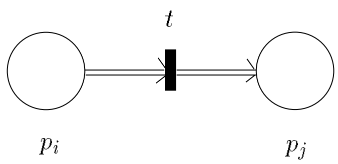


qui autoriserait de vider la place uniquement dans le cas ou la place n'est pas vide. On ne peut pas exprimer cet effet avec un seul polynôme à coefficients entiers.

Notre étude porte principalement sur les sous-classes des G-réseaux issues des combinaisons des arcs de Petri, des arcs G-Post et des arcs Reset. Une combinaison particulière est le *Transfert*. On peut le voir comme une variante du Reset avec conservation des jetons. Un Transfert déplace en un coup le contenu d'une place vers une autre. Cette action est implémentée par le jumelage d'un pré-arc et d'un post-arc. Pour transférer le contenu de p_i dans p_j, la transition t vide p_i par un arc $(p_i; p_i; t)$, et ajoute p_i à p_j par un arc $(t; p_i; p_j)$:



Nous considérerons le Reset d'une place et son ajout dans une autre place en un tout, comme une seule action. À des fins de simplicité et dans les cas où cela ne créera pas d'ambiguïté, nous représenterons graphiquement cette action avec un *arc de Transfert*. Nous dessinons l'arc de Transfert à l'aide de deux flèches doubles. Par exemple, ici le Transfert se fait de p_i vers p_j :



2.2.1 Hiérarchie des G-réseaux

Nous définissons les classes qui constituent la hiérarchie des G-réseaux.

- Un *Self-Modifying net* de Valk est un G-réseau tel que la fonction de flot n'utilise que des polynômes de degré inférieur ou égal à 1.
- Un *réseau de Petri* est un G-réseau tel que la fonction de flot n'utilise que des polynômes de degré 0.
- Un *réseau G-Post* est un G-réseau dont les pré-arcs sont de type réseau de Petri.
- Un *Post Self-Modifying* de Valk est un *Self-Modifying net* dont les pré-arcs sont de type réseau de Petri.
- Un *réseau G-Post + Reset* est un G-réseau dont les pré-arcs sont soit de type arc de Reset (c'est-à-dire $(p_i; p_i; t)$), soit de type réseau de Petri.
- Un *réseau de Petri + Reset* est un G-réseau dont les pré-arcs sont soit de type arc de Reset, soit de type réseau de Petri, et dont les post-arcs sont de type réseau de Petri.
- Un *réseau G-Post + Transfert* est un réseau G-Post + Reset tel que pour chaque pré-arc $(p_i; p_i; t)$ il existe un post-arc $(t; Q; p_j)$ tel que :

  $$Q = \lambda {p_i}^n + Q' \text{ avec } \lambda, n \in \mathbb{N}^+ \text{ et } Q' \in \mathbb{N}[P]$$

  (où P est l'ensemble des places du réseau). Ainsi, les jetons retirés de p_i sont ajoutés dans une autre place et donc restent dans le réseau.
- Un *réseau de Petri + Transfert* est un G-Post + Transfert tel que tout post-arc est soit de type réseau de Petri, soit de la forme $(t; Q; p_j)$ tel que :

  $$Q = \Sigma_{p_i \in P} \lambda_i p_i \text{ où } \lambda_i \in \{0, 1\}$$

  (où P est l'ensemble des places du réseau) et tel que :

  - pour chaque pré-arc $(p_i; p_i; t)$, il existe un unique post-arc $(t; Q; p_j)$ tel que $Q = Q' + p_i$.
  - pour chaque post-arc $(t; Q; p_j)$ tel que $Q = Q' + p_i$, il existe un pré-arc $(p_i; p_i; t)$.
- Un *réseau de Petri + Double* est un *Post Self-Modifying net* dont les post-arcs sont soit de type $(t; p_i; p_i)$, dit *arc Double*, soit de type réseau de Petri.
- Un *réseau de Petri normalisé* est un réseau de Petri dont les arcs sont étiquetés par 0 ou 1, et dont le marquage initial est un vecteur dans $\{0, 1\}^{|P|}$, où P est l'ensemble des places du réseau.

Nous mettons explicitement la classe comprenant les arcs Doubles dans notre hiérarchie car ces arcs constituent la plus "faible" extension de réseaux de Petri de type *Post Self-Modifying*. Tout comme l'arc de Reset, l'arc Double est implémenté à l'aide du polynôme identité et opère une action locale. Par contre, l'arc Double est un post-arc. Pour la classe des réseaux de Petri avec *deux* arcs Double, l'accessibilité est indécidable [DFS98].

La hiérarchie des G-réseaux est représentée à la figure 2.2. Une flèche d'une classe vers une autre indique l'inclusion de la première classe dans la seconde.

Remarque : *pour désigner les réseaux de Petri avec Reset ou réseaux de Petri avec Transfert, nous utiliserons les noms abrégés "réseaux Reset" ou "réseaux Transfert".*

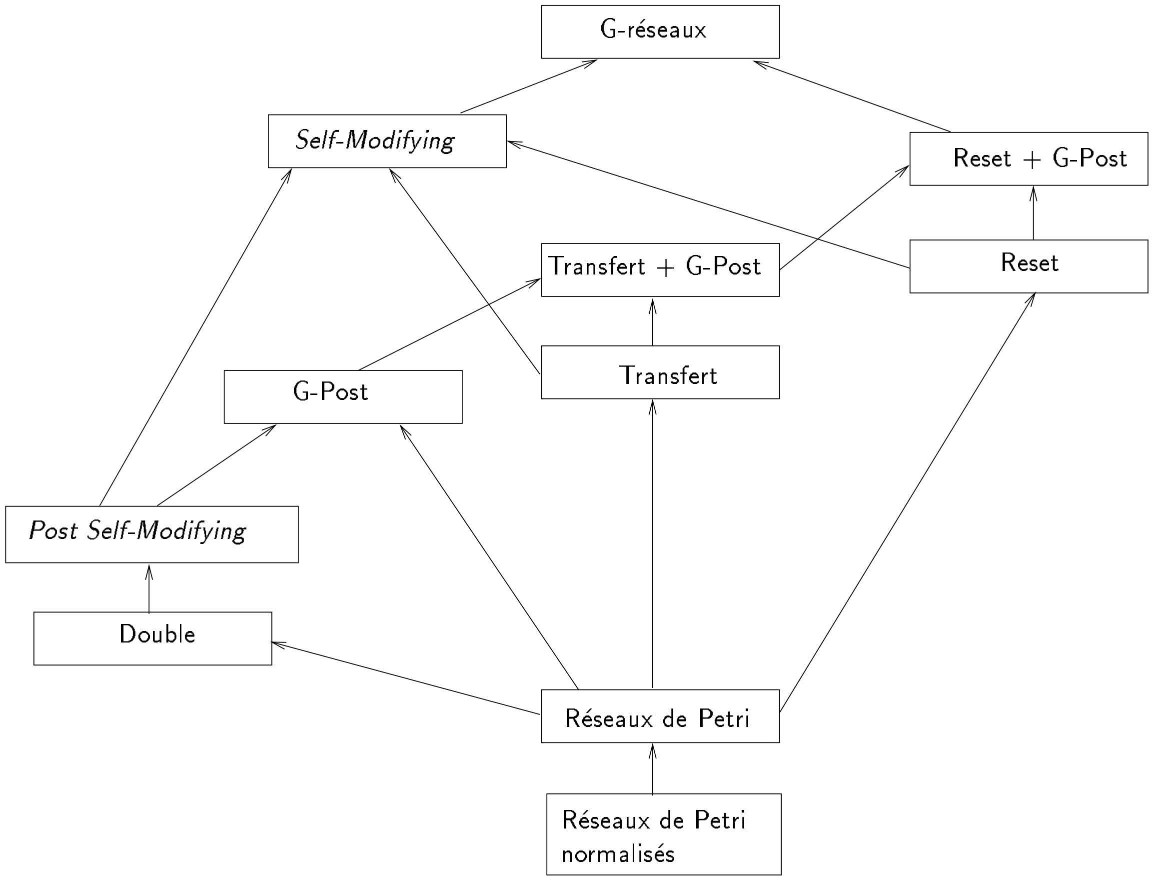
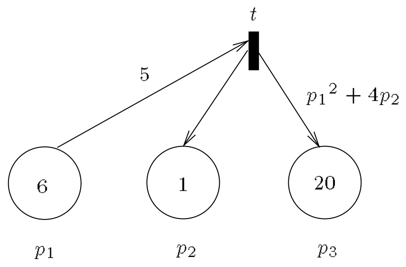


FIG. 2.2 – *Hiérarchie des sous-classes des G-réseaux*

2.2.2 Effets des arcs étendus

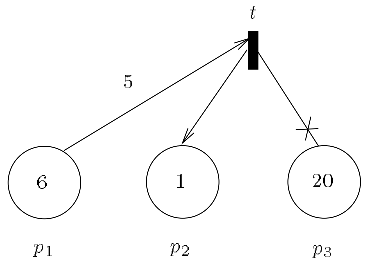
Voici quelques exemples de franchissements de transitions étendues. Rappelons qu'une même transition peut impliquer plusieurs arcs d'un même type (étendu ou Petri). Par contre, pour tout couple (p, t) ou (t, p), il n'y a qu'un arc étiqueté par un polynôme.

G-Post :



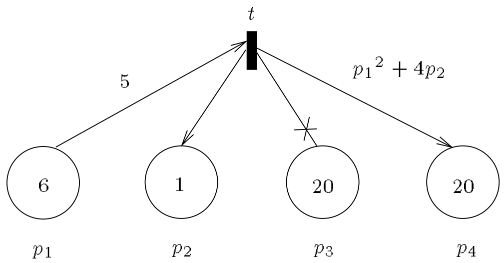
Ici, nous avons $(6, 1, 20) \xrightarrow{t} (1, 2, 60)$.

Petri + Reset :



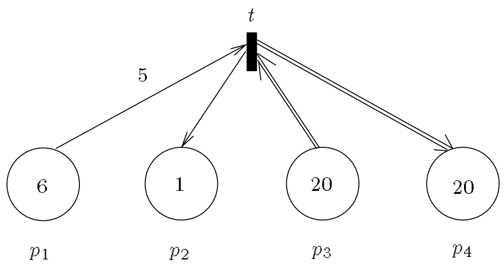
Ici, nous avons $(6, 1, 20) \xrightarrow{t} (1, 2, 0)$. Remarquons d'ailleurs que pour tout $i \in \mathbb{N}$, nous avons $(6, 1, i) \xrightarrow{t} (1, 2, 0)$.

G-Post + Reset :



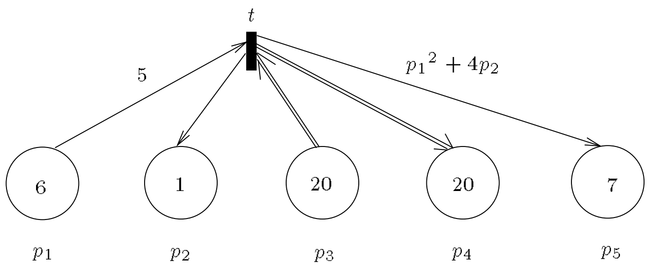
Ici, nous avons $(6, 1, 20, 20) \xrightarrow{t} (1, 2, 0, 60)$.

Petri + Transfert :



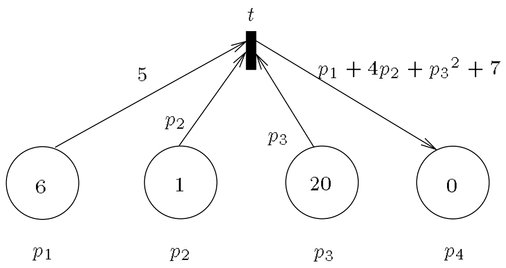
Ici, nous avons $(6,1,20,20) \xrightarrow{t} (1,2,0,40)$.

G-Post + Transfert :



Ici, nous avons $(6,1,20,20,7) \xrightarrow{t} (1,2,0,40,47)$.

G-Post + Transfert :



En voici un second exemple. Nous utilisons cette fois la description explicite avec les polynômes. Ici, les places p_2 et p_3 sont vidées, mais leur contenu est sauvegardé dans p_4. Nous avons $(6,1,20,0) \xrightarrow{t} (1,0,0,417)$.

2.3 Classification des G-réseaux bien structurés

Les systèmes que décrivent certaines sous-classes de G-réseaux sont *bien structurés* sous la relation $\leq$ (l'ordre habituel pour les vecteurs d'entiers). Ceci veut dire que si l'on a $m_1 \xrightarrow{t} m_2$ et $m_1 \leq m_1'$ alors on a $m_1' \xrightarrow{t} m_2'$ avec $m_2 \leq m_2'$. Intuitivement, ce qui peut être effectué d'un marquage peut aussi l'être de tout marquage plus grand. Ces systèmes entrent dans le cadre des *systèmes de transitions bien structurés*.

Les *systèmes de transitions (bien) structurés* ont été définis en 1987 par FINKEL [Fin87, Fin90] (voir aussi [FS97]), puis sous une forme un peu différente en 1996 par ABDULLA et al [AČJY96]. Ils définissent un concept général d'abstraction de structures particulières et permettent d'obtenir des résultats de décidabilité pour certains problèmes tels que, par exemple, la terminaison ou le caractère borné.

Dans cette section nous définissons les systèmes de transitions bien structurés, et nous répertorions les sous-classes de la hiérarchie de G-réseaux selon trois catégories : les sous-classes *bien structurées*, qui incluent les sous-classes *strictement structurées*, qui incluent elles-mêmes les sous-classes *fortement structurées*. Les notions de "strictement" et "fortement" structuré raffinent la notion de "bien" structuré et permettent d'obtenir des résultats supplémentaires.

2.3.1 Systèmes de transitions bien structurés

Nous commençons par rappeler la définition des systèmes de transitions (voir aussi [Arn92]).

Définition 12

- *Un système de transitions (infini) est une structure* $\mathcal{T} =< S, \mapsto >$ *où* S *est un ensemble (infini) d'états et* $\mapsto \subseteq S \times S$ *est une relation de transition.*

- *Étant donnés deux états* s *et* $s' \in S$, s' *est accessible de* s*, noté* $s \mapsto s'$*, si* $(s, s') \in \mapsto$.

Exemple : un G-réseau $N =< P, T, F, m_0 >$ peut représenter de façon compacte et finie un système de transitions infini, où $S = Acc(N)$ et où $m \mapsto m'$ si et seulement si $\exists\, t \in T$ telle que $m \xrightarrow{t} m'$.

Nous ne considérons que les systèmes de transitions à branchement fini, c'est-à-dire tels que chaque état a un nombre fini de successeurs immédiats. C'est le cas des G-réseaux.

Définition 13 *(voir aussi [Hig52, Kru72]) Un bel ordre* $\preceq$ *sur un ensemble* X *est une relation réflexive et transitive, telle que pour toute séquence infinie* $x_0, x_1, \cdots$ *de* X*, il existe deux indices* i *et* j*,* $i < j$*, tels que* $x_i \preceq x_j$.

Un bel ordre est *bien fondé*, c'est-à-dire qu'il n'admet pas de séquence infinie strictement décroissante.

Exemple : la relation $\leq$ sur les vecteurs d'entiers naturels est un bel ordre.

Définition 14 *Un système de transitions* $\mathcal{T} =< S, \mapsto >$ *muni d'une relation* $\preceq \subseteq S \times S$ *est bien structuré si :*

1. $\preceq$ *est un bel ordre,*

2. *pour tout* $s_1, s_2 \in S$ *tels que* $s_1 \preceq s_2$*, nous avons :*

$$\exists s'_1 \in S,\ s_1 \mapsto s'_1 \ \Rightarrow\ \exists\, s'_2 \in S,\ s_2 \mapsto s'_2 \ et\ s'_1 \preceq s'_2$$

L'existence d'un bel ordre sur l'ensemble infini des états d'un système de transitions peut garantir la terminaison de méthodes algorithmiques utilisées pour analyser ce système.

2.3.2 Premiers résultats

Proposition 1 *Les G-Post + Reset munis de la relation $\leq$ sur les marquages forment des systèmes de transitions bien structurés.*

Preuve : étant donné un réseau G-Post + Reset $N =< P, T, F, m_0 >$, le système de transitions qui lui correspond a pour ensemble d'états $S = Acc(N)$, avec $\forall\, m_1, m_2 \in S$:

$$m_1 \mapsto m_2 \Leftrightarrow \exists\, t \in T,\ m_1 \xrightarrow{t} m_2$$

Considérons deux marquages m_1 et m_2 dans $Acc(N)$ tels que $m_1 \leq m_2$ et une transition $t \in T$. Si $m_1 \xrightarrow{t}$, clairement $m_2 \xrightarrow{t}$. Ainsi, si $m_1 \xrightarrow{t} m_1'$, alors il existe un marquage m_2' tel que $m_2 \xrightarrow{t} m_2'$.

Vérifions que $m_1' \leq m_2'$. Pour chaque place p, deux cas peuvent se produire :

1. Si t ne met pas p à zéro, l'action opérée sur p par t à partir de m_1 est de la forme :

   $$m_1'(p) = m_1(p) - c + Q(m_1)$$

   avec $c \in \mathbb{N}$, $c \leq m_1(p)$ et $F(t,p) = Q \in \mathbb{N}[P]$. Puisque $m_1(p) \leq m_2(p)$ et puisque l'évaluation du polynôme, l'addition et la soustraction sont des opérations monotones, nous avons :

   $$m_2'(p) = m_2(p) - c + Q(m_2) \geq m_1(p) - c + Q(m_1) = m_1'(p)$$

2. Si t met p à zéro, l'effet de t sur p à partir de m_1 est de la forme :

   $$m_1'(p) = m_1(p) - m_1(p) + Q(m_1)$$

   Et nous avons :

   $$m_2'(p) = m_2(p) - m_2(p) + Q(m_2) \geq Q(m_1) = m_1'(p)$$

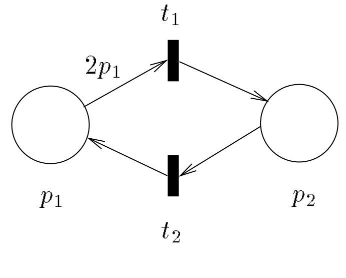
Ainsi, nous avons $\forall p \in P$, $m_1'(p) \leq m_2'(p)$, et donc $m_1' \leq m_2'$. De plus, la relation $\leq$ est un bel ordre. Nous en concluons que les G-Post + Reset sont bien structurés sous la relation $\leq$.

□

Nous voyons à présent que les G-réseaux ne sont pas bien structurés pour l'ordre habituel sur les vecteurs d'entiers.

Proposition 2 *Les G-réseaux munis de la relation $\leq$ ne forment pas des systèmes bien structurés.*

Preuve : considérons le réseau suivant :



Une exécution de ce réseau est :

$$(0,0) \xrightarrow{t_1} (0,1) \xrightarrow{t_2} (1,0)$$

où $(1,0)$ est un blocage. Ainsi, ce qui est possible à partir de $(0,0)$ n'est pas toujours possible à partir d'un marquage accessible, plus grand. Le système n'est donc pas bien structuré pour $\leq$.

□

Dans notre hiérarchie de G-réseaux, la plus grande classe bien structurée sous $\leq$ est donc celle des G-Post + Reset.

2.3.3 Raffinement des classes bien structurées

Afin de séparer plus finement les différentes sous-classes de la hiérarchie, nous raffinons la notion de “bien structuré” pour la relation $\leq$ et nous retenons trois caractérisations. En accord avec la définition de [Fin90], nous posons :

Définition 15 *Étant donné un G-réseau* $N = < P, T, F, m_0 >$ *:*

- *N est bien structuré si $\forall t \in T$, $\forall\, m_1,\, m_2 \in Acc(N)$ tel que $m_1 \leq m_2$:*

$$\exists m_1' \in \mathbb{N}^{|P|},\ m_1 \xrightarrow{t} m_1' \ \Rightarrow\ \exists\, m_2' \in \mathbb{N}^{|P|},\ m_2 \xrightarrow{t} m_2' \text{ et } m_1' \leq m_2'$$

- *N est strictement structuré si N est bien structuré et $\forall t \in T$, $\forall\, m_1,\, m_2 \in Acc(N)$ tel que $m_1 < m_2$:*

$$\exists m_1' \in \mathbb{N}^{|P|},\ m_1 \xrightarrow{t} m_1' \ \Rightarrow\ \exists\, m_2' \in \mathbb{N}^{|P|},\ m_2 \xrightarrow{t} m_2' \text{ et } m_1' < m_2'$$

- *N est fortement structuré si N est strictement structuré et $\forall\, m_1,\, m_2 \in Acc(N)$ tel que $m_1 < m_2$, $m_1 \xrightarrow{t} m_1'$ et $m_2 \xrightarrow{t} m_2'$:*

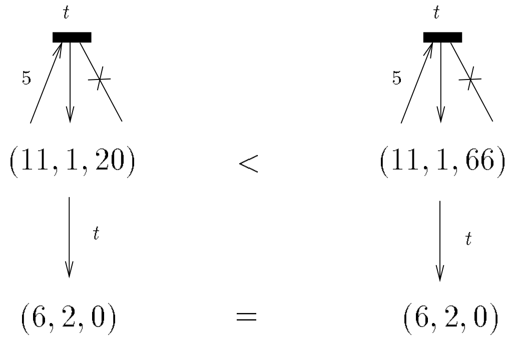
$$m_1(p) < m_2(p) \Rightarrow m_1'(p) < m_2'(p)$$

Ces définitions s'étendent naturellement aux séquences de transitions. Les sous-classes fortement structurées sont strictement structurées. Et les sous-classes strictement structurées sont bien structurées.

Les G-Post + Reset sont seulement bien structurés, mais pas strictement structurés.

Proposition 3 *Les réseaux de Petri avec Reset ne sont pas strictement structurés.*

Preuve : considérons les franchissements de la transition suivante :



De deux marquages différents m_1 et m_2 tels que $m_1 < m_2$, on peut atteindre le même marquage en franchissant une transition qui admet des arcs Reset. Nous en concluons que les réseaux de Petri avec Reset ne sont pas strictement structurés. □

Corollaire 4 *Les G-Post + Reset ne sont pas strictement structurés.*

Considérons à présent les G-Post + Transfert. Cette classe est bien structurée puisqu'elle est incluse dans les G-Post + Reset. De plus :

Proposition 5 *Les G-Post + Transfert sont strictement structurés.*

Preuve : étant donné un G-Post + Transfert $N =< P, T, F, m_0 >$, considérons les marquages m_1, $m_2 \in Acc(N)$ tels que $m_1 < m_2$ et $t \in T$. Si :

$$\exists m'_1 \in \mathbb{N}^{|P|} \text{ tel que } m_1 \xrightarrow{t} m'_1$$

alors, puisque N est bien structuré,

$$\exists\, m'_2 \text{ tel que } m_2 \xrightarrow{t} m'_2 \text{ et } m'_1 \leq m'_2$$

Vérifions que $m'_1 < m'_2$. Puisque $m_1 < m_2$, alors $\exists\, p_i \in P$ telle que $m_1(p_i) < m_2(p_i)$. Deux cas se présentent pour p_i :

1. p_i n'est pas mise à zéro par t (l'arc de p_i vers t est de Petri). Nous avons pour m'_1 :

   $$m'_1(p_i) = m_1(p_i) - c + Q(m_1) \text{ avec } F(p_i, t) = c \in \mathbb{N} \text{ et } F(t, p_i) = Q \in \mathbb{N}[P]$$

   et pour m'_2 :

   $$m'_2(p_i) = m_2(p_i) - c + Q(m_2) > m_1(p_i) - c + Q(m_2) \geq m'_1(p_i)$$

   donc $m'_2(p_i) > m'_1(p_i)$.

2. p_i est à l'origine d'un transfert, et donc est vidée par t, et il existe une place p_j telle que $F(t, p_j) = \lambda {p_i}^n + Q'$ avec $\lambda, n > 0$ et $Q' \in \mathbb{N}[P]$. Nous avons pour m'_1 :

   $$m'_1(p_j) = m_1(p_j) - X + \lambda m_1(p_i)^n + Q'(m_1)$$

   où soit $X = c \in \mathbb{N}$, soit $X = m_1(p_j)$ (une mise à zéro). Et pour m'_2 :

   $$\begin{aligned} m'_2(p_j) &= m_2(p_j) - X' + \lambda m_2(p_i)^n + Q'(m_2) \\ &> m_2(p_j) - X' + \lambda m_1(p_i)^n + Q'(m_2) \geq m'_1(p_j) \end{aligned}$$

   où $X' = c$ si $X = c$, et $X' = m_2(p_j)$ sinon. Nous en concluons que $m'_1(p_j) < m'_2(p_j)$.

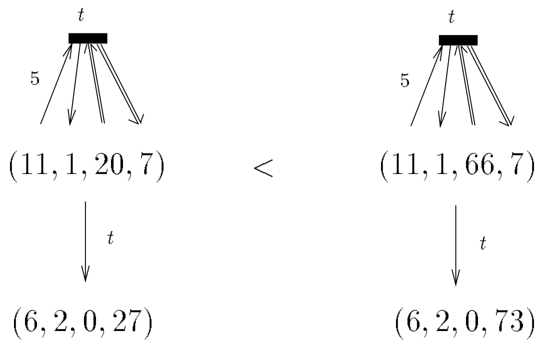
Dans les deux cas, nous obtenons au moins une place p telle que $m'_1(p) < m'_2(p)$, et puisque $m'_1 \leq m'_2$, alors $m'_1 < m'_2$. Nous en concluons que les G-Post + Transfert sont strictement structurés.

□

Cependant, les G-Post+ Transfert ne sont pas fortement structurés.

Proposition 6 *Les réseaux de Petri avec Transfert ne sont pas fortement structurés.*

Preuve : considérons les franchissements de la transition suivante :



Ici, $m_1 = (11, 1, 20, 7)$ et $m_2 = (11, 1, 66, 7)$ sont tels que $m_1 < m_2$ et, en particulier, $m_1(p_3) < m_2(p_3)$. Nous avons $m_1 \xrightarrow{t} m'_1 = (6, 2, 0, 27)$ et $m_2 \xrightarrow{t} m'_2 = (6, 2, 0, 73)$ tels que $m'_1(p_3) = m'_2(p_3) = 0$. Ainsi, $m_1(p_3) < m_2(p_3)$ mais $m'_1(p_3) = m'_2(p_3)$. Nous en concluons que les réseaux de Petri avec Transfert ne sont pas fortement structurés. □

Corollaire 7 *Les G-Post + Transfert ne sont pas fortement structurés.*

Considérons enfin les G-Post. Ils sont strictement structurés puisqu'ils sont inclus dans les G-Post + Transfert. De plus :

Proposition 8 *Les G-Post sont fortement structurés.*

Preuve : reprise directe du cas (1) de la preuve de la proposition 5. □

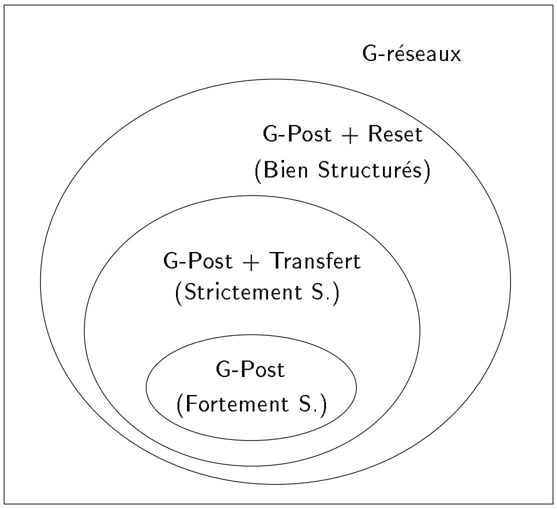


FIG. 2.3 – *Caractérisation de la hiérarchie des G-réseaux.*

Ainsi, les réseaux de Petri ou les *Post Self-Modifying nets* sont fortement structurés et de nombreuses preuves de décidabilité s'appuient sur cette caractérisation.

La figure 2.3 résume en un diagramme la caractérisation de la hiérarchie des G-réseaux.

2.4 Taille d'un G-réseau

Dans le chapitre 4, nous donnerons plusieurs réductions polynomiales. Avant de parler de complexité, il est nécessaire de définir ce qu'est la taille d'un G-réseau. À des fins de simplicité, la borne que nous choisissons est calculée largement. Les polynômes sont représentés sous leur forme développée.

Définition 16 *Soit P un ensemble de variables et soit $Q \in \mathbb{N}[P]$, présenté sous sa forme développée :*

$$Q(p_1..p_k) = \Sigma_{j \in J} \lambda_j \Pi_{i=1}^{|P|} p_i^{n_{i_j}}, \; p_i \in P, \; \lambda_j, n_{i_j} \in \mathbb{N}, \; J \text{ ensemble fini}$$

- $V(Q) = \mathtt{max}_{j \in J}(\{\lambda_j\} \cup \mathtt{max}_{i=1}^{|P|}\{n_{i_j}\})$.
- $L(Q) = |J|$

$V(Q)$ est le maximum de toutes les valuations de la description de Q (coefficients, exposants et constantes) et $L(Q)$ est le nombre de termes de Q.

Définition 17 *Soit $N =< N, T, F, m_0 >$ un G-réseau.*

- $V(N) = \ \mathtt{max} \ (\ \{ V(Q) \,|\, \exists\, (x,y) \in (P \times T) \cup (T \times P), \, F(x,y) = Q\} \cup \{m_0(p_i) \,|\, p_i \in P\} \ )$
- $L(N) = \ \mathtt{max} \ (\ \{ L(Q) \,|\, \exists\, (x,y) \in (P \times T) \cup (T \times P), \, F(x,y) = Q\} \ )$

$V(N)$ est la plus grande valuation de la définition de N parmi les coefficients, les exposants et les constantes de tous les polynômes et les valeurs du marquage initial. $L(N)$ est le nombre maximum de termes nécessaires à la description d'un polynôme apparaissant dans F.

Nous codons $N =< P, T, F, m_0 >$ avec deux matrices et un vecteur. La première matrice M_{pre} correspond aux pré-arcs. La seconde matrice M_{post} correspond aux post-arcs. Le vecteur est le marquage initial. Chaque matrice est de dimension $|P| \times |T|$ et contient la description des polynômes, dont chacun comprend au plus $L(N)$ termes. Nous réservons l'espace nécessaire en pire cas pour décrire un polynôme, c'est-à-dire $(L(N)\,|P| \log V(N))$ et nous définissons la taille de N comme étant :

$$|N| = |M_{pre}| + |M_{post}| + |m_0|$$

c'est-à-dire :

$$|N| \in \Theta(\,|P| \ \ |T| \ \ L(N)\,|P| \log V(N) \ + \ |P| \log V(N)\,)$$

ou encore :

$$|N| \in \Theta(\,|P|^2 \ |T| \ L(N) \ \log V(N)\,)$$

2.5 État de l'art

Carl Adam PETRI définit son modèle en 1962 dans sa thèse de doctorat [Pet62]. KARP et MILLER étudient en 1969 un modèle équivalent : les *systèmes d'addition de vecteurs* [KM69]. Aujourd'hui, les réseaux de Petri sont à la base de nombreux outils de vérification utilisés dans de nombreux sites universitaires et dans de grands laboratoires industriels. Citons Design-CPN [Jen97] pour les *réseaux de Petri colorés*, PEP [GRT+95] pour les *réseaux de Petri avec algèbres de processus* ou encore PAPETRI [BJP90].

Nous allons passer en revue les principaux résultats de complexité et de décidabilité des réseaux de Petri (voir [EN94] pour une synthèse plus complète).

De nombreux résultats de décidabilité, et en particulier celui du caractère borné, sont dus à KARP et MILLER. On montre qu'un réseau de Petri $N =< P, T, F, m_0 >$ est non borné si et seulement s'il existe un "générateur de jetons", c'est-à-dire une séquence de transitions $\sigma \in T^*$ et $m_1,\ m_2 \in \mathbb{N}^{|P|}$ tels que :

$$m_0 \xrightarrow{*} m_1 \xrightarrow{\sigma} m_2, \text{ avec } m_1 < m_2$$

La séquence σ peut être itérée une infinité de fois, rajoutant à chaque fois de nouveaux jetons dans le réseau (nous rejoignons des concepts liés aux systèmes bien structurés avec "compatibilité stricte"). En 1978, RACKOFF [Rac78] prouve que si un réseau n'est pas borné, alors il existe une telle séquence dont la longueur est bornée par une double exponentielle en fonction de la taille du réseau. De là, RACKOFF déduit un résultat de complexité-espace pour décider du caractère borné d'un réseau N (valable aussi pour la couverture) de $2^{c.|N| \log |N|}$ où $|N|$ est la taille du réseau N et c une constante.

Décider du caractère structurellement borné pour les réseaux de Petri s'effectue par contre en temps polynomial [MR80]. Pour montrer qu'un réseau n'est pas structurellement borné, il suffit d'exhiber une séquence $\sigma \in T^*$ dont l'effet total est positif sur toutes les places de N et strictement positif sur au moins une place. Ce problème se ramène à la résolution d'un système d'inéquations linéaires.

KARP et MILLER introduisent le concept d'*arbre de couverture*, construit par une technique de *chaînage avant* (voir aussi FINKEL [Fin93]) dont le principe consiste à essayer systématiquement tous les franchissements de transitions possibles à partir du marquage initial. Une approximation "couvrante" de l'ensemble des états accessibles est construite, permettant de décider de nombreux problèmes dont la couverture ou le caractère borné d'une place. Mais LIPTON donne une borne inférieure de complexité-espace pour la couverture et le caractère borné de $2^{\sqrt{|N|}}$ et BOUZIANE [Bou96] affine la borne pour le caractère borné avec $2^{|N|}$.

De nombreux chercheurs se sont penchés sur le problème de l'accessibilité dans les réseaux de Petri qui est très célèbre pour sa difficulté d'analyse. Ce problème est EXPSPACE-difficile [CLM76, Lip76]; il a été montré décidable par KOSARAJU [Kos82] et MAYR [May84] (voir aussi [Lam92]), cependant les algorithmes proposés étaient NON-primitif-récursifs. Le blocage et la vivacité sont récursivement équivalents à l'accessibilité [Hac76] et donc sont aussi décidables.

Pendant 36 ans, la question de la complexité intrinsèque du problème d'accessibilité est restée ouverte. Très récemment, Zakariae Bouziane a fermé cette question en montrant que le problème d'accessibilité est primitif-récursif [Bou98b]. De plus, la borne supérieure de complexité est exponentielle en espace [Bou98a]. Ce résultat apporte un atout supplémentaire aux réseaux de Petri, car l'accessibilité est une propriété clé de l'analyse des systèmes.

Les extensions des réseaux de Petri

Pour faciliter le travail de modélisation des systèmes et pour enrichir les descriptions, des modèles étendus ont été proposés. Dans les modèles de *haut niveau*, on associe de l'information aux jetons et on leur donne une identité. Citons les réseaux de Petri colorés de Jensen [Jen92, Jen95, Jen97] ou encore les réseaux à prédicats [GL81] (voir aussi [Smi96] pour une synthèse).

Dans notre étude, nous nous intéresserons plutôt aux extensions relatives aux fonctions qui étiquettent les arcs. [FMP96] est une étude générale de ce type d'extensions qui considère des fonctions vérifiant de "bonnes propriétés" (telle la monotonie).

Les réseaux de Petri avec *deux* arcs inhibiteurs peuvent simuler les machines de Turing [Hac75], rendant ainsi toutes les propriétés non triviales indécidables. Par contre, pour les réseaux de Petri avec un seul arc inhibiteur, l'accessibilité est décidable [Rei95].

En 1978, Valk [Val78a, Val78b, Val81] a proposé deux extensions strictes des réseaux de Petri. L'extension la plus faible, les *Post Self-Modifying nets*, autorise d'ajouter aux places non seulement des constantes, mais des combinaisons linéaires du marquage courant du réseau. Pour ce modèle, la couverture et le caractère borné du réseau (et d'une place) restent décidables, mais l'accessibilité devient indécidable.

La seconde extension, les *Strong Self-Modifying nets* [Val78b], est plus puissante. Elle autorise une transition à *soustraire* en plus d'additionner aux places, des combinaisons linéraires du marquage courant du réseau. On obtient cette fois un modèle pour lequel toutes les propriétés non triviales sont indécidables, car les *Strong Self-Modifying nets* permettent d'implémenter les arcs inhibiteurs.

Les Reset constituent un cas particulier des *Self-Modifying nets*. En 1977, Araki et Kasami [AK77] montrent que pour la classe des réseaux de Petri avec arcs de Reset, l'accessibilité est indécidable. La preuve consiste à simuler une machine à compteurs et elle est très semblable à la preuve d'indécidabilité de l'accessibilité pour les *Post Self-Modifying nets*. En fait, *deux* arcs *Reset*, *Post* ou *Transfert* suffisent à rendre l'accessibilité indécidable [DFS98]. Billington utilise les Reset pour modéliser le protocole du *"Cambridge fast ring network"* [Bil91]. Récemment, Kishinevsky et al. [KCK+97] introduisent le modèle des *Place-chart nets* qui autorise des mises à zéro sur des ensembles de places.

[Ajouté en novembre 1998] En 1976, Arnold et Latteux [AL76, AL78] montrent, dans un cadre de théorie des langages, que la couverture est décidable pour les systèmes VASE (*Vector Addition Systems with Erasing*) qui s'apparentent en fait aux réseaux de Petri avec Reset. La technique est de type *chaînage arrière* car l'algorithme de construction de

l'arbre de Karp et Miller, de type *chaînage avant*, ne peut pas s'étendre aux VASE, à cause des *effacements* (mises à zéro).

On trouve aussi d'autres études d'arcs étendus. Ciardo étudie en [Cia94] les arcs Reset, Transfert et les Post arcs de Valk (entre autres) du point de vue des *semi-flots* (places et transitions) et des classes de langage de ces réseaux étendus. Lakos et Christensen [LC94] étudient et comparent une collection d'arcs étendus, dont les Reset, les Transfert et les Double, pour les réseaux de Petri colorés.

D'autres restrictions des *Self-Modifying nets* ont été proposées. Badouel et Darondeau ont défini les *réseaux stratifiés* [BD97] qui sont des *Self-Modifying nets* dont les places $p_1, .., p_k$ sont ordonnées de telle sorte que le contenu de p_i ne puisse être ajouté ou retiré que d'une autre place p_j telle que $j < i$. Mais ce modèle peut implémenter les arcs inhibiteurs. Enfin, les *réseaux reconfigurables* [BO98] s'apparentent aux réseaux Transfert et pour ce modèle, le caractère borné est décidable.

Tableau récapitulatif

Voici un résumé de la situation au mieux de notre connaissance au début de cette thèse, des problèmes de décision des G-réseaux et des sous-classes que nous avons définies :

	Terminaison	Couverture	Réseau borné	Place bornée	Réseau structurellement borné	Accessibilité	Blocage Vivacité t-Vivacité
G-Réseaux *Strong Self-Modifying*	**I**	**I**	**I**	**I**	**I**	**I**	**I**
G-Post + Reset	?	?	?	?	?	**I**	?
Reset	?	?	?	?	?	**I**	?
G-Post + Transfert	?	?	?	?	?	**I**	?
Transfert	?	?	?	?	?	**I**	?
G-Post	?	?	?	?	?	**I**	?
Post Self-Modifying	**D**	**D**	**D**	**D**	?	**I**	?
Petri	**D**	**D**	**D**	**D**	**D**	**D**	**D**

Légende : D=Décidable et I=Indécidable.

[Ajouté en novembre 1998] Il manquait à notre liste le résultat de la décidabilité de la couverture pour les Reset en [AL76, AL78].

L'objet de cette thèse est principalement de compléter ce tableau et nous obtiendrons les résultats suivants :

	Terminaison	Couverture	Réseau borné	Place bornée	Réseau structurellement borné	Blocage Accessibilité	Vivacité t-Vivacité
G-Réseaux *Strong Self-Modifying*	**I**	**I**	**I**	**I**	**I**	**I**	**I**
G-Post + Reset	**D**	**D**	**I**	**I**	**I**	**I**	**I**
Reset	**D**	**D**	**I**	**I**	**I**	**I**	**I**
G-Post + Transfert	**D**	**D**	**D**	**I**	**I**	**I**	**I**
Transfert	**D**	**D**	**D**	**I**	**I**	**I**	**I**
G-Post	**D**	**D**	**D**	**D**	**D**	**I**	**I**
Post Self-Modifying	**D**	**D**	**D**	**D**	**D**	**I**	**I**
Petri	**D**	**D**	**D**	**D**	**D**	**D**	**D**

Chapitre 3

Dépliage et normalisation

Une stratégie d'attaque pour étudier une nouvelle classe de réseau consiste à trouver une représentation simplifiée de cette classe. Dans cette section, nous montrerons deux constructions, dont la première simplifie la représentation des G-Post et la seconde simplifie la représentation des réseaux de Petri avec arcs de Reset ou avec arcs de Transfert.

Nous montrerons en section 3.1 comment *déplier* un réseau G-Post (resp. avec Reset; resp. avec Transfert) pour obtenir un réseau de Petri (resp. avec Reset; resp. avec Transfert). Ce dépliage préserve la couverture et le caractère borné du réseau et d'une place. Ces problèmes sont donc décidables pour les G-Post, puisqu'ils le sont pour les réseaux de Petri. Ainsi, ce dépliage donne la possibilité de vérifier les G-Post, par la traduction en un réseau de Petri suivie de l'utilisation des outils existant pour les réseaux de Petri, sans besoin de les modifier. De plus, nous obtiendrons des complexités pour les *Post Self-Modifying nets* pour lesquels le dépliage s'effectue en temps polynomial.

Nous montrerons ensuite en section 3.2 comment *normaliser* en temps polynomial les réseaux de Petri (et les réseaux avec arcs de Reset ou de Transfert). Cette construction préserve la couverture, les trois caractères bornés (réseau, place et structurellement borné), l'accessibilité et la t-vivacité. La contribution principale de notre normalisation est sa complexité car toutes les normalisations connues jusqu'ici sont exponentielles. Les normalisations ont l'avantage de permettre de simplifier des preuves puisqu'on manipule une fonction de flot à valeurs dans $\{0, 1\}$.

3.1 Dépliage d'un réseau G-Post en un réseau de Petri

Déplier un réseau N d'une classe de réseaux $\mathcal{C}$ vers une autre classe $\mathcal{C}'$ consiste en une transformation (réalisable) de N en un réseau N' de $\mathcal{C}'$. Le dépliage a de l'intérêt lorsque la transformation préserve des propriétés qui sont "plus faciles" à analyser dans la classe $\mathcal{C}'$ que dans la classe $\mathcal{C}$.

Considérons le dépliage de N vers N', d'une classe $\mathcal{C}$ vers une classe $\mathcal{C}'$, qui préserve la propriété P et voyons les intérêts de cette transformation :

- Le dépliage permet de donner des *résultats de décidabilité*. Si P est décidable pour $\mathcal{C}'$, alors on déduit que P est décidable pour $\mathcal{C}$.

Exemple : dans cette section, nous montrons comment déplier un G-Post vers un réseau de Petri par une transformation qui préserve la couverture et le caractère borné (du réseau ou d'une place). Puisque ces propriétés sont décidables pour les réseaux de Petri, le dépliage permet de conclure qu'elles le sont aussi pour les G-Post.

- Le dépliage permet aussi d'utiliser sans modification des outils existants pour vérifier des classes étendues.

  Exemple : grâce au dépliage, on peut utiliser les outils de vérification des réseaux de Petri pour analyser les G-Post.

- Le dépliage permet encore de donner des *complexités*. Si l'on connaît la complexité de la transformation de N vers N', et si l'on connaît la complexité de P pour la classe $\mathcal{C}'$, alors on obtient une borne supérieure de complexité de P pour $\mathcal{C}$.

  Exemple : le dépliage des *Post Self-Modifying nets* vers les réseaux de Petri s'effectue en temps polynomial. Ceci nous permet d'obtenir pour les *Post Self-Modifying nets* des complexités proches de celles des réseaux de Petri pour la couverture ou encore pour le caractère borné du réseau.

- Le dépliage permet enfin de *simplifier des preuves*, lorsque la classe d'arrivée $\mathcal{C}'$ est "plus facile" à étudier que la classe de départ $\mathcal{C}$.

  Exemple : au chapitre 4, nous montrerons que la couverture est décidable pour les réseaux de Petri avec Reset (section 4.4). Nous en conclurons que la couverture est décidable pour les G-Post + Reset, en utilisant un dépliage des G-Post + Reset vers les réseaux de Petri avec Reset.

Nous allons montrer comment *déplier* un G-Post (+ Reset) vers un réseau de Petri (+ Reset). En particulier, nous allons devoir simuler l'effet d'un arc G-Post par des réseaux de Petri et donc calculer l'évaluation de polynômes avec des réseaux de Petri.

On trouve dans [Hac75] et [Pet81] une construction qui permet d'évaluer des polynômes avec des réseaux de Petri. L'évaluation d'un polynôme Q à k variables (Q est codé "en dur" dans le réseau) sur l'entrée $x \in \mathbb{N}^k$ (x apparaît dans le marquage de départ) consiste à ajouter au plus $Q(x)$ jetons dans une place particulière que nous appelons p_Q (vide au départ). Avec des réseaux de Petri, il n'est pas possible de calculer exactement $Q(x)$, c'est-à-dire d'assurer le fait que $Q(x)$ jetons seront ajoutés dans p_Q. Cependant, le calcul peut être fait dans un *sens faible* :

- p_Q reçoit exactement $Q(x)$ au terme d'une exécution qui est "la bonne";
- p_Q reçoit au plus $Q(x)$ jetons quelle que soit l'exécution.

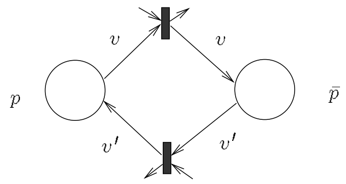
Nous allons exploiter cette idée, mais la difficulté résidera dans l'enchaînement des évaluations successives des polynômes. On peut comprendre déjà ici pourquoi le caractère borné du réseau, par exemple, sera préservé : une évaluation du polynôme d'un post-arc est soit l'évaluation exacte, soit une sous-estimation de l'évaluation.

Nous allons présenter le dépliage de la façon suivante : dans un premier temps, nous montrerons comment construire un réseau qui simule l'évaluation d'un terme d'un polynôme et nous prouverons quelques propriétés sur ce réseau. Sachant évaluer les termes, nous montrerons comment évaluer un polynôme. Ensuite, sachant évaluer les polynômes, nous montrerons comment simuler l'effet d'une transition d'un G-Post. Enfin, nous donnerons le dépliage complet d'un G-Post vers un réseau de Petri.

Au cours de la construction, nous nous servirons de la notion de places complémentaires :

Définition 18 *Soient un G-réseau $N =< P, T, F, m_0 >$ et $T' \subseteq T$.*
Deux places p et $\bar{p}$ sont complémentaires sur T', si tout marquage m' tel que $m \xrightarrow{\sigma} m'$, avec $\sigma \in T'^*$, vérifie l'invariant $m(p) + m(\bar{p}) = m'(p) + m'(\bar{p})$.

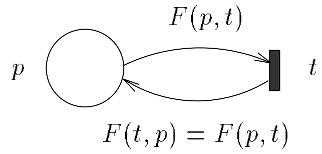
Exemple : pour un réseau de Petri, si pour tout $t \in T,\ F(p,t) = F(t,\bar{p})$ et $F(\bar{p},t) = F(t,p)$, alors p et $\bar{p}$ sont complémentaires pour T :



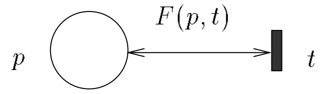
Nous nous servirons aussi de la notion de place *testée* (implémentée avec une boucle élémentaire sur une place) :

Définition 19 *Soit un G-réseau $N =< P, T, F, m_0 >$.*
Une place p est testée par une transition t si $F(p,t) \in \mathbb{N}$ et $F(p,t) = F(t,p)$.

Franchir t dépend du contenu de p, mais ne modifie pas le contenu de p :



Nous représentons aussi cette action de façon abrégée avec un seul arc :



3.1.1 Évaluation d'un terme par un réseau de Petri

Nous commençons par décrire un réseau qui, étant donné $n \in \mathbb{N}$ et $v \in \mathbb{N}^n$, calcule :

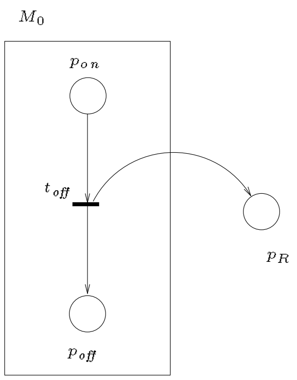
$$R(v) = \Pi_{i=1}^{n} v(i)$$

Rappelons que pour le cas $n = 0$, par convention, nous obtenons $R = 1$. Ce calcul est effectué dans une place p_R, qui est vide au départ, par le biais des n multiplications élémentaires suivantes :

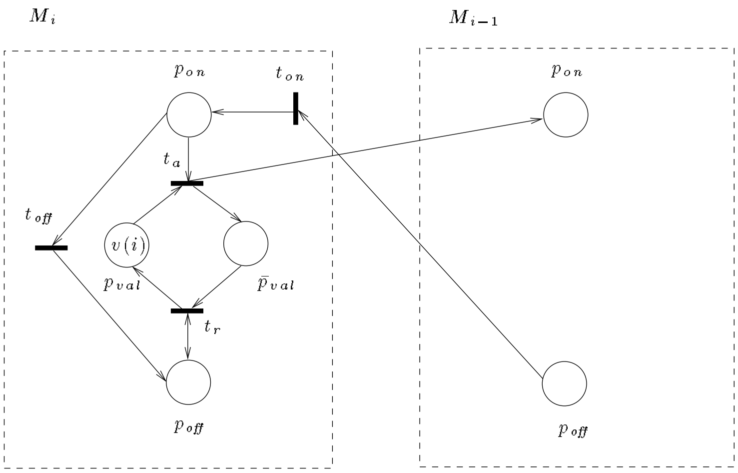
$$p_R \leftarrow v(n) * v(n-1) * \cdots * v(1) * 1$$

La construction sera décrite de façon récursive. Nous allons associer un réseau de Petri M_i (simple) à chaque $v(i)$ et un autre réseau de Petri M_0 (simple) pour la multiplication par 1. Nous appelons ces réseaux des *modules*. Pour évaluer R, nous mettrons les n modules M_i en une série terminée par M_0.

Ajout d'un jeton dans p_R : le réseau M_0 illustré ci-dessous ajoute un jeton dans p_R. Il est composé de deux places : $P_0 = \{p_{on},\, p_{off}\}$ et d'une transition $T_0 = \{t_{off}\}$. Lorsque p_{on} est marquée, t_{off} est franchissable. Franchir t_{off} ajoute 1 jeton dans p_R et fait passer un jeton de p_{on} à p_{off}.



Multiplication par $v(i)$: on associe à $v(i)$ le module M_i de type réseau de Petri. La figure ci-dessous représente le module M_i et montre comment M_i est interconnecté à sa droite avec le module M_{i-1} :



M_i, lorsqu'il sera correctement initialisé en fonction de $v(i)$, permettra d'effectuer une multiplication de $v(i)$ par $\Pi_{j=1}^{i-1} v(j)$ dans p_R.

Définissons $M_i =< P_i, T_i, F_i, m_{0i} >$ $(1 \le i \le n)$:

- $P_i = \{p_{on}, p_{off}, p_{val}, \bar{p}_{val}\}$.

– $T_i = \{t_r, t_a, t_{on}, t_{off}\}$ ($_a$ pour *avancer* et $_r$ pour *renflouer*).

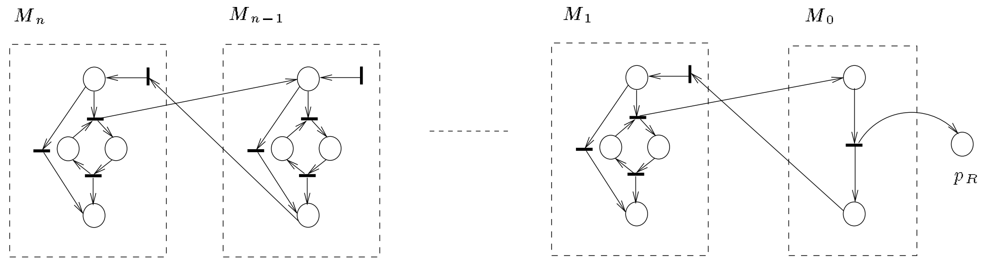
– La fonction F_i est décrite par le schéma.

– $m_{0i} = (0, 0, v(i), 0)$ pour l'ordonnancement suivant des places : p_{on}, p_{off}, p_{val} et $\bar{p}_{val}$

Les places p_{val} et $\bar{p}_{val}$ sont complémentaires pour les transitions de M_i.

Évaluation de $R(v)$.

Soient $n \in \mathbb{N}$ et $v \in \mathbb{N}^n$. Pour calculer $R(v) = \Pi_{i=1}^{n} v(i)$, nous construisons un réseau N_R en mettant les n modules de multiplication M_i et le module M_0 en série :



Notation : *pour désigner une place p (resp. une transition t) spécifique à un module M_i, nous utiliserons la notation $M_i[p]$ (resp. $M_i[t]$).*

Définissons $N_R =< P_R, T_R, F_R, m_{0R} >$ pour le calcul de $R(v)$:

– $P_R = \cup_{i=0}^{n} P_i \cup \{p_R\}$;

– $T_R = \cup_{i=0}^{n} T_i$;

– F_R est décrite par le schéma;

– $\forall p \in P_R,\ m_{0R}(p) = \begin{cases} 1 & \text{si } p = M_n[p_{on}] \\ v(i) & \text{si } p = M_i[p_{val}]\ (1 \leq i \leq n) \\ 0 & \text{sinon} \end{cases}$

Dans m_{0R}, chaque module M_i est correctement initialisé pour $v(i)$ et $M_n[p_{on}]$ contient un jeton pour "démarrer" le calcul. Lorsque le calcul de $R(v)$ dans p_R est effectué correctement, N_R termine dans un marquage particulier :

Définition 20 *Le marquage m_{f_R} tel que :*

$$\forall p \in P_R,\ m_{f_R}(p) = \begin{cases} 1 & \textit{si } p = M_n[p_{off}] \\ v(i) & \textit{si } p = M_i[p_{val}]\ (1 \leq i \leq n) \\ R(v) & \textit{si } p = p_R \\ 0 & \textit{sinon} \end{cases}$$

est un marquage final après calcul de $R(v)$ par N_R.

Le marquage final m_{f_R} est presque le même que m_{0R}, mais cette fois c'est $M_n[p_{off}]$ qui contient un jeton et p_R contient $R(v)$ jetons. Le calcul de $R(v)$ dans N_R est effectué par la séquence *nominale*.

Définition 21 *Une séquence $\sigma \in T_R{}^*$ telle que $m_{0R} \xrightarrow{\sigma} m_{f_R}$ est une séquence nominale.*

Lemme 9 *Pour tout $n \in \mathbb{N}$, pour tout $v \in \mathbb{N}^n$, il existe une séquence nominale dans N_R (le réseau associé à n) qui calcule correctement $R(v)$ dans p_R.*

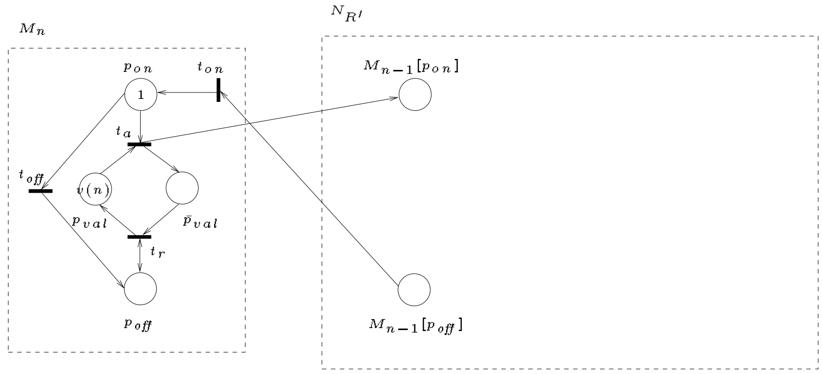
Preuve : nous donnons une preuve par induction sur n.

Considérons le cas où $n = 0$; alors $R = 1$ et le réseau N_R est composé du module M_0 qui contient deux places $M_0[p_{on}]$ et $M_0[p_{off}]$. Nous avons $m_{0R} = (1, 0, 0)$ et $m_{f_R} = (0, 1, 1)$, et en franchissant t_{off}, nous avons bien $m_{0R} \xrightarrow{t_{off}} m_{f_R}$. Le cas $n = 0$ est donc vérifié.

Supposons que pour tout $k \leq n - 1$, pour tout $v \in \mathbb{N}^k$, il existe une séquence nominale dans N_R (le réseau associé à k) pour calculer $R(v) = \Pi_{i=1}^{k} v(i)$ dans p_R. Montrons que cela est vrai pour n.

Soit N_R le réseau associé à n ($n \geq 1$) et soit $v \in \mathbb{N}^n$.

Définissons $v' \in \mathbb{N}^{n-1}$ tel que $\forall i \in [1..(n-1)]$, $v'(i) = v(i)$ et considérons le terme $R'(v') = \Pi_{i=1}^{n-1} v'(i)$. Nous avons $R(v) = v(n) * R'(v')$ et le réseau $N_{R'}$ (associé à $n - 1$) est exactement le réseau N_R dont on a retiré le module M_n. Nous avons la situation suivante :



Appliquons notre hypothèse d'induction et notons $\sigma' \in T_{R'}{}^*$ la séquence nominale qui calcule $R'(v')$ par $N_{R'}$. Considérons m_{0R} un marquage initial pour calculer $R(v)$ par N_R et considérons la séquence σ_1 suivante pour N_R :

$$\sigma_1 = (M_n[t_a]\,\sigma' M_n[t_{on}])^{v(n)}$$

Cette séquence est franchissable à partir de m_{0R} et ajoute $v(n) * R'(v')$ jetons dans p_R. De plus, nous avons $m_{0R} \xrightarrow{\sigma_1} m'$ avec :

$$\forall p \in P_R,\ m'(p) = \begin{cases} 1 & \text{si } p = M_n[p_{on}] \\ v(n) & \text{si } p = M_n[\bar{p}_{val}] \\ v(i) & \text{si } p = M_i[p_{val}]\ (1 \leq i \leq n-1) \\ R(v) & \text{si } p = p_R \\ 0 & \text{sinon} \end{cases}$$

Considérons à présent la séquence :

$$\sigma_2 = M_n[t_{off}]\, M_n[t_r]^{v(n)}$$

Nous avons $m' \xrightarrow{\sigma_2} m''$ avec :

$$\forall p \in P_R,\ m''(p) = \begin{cases} 1 & \text{si } p = M_n[p_{off}] \\ v(i) & \text{si } p = M_i[p_{val}]\ (1 \leq i \leq n) \\ R(v) & \text{si } p = p_R \\ 0 & \text{sinon} \end{cases}$$

qui est un marquage final après calcul de $R(v)$ par N_R. Nous avons donc une séquence nominale qui calcule $R(v)$ dans p_R et nous en concluons que le lemme est vrai pour n.

□

Avec le lemme 9, nous venons de mettre en évidence l'existence d'une séquence qui effectue le "bon" calcul dans p_R. Nous voyons à présent ce qu'il advient lors du franchissement d'une séquence non nominale. Nous considérons ici des marquages plus généraux que les précédents.

Un *représentant de* $v \in \mathbb{N}^n$ dans N_R est un marquage tel que pour tout M_i de N_R, la somme du contenu des places $M_i[p_{val}]$ et $M_i[\bar{p}_{val}]$ est égale à $v(i)$. De plus, toutes les autres places sont vides, exceptée $M_n[p_{on}]$ qui contient un jeton.

Définition 22 *Un marquage m est un représentant de v pour N_R si :*

$$\forall p \in P_R,\ m(p) = \begin{cases} 1 & si\ p = M_n[p_{on}] \\ x_i & si\ p = M_i[p_{val}]\ (1 \leq i \leq n) \\ y_i & si\ p = M_i[\bar{p}_{val}]\ (1 \leq i \leq n) \\ 0 & sinon \end{cases}$$

avec $\forall i\ (1 \leq i \leq n),\ x_i + y_i = v(i)$.

Remarque : *soit m_v un représentant de v pour N_R; par construction de N_R, à tout moment il n'y a qu'un jeton qui circule dans l'ensemble des places p_{on} et p_{off} de tous les M_i. Nous l'appelons le jeton de contrôle. Plus formellement, pour tout $m \in Acc(N_R, m_v)$, nous avons :*

$$\Sigma_{p \in P'} m(p) = 1$$

avec $P' \subseteq P_R$ et $P' = \cup_{i=0}^{n}\{M_i[p_{on}], M_i[p_{off}]\}$.

Nous allons montrer qu'à partir d'un représentant de v, toute exécution ne produit pas plus de $R(v)$ jetons dans p_R.

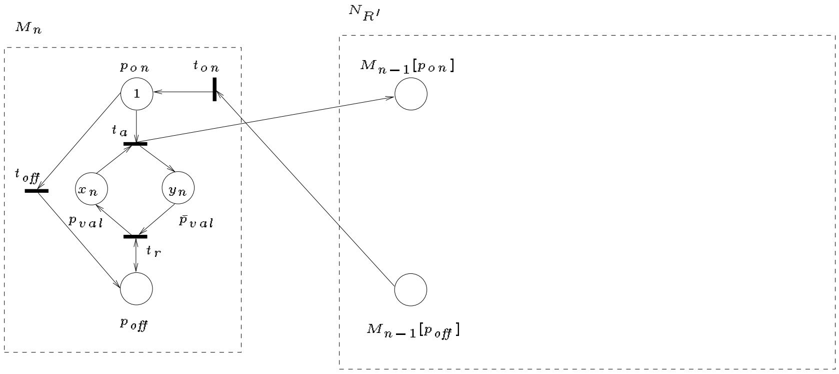
Lemme 10 *Tout marquage m accessible à partir d'un marquage m_v représentant de v pour N_R satisfait $m(p_R) \leq R(v)$.*

Preuve : nous donnons une preuve par induction sur n.

Considérons le cas où $n = 0$; alors $R = 1$ et le réseau N_R est le module M_0. Le seul marquage représentant d'un vecteur est $(1, 0, 0)$ à partir duquel la seule exécution possible est $(1,0,0) \stackrel{t_{\mathit{off}}}{\rightarrow} (0,1,1)$. Nous avons donc bien au plus un jeton dans p_R et le cas $n=0$ est donc vérifié.

Supposons que pour tout $k \leq n-1$, un réseau N_R (associé à k) ne puisse pas incrémenter p_R plus de $R(v)$ fois à partir d'un représentant de v. Montrons que cela est vrai pour n.

Soit N_R le réseau associé à n ($n \geq 1$) et soient $v \in \mathbb{N}^n$ et m_v un représentant de v. Définissons $v' \in \mathbb{N}^{n-1}$ tel que $\forall i \in [1..(n-1)]$, $v'(i) = v(i)$ et considérons le terme $R'(v') = \Pi_{i=1}^{n-1} v'(i)$. Nous avons $R(v) = v(n) * R'(v')$ et le réseau $N_{R'}$ (associé à $n-1$) est exactement le réseau N_R dont on a retiré le module M_n. Nous avons la situation suivante :



La seule transition de M_n susceptible d'entraîner l'ajout de jetons dans p_R est t_a car les autres transitions ont une action "locale" à M_n. Pour franchir $M_n[t_a]$, il faut décrémenter $M_n[p_{val}]$. La seule façon de "renflouer" $M_n[p_{val}]$ est de franchir $M_n[t_r]$, mais pour cela, il faut franchir $M_n[t_{\mathit{off}}]$ et dès que $M_n[t_{\mathit{off}}]$ est franchie, $M_n[t_a]$ n'est plus franchissable.

À partir de m_v, N_R pourra donc franchir au plus x_n fois $M_n[t_a]$ par une séquence de la forme :

$$\sigma = M_n[t_a]\,\sigma'_1\,M_n[t_{on}]\;M_n[t_a]\,\sigma'_2\,M_n[t_{on}]\;M_n[t_a]\,\sigma'_3\,M_n[t_{on}]\;\cdots$$

où $\sigma'_i \in {T_{R'}}^*$. Puisqu'il n'y a qu'un jeton de contrôle et puisque toutes les $M_i[p_{val}]$ et $M_i[\bar{p}_{val}]$ sont complémentaires, le réseau $N_{R'}$ est forcément dans un marquage représentant de v' avant chaque σ'_i. Par hypothèse d'induction, chaque σ'_i ajoute au plus $R'(v')$ jetons dans p_R.

Nous en déduisons que σ ajoute au plus $x_n * R'(v')$ jetons dans p_R, c'est-à-dire au plus $R(v)$ jetons puisque $x_n \leq v(n)$. Le lemme est donc vérifié pour n. $\square$

3.1.2 Évaluation d'un polynôme par un réseau de Petri

Considérons un ensemble P de k places et le polynôme $Q \in \mathbb{N}[P]$:

$$Q(p_1, .., p_k) = \Sigma_{j\in J} \lambda_j \Pi_{i=1}^{k} p_i^{n_{i_j}} = \Sigma_{j \in J} R_j$$

avec J un ensemble fini d'indices, λ_j, $n_{i_j} \in \mathbb{N}$ et $R_j = \lambda_j \Pi_{i=1}^{k} p_i^{n_{i_j}}$.

Pour construire le réseau N_{R_j} correspondant à chaque terme R_j, nous mettons en série $\Sigma_{i=1}^{k} n_{i_j} + 1$ modules, c'est-à-dire n_{1_j} modules pour p_1, n_{2_j} modules pour p_2, .., n_{k_j} modules pour p_k et enfin un module M_0 (associé à λ_j et non plus à 1). Nous dirons qu'un module implémentant une multiplication du contenu de la place p_i est *de type* M_i.

Exemple : pour $Q(p_1, .., p_k) = 3\, {p_1}^4\, p_2$, nous utilisons 4 modules de type M_1 (pour p_1), un module de type M_2 (pour p_2) et un module M_0 pour 3.

Pour construire le réseau N_Q qui évalue le polynôme Q dans p_Q, nous mettons en parallèle les $|J|$ réseaux N_{R_j}, dont on a fusionné les $|J|$ places p_{R_j} en une seule place p_Q (voir figure 3.1). Pour implémenter Q, on utilise en tout $\Sigma_{j \in J} \Sigma_{i=1}^{k} n_{i_j}$ modules, dont $\Sigma_{j \in J} n_{i_j}$ modules de type M_i.

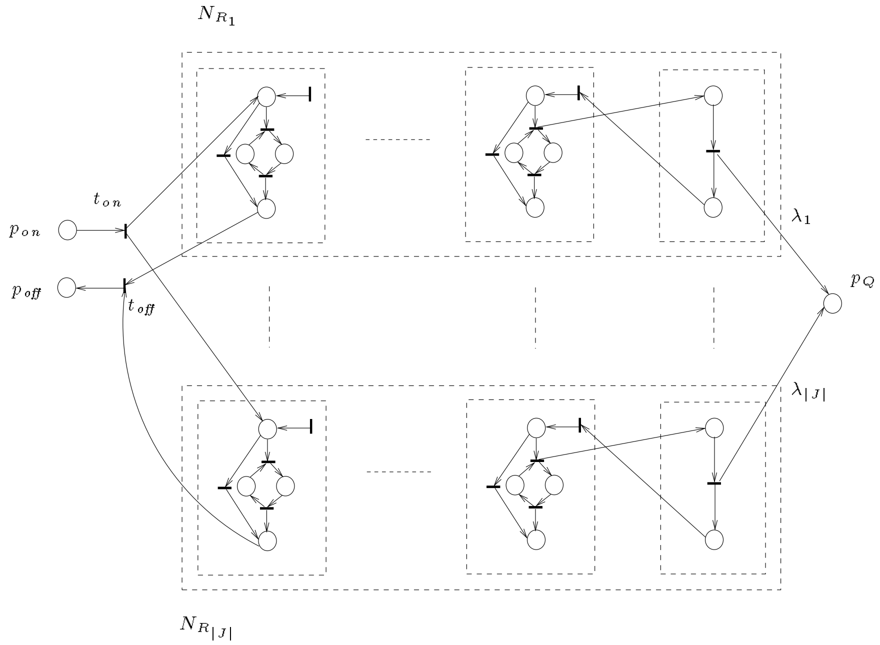


FIG. 3.1 – *Réseau N_Q pour évaluer le polynôme $Q = \Sigma_{j \in J} R_j$ dans p_Q.*

Soit $v \in \mathbb{N}^k$. Définissons $N_Q =< P_Q, T_Q, F_Q, m_{0Q} >$ pour l'évaluation de $Q(v)$:

- $P_Q = \{p_{on}, p_{off}, p_Q\} \cup_{j \in J} P_{R_j}$ où P_{R_j} est l'ensemble des places de R_j.
- $T_Q = \{t_{on}, t_{off}\} \cup_{j \in J} T_{R_j}$ où T_{R_j} est l'ensemble des transitions de R_j
- F_Q est décrite par la figure 3.1.
- $\forall p \in P_Q,\ m_{0Q}(p) = \begin{cases} 1 & \text{si } p = p_{on} \\ v(i) & \text{si } p \text{ est la place } p_{val} \text{ d'un module de type } M_i \\ 0 & \text{sinon} \end{cases}$

Nous généralisons la définition 20 d'un marquage final pour calculer N_Q dans p_Q.

Définition 23 *Le marquage* m_{f_R} *tel que :*

$$\forall p \in P_Q,\ m_{f_R}(p) = \begin{cases} 1 & si\ p = p_{off} \\ v(i) & si\ p\ est\ la\ place\ p_{val}\ d'un\ module\ de\ type\ M_i \\ Q(v) & si\ p = p_Q \\ 0 & sinon \end{cases}$$

est un marquage final après calcul de $Q(v)$ *dans* p_Q *par* N_Q*.*

Le lemme suivant est une généralisation directe du lemme 9 :

Lemme 11 *Soit* N_Q *le réseau calculant* $Q(v)$*. Il existe une séquence nominale pour calculer* $Q(v)$ *dans* p_Q*.*

Preuve : il suffit que N_Q franchisse t_{on}, que tous les réseaux N_{R_j} exécutent leur séquence nominale, puis que N_Q franchisse t_{off}. N_Q accèdera alors à un marquage final avec la valeur $Q(v)$ dans p_Q. □

Nous allons généraliser la définition 22. Un représentant de v pour N_Q est un marquage tel que pour tout module de type M_i, la somme du contenu des places p_{val} et $\bar{p}_{val}$ est $v(i)$. De plus, toutes les autres places sont vides, exceptée p_{on} qui contient un jeton.

Définition 24 *Le marquage* m *est un représentant de* v *pour* N_Q *si :*

$$\begin{cases} m(p_{on}) = 1 \\ Pour\ tout\ M\ de\ type\ M_i,\ m(M[p_{val}]) + m(M[\bar{p}_{val}]) = v(i) \\ m(p) = 0\ pour\ toutes\ les\ autres\ places \end{cases}$$

Le lemme suivant est une généralisation directe du lemme 10.

Lemme 12 *Soit* m_v *un représentant de* v *pour* N_Q*. On a :*

$$\forall \sigma \in {T_Q}^*,\ \forall m \in \mathbb{N}^n,\ m_v \xrightarrow{\sigma} m\ \Rightarrow\ m(p_Q) \leq Q(v)$$

Ainsi, par le lemme 11, à partir de m_{0Q}, si tous les N_{R_j} exécutent leur séquence nominale, p_Q recevra la valeur $Q(v)$. Par contre, par le lemme 12, si un réseau N_{R_j} exécute une séquence non nominale, alors p_Q recevra moins de $Q(v)$ jetons. On dit alors que $Q(v)$ est évalué dans *un sens faible.*

3.1.3 Simulation d'un réseau G-Post par un réseau de Petri

Soit $N =< P, T, F, m_0 >$ un G-Post à k places. Nous allons montrer comment déplier N en un réseau de Petri $N_D =< P_D, T_D, F_D, m_{0D} >$.

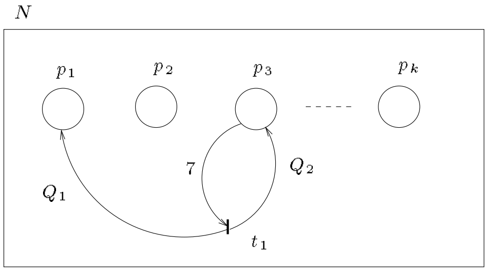
Nous ne décrirons pas tout de suite le réseau N_D. Annonçons simplement que N_D comprendra au moins deux ensembles de k places : P et P_{res}. L'ensemble P de N_D (que nous noterons $N_D[P]$) correspond à l'ensemble P de N (que nous noterons $N[P]$), et contient les valeurs sur lesquelles les polynômes doivent être évalués. Les places de P_{res} recevront les évaluations des polynômes : pour chaque i, la place p_{res_i} recevra les valeurs destinées à être ajoutées à la place p_i de $N[P]$.

Nous allons donner la description de N_D en deux étapes. D'abord, nous allons montrer comment simuler le franchissement d'une transition. Ensuite, nous allons montrer comment enchaîner les simulations successives des transitions de N.

Simulation du franchissement d'une transition

Pour simuler l'effet d'une transition de N, on associera à chaque transition t de T un réseau de Petri N_t dans N_D. Tous les réseaux N_t interagiront avec $N_D[P]$ (dont les places contiennent les valeurs sur lesquelles simuler la prochaine transition) et P_{res} (dont les places recevront les évaluations des polynômes).

Nous décrivons sur un exemple la construction d'un N_t. Considérons la transition t_1 suivante :



La partie N_{t_1} du réseau N_D qui simule l'effet de t_1 apparaît à la figure 3.2. Sur cette même figure apparaît l'ébauche d'un autre réseau N_t simulant l'effet d'une autre transition. Tout réseau N_t comprend les deux places deb_t et fin_t, et utilise les réseaux de Petri qui évaluent les polynômes étiquetant les post-arcs de t.

Exemple : pour N_{t_1}, nous avons le réseau N_{Q_1} qui évalue Q_1 et le réseau N_{Q_2} qui évalue Q_2.

Les places p_Q des réseaux qui évaluent les polynômes sont fusionnées avec les places de l'ensemble P_{res} de la façon suivante : un arc $(t; Q; p_j)$ évalue Q dans la place p_{res_j}.

Exemple : ici, N_{Q_1} évalue Q_1 dans p_{res_1} et N_{Q_2} évalue Q_2 dans p_{res_3}.

Soit $m \in \mathbb{N}^k$ (un marquage pour N) et $t \in T$. Définissons le marquage m_D qui correspond à m dans N_D et qui permettra de simuler le franchissement de t à partir de m :

Définition 25 *Le marquage m_D tel que :*

$$\forall p \in P_D,\ m_D(p) = \begin{cases} 1 & si\ p = deb_t \\ m(i) & si\ p = p_i\ (p_i \in N_D[P]) \\ m(i) & si\ p = M[p_{val}]\ pour\ tout\ M\ de\ type\ M_i \\ 0 & sinon \end{cases}$$

est un marquage de N_D pour simuler le franchissement de t à partir de m.

Définissons à présent le marquage final, attendu au terme d'une séquence nominale qui effectue la "bonne" simulation :

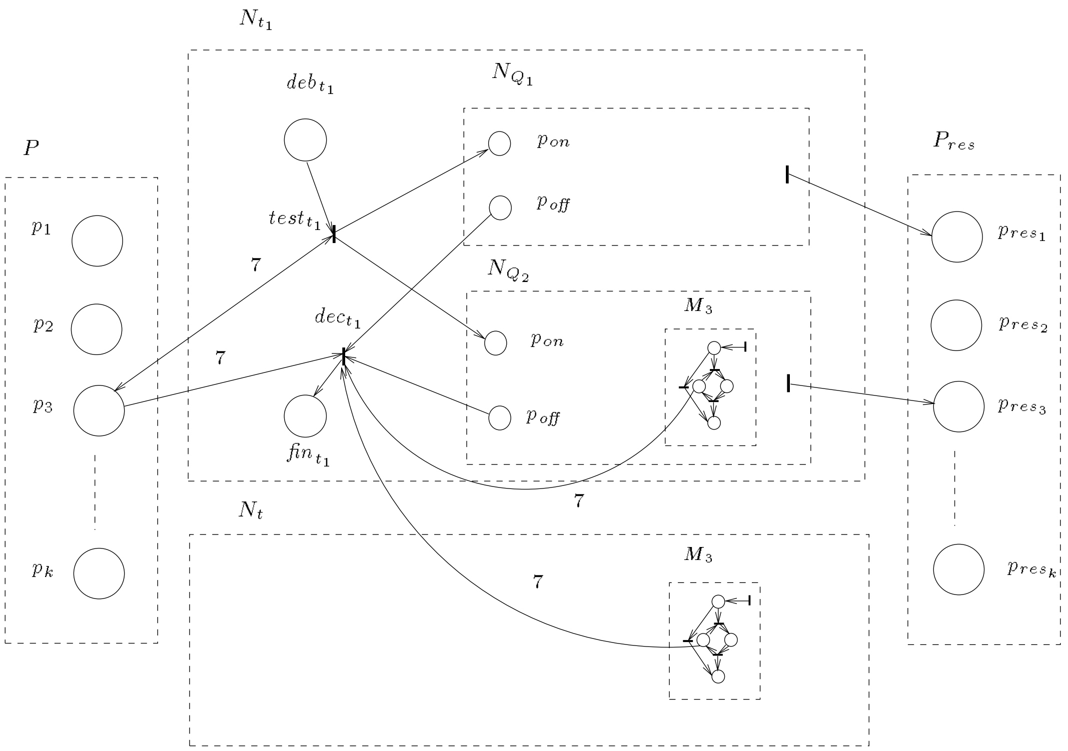


FIG. 3.2 – *La partie de N_D qui simule la transition t_1 de N.*

Définition 26 *Le marquage m_{f_D} tel que :*

$$\forall p \in P_D,\ m_{f_D}(p) = \begin{cases} 1 & si\ p = fin_t \\ m(i) - F(p_i, t) & si\ p = p_i\ (p_i \in N_D[P]) \\ m(i) - F(p_i, t) & si\ p = M[p_{val}]\ pour\ tout\ M\ de\ type\ M_i \\ Q(m)\ tel\ que\ Q = F(t, p_i) & si\ p = p_{res_i} \\ 0 & sinon \end{cases}$$

est un marquage final après simulation dans N_D du franchissement de t à partir de m.

Ainsi, nous décrivons un processus qui évalue dans P_{res} les sorties de t et qui met à jour les retraits engendrés par t dans $N_D[P]$ et dans les places p_{val} de tous les modules de N_D. Le processus commence lorsque deb_t est marquée et s'achève lorsque fin_t est marquée. Voyons les étapes successives :

- La simulation commence par le test de franchissement de t.

  Exemple : lorsque deb_{t_1} est marquée, la transition $test_{t_1}$ peut tester si p_3 contient au moins 7 jetons.

  Ce test valide la condition de franchissement de t, mais laisse intact le contenu des places de $N_D[P]$. Si t est franchissable dans N, alors deb_t est franchissable dans N_D.

- Lorsque $test_t$ est franchie, tout réseau N_Q construit pour un arc (t, Q, p_j) est dans un "bon" marquage initial pour calculer $Q(m)$ dans p_{res_j}.

  Exemple : franchir $test_{t_1}$ marque $N_{Q_1}[p_{on}]$ et $N_{Q_2}[p_{on}]$. Le réseau N_{Q_1} calculera $Q_1(m)$ dans p_{res_1} et le réseau N_{Q_2} calculera $Q_2(m)$ dans p_{res_3}.

- À la fin des évaluations des réseaux N_Q, lorsque toutes les places $N_Q[p_{off}]$ sont marquées, la transition dec_t est franchissable et met à jour les retraits de jetons entrainés par t : pour toute place $p_i \in N[P]$ en entrée de t, dec_t retire $F(p_i, t)$ jetons à $p_i \in N_D[P]$ et à toute place $M[p_{val}]$, pour tout M de type M_i (parmi **tous** les réseaux N_t de N_D). Franchir dec_t marque la place fin_t qui indique que le calcul des jetons produits pour t est achevé. Si dec_t n'est pas franchissable, alors N_D tombera dans un blocage (ceci peut se produire lorsque les réseaux N_Q n'ont pas effectué leur séquence nominale).

  Exemple : dec_{t_1} retire 7 jetons de p_3 et de toutes les places p_{val} des modules de type M_3 du réseau N_D au complet (ici, dans N_{t_1} et dans N_t).

Remarque : *il faut d'abord calculer le nombre de jetons produits, puis ensuite retirer les jetons des places d'entrée, car les évaluations des post-arcs sont faites sur le contenu des places a priori.*

Au terme d'une simulation de t, toute place p_{res_i} contiendra au plus $Q(m)$ jetons où $F(t, p_i) = Q$. Remarquons enfin qu'à tout moment, par construction de N_D, le contenu d'une place $p_i \in N_D[P]$ est égal à la somme du contenu des places $M[p_{val}]$ et $M[\bar{p}_{val}]$ pour tout M de type M_i.

Mise à jour de N_D et enchaînement des simulations

Voici à présent la description de $N_D =< P_D, T_D, F_D, m_{0D} >$ à partir d'un réseau G-Post $N =< P, T, F, m_0 >$:

- $P_D = P \cup P_{res} \cup \{p_{choix}\} \cup_{t \in T} P_t$
  où $P_{res} = \cup_{i=1}^{k} \{p_{res_i}\}$ et où P_t est l'ensemble des places du réseau N_t qui simule t.
- $T_D = T_{res} \cup_{t \in T} \{c_t, f_t\} \cup_{t \in T} T_t$
  où $T_{res} = \cup_{i=1}^{k} \{t_{res_i}\}$ et où T_t est l'ensemble des transitions du réseau N_t qui simule t.
- F_D est décrite à l'aide des figures 3.2 et 3.3.
- $\forall p \in P_D,\ m_{0D}(p) = \begin{cases} 1 & \text{si } p = p_{choix} \\ m_0(i) & \text{si } p = p_i \ (p_i \in N_D[P]) \\ m_0(i) & \text{si } p = M[p_{val}] \text{ avec } M \text{ de type } M_i \\ 0 & \text{sinon} \end{cases}$

Les places P_{res} sont des places de "réserve" pour P. Pour un marquage $m \in \mathbb{N}^{|P_D|}$, la place p_i possède à tout moment "virtuellement" $m(p_i) + m(p_{res_i})$ jetons. Il reste à faire en sorte que p_i récupère ses jetons, puis à lancer la prochaine simulation de transition. C'est la place p_{choix} qui contrôle cette partie qui est illustrée à la figure 3.3.

Lorsque p_{choix} est marquée, il y a deux possibilités :

- Le franchissement de T_{res_i} retire un jeton de p_{res_i} et ajoute un jeton dans $p_i \in N_D[P]$ et dans toute place $M[p_{val}]$ pour tout M de type M_i.

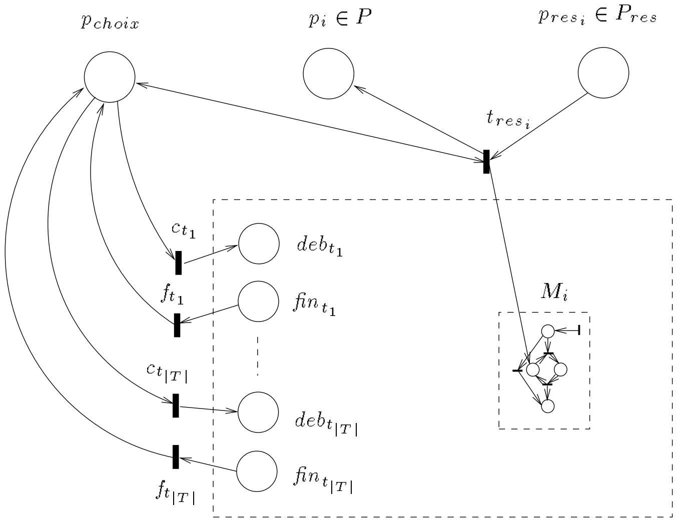


FIG. 3.3 – *Enchaînement des simulations de transitions dans N_D*

– Le franchissement de c_t lance la simulation de la transition t.

Les lemmes suivants étendent directement les lemmes et les propriétés précédentes :

Lemme 13

1. *Pour tout $m \in Acc(N, m_0)$, il existe $m_D \in Acc(N_D, m_{0D})$ tel que :*

$$\forall p \in P_D,\ m_{0D}(p) = \begin{cases} 1 & si\ p = p_{choix} \\ m(i) & si\ p = p_i\ (p_i \in P) \\ m(i) & si\ p = M[p_{val}]\ pour\ tout\ M\ de\ type\ M_i \\ 0 & sinon \end{cases}$$

2. *Pour tout $m_D \in Acc(N_D, m_{0D})$, il existe $m \in Acc(N, m_0)$ tel que :*

$$\forall i \in [1..k],\ m_D(p_i) + m_D(p_{res_i}) \leq m(p_i)$$

3. *Soit $m_D \in Acc(N_D, m_{0D})$. Pour tout module M de type M_i, nous avons l'invariant $m_D(M[p_{p_{val}}]) + m_D(M[\bar{p}_{val}]) = m_D(p_i)$.*

Le lemme 13.1 implique que si N n'est pas borné, alors N_D n'est pas borné. Le lemme 13.2 implique que si N est borné, alors les places de $N_D[P]$ et $N_D[P_{res}]$ sont bornées. Enfin, le lemme 13.3 implique que dans N_D, les places des modules M_i sont bornées si et seulement si les places de P et de P_{res} sont bornées. Les autres places étant bornées par la construction de N_D, nous obtenons le corollaire suivant :

Corollaire 14 *N est borné si et seulement si N_D est borné.*

Adaptation aux arcs Reset

La construction s'adapte aux arcs Reset et l'on peut déplier un réseau G-Post + Reset N_r vers un réseau de Petri + Reset N_{rD}. Il faut modifier la simulation d'une transition t de la façon suivante :

– Pour toute transition t, la transition c_t met toutes les places de P_{res} à zéro (si les simulations précédentes ont été accomplies correctement, ces places sont déjà vides).

- Pour tout arc de mise à zéro $(p_i; p_i; t)$, la transition dec_t met à zéro $p_i \in N_{rD}[P]$ et toutes places $M[p_{val}]$ et $M[\bar{p}_{val}]$ d'un module de type M_i.

3.1.4 Propriétés préservées

Soit N (resp. N_r) un réseau G-Post (resp. G-Post + Reset) à k places et N_D (resp. N_{rD}) le réseau de Petri (resp. réseau de Petri avec Reset) associé au dépliage de N (resp. N_r).

Par le lemme 13, nous savons que le dépliage préserve le caractère borné du réseau. Par contre, le dépliage peut entraîner des blocages dans N_D alors qu'il n'y en avait pas dans N; aussi, la propriété de blocage n'est pas préservée. La propriété d'accessibilité n'est pas préservée non plus, car l'accessibilité nécessite une correspondance exacte entre les marquages accessibles par N_D et les marquages accessibles par N. Notre dépliage propose une correspondance de "couverture" de N_D par N (qui suffit toutefois pour le caractère borné). De toute façon, un tel dépliage ne peut exister, car l'accessibilité est indécidable pour les G-Post, mais décidable pour les réseaux de Petri.

Voici les propriétés préservées :

Lemme 15

- *N (resp. N_r) est borné si et seulement si N_D (resp. N_{rD}) est borné.*
- *Soit $p_i \in P$ (resp. P_r) une place de N (resp. N_r); p_i est bornée dans N (resp. N_r) si et seulement si p_i est bornée dans $N_D[P]$ (resp. $N_{rD}[P]$).*
- *Soit $m \in \mathbb{N}^k$; m est couvert dans N (resp. N_r) si et seulement si m_D tel que :*
$$\forall p \in P_D,\ m_D(p) = \begin{cases} m(i) & si\ p = p_i \in N_D[P]\ (resp. N_{rD}[P]) \\ 0 & sinon \end{cases}$$
est couvert dans N_D (resp. N_{rD}).

Corollaire 16

- *Les propriétés de couverture, caractère borné du réseau et caractère borné d'une place sont décidables pour les réseaux G-Post.*
- *Il existe un dépliage des réseaux G-Post + Reset vers les réseaux de Petri + Reset qui préserve la couverture, le caractère borné du réseau et le caractère borné d'une place.*

Pour obtenir des résultats de décidabilité, cette approche en est une parmi d'autres. Nous verrons à la section 4.3 que le fait que les G-Post soient fortement structurés suffit pour que ces propriétés soient décidables.

3.1.5 Complexité pour les *Post Self-Modifying nets*

Le dépliage que nous venons de décrire est clairement exponentiel. En effet, pour un terme :

$$R = \lambda \Pi_{i=1}^{k} p_i^{n_i}$$

il faut construire n_i modules pour p_i, c'est-à-dire $\Omega(2^{|n_i|})$ modules où $|n_i| = \log n_i$ est la taille de n_i. Observons par contre ce que l'on peut faire dans le cas particulier des *Post Self-Modifying nets* (Post-SM) dont les polynômes sont de la forme :

$$c + \Sigma_{i=1}^{k} \lambda_i p_i \text{ avec } c,\ \lambda_i \in \mathbb{N}$$

Il y a au plus $|P|$ termes sur un arc et il y a au plus $|P|$ post-arcs pour une transition t. Il y a donc au plus $|P|^2$ termes par transition et puisqu'un terme n'utilise qu'un module M_i (qui est lui-même de taille constante), on en conclut que simuler une transition nécessite un nombre polynomial d'objets et non un nombre exponentiel.

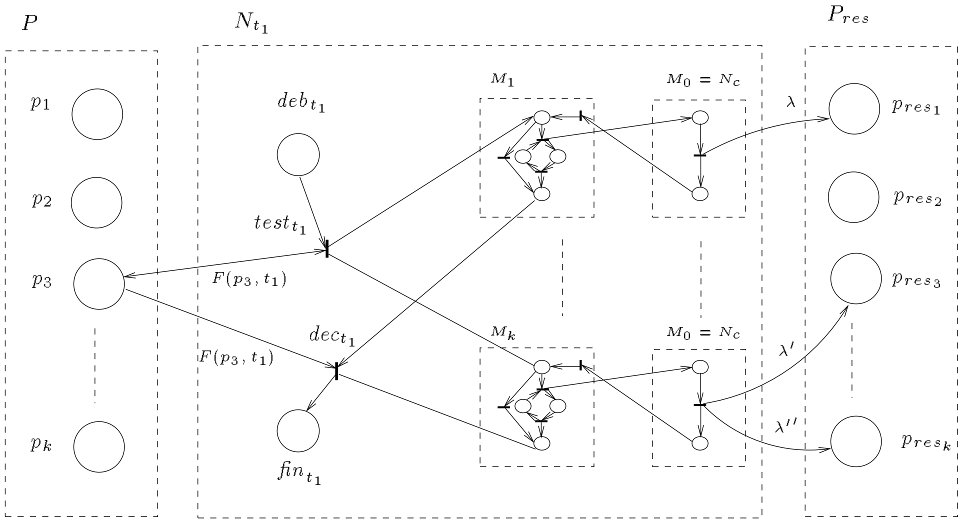


FIG. 3.4 – Post Self-Modifying *: simulation des effets des post-arcs de la transition t*

On peut encore améliorer la construction, en mettant à jour simultanément toutes les places de P_{res} qui sont incrémentées en fonction de la valeur d'une même place. Nous illustrons cette construction en figure 3.4 où, par exemple, p_{res_3} et p_{res_k} reçoivent en même temps les jetons que p_k produit pour p_3 et p_k. Pour simuler une transition t, un réseau de $O(|P|)$ places suffit.

La taille d'un *Post Self-Modifying net* est en $\Omega(|P| * |T| * \log V(N))$ où $V(N)$ est le plus grand entier parmi les exposants, les coefficients des polynômes et les valeurs du marquage initial. Voyons à présent la taille du réseau N_D en version compacte :

- *Places :* la simulation d'une transition utilise $O(|P|)$ places. Puisqu'il y a en tout $|T|$ transitions à simuler, nous comptons donc $|P|.|T|$ places pour les réseaux N_t. Par ailleurs, il y a $|P|$ places pour $N_D[P]$, $|P|$ places pour P_{res} et $O(|P| + |T|)$ places de contrôle pour enchaîner les simulations de transitions successives. En tout, N_D possède $O(|P|.|T|)$ places.

- *Transitions :* la simulation d'une transition utilise $O(|P|)$ transitions. Il y a $|T|$ transitions à simuler; nous comptons donc $O(|P|.|T|)$ transitions pour les réseaux N_t.

Par ailleurs, il y a $O(|T|)$ transitions de type dec_t ou $test_t$ et $O(|P| + |T|)$ transitions diverses pour assurer la gestion des contrôles. En tout, N_D possède $O(|P|.|T|)$ transitions.

– *Valuation :* la valuation maximale de N_D reste $V(N)$.

Au total, la taille de N_D est en $O(|P_D|.|T_D|.\log V(N_D))$, c'est-à-dire :

$$\mathtt{Taille}(N_D) \in O((|P|.|T|).(|P|.|T|).\log V(N))$$

Nous en déduisons donc que :

$$\mathtt{Taille}(N_D) \in O(|N|^2)$$

La construction est directe et ne nécessite aucun calcul qui fasse éclater cette borne. Le temps de dépliage d'un *Post Self-Modifying net* N en un réseau de Petri est donc en $O(|N|^2)$ (la construction est décrite aussi dans [DF97]).

Nous en déduisons que la complexité d'analyse des propriétés préservées par le dépliage est proche de celle des réseaux de Petri. Puisque la couverture et le caractère borné du réseau se résolvent en espace $2^{c_1.|N|.\log|N|}$ (pour une constante c_1) pour les réseaux de Petri [Rac78], nous obtenons le corollaire suivant :

Corollaire 17 *Pour les* Post Self-Modifying nets*, les problèmes de couverture et de caractère borné se résolvent en espace* $2^{c_2.|N|^2.\log|N|}$ *(pour une constante* c_2*).*

3.2 Normalisation des réseaux Reset et Transfert

Nous donnons ici une normalisation polynomiale en temps, applicable aussi bien aux réseaux de Petri, aux Reset ou aux Transfert. Rappelons qu'un réseau de Petri est *normalisé* si la fonction de flot est à valeurs dans $\{0, 1\}$ et le marquage initial est dans $\{0, 1\}^{|P|}$ (où P est l'ensemble des places). Cette définition s'étend directement aux réseaux de Petri avec Reset et Transfert.

Définition 27 *Un réseau de Petri avec arcs de Reset ou avec arcs de Transfert est normalisé, si tout arc de type réseau de Petri est étiqueté par* 0 *ou* 1 *et* $m_0 \in \{0, 1\}^{|P|}$.

Remarque : *les réseaux normalisés ne doivent pas être confondus avec les réseaux* 1-safe *dont tous les marquages accessibles sont* 1*-bornés. La notion de normalisation ne concerne que la description d'un réseau et ne donne aucune indication quant au comportement de ce réseau. Ainsi, au contraire des réseaux* 1*-safe, les réseaux normalisés peuvent être non bornés.*

L'intérêt pratique d'une normalisation tient au fait que les réseaux normalisés sont plus faciles à manipuler et à étudier que les réseaux généraux.

L'intérêt théorique de cette nouvelle normalisation est la borne de complexité, car les normalisations de réseaux de Petri connues jusqu'à présent étaient exponentielles. Pour les réseaux de Petri et les Reset, la complexité en temps de la normalisation est en $O(|N|^2)$, où

$|N|$ est la taille du réseau. De plus, lorsque nous avons $O(|T|) = O(|P|)$, le temps de normalisation est quasi-linéaire, c'est-à-dire en $O(\log V(N)\ |N|)$. Pour les réseaux Transfert, la normalisation est quasi-quadratique en temps. L'idée de base est d'enrayer l'explosion combinatoire en utilisant le *codage binaire* d'une valuation au lieu d'utiliser sa *valeur*.

À partir d'un réseau N (Petri, resp. Reset, resp. Transfert), nous construisons un réseau normalisé (Petri, resp. Reset, resp. Transfert) N' qui est λ-bisimilaire à N. Ceci signifie que N et N' ont des comportements très similaires, moyennant des franchissements de transitions invisibles. Nous choisissons la définition de la bisimulation par branchement [GW89], qui est la plus forte des bisimulations avec événements invisibles. Rappelons que nous notons le mot vide par λ.

Définition 28

- *Un réseau étiqueté (N, l) est composé d'un réseau N et d'une fonction d'étiquetage par le morphisme $l : T^* \to A^*$ où A est un alphabet fini.*
- *Deux réseaux étiquetés (N, l) et (N', l') avec $N =< P, T, F, m_0 >$ et $N' =< P', T', F', m'_0 >$ sont λ-bisimilaires s'il existe une relation $\mathcal{R} \subseteq \mathbb{N}^{|P|} \times \mathbb{N}^{|P'|}$ telle que :*

  1. $m_0\, \mathcal{R}\, m'_0$
  2. *Pour tous $m_1 \in \mathbb{N}^{|P|}$, $m'_1 \in \mathbb{N}^{|P'|}$ tels que $m_1\, \mathcal{R}\, m'_1$ et pour toute transition $t \in T$ tels que $\exists\, m_2 \in \mathbb{N}^{|P|}$, $m_1 \xrightarrow{t} m_2$:*
     - *Soit $l(t) = \lambda$ et $m_2\, \mathcal{R}\, m'_1$,*
     - *Soit $\exists\, u \in T'^*$, $t' \in T'$, m'_2 et $m'_3 \in \mathbb{N}^{|P'|}$ tels que $l'(u) = \lambda$, $l'(t') = l(t)$ et $m'_1 \xrightarrow{u} m'_2 \xrightarrow{t'} m'_3$ avec $(m_1\, \mathcal{R}\, m'_2)$ et $(m_2\, \mathcal{R}\, m'_3)$.*
  3. *Pour tous $m_1 \in \mathbb{N}^{|P|}$, $m'_1 \in \mathbb{N}^{|P'|}$ tels que $m_1\, \mathcal{R}\, m'_1$ et pour toute transition $t' \in T'$ tels que $\exists\, m'_2 \in \mathbb{N}^{|P'|}$, $m'_1 \xrightarrow{t'} m'_2$:*
     - *Soit $l'(t') = \lambda$ et $m_1\, \mathcal{R}\, m'_2$,*
     - *Soit $\exists\, u \in T^*$, $t \in T$, m_2 et $m_3 \in \mathbb{N}^{|P|}$ tels que $l(u) = \lambda$, $l(t) = l'(t')$ et $m_1 \xrightarrow{u} m_2 \xrightarrow{t} m_3$ avec $(m_2\, \mathcal{R}\, m'_1)$ et $(m_3\, \mathcal{R}\, m'_2)$.*

Autrement dit, deux réseaux sont λ-*bisimilaires* si, quoi que fasse l'un, l'autre peut l'imiter, avec toutefois la possibilité d'utiliser quelques transitions invisibles.

Notre normalisation préserve les propriétés de couverture, caractères bornés (réseau, place et structurellement borné), accessibilité et t-vivacité.

Nous commençons par faire le tour d'horizon des normalisations déjà connues pour les réseaux de Petri. Par la suite, nous donnerons notre normalisation, sa preuve d'exactitude et sa complexité.

Définition 29 *Soient un réseau de Petri $N =< P, T, F, m_0 >$ et $p \in P$, nous avons :*

$$V(p) = \mathtt{max}(m_0(p), \mathtt{max}_{t \in T}(F(p, t), F(t, p)))$$

Nous définissons $V(p)$ comme étant la valeur maximale parmi $m_0(p)$ et les valuations des arcs adjacents à p.

3.2.1 Normalisations connues pour les réseaux de Petri

Hack [Hac76, Pet81] propose une normalisation d'un réseau de Petri *par anneau*. Chaque place p est transformée en un anneau de $V(p)$ places (voir figure 3.5) dont $m_0(p)$ places sont marquées d'un jeton (les autres places sont vides). Tout arc adjacent à p valué par v (≥ 2) est transformé en v arcs valués par 1, dont chacun touche une place différente, arbitraire, de l'anneau. Les jetons sont libres de circuler le long de l'anneau et peuvent à tout moment rejoindre les places en entrée d'une transition, permettant ainsi son franchissement si celui-ci est possible. Cette normalisation est exponentielle puisque $V(p)$ est codé sur $\log(V(p))$ bits alors que l'anneau construit pour p comporte $\Omega(2^{\log(V(p))})$ places. À tout moment, la somme des jetons de l'anneau correspond à un marquage accessible de p dans le réseau de départ.

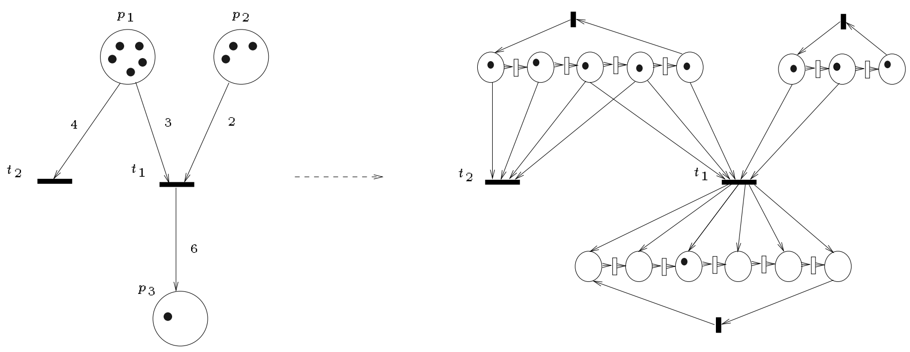


Fig. 3.5 – *Normalisation par anneau de Hack*

Une seconde approche est la normalisation *en profondeur*. L'idée principale est de créer v niveaux pour une valuation d'arc v, le $i^{\text{ème}}$ niveau correspondant au $i^{\text{ème}}$ jeton à retirer ou à ajouter. Les niveaux sont activés séquentiellement, simulant ainsi l'effet originel de l'arc. Kasai et Miller [KM82] proposent une normalisation "simulation" axée sur la préservation de la vivacité, des blocages et de la terminaison. Schwer [Sch85] a pour objectif la préservation de la régularité et l'algébricité du langage.

Une autre approche, *en largeur*, adopte une stratégie avec retrait ou ajout des jetons en un seul coup. Sifakis [Sif79] associe à chaque place p une file d'une capacité de $V(p)$ jetons, implémentée par $V(p)$ places. Lorsque vient le temps de retirer v jetons à p, les v premiers jetons de la file sont prélevés en parallèle à l'aide de v arcs, chacun valué à 1. La file est ensuite mise à jour. Cette construction préserve la persistence. Pelz [Pel92] donne la seule normalisation, parmi celles de cette section, qui préserve le langage sans utiliser de λ-transitions; par contre, sa complexité est doublement exponentielle. L'idée principale est de transformer d'abord toute place p en un complexe de $V(p)$ places (on obtient une première exponentielle dans la complexité). Ensuite, tout arc adjacent à p, valué par v, est transformé en autant de transitions que de façons de prendre v parmi $V(p)$: chacune de ces nouvelles transitions est munie de v arcs, chacun étant valué à 1 et touchant une place différente (on obtient ici la seconde exponentielle dans la complexité).

Kasai et Miller, toujours en [KM82], proposent une normalisation *arborescente*, dite "émulation". Le principe consiste tout d'abord à diviser par deux le marquage initial d'une place p, transformant ainsi p en deux nouvelles places. Selon le cas, une place `pair` ou une place `impair` est marquée afin de rendre compte de l'état courant de la place p. Ensuite, tout arc adjacent à p est séparé en deux arcs, chacun valué de la moitié de sa valeur et touchant une des deux places. Cette transformation d'arcs est dupliquée pour prendre en compte le cas `pair` et le cas `impair`. La construction est itérée jusqu'à la normalisation complète du réseau. La couverture, le caractère borné, l'accessibilité, la vivacité et les blocages sont préservés.

Les normalisations précédemment décrites ont toutes une borne inférieure de complexité en temps exponentiel. Notre normalisation [DF98] par *décomposition binaire* a une complexité-temps polynomiale. L'idée principale est d'utiliser le codage en base 2 des valuations et d'engendrer $O(\log V(p))$ objets (places ou transitions) pour une place p, au lieu de $\Omega(V(p))$. Mentionnons que cette idée peut s'appliquer à certaines des stratégies de normalisation décrites ci-dessus et améliorer leur complexité.

3.2.2 Normalisation de v-valué vers 2-valué.

Nous partons d'un réseau de Petri v-valué, c'est-à-dire dont la plus grande valuation est v, avec v arbitraire, et nous présentons la normalisation en deux temps. Dans un premier temps, nous construisons un réseau 2-valué, sur lequel nous montrons la préservation des propriétés. Nous avons choisi de faire les preuves sur le réseau 2-valué afin d'alléger les notations. Dans un second temps, essentiellement technique, nous éliminons les valuations à 2 et nous obtenons le réseau normalisé.

Au cours de notre construction, nous utilisons la notion de *fusion par transitions*:

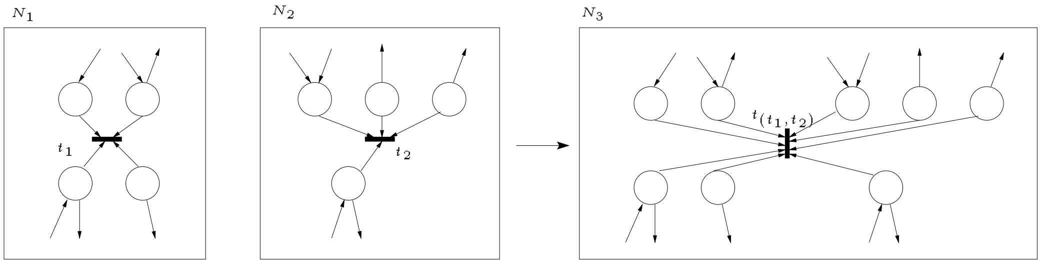


FIG. 3.6 – *Fusion de deux réseaux sur t_1 et t_2: $(N_1 \underset{(t_1,t_2)}{\|} N_2) = N_3$*

Définition 30 *Soient $N_1 =< P_1, T_1, F_1, m_{01} >$ et $N_2 =< P_2, T_2, F_2, m_{02} >$ deux réseaux.*

La fusion par transitions de N_1 et N_2 sur les deux transitions $t_1 \in T_1$ et $t_2 \in T_2$, notée $(N_1 \underset{(t_1,t_2)}{\|} N_2)$, produit un nouveau réseau $N_3 =< P_3, T_3, F_3, m_{03} >$ défini par :

- $P_3 = P_1 \cup P_2$
- $T_3 = (T_1 \setminus \{t_1\}) \cup (T_2 \setminus \{t_2\}) \cup \{t_{(t_1,t_2)}\}$
- *F_3 est construit directement à partir de F_1 et F_2. Les transitions t_1 et t_2 sont remplacées par une transition $t_{(t_1,t_2)}$ (voir figure 3.6).*

- $\forall p \in P_3,\ m_{0_3}(p) \ = \ \begin{cases} m_{01}(p) & si\ p \in P_1 \\ m_{02}(p) & si\ p \in P_2 \end{cases}$

Remarque : *la fusion s'étend aux ensembles de couples de transitions et aux réseaux étiquetés.*

Nous décrivons notre construction sur les réseaux de Petri et nous l'adaptons ensuite aux Reset et aux Transfert.

Définition 31 *Soit un réseau de Petri général $N = < P, T, F, m_0 >$.*

- *$K(p)$ est le nombre de bits nécessaires pour coder $V(p)$:*

$$K(p) = \lfloor \log(V(p)) \rfloor + 1$$

- *La fonction $B_k(n)$, avec $k \in \mathbb{N}$, prend en entrée un entier n (tel que $\log(n+1) \leq k$) et retourne un vecteur de bits de dimension k, correspondant au codage binaire de n complété à gauche par des 0 si nécessaire.*

Nous indexons le vecteur retourné par B_k de façon descendante partant de $k-1$, et non de façon ascendante comme c'est le cas habituellement. Ainsi, nous considérons l'intervalle de valeurs $[k-1 \,..\, 0]$, ceci afin de conserver la relation entre une position et une puissance de deux lors d'une conversion de binaire en décimal.

Exemple: $23 = 10111_2$, et donc $b = B_7(23) = (0,0,1,0,1,1,1)$ avec $b(0) = 1$, $b(1) = 1$, $b(2) = 1$, $\cdots$, $b(6) = 0$.

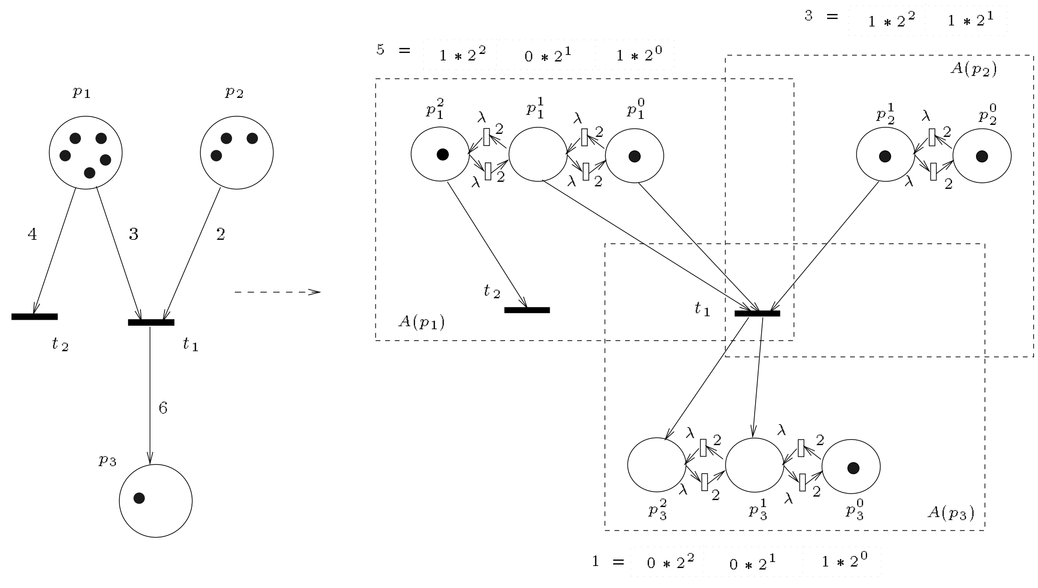


FIG. 3.7 – *Construction de v-valué vers 2-valué*

Nous partons d'un réseau de Petri $N = < P, T, F, m_0 >$ et nous construisons un réseau étiqueté (N', l') avec $N' = < P', T', F', m'_0 >$ 2-valué et tel que $l' : T^* \rightarrow T'^*$. La construction commence avec la transformation de chaque place p en un réseau $A(p)$ (pour "Associé à p"), dont la taille est en $O(|T| * K(p))$. La construction s'achève avec l'assemblage par fusion des réseaux $A(p)$ pour former le réseau N'. Nous illustrons les deux étapes avec un exemple (figure 3.7).

Construction des réseaux A(p)

Sur la figure 3.7, les réseaux $A(p)$ sont encadrés d'une ligne pointillée. Le réseau $A(p)$ possède $K(p)$ places, $O(K(p))$ transitions invisibles et $|^\bullet p \cup p^\bullet|$ transitions visibles.

Détaillons la construction de $A(p) =< P_{A(p)}, T_{A(p)}, F_{A(p)}, m_{A(p)} >$:

1. On calcule $K(p)$.

   Exemple : $K(p_1) = 3$ puisque :

   $$\mathtt{max}(m_0(p_1), F(p_1, t_1), F(p_1, t_2)) = \mathtt{max}(5, 3, 4) = 5$$

   et $K(p_1) = \lfloor \log 5 \rfloor + 1 = 3$.

2. *Places :* les $K(p)$ places de $A(p)$ sont étiquetées de p^0 à $p^{K(p)-1}$. Elles sont alignées de droite à gauche sur la figure.

   Exemple : nous avons $P_{A(p_1)} = \{p_1^2, p_1^1, p_1^0\}$.

3. *Marquage initial :* le réseau $A(p)$ est initialisé par $B_{K(p)}(m_0(p))$, c'est-à-dire qu'on projette tel quel le codage binaire de $m_0(p)$ sur $P_{A(p)}$.

   Exemple : pour $A(p_3)$, la place p_3^0 reçoit 1 jeton, mais p_3^1 et p_3^2 sont vides (en effet, $m_0(p_3) = 1 = 001_2$ est codé sur 3 bits ici, car $K(p_3) = 3$).

4. *Transitions invisibles :* les transitions invisibles de $A(p)$ sont étiquetées par λ sur le dessin. Il y a deux types de transitions invisibles. Nous appelons transitions de type D celles qui retirent 2 jetons de p^i pour ajouter 1 jeton dans p^{i+1} $(K(p)-1 > i \geq 0)$. Si l'on se réfère à la figure, ces transitions *diffusent* les jetons dans un mouvement de droite à gauche. Nous appelons transitions de type G celles qui retirent 1 jeton de p^{i+1} pour ajouter 2 jetons dans p^i $(K(p)-1 > i \geq 0)$. Si l'on se réfère à la figure, ces transitions *groupent* les jetons dans un mouvement de gauche à droite.

5. *Transitions visibles :* les transitions visibles correspondent aux transitions de N et en conservent l'étiquette d'origine. Ce sont elles qui seront impliquées dans la fusion de tous les réseaux $A(p)$.

   Pour tout couple $(t, p) \in (T \times P)$, on calcule $b_{t,p} = B_{K(p)}(F(t, p))$. Ensuite, pour tout $b_{t,p}(i) = 1$, on ajoute l'arc $(t; 1; p^i)$ dans $A(p)$.

   Exemple : $F(t_1, p_3) = 6 = 110_2$. Le vecteur $b_{t_1,p_3} = (1, 1, 0)$ et donne lieu à la création de deux arcs valués à 1 dans $A(p_3)$ soient $(t_1; 1; p_3^2)$ et $(t_1; 1; p_3^1)$.

   Les couples $(p, t) \in (P \times T)$ sont traités de façon semblable : on calcule le vecteur $b_{p,t} = B_{K(p)}(F(p, t))$ et on ajoute autant d'arcs $(p^i; 1; t)$ que de positions $b_{p,t}(i) = 1$.

   Exemple : $F(p_1, t_2) = 4$ donne lieu à la création d'un arc $(p_1^2; 1; t_2)$ dans $A(p_1)$ puisque $4 = 100_2$.

Assemblage des réseaux $A(p)$.

La fusion de tous les sous-réseaux est opérée sur les transitions visibles. Nous obtenons :

$$N' = [\,(\,A(p_1)\,\underset{(t_1,t_2)}{||}\,A(p_2)\,)\,\underset{(t_1,t_2)}{||}\,A(p_3)\,]$$

Propriétés des réseaux $A(p)$

Nous énonçons quelques définitions et propriétés des réseaux $A(p)$. Un marquage d'un réseau $A(p)$ à k places sera "interprété" par la fonction φ_k définie comme suit :

Définition 32 *Soient m_1 et m_2 deux marquages dans $\mathbb{N}^k$ ($k \geq 1$) que nous indexons de façon descendante à partir de $k-1$, c'est-à-dire par l'intervalle de valeurs $[k-1 \,..\, 0]$.*

- *La fonction $\varphi_k \,:\, \mathbb{N}^k \rightarrow \mathbb{N}$ est telle que :*

$$\varphi_k(m) = \Sigma_{i=0}^{k-1} 2^i * m(i)$$

- *On pose $m_1 \gg m_2 \Leftrightarrow \varphi_k(m_1) \geq \varphi_k(m_2)$.*
- *On pose $m_1 \sim m_2 \Leftrightarrow m_1 \gg m_2$ et $m_2 \gg m_1$.*

La fonction φ_k appliquée sur un vecteur dans $\{0,1\}^k$ retourne la valeur en base dix qu'on associe à un registre de k bits. Ici, la notion est étendue à un vecteur d'entiers.

Lemme 18 *Soient m_1 et $m_2 \in \mathbb{N}^k$, $k \in \mathbb{N}$, avec $m_1 \geq m_2$.*

1. *$\varphi_k(m_1 + m_2) = \varphi_k(m_1) + \varphi_k(m_2)$.*
2. *$\varphi_k(m_1 - m_2) = \varphi_k(m_1) - \varphi_k(m_2)$.*

Preuve :

1. Immédiat puisque :

$$\Sigma_{i=0}^{k-1}\, 2^i[m_1(i) + m_2(i)] \;=\; \Sigma_{i=0}^{k-1}\, 2^i m_1(i) \;+ \Sigma_{i=0}^{k-1}\, 2^i m_2(i)$$

2. Immédiat de la même façon :

$$\Sigma_{i=0}^{k-1}\, 2^i[m_1(i) - m_2(i)] \;=\; \Sigma_{i=0}^{k-1}\, 2^i m_1(i) \;-\; \Sigma_{i=0}^{k-1}\, 2^i m_2(i)$$

$\square$

Le lemme qui suit signifie que le marquage d'un réseau $A(p)$ ne change pas de classe d'équivalence par le franchissement des transitions invisibles. Par ailleurs, si un marquage $m_1 \gg m_2$, alors il est possible, à l'aide des transitions invisibles, de couvrir m_2 à partir de m_1.

Lemme 19 *Soient un réseau $A(p)$ à k places et l' sa fonction d'étiquetage, et soient $m_1,\ m_2 \in \mathbb{N}^k$.*

1. *$m_1 \xrightarrow{\sigma} m_2$ avec $l'(\sigma) = \lambda \Rightarrow m_1 \sim m_2$.*

2. $m_1 \gg m_2 \;\Rightarrow\; \exists \sigma, l'(\sigma) = \lambda$ *et* $\exists\, m_3 \in \mathbb{N}^k$ *tels que* $m_1 \xrightarrow{\sigma} m_3$ *et* $m_3 \geq m_2$*.*

Preuve :

1. Il suffit de prouver que $m_1 \xrightarrow{t} m_2 \Rightarrow m_1 \sim m_2$, pour tout t telle que $l'(t) = \lambda$. Il y a deux cas :
   (a) t est de type G et retire 2 jetons de p^j pour ajouter 1 jeton dans p^{j+1}. Nous avons :
   $$\varphi_k(m_2) = (\Sigma_{i=0}^{k-1} 2^i[m_1(i)]) - 2^j * 2 + 2^{j+1} = \varphi_k(m_1)$$
   (b) t est de type D et retire un jeton de p^{j+1} pour ajouter un jeton dans p^j. Nous avons :
   $$\varphi_k(m_2) = (\Sigma_{i=0}^{k-1} 2^i[m_1(i)]) - 2^{j+1} + 2^j * 2 = \varphi_k(m_1)$$
2. À partir d'un marquage $m \in \mathbb{N}^k$ tel que $\varphi_k(m) = i$, il est toujours possible d'accéder au marquage $(0,..,0,i)$, qui est le marquage contenant i dans sa composante la plus à droite et 0 dans toutes les autres composantes. En effet, il suffit de franchir tant que possible les transitions de type G.

   Par les transitions de type G, nous avons $m_1 \xrightarrow{\sigma_1} (0,..,0,\varphi_k(m_1))$ et nous avons $m_2 \xrightarrow{\sigma_2} (0,..,0,\varphi_k(m_2))$. Or, les transitions de type G sont réversibles par celles de type D, et ceci rend possible $m_2 \xrightarrow{\sigma_2} (0,..,0,\varphi_k(m_2)) \xrightarrow{\sigma_2^r} m_2$ où σ_2 est une suite de transitions de type G (invisibles) et σ_2^r est sa suite inverse, composée de transitions de type D (invisibles). Enfin, puisque $\varphi_k(m_1) \geq \varphi_k(m_2)$, nous en concluons qu'il existe $m_3 \in \mathbb{N}^k$ tel que $m_1 \xrightarrow{\sigma_1} (0,..,0,\varphi_k(m_1)) \xrightarrow{\sigma_2^r} m_3$ avec $m_3 \geq m_2$ et $l'(\sigma_1\sigma_2^r) = \lambda$.
   □

Corollaire 20 *D'un marquage donné, on peut accéder à tout marquage appartenant à la même classe d'équivalence et ce, en n'utilisant que des transitions invisibles.*

3.2.3 λ-bisimulation du réseau v-valué et du réseau 2-valué

Nous définissons d'abord la relation $\mathcal{R}$ qui est une extension de la fonction φ au réseau N' entier. Ensuite, nous prouvons que les deux réseaux N et N' sont λ-bisimilaires par $\mathcal{R}$.

Soit $N =< P, T, F, m_0 >$ un réseau de Petri et soit (N', l'), avec $N' =< P', T', F', m'_0 >$, le réseau étiqueté par l' et 2-valué associé :

Définition 33

- *Soient* $p \in P$ *et* $m' \in \mathbb{N}^{|P'|}$*. Nous notons* $m'[A(p)]$ *le sous-marquage de* m' *correspondant au réseau* $A(p)$*.*
- *Soient* $m \in \mathbb{N}^{|P|}$ *et* $m' \in \mathbb{N}^{|P'|}$*. La relation* $\mathcal{R}$ *est telle que :*

$$m\,\mathcal{R}\,m' \Leftrightarrow \forall p \in P,\ \varphi_{K(p)}(m'[A(p)]) = m(p)$$

Exemple : (figure 3.7) le marquage initial de N' est $m'_0 = (1,0,1,1,1,0,0,1)$ avec, plus particulièrement, $m'_0[A(p_1)] = (1,0,1)$, $m'_0[A(p_2)] = (1,1)$ et $m'_0[A(p_3)] = (0,0,1)$. Nous avons $(5,3,1)\,\mathcal{R}\,m'_0$, puisque $\varphi_3((1,0,1)) = 5$, $\varphi_2((1,1)) = 3$ et $\varphi_3((0,0,1)) = 1$ (où $(5,3,1)$ est le marquage initial de N).

Remarque : *par construction des $A(p)$, nous avons toujours $m_0\,\mathcal{R}\,m'_0$.*

Nous prouvons maintenant les deux propositions principales. Tout d'abord, tout ce que peut faire N peut être imité par N', tout en conservant la relation $\mathcal{R}$.

Proposition 21 *Soient $m_1, m_2 \in \mathbb{N}^{|P|}$, $m'_1 \in \mathbb{N}^{|P'|}$ et $t \in T$ tels que $(m_1\,\mathcal{R}\,m'_1)$ et $m_1 \xrightarrow{t} m_2$.*

Il existe $u \in T'^*$, $t' \in T'$ et $m'_2, m'_3 \in \mathbb{N}^{|P|'}$ tels que $l'(u) = \lambda$, $l'(t') = t$ et $m'_1 \xrightarrow{u} m'_2 \xrightarrow{t'} m'_3$ avec $(m_1\,\mathcal{R}\,m'_2)$ et $(m_2\,\mathcal{R}\,m'_3)$.

Preuve : puisque $m_1 \xrightarrow{t} m_2$, alors $\forall p \in P$, $m_1(p) \geq F(p,t)$. Par la relation $\mathcal{R}$ nous savons que :

$$\forall\, p \in P,\ \varphi_k(m'_1[A(p)]) \geq F(p,t) \text{ donc } m'_1[A(p)] \gg B_{K(p)}F(p,t)$$

Par le lemme 19, nous savons que $\forall p \in P$, $\exists\, \sigma_p \in T'^*, \exists\, m'_p \in \mathbb{N}^{|K(p)|}$ tels que :

$$l'(\sigma_p) = \lambda \text{ et } m'_1[A(p)] \xrightarrow{\sigma_p} m'_p \geq B_{K(p)}F(p,t)$$

Considérons le marquage $m'_2 \in \mathbb{N}^{|P'|}$ tel que $m'_1 \xrightarrow{\sigma_{p_1}..\sigma_{p_{|P|}}} m'_2$. Nous avons:

$$\forall p \in P,\ m'_2[A(p)] \sim m'_1[A(p)]$$

et donc $m_1\,\mathcal{R} m'_2$. De plus :

$$\forall p \in P,\ m'_2[A(p)] = m'_p \geq B_{K(p)}F(p,t)$$

Considérons à présent (l'unique) $t' \in T'$ telle que $l'(t') = t$. Par construction de N', nous avons $m'_2 \xrightarrow{t'}$. Donc, $\exists\, m'_3 \in \mathbb{N}^{|P'|}$ tel que $m'_2 \xrightarrow{t'} m'_3$ et :

$$\forall p \in P,\ m'_3[A(p)] = m'_2[A(p)] - B_{K(p)}F(p,t) + B_{K(p)}F(t,p)$$

Nous avons, $\forall p \in P$, $m_2(p) = m_1(p) - F(p,t) + F(t,p)$ et donc, par le lemme 18 :

$$\forall p \in P,\ m_2(p) = \varphi_{K(p)}(m'_3[A(p)])$$

Nous en déduisons que $(m_2\,\mathcal{R}\,m'_3)$.

Nous avons donc bien : $m'_1 \xrightarrow{\sigma_{p_1}..\sigma_{p_{|P|}}} m'_2 \xrightarrow{t'} m'_3$ avec $l'(\sigma_{p_1}..\sigma_{p_{|P|}}) = \lambda$, $l(t') = t$, $(m_1\,\mathcal{R}\,m'_2)$ et $(m_2\,\mathcal{R}\,m'_3)$, ce qui termine la preuve de la proposition 21. □

Exemple : (figure 3.7) dans N, nous avons $(5,3,1) \xrightarrow{t_1}$ puisque p_1 contient au moins 3 jetons et p_2 contient au moins 2 jetons. Dans le réseau N' par contre, la transition qui lui correspond, étiquetée aussi par t_1, n'est pas immédiatement franchissable, puisque la place p_1^1 de $A(p_1)$ est vide. Cependant, il suffit de franchir d'abord la transition de type G entre p_1^2 et p_1^1 pour incrémenter p_1^1 de deux jetons, afin de permettre le franchissement de t_1 dans N'. Dans N, nous obtenons la séquence suivante :

$$m_0 = (5,3,1) \xrightarrow{t_1} (2,1,7) = m$$

Dans N', la séquence correspondante est :

$$m'_0 = (1,0,1,1,1,0,0,1) \xrightarrow{\lambda} (0,2,1,1,1,0,0,1) \xrightarrow{t_1} (0,1,0,0,1,1,1,1) = m'$$

avec $m'[A(p_1)] = (0,1,0)$, $m'[A(p_2)] = (0,1)$ et $m'[A(p_3)] = (1,1,1)$. Nous constatons bien que $(m\,\mathcal{R}\,m')$, puisque $\varphi_2((0,1,0)) = 2$, $\varphi_2((0,1)) = 1$ et $\varphi_3((1,1,1)) = 7$.

Nous prouvons maintenant la seconde proposition. Tout ce que peut faire N' peut être imité par N, tout en conservant la relation $\mathcal{R}$.

Proposition 22 *Soient $m_1 \in \mathbb{N}^{|P|}$, $m'_1, m'_2 \in \mathbb{N}^{|P'|}$ et $t' \in T'$ tels que $(m_1\,\mathcal{R}\,m'_1)$ et $m'_1 \xrightarrow{t'} m'_2$, nous avons :*

- *soit $l'(t') = \lambda$ et $m_1\,\mathcal{R}\,m'_2$*
- *soit $l'(t') = t \in T$ et $\exists\, m_2 \in \mathbb{N}^{|P|}$ tel que $m_1 \xrightarrow{t} m_2$ avec $m_2\,\mathcal{R}\,m'_2$*

Preuve : voyons les deux cas :

- Si $l'(t') = \lambda$, alors par le lemme 19, nous savons que

  $$\forall p \in P,\ m'_2[A(p)] \sim m'_1[A(p)]$$

  et nous en concluons que $m_1\,\mathcal{R}\,m'_2$.

- Si t' est telle que $l'(t') = t$ avec $t \in T$ alors, puisque $m'_1 \xrightarrow{t'} m'_2$, nous avons :

  $$\forall p \in P,\ m'_1[A(p)] \geq B_{K(p)}F(p,t)$$

  Par la relation $\mathcal{R}$, nous savons que :

  $$\forall\, p \in P,\ m_1(p) = \varphi_{K(p)} m'_1[A(p)] \geq F(p,t)$$

  La transition t est donc franchissable de m_1 dans N, et mène à un marquage m_2 tel que $\forall p \in P$, $m_2(p) = m_1(p) - F(p,t) + F(t,p)$. Dans N' nous avons :

  $$\forall p \in P,\ m'_2[A(p)] = m'_1[A(p)] - B_{K(p)}F(p,t) + B_{K(p)}F(t,p)$$

  et donc :

  $$\forall p \in P,\ m_2(p) = \varphi_{K(p)} m'_2[A(p)]$$

  Nous en concluons que $m_2\,\mathcal{R}\,m'_2$ et ceci termine la preuve de la proposition 22.

$\square$

Des propositions 21 et 22, et de $m_0\,\mathcal{R} m'_0$, on en déduit le théorème suivant :

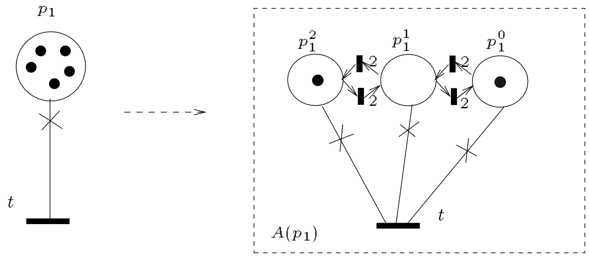
Théorème 23 *N et N' sont λ-bisimilaires par $\mathcal{R}$.*

Remarque : *implicitement, N est étiqueté par la fonction l telle que $\forall t \in T$, $l(t) = t$.*

3.2.4 Adaptation aux réseaux Reset et Transfert

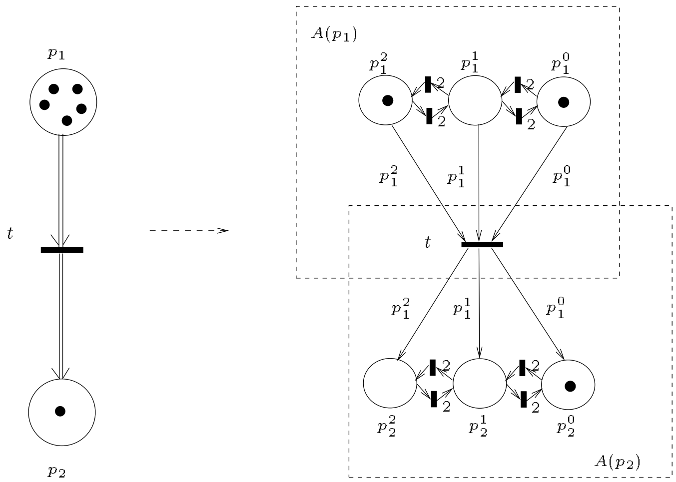
Voyons à présent les modifications qu'il faut apporter pour adapter cette construction aux réseaux étendus.

Étant donné un réseau de Petri avec Reset N_R, nous construisons un réseau normalisé N'_R. Nous "2-normalisons" d'abord les arcs de Petri selon la méthode précédente. Ensuite, pour tout arc Reset $(p; p; t)$, nous ajoutons autant d'arcs Reset que de places dans $A(p)$. Plus exactement, nous ajoutons les arcs $(p^i; p^i; t)$ avec $0 \leq i \leq K(p)$:



Ainsi, mettre p à zéro dans N_R correspond à mettre $A(p)$ à zéro dans N'_R.

Étant donné un réseau de Petri avec Transfert N_T, nous construisons un réseau normalisé N'_T. Il faut ajuster un peu la construction. Nous considérons d'abord les arcs de Petri et nous construisons les réseaux $A(p)$ avec la taille maximale possible, c'est-à-dire que $\forall p \in P$, $K(p) = K = \lfloor \log V(N) \rfloor + 1$ places. Ensuite, nous ajoutons K arcs de transfert pour chaque transfert de p vers p', opéré par une transition t. Par exemple, pour un transfert de p_1 vers p_2, nous ajoutons les arcs de transfert suivants : $(p_1^i; p_1^i; t)$ et $(t; p_2^i; p_2^i)$ avec $0 \leq i \leq K$:



Ainsi, transférer p dans p' correspond à transférer toute place de $A(p)$ dans la place de $A(p')$ qui lui correspond.

Remarque : *sur le dessin, nous avons repris la représentation des transferts par les polynômes et non les flèches doubles, afin qu'il n'y ait pas d'ambiguïté quant aux directions des transferts.*

La preuve de λ-bisimulation des réseaux de Petri s'étend sans difficulté et nous obtenons la proposition suivante :

Proposition 24 *Soient N_R un réseau de Petri avec Reset et N'_R son réseau 2-valué associé, et soient N_T un réseau de Petri avec Transfert et N'_T son réseau 2-valué associé :*

- *Les réseaux N_R et N'_R sont λ-bisimilaires par la relation $\mathcal{R}$.*
- *Les réseaux N_T et N'_T sont λ-bisimilaires par la relation $\mathcal{R}$.*

3.2.5 Préservation des propriétés

Voyons quelles sont les propriétés préservées par la "2-normalisation" (pour les réseaux de Petri, Reset ou Transfert).

Proposition 25 *Étant donné un réseau N (Petri, Reset ou Transfert) à k places, la 2-normalisation de N vers N' préserve :*

- *Le langage via l' et la t-vivacité.*
- *La couverture, le caractère borné du réseau, le caractère borné d'une place, le caractère structurellement borné et l'accessibilité.*

Preuve : soit $m \in \mathbb{N}^{|P|}$, nous notons par m' le marquage dans $\mathbb{N}^{|P'|}$ tel que :

$$\forall p \in P,\ m'(p) \quad = \quad \begin{cases} m(p_i) & \text{si } p = p_i^0 \ (p \in A(p_i)) \\ 0 & \text{sinon} \end{cases}$$

La place p^0 d'un réseau $A(p)$ est la place la plus à droite, dans laquelle on peut grouper tous les jetons de $A(p)$ à tout moment, par les transitions de type G. Nous avons $(m\,\mathcal{R}\,m')$.

- La préservation du langage et de la t-vivacité sont une conséquence directe de la λ-bisimulation.
- *Couverture :* $m \in \mathbb{N}^k$ est couvert dans N si et seulement si $m' \in \mathbb{N}^{|P'|}$ est couvert dans N'.
- *Caractère borné du réseau :* N est borné si et seulement si N' est borné.
- *Caractère borné d'une place :* une place p_i de N n'est pas bornée si et seulement si le sous-réseau $A(p_i)$ de N' n'est pas borné, si et seulement si la place p_i^0 n'est pas bornée dans N'.
- *Caractère structurellement borné :* le réseau N n'est pas structurellement borné si et seulement si $\exists m \in \mathbb{N}^{|P|}$, tel que $Acc(N, m)$ est infini si et seulement si $\forall m'' \in \mathbb{N}^{|P'|}$ tel que $m\mathcal{R}m''$, $Acc(N', m'')$ est infini.
- *Accessibilité :* $m \in Acc(N)$ si et seulement si $m' \in \mathbb{N}^{|P'|}$ est accessible dans N'.

Toutes ces préservations de propriétés sont une conséquence directe de la λ-bisimilarité et des propriétés de la relation $\mathcal{R}$. □

3.2.6 Normalisation de 2-valué vers 1-valué

Nous partons d'un réseau 2-valué $N' =< P', T', F', m'_0 >$ (Petri, Reset ou Transfert) et nous éliminons les valuations 2, qui se trouvent uniquement sur les arcs de transitions invisibles des réseaux $A(p)$. La valeur 2 est vue comme une constante. Nous construisons un réseau N'' comme suit :

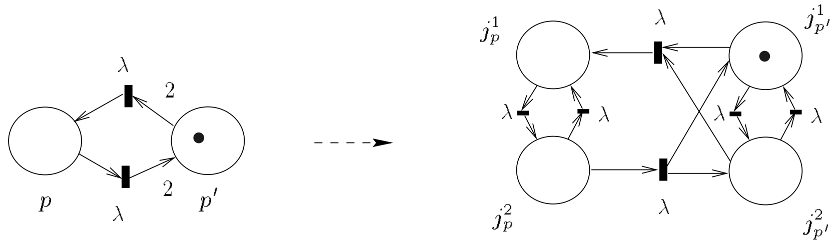


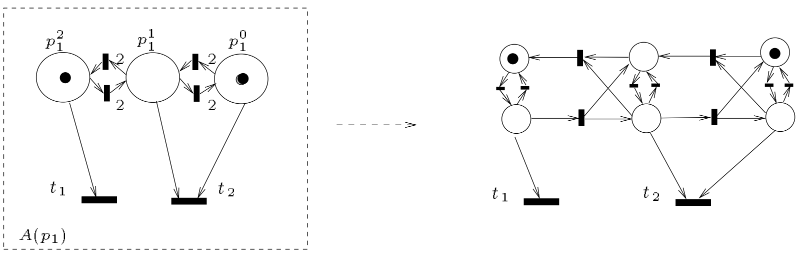
FIG. 3.8 – *Construction de 2-valué vers 1-valué*

La figure 3.8 illustre l'algorithme appliqué sur tout fragment 2-valué d'un réseau $A(p)$. L'idée est de séparer toute place $p \in |P'|$ en deux places jumelles, que nous notons j_p^1 et j_p^2. Deux places jumelles communiquent librement, c'est-à-dire qu'à tout moment un jeton de l'une peut bouger dans l'autre par des transitions qui sont invisibles. Si $m'_0(p) = 1$ dans N', alors $m''_0(j_p^1) = 1$ dans N'' (mais j_p^2 est vide).

- *Transitions invisibles :* retirer (resp. ajouter) 1 jeton dans une place p de N' correspond à retirer (resp. ajouter) 1 jeton dans la place j_p^2 (resp. j_p^1) de N''. Retirer (resp. ajouter) 2 jetons dans une place p de N' correspond à retirer (resp. ajouter) 1 jeton dans chacune des places jumelles j_p^1 et j_p^2 de N''. Les transitions de N'' qui effectuent ces opérations sont toutes invisibles.

- *Transitions visibles :* tout arc $(p; Q; t)$ (resp. $(t; Q; p)$) de N' correspond à un arc $(j_p^1; Q; t)$ (resp. $(t; Q; j_p^1)$) dans N''. Les transitions visibles de N' restent visibles dans N''.

Voici, par exemple, la normalisation du réseau $A(p_1)$. Toutes les transitions non étiquetées sur le dessin sont invisibles.



Un réseau 2-valué N' et le réseau normalisé N'' qui lui est associé sont λ-bisimilaires via la relation $\mathcal{T}$ que nous définissons ici.

Définition 34 *Soient un réseau v-valué N, son réseau 2-valué N' et son réseau normalisé N'' associé et soient $m'_1 \in \mathbb{N}^{|P'|}$ et $m''_1 \in \mathbb{N}^{|P''|}$.*

La relation $\mathcal{T}$ est définie par :

$$m'_1 \,\mathcal{T}\, m''_1 \Leftrightarrow \exists m'_2 \in \mathbb{N}^{|P'|} \text{ tel que :}$$

– $\forall p \in P,\ m'_1[A(p)] \sim m'_2[A(p)]$

– $\forall p' \in P',\ m'_2(p) = m''_1(j_p^1) + m''_1(j_p^2)$

Il s'agit de regarder la valeur correspondant à la somme sur les couples de places jumelles. Nous définissons enfin la relation entre le réseau v-valué et le réseau normalisé :

Définition 35 *Soient un réseau v-valué N, son réseau 2-valué N' et son réseau normalisé N'' associé et soient $m \in \mathbb{N}^{|P|}$ et $m'' \in \mathbb{N}^{|P''|}$.*

La relation $\mathcal{S}$ est définie par :

$$m \,\mathcal{S}\, m'' \Leftrightarrow \exists m' \in \mathbb{N}^{|P'|},\ \text{tel que } (m \,\mathcal{R}\, m') \text{ et } (m' \,\mathcal{T}\, m'')$$

Théorème 26 *Étant donné un réseau de Petri N (resp. un réseau de Petri N_R avec Reset, un réseau de Petri N_T avec Transfert), il existe une normalisation qui préserve les caractères bornés du réseau et d'une place, le caractère structurellement borné, l'accessibilité et la t-vivacité.*

Preuve : le réseau N (resp. N_R, resp. N_T) est λ-bisimilaire avec N'' (resp. N''_R, resp. N''_T) par la relation $\mathcal{S}$, ce qui assure la préservation des propriétés décrites précédemment (la preuve est directe). □

3.2.7 Complexité

Nous donnons une borne supérieure sur la complexité en temps. Rappelons que nous encodons la fonction de flot d'un réseau par deux matrices $(P \times T)$: une matrice des polynômes des pré-arcs et une matrice des polynômes des post-arcs. Nous raffinons la définition de taille donnée en section 2.4, car les réseaux de Petri, Reset et Transfert sont des cas particuliers des G-réseaux.

La taille d'un réseau de Petri N est :

$$\texttt{Taille}\ (N) \in \Theta(|P| * |T| * \log V(N))$$

car les polynômes sont en fait des constantes. La taille d'un réseau de Petri avec Reset N_R est la même :

$$\texttt{Taille}\ (N_R) \in \Theta(|P_R| * |T_R| * \log V(N_R))$$

car nous pouvons conserver deux matrices d'entiers et ajouter une matrice de booléens pour identifer les arcs de mise à zéro. La taille d'un réseau de Petri avec Transfert N_T est :

$$\texttt{Taille}\ (N_T) \in \Theta(|P_T| * |T_T| * \log V(N_T) + |P|^2 * |T|)$$

car nous pouvons coder les arcs de Petri par deux matrices d'entiers, les arcs de mise à zéro par une matrice de booléens et les post-arcs qui rendent compte du transfert par une matrice $|P| \times |T| \times |P|$ de booléens qui indique quelles places sont déversées dans la place courante par la transition courante (plusieurs places peuvent être transférées dans une même place avec une même transition).

Nous partons d'un réseau N (Petri, Reset ou Transfert) et nous évaluons le nombre total de places et de transitions produites pour le réseau 2-normalisé N'. Posons $V = V(N)$ (valuation maximale du réseau parmi tous les coefficients, exposants et valeurs dans m_0).

- *Places*: pour chaque place p, on construit le réseau $A(p)$ de $\lfloor \log V \rfloor + 1$ places au maximum. Nous avons $O(|P| * \log V)$ places.
- *Transitions*: chaque réseau $A(p)$ contient $O(K(p))$ transitions invisibles. Nous avons en effet deux transtions (type G et type D) associées à chaque place. En tout, nous avons $O(|P| * \log V)$ transitions invisibles. Par ailleurs, nous avons $|T|$ transitions visibles correspondant aux transitions de N. En tout, le nombre de transitions de N' est en $O(|P| * \log V + |T|)$.
- *Valuation*: la valuation maximale est égale à 2 (et considérée comme une constante).

Pour les réseaux de Petri ou les Reset, nous obtenons la complexité qui suit :

$$\texttt{Taille}(N') \in O((|P| * \log V) * (|P| * \log V + |T|) * \log 2)$$

c'est-à-dire :

$$\texttt{Taille}(N') \in O((|P| * \log V)^2 + (|T| * |P| * \log V))$$

Et puisque $|N| \in \Theta(|P| * |T| * \log V)$, alors chaque terme est dans l'ordre de $|N|^2$. Nous en concluons une borne (large) supérieure pour N' de :

$$\texttt{Taille}(N') \in O(|N|^2)$$

De plus, lorsque dans N nous avons $O(|P|) = O(|T| / \log V)$, alors la taille de N' est **linéaire** : $\texttt{Taille}(N') \in O(|P| * \log V) * (|T| + |T|)) = O(|N|)$. Par ailleurs, si l'on suppose que $O(|P|) = O(|T|)$, nous obtenons cette fois une relation quasiment linéraire : $\texttt{Taille}(N') \in \log V * |N|$.

Pour les Transfert, nous obtenons :

$$\texttt{Taille}(N') \in O((|P| * \log V) * (|P| * \log V + |T|) + (((|P| * \log V)^2 * (|P| * \log V + |T|)))$$

ou encore :

$$\texttt{Taille}(N') \in O((|P| * \log V) * [(|P| * \log V + |T|) + (|P| * \log V)(|P| * \log V + |T|)])$$

d'où :

$$\texttt{Taille}(N') \in O((|P| * \log V) * (|P| * \log V + |T|) * (1 + |P| * \log V))$$

nous obtenons donc une borne large :

$$\texttt{Taille}(N') \in O(\log V * |N|^2)$$

qui est quasiment quadratique.

Conclusion.

La construction est directe et ne nécessite aucun calcul faisant éclater la borne supérieure quadratique que nous venons de donner. Le temps total de construction de N' est dans l'ordre de $|N|^2$ ($\log V * |N|^2$ pour les Transfert). La seconde transformation, de 2-valué à 1-valué, est aussi directe et prend un temps linéraire en $|N'|$.

Nous en concluons que le temps total de la normalisation des Petri et Reset appartient à $O(|N|^2)$. Pour les Transfert, nous obtenons une borne quasi-quadratique. Dans tous les cas, la construction est polynomiale.

Nous pouvons alors énoncer le théorème suivant :

Théorème 27 *Étant donné un réseau de Petri (resp. un réseau de Petri avec Reset, un réseau de Petri avec Transfert), il existe une normalisation polynomiale qui préserve la couverture, les caractères bornés du réseau et d'une place, le caractère structurellement borné, l'accessibilité et la t-vivacité.*

Ainsi, par le dépliage de la section précédente et par la normalisation, on obtient un exemple de simplification de la représentation des *Post Self-Modifying nets*. Bien qu'ils soient plus puissants que les réseaux de Petri (puisqu'en particulier l'accessibilité est indécidable), les *Post Self-Modifying net* peuvent se déplier en temps polynomial dans la classe des réseaux de Petri normalisés. La transformation préserve la couverture et les caractères bornés d'un réseau et d'une place.

Chapitre 4

Analyse des G-réseaux bien-structurés

Dans ce chapitre, nous allons traiter de l'analyse des classes bien structurées de G-réseaux, dont la plus grande est la classe des G-Post + Reset. Les sous-classes intermédiaires sont les G-Post + Transfert, les Transfert, les Reset, les G-Post, les *Post Self-Modifying*, les Double et les réseaux de Petri. Pour toutes ces classes, les gardes effectives des transitions sont de type réseau de Petri, car les arcs de Reset ou de Transfert ne sont pas contraignants. Par contre, les arcs de Reset ou de Transfert ne peuvent pas être simulés par les réseaux de Petri.

Nous allons passer en revue les problèmes définis à la section 2.1.2 et nous allons étudier ces problèmes de deux manières :

- D'abord, nous mettrons en évidence les relations qui existent entre ces problèmes et nous établirons une hiérarchie basée sur leurs "difficultés relatives". Pour cela, nous utiliserons les *réductions multivoques polynomiales*. Nous regrouperons les problèmes polynomialement équivalents en classes et nous ordonnerons ces classes.

- Ensuite, nous rappellerons les solutions et les techniques connues (habituellement appliquées dans le contexte des réseaux de Petri) et nous confronterons les réseaux étendus à ces méthodes. Parfois, les théorèmes ou les algorithmes s'étendent naturellement aux sous-classes des G-réseaux.

En section 4.1, nous définirons notre outil de comparaison des problèmes, c'est-à-dire la réduction multivoque polynomiale. En section 4.2, nous mettrons en évidence la richesse du problème de caractère borné d'une place, auquel la terminaison, la couverture et le caractère borné du réseau se réduisent polynomialement. En section 4.3, nous étudierons les extensions et les limites de la construction de l'arbre de Karp et Miller et nous verrons que pour les réseaux Reset, cette approche ne permet pas de résoudre la couverture. En section 4.4, nous montrerons comment résoudre la couverture pour les Reset + G-Post par un algorithme de chaînage arrière. En section 4.5, nous étudierons la propriété d'accessibilité. En section 4.5.2 nous montrerons que l'accessibilité et le blocage sont polynomialement équivalents. En section 4.5.3, nous montrerons pour quelles sous-classes de réseaux la terminaison, le caractère borné du réseau et le caractère structurellement borné se réduisent polynomialement à l'accessibilité. En section 4.6, nous montrerons que la vivacité du réseau et la vivacité d'une transition sont polynomialement équivalents. Enfin, nous terminerons ce chapitre avec un tableau récapitulatif des nouveaux résultats apportés par cette première étude.

4.1 Réductions multivoques polynomiales

Le but des réductions est de comparer, ordonner et regrouper les problèmes selon leur degré de difficulté. Notre outil est la réduction multivoque polynomiale [HU79]. L'idée consiste à prendre une instance d'un problème A et à la transformer de façon efficace, c'est-à-dire en temps polynomial, en une instance d'un problème B. Ainsi, si l'on sait résoudre B, on sait résoudre A.

Réduire de façon multivoque polynomiale un problème A à un problème B a plusieurs intérêts :

- L'intérêt pratique est de pouvoir utiliser sans modification un programme résolvant B pour résoudre A. Le critère de temps polynomial assure la réalisation pratique de la transformation.
- Si l'on sait que A se réduit à B et que A est indécidable, alors on en déduit que B est indécidable. De même, si l'on sait que A se réduit à B et que B est décidable, alors A est décidable.
- La position d'un problème dans une hiérarchie donne des indications quant à son degré de difficulté.
- Réduire plusieurs problèmes à un même problème met en évidence la richesse de ce problème.
- Dans un contexte de décidabilité, les réductions polynomiales permettent d'obtenir des résultats de complexité.

Nous travaillons sur des problèmes de décision, c'est-à-dire dont la réponse attendue est `OUI` ou `NON`. Par exemple, nous voulons savoir si un réseau N est (ou non) borné. Nous notons par I_P un candidat à un problème P. La plupart du temps, nos instances sont des G-réseaux, mais il se peut qu'une instance soit un couple (un réseau et une transition, ou bien encore un réseau et une place). Nous parlons indifféremment de "problème" ou de "propriété". Soient A et B deux problèmes de décision :

Définition 36 *A est réductible de façon multivoque polynomiale à B, noté $A \leq_{poly} B$, s'il existe une fonction f calculable en temps polynomial telle que :*

$$I_A \textit{ possède la propriété } A \Leftrightarrow f(I_A) \textit{ possède la propriété } B$$

Si $A \leq_{poly} B$ et $B \leq_{poly} A$, alors les deux problèmes sont polynomialement équivalents (noté $A \equiv_{poly} B$).

Définition 37 *Une transformation d'un G-réseau N vers un G-réseau N' est forte si elle préserve pour N' la classe dont N fait partie.*

Exemple : ajouter des post-arcs étiquetés par des polynômes de degré 2 à un réseau de Petri avec arcs de Reset ne constitue pas une transformation forte. Par contre, le dépliage d'un G-Post + Reset vers un réseau de Petri est une transformation forte.

4.2 Richesse du caractère borné d'une place

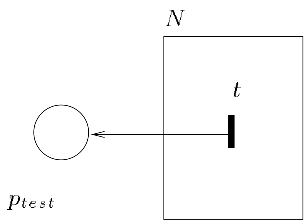
Nous comparons la terminaison, la couverture et les deux problèmes de caractère borné du réseau et d'une place, à l'aide des réductions polynomiales. Au terme de ces comparaisons, nous mettons en évidence la richesse de la propriété de caractère borné d'une place, à laquelle les trois autres problèmes se réduisent polynomialement. Dans cette section, toutes les transformations que nous donnons sont fortes.

De la terminaison vers le caractère borné du réseau

Nous réduisons le problème de terminaison au problème de caractère borné du réseau.

Proposition 28 *Pour les G-Post + Reset, Terminaison $\leq_{poly}$ Caractère Borné du Réseau.*

Preuve : soit $N =< P, T, F, m_0 >$ un G-Post + Reset à k places. Nous voulons décider de la terminaison de N. Nous construisons N' à partir de N, en ajoutant une place p_{test} et en mettant à jour F' de la façon suivante : $\forall t \in T,\ F'(t, p_{test}) = 1$. La place p_{test} est vide au départ, puis est incrémentée à chaque fois qu'une transition est franchie :



Correction : N termine si et seulement si N' est borné.

($\Rightarrow$) Si N termine, alors toutes ses exécutions sont finies. Puisque le système est finiment branché, le nombre d'exécutions est fini. Il en est de même pour N'. Nous notons l_{max} la longueur de la plus longue exécution de N. La valeur maximale que peut contenir toute place p de N' est inférieure ou égale à $m'_0(p) + l_{max}$ et donc N' est borné.

($\Leftarrow$) Si N ne termine pas, alors il existe dans N une exécution infinie. Dans N', l'exécution correspondante ajoutera une infinité de jetons dans p_{test}, et N' n'est donc pas borné.

Complexité : la transformation est linéaire.

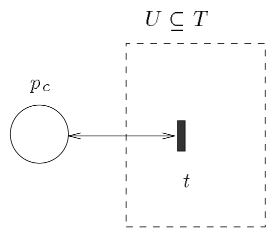
Nous en déduisons que pour les G-Post + Reset, Terminaison $\leq_{poly}$ Caractère Borné du Réseau. □

Remarque : *cette transformation est forte, puisqu'elle n'implique que des ajouts d'arcs de type réseau de Petri.*

Du caractère borné du réseau vers le caractère borné d'une place

Nous réduisons à présent le problème de caractère borné du réseau vers le problème de caractère borné d'une place. Dans la transformation, nous utilisons une place de contrôle que nous définissons ici.

Définition 38 *Une place de contrôle p_c pour un sous-ensemble de transitions $U \subseteq T$ est une place telle que $\forall t,\ t \in U,\ F(p_c, t) = F(t, p_c) = 1$.*



Une transition de U est franchissable sous la condition que p_c soit marquée, mais ne modifie pas son contenu. Autrement dit, la place p_c est testée par toute transition $t \in U$.

Théorème 29 *Pour les Reset+G-Post, Caractère Borné du Réseau $\leq_{poly}$ Caractère Borné d'une Place.*

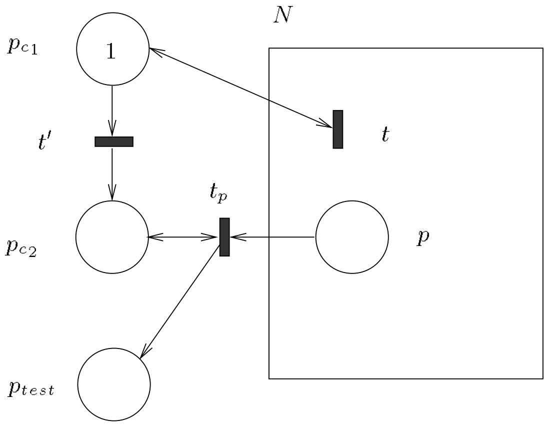


FIG. 4.1 – *Du caractère borné d'un réseau vers le caractère borné d'une place.*

Preuve : soit $N =< P, T, F, m_0 >$ un G-Post + Reset à k places. Nous voulons décider si N est borné. Nous construisons un réseau N' à partir du réseau N, tel que défini ci-dessous et illustré à la figure 4.1.

- $P' = P \cup \{p_{c1}, p_{c2}, p_{test}\}$
- $T' = T \cup \{t'\} \cup T_p$ où $T_p = \{t_p \mid p \in P\}$
- $F'(x, y) = F(x, y)$ pour $(x, y) \in (P \times T) \cup (T \times P)$. De plus :
  - $\forall p \in P,\ F'(p, t_p) = F'(p_{c2}, t_p) = F'(t_p, p_{c2}) = F'(t_p, p_{test}) = 1.$
  - $\forall t \in T,\ F'(t, p_{c1}) = F'(p_{c1}, t) = 1.$

- $F'(p_{c1}, t') = F'(t', p_{c2}) = 1$

- $m'_0 = (m_0(1), .., m_0(k), 1, 0, 0)$ où les k premières places sont celles de P, suivies de p_{c1}, p_{c2} et p_{test}.

La place p_{c1} est une place de contrôle pour T et la place p_{c2} pour T_p.

Correction : nous montrons que N est borné si et seulement si p_{test} est bornée.

($\Rightarrow$) Si N est borné, alors $\exists n \in \mathbb{N}, \forall m \in Acc(N), \forall p \in P, \ m(p) \leq n$. La seule façon d'incrémenter p_{test} est de franchir les transitions de T_p qui sont contrôlées par p_{c2}. Pour marquer p_{c2}, la seule possibilité est de franchir t' qui retire le jeton de la place de contrôle p_{c1}. Dès que t' est franchie, seules les transitions de T_p sont franchissables. Mais chaque franchissement de t_p décrémente la place p qui contient au plus n jetons. On conclut donc qu'on pourra franchir chaque t_p au maximum n fois, et donc qu'on pourra ajouter au plus $n.k$ jetons dans p_{test}. La place p_{test} est donc bornée.

($\Leftarrow$) Si N n'est pas borné, alors N peut produire un nombre arbitrairement grand de jetons et les déverser dans p_{test}. La place p_{test} n'est donc pas bornée.

Complexité : la transformation est polynomiale.

Nous en déduisons que pour les Reset+G-Post, Caractère Borné du Réseau $\leq_{poly}$ Caractère Borné d'une Place.

$\square$

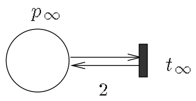
Remarque : *cette transformation est forte.*

De la couverture vers le caractère borné d'une place

Nous réduisons maintenant le problème de couverture au problème de caractère borné d'une place. Nous utilisons dans la transformation une place "inondable" que nous définissons ici.

Définition 39 *Une place inondable p_∞ est une place associée à une transition t_∞ telle que $F(p_\infty, t_\infty) = 1$ et $F(t_\infty, p_\infty) = 2$.*

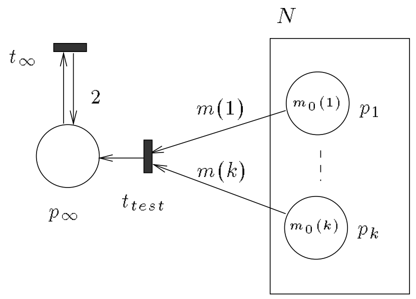
Dès qu'une place inondable contient un jeton, elle peut s'ajouter à elle-même une infinité de jetons :



Théorème 30 *Pour les G-Post+Reset, Couverture $\leq_{poly}$ Caractère Borné d'une Place.*

Preuve : soit $N =< P, T, F, m_0 >$ un G-Post + Reset à k places. Nous voulons décider si un marquage m est couvert dans N. Nous construisons N_m en ajoutant à N une place inondable p_∞ initialisée à 0, la transition t_∞ associée et une transition t_{test}. Nous définissons F_m à partir de F, avec en plus : $\forall p \in P, \ F_m(p, t_{test}) = m(p)$ et $F_m(t_{test}, p_\infty) = 1$.

Nous obtenons le réseau suivant :



Correction : m est couvert dans $N \Leftrightarrow t_{test}$ peut être franchie dans $N_m \Leftrightarrow p_\infty$ peut contenir un jeton $\Leftrightarrow p_\infty$ est non bornée.

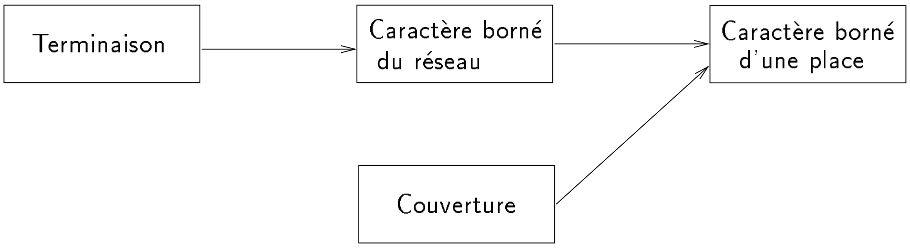
Complexité : la transformation est linéaire.

Nous en déduisons que pour les G-Post+Reset, Couverture $\leq_{poly}$ Caractère Borné d'une Place. □

Remarque : *cette transformation est forte.*

Diagramme récapitulatif des réductions polynomiales

Les trois réductions que nous venons de donner sont prouvées avec des transformations fortes et sont valables pour les G-Post + Reset qui constituent la plus grande classe de G-réseaux bien structurés. Nous résumons ci-dessous les relations entre ces problèmes dont le caractère borné d'une place est le plus général.



4.3 Extensions et limites de l'arbre de couverture

L'*arbre de couverture* d'un réseau de Petri (ou d'un G-réseau) est une structure finie qui rend compte, avec perte d'information, de l'aspect global de l'ensemble d'accessibilité du réseau. De nombreuses propriétés peuvent être vérifiées par la construction d'un arbre de couverture dont la terminaison, la couverture, le caractère borné du réseau ou d'une place.

La construction de l'arbre de couverture est de type chaînage avant : partant du marquage initial, on explore de façon systématique les exécutions possibles. On stoppe l'exploration d'une exécution sous certaines conditions qui assurent une construction finie. Nous devons cette méthode à Karp et Miller [KM69].

Dans cette section, nous commencerons par donner deux théorèmes qui expliquent l'intérêt d'une méthode de chaînage avant pour les systèmes bien-structurés, puis nous présenterons la construction de l'arbre de couverture dans le contexte des réseaux de Petri. Ensuite, nous verrons comment étendre cette approche aux G-réseaux.

Nous avons choisi de décrire explicitement l'algorithme de construction de l'arbre de couverture pour les réseaux de Petri pour deux raisons. D'une part, cela nous permet d'expliciter l'extension que l'on peut en faire, lorsque cela est possible, aux G-réseaux; d'autre part, cela nous permet d'expliciter les difficultés que l'on rencontre lorsque l'on tente d'étendre l'algorithme trop loin. Cependant, nous appliquons essentiellement des résultats connus et c'est la raison pour laquelle nous n'entrerons pas dans les détails des preuves de correction.

4.3.1 Terminaison et caractère borné du réseau

Pour résoudre la NON-terminaison, il suffit de savoir que l'on est en présence d'un G-réseau bien structuré (voir [Fin90, FS97]) :

Théorème 31 *Un G-réseau bien structuré $N =< P, T, F, m_0 >$ ne termine pas si et seulement s'il existe une exécution telle que :*

$$m_0 \xrightarrow{*} m \xrightarrow{\sigma} m' \text{ avec } \sigma \in T^+ \text{ et } m \leq m'$$

Preuve :

- (⇒) Si le réseau ne termine pas, alors il existe une exécution infinie. Puisque la relation $\leq$ est un bel ordre sur les vecteurs d'entiers, il est donc inévitable de rencontrer sur cette exécution deux marquages m puis m' tels que $m \xrightarrow{\sigma} m'$, $\sigma \in T^+$ et $m \leq m'$.
- (⇐) S'il existe une exécution telle que décrite par l'énoncé du théorème, alors la séquence σ peut être itérée une infinité de fois à partir de m', puisque le réseau est bien structuré. Le réseau ne termine donc pas.

□

Ainsi, pour répondre à la NON-terminaison d'un réseau bien structuré, dont les G-Post + Reset, il suffit d'exhiber une telle exécution.

Pour résoudre le problème du caractère NON-borné du réseau, il suffit de savoir que l'on est en présence d'un G-réseau strictement structuré (voir [Fin90, FS97]) :

Théorème 32 *Un G-réseau strictement structuré $N =< P, T, F, m_0 >$ est non borné si et seulement s'il existe une exécution telle que :*

$$m_0 \xrightarrow{*} m \xrightarrow{\sigma} m' \text{ avec } \sigma \in T^* \text{ et } m < m'$$

Preuve :

- (⇒) Généralisation du théorème de Karp et Miller [KM69] pour les réseaux de Petri.

($\Leftarrow$) La séquence σ peut être itérée une infinité de fois à partir de m', menant, à chaque itération, à un marquage de plus en plus grand, puisque le système est strictement structuré.

□

Pour répondre au caractère NON-borné d'un réseau strictement structuré, dont les G-Post + Transfert, il suffit d'exhiber une telle exécution.

Les réseaux de Petri sont strictement structurés, donc les deux théorèmes précédents s'appliquent à leur analyse. Mais puisqu'ils sont aussi fortement structurés, il existe un algorithme pour construire un arbre de couverture par lequel on peut décider en plus, de la couverture et du caractère borné d'une place.

4.3.2 Construction de l'arbre de couverture

Nous donnons rapidement une procédure pour construire l'arbre de couverture d'un réseau de Petri (voir les détails dans [KM69, Fin90]).

Nous travaillons dans $\mathbb{N} \cup \{\omega\}$, noté $\mathbb{N}_\omega$, où ω représente l'infini. Nous étendons l'addition et la soustraction sur $\mathbb{N}_\omega$, avec les règles : $\omega = \omega + \omega = \omega + n = \omega - n$ où $n \in \mathbb{N}$. De même, nous étendons la relation d'ordre $\leq$ sur $\mathbb{N}_\omega$ par $n < \omega$, $\forall n \in \mathbb{N}$.

Définition 40 *Considérons une séquence infinie croissante $\{u_n\}_{n\geq 1}$ d'entiers. Cette séquence est bornée s'il existe un K tel que $\forall n,\ u_n \leq K$.*

Nous définissons la limite de la séquence u_n :

$$lim\ u_n = \begin{cases} \omega & si\ \{u_n\}_{n\geq 1}\ n'est\ pas\ born\acute{e}e \\ \mathtt{max}\{u_n\} & sinon \end{cases}$$

Considérons une séquence infinie $\{u_n\}_{n\geq 1}$ dans $\mathbb{N}^k$. Supposons que $\{u_n\}_{n\geq 1}$ soit croissante, alors on pose :

$$limite\ de\ \ u_n = (lim\ u_n(1), lim\ u_n(2), .., lim\ u_n(k))$$

Étant donné un réseau de Petri $N =< P, T, F, m_0 >$ à k places, nous construisons un arbre de couverture $AC(N)$. Les nœuds de l'arbre sont des marquages dans $\mathbb{N}_\omega^k$ et sont tels que :

- Pour tout marquage $m \in Acc(N)$, il existe un marquage $m' \in AC(N)$ tel que $m' \geq m$.
- Tout marquage $m \in \mathbb{N}^k$ de $AC(N)$ appartient à $Acc(N)$.
- Tout marquage $m \in AC(N)$ tel que $m \notin \mathbb{N}^k$ est la limite d'une suite infinie de marquages accessibles dans N (pas forcément lors d'une même exécution).

  Exemple : si $(3, \omega, 5)$ fait partie de l'arbre de couverture, alors on pourra conclure que dans $Acc(N)$ il existe une suite infinie de marquages $(3, i, 5)$, avec i arbitrairement grand.

On commence la construction avec le marquage initial m_0 à la racine de l'arbre. On ajoute à m_0 tous ses successeurs, c'est-à-dire les m pour lesquels $\exists t \in T$ tel que $m_0 \xrightarrow{t} m$, puis on recommence l'opération sur chacun des successeurs. On essaye systématiquement les séquences de transitions franchissables. Les branches de l'arbre sont des exécutions du réseau.

Dès que l'on détecte une exécution $m_0 \xrightarrow{*} m_1 \xrightarrow{\sigma} m_2$ telle que $m_1 < m_2$, on opère sur m_2 une mise-à-jour de m_2 selon la procédure suivante. Pour tout i $(1 \leq i \leq k)$:

$$m_2(p_i) = \begin{cases} \omega & \text{si } m_1(p_i) < m_2(p_i) \\ m_2(p_i) & \text{sinon} \end{cases}$$

Ainsi, on remplace m_2 avec la limite de la suite infinie de marquages accessibles à partir de m_2 par la séquence σ (dans l'algorithme plus bas, c'est la fonction `limite` qui fait cette mise-à-jour). En effet, puisque les réseaux de Petri sont fortement structurés, on peut itérer σ à partir de m_2 une infinité de fois et produire de plus en plus de jetons dans les places telles que $m_1(p_i) < m_2(p_i)$.

Voici les deux critères qui assurent la terminaison de l'algorithme :

- Une exécution peut aboutir à un blocage et elle est donc finie.
- Une exécution peut avoir la forme $m_0 \xrightarrow{*} m \xrightarrow{\sigma} m'$ où $\sigma \in T^+$ avec $m = m'$. Dans ce cas, on arrête de développer l'arbre à partir de m'.

Algorithme de l'arbre de couverture

Définition 41 *Étant donnée une exécution $m_0 \xrightarrow{*} m \xrightarrow{\sigma} m'$ où $\sigma \in T^+$, m est l'ancêtre de m'.*

Voici l'algorithme principal de la construction de l'arbre de couverture (décrit récursivement) qui utilise la procédure `Construction` :

Procédure Construction (*in* $N =< P, T, F, m_0 >$: réseau de Petri;
inout AC : arbre;
in m : marquage;
inout Termine : booléen);

var m_a, m' : marquage;

{ **si** m a un ancêtre m_a tel que $m_a \leq m$ **alors**
(1) { Termine := `faux`;
(2) **si** $m_a < m$ **alors**
$m :=$ limite(m_a, m) }
si m n'a pas d'ancêtre m_a tel que $m = m_a$ **alors**
{ **pour tout** $m' \in \mathbb{N}^k$ pour lequel $\exists t \in T$ tel que $m \xrightarrow{t} m'$ **faire**
{ Ajout-fils(AC, m, m');
Construction(N, AC, m', Termine) } } }

Procédure Arbre-de-Couverture (*in* $N = < P, T, F, m_0 >$: réseau de Petri;
out AC : arbre;
out Termine : booléen);

{ Termine := `vrai`;
AC := CréerArbre(m_0);
Construction (N, AC, m_0, Termine) }

Voici comment utiliser les résultats de l'algorithme :

- *Terminaison* : N termine si et seulement si `Termine` a pour valeur `vrai`. Remarquons qu'en effet dès que l'on détecte une exécution $m_0 \xrightarrow{*} m \xrightarrow{\sigma} m'$ avec $|\sigma| > 0$ et $m \leq m'$, `Termine` est mise à `faux`.
- *Couverture* : un marquage $m \in \mathbb{N}^{|P|}$ est couvert dans N si et seulement s'il existe dans $AC(N)$ un marquage m' tel que $m' \geq m$.
- *Caractère borné du réseau* : N est non borné si et seulement s'il existe un marquage de $AC(N)$ qui contienne un ω. Remarquons qu'en effet nous plaçons les ω uniquement dans le cas où nous avons repéré une exécution de type $m_0 \xrightarrow{*} m \xrightarrow{*} m'$ avec $m < m'$.
- *Caractère borné d'une place* : une place p_i est non bornée si et seulement s'il existe un marquage m dans $AC(N)$ tel que $m(p_i) = \omega$.

4.3.3 Application aux G-réseaux

Pour décider de la NON-terminaison d'un réseau bien structuré, on peut stopper la construction de l'arbre de couverture dès que l'on rencontre une séquence de type $m_0 \xrightarrow{*} m \xrightarrow{\sigma} m'$ avec $|\sigma| > 0$ et $m \leq m'$ (théorème 31). Nous avons repéré la ligne correspondante de l'algorithme par (1). Si le réseau termine, alors l'ensemble d'accessibilité est fini, et l'algorithme stoppe avec la construction complète de l'*arbre d'accessibilité* qui contient tous les marquages du réseau (et aucun ω). Cet algorithme abrégé est correct pour les G-Post + Reset.

Corollaire 33 *Le problème de terminaison est décidable pour G-Post + Reset.*

Pour décider du caractère NON-borné d'un réseau strictement structuré, on peut stopper la construction de l'arbre dès l'apparition du premier ω, c'est-à-dire dès que l'on repère une séquence de la forme $m_0 \xrightarrow{*} m \xrightarrow{*} m'$ avec $m < m'$ (théorème 32). Nous avons repéré la ligne correspondante de l'algorithme par (2). Encore une fois, si le réseau est borné, alors l'ensemble d'accessibilité est fini et l'algorithme stoppe avec la construction complète de l'arbre d'accessibilité. Cet algorithme abrégé est correct pour les G-Post + Transfert.

Corollaire 34 *Le problème du caractère borné du réseau est décidable pour les G-Post + Transfert.*

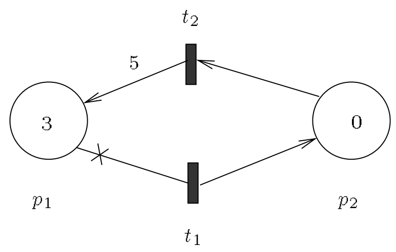
L'algorithme qui construit l'arbre de couverture s'étend directement aux G-Post qui sont, comme les réseaux de Petri, fortement structurés. Nous en déduisons que :

Corollaire 35 *Les problèmes de terminaison, couverture, caractère borné du réseau et caractère borné d'une place sont décidables pour les G-Post.*

Pour décider de la couverture d'un marquage m, on pourrait stopper la construction de l'arbre de couverture dès que l'on rencontre un marquage $m' \geq m$. Mais cet algorithme abrégé ne s'étend pas, du moins pas directement, à une classe plus grande que les G-Post.

Le problème vient de la gestion des ω. Pour les problèmes de terminaison ou de caractère borné du réseau, l'algorithme abrégé stoppe dès la rencontre du premier ω, ou avant. Par contre, pour répondre à un problème de couverture, il se peut qu'il faille construire un chemin de plusieurs marquages contenant des ω. Mais lorsque l'on est confronté à un système qui admet des mises à zéro, on ne peut pas calculer les limites des suites infinies de marquages comme pour les réseaux de Petri.

Considérons, par exemple, le réseau N suivant qui comporte un arc Reset :



L'exécution $(3,0) \xrightarrow{t_1} (0,1) \xrightarrow{t_2} (5,0)$, avec $(3,0) < (5,0)$, est possible. L'algorithme remplacerait dans ce cas $(5,0)$ par $(\omega,0)$, mais ici, ce serait une faute. L'itération infinie de la séquence t_1t_2, ne remplit pas p_2 d'une infinité de jetons. Nous avons plutôt :

$$(3,0) \xrightarrow{t_1} (0,1) \xrightarrow{t_2} (5,0) \xrightarrow{t_1} (0,1) \xrightarrow{t_2} (5,0) \xrightarrow{t_1} (0,1) \xrightarrow{t_2} (5,0) \cdots$$

et en fait, ici la place p_1 est bornée par 5.

Corollaire 36 *Pour les réseaux Reset (donc les G-Post + Reset) et les réseaux Transfert (donc les G-Post + Transfert), l'algorithme qui construit l'arbre de couverture donne un résultat faux.*

En fait, nous verrons à la section 5.3 qu'il n'est pas possible de construire de façon effective un arbre de couverture pour les classes des G-réseaux qui comportent des Reset, car pour celles-ci le problème du caractère borné du réseau est indécidable. Par contre, à la prochaine section nous verrons qu'il est possible de décider de la couverture pour les G-Post + Reset, mais par une autre approche.

Tableau récapitulatif

Au terme de cette première investigation des techniques habituellement appliquées aux réseaux de Petri, nous pouvons ajouter dans notre tableau récapitulatif quelques résultats de décidabilité.

	Terminaison	Couverture	Réseau borné	Place bornée	Réseau structurellement borné	Accessibilité	Blocage Vivacité t-Vivacité
G-Réseaux *Self-Modifying*	**I**	**I**	**I**	**I**	**I**	**I**	**I**
G-Post + Reset	**D**	?	?	?	?	**I**	?
Reset	**D**	?	?	?	?	**I**	?
G-Post + Transfert	**D**	?	**D**	?	?	**I**	?
Transfert	**D**	?	**D**	?	?	**I**	?
G-Post *Post Self-Modifying*	**D**	**D**	**D**	**D**	?	**I**	?
Petri	**D**	**D**	**D**	**D**	**D**	**D**	**D**

4.4 Algorithme de chaînage arrière pour résoudre la couverture

Pour résoudre le problème de couverture pour les G-Post + Reset, il existe une approche autre que celle de l'arbre de couverture, beaucoup plus récente, qui provient des travaux de Abdulla, Čerāns, Jonsson et Yih-Kuen [AČJY96] et de Finkel et Schnoebelen [FS97]. Cette méthode est dite de *chaînage arrière* ("*backward strategy*").

[Ajouté en novembre 1998] Rappelons que la couverture est en fait connue décidable pour les Reset depuis 1976. L'algorithme utilisé par Arnold et Latteux [AL76, AL78] est le même que celui que nous présentons, et redécouvrons. Il est de type *chaînage arrière* : les auteurs construisent un arbre de chemins "inversés" partant du marquage que l'on souhaite couvrir et allant vers un ensemble de marquages susceptible de le couvrir.

Définition 42 *Un prédécesseur couvrant de m par N est un marquage m' tel que $m' \xrightarrow{*} m''$ et $m'' \geq m$.*

Répondre à la question de la couverture revient à regarder si le marquage initial m_0 est un prédécesseur couvrant de m. L'idée de l'algorithme consiste donc à calculer l'ensemble de tous les *prédécesseurs couvrants* de m par N. Le point de départ de cet algorithme est m, et non m_0, et le travail consiste à "remonter" dans les exécutions possibles, et non à avancer comme c'est le cas dans la construction de l'arbre de couverture.

Dans un premier temps, nous expliquerons l'algorithme, puis nous montrerons comment l'appliquer à la classe des G-Post + Reset. Ensuite, nous donnerons un exemple d'application sur un réseau de Petri avec arcs de Reset.

4.4.1 Algorithme de chaînage arrière

Voici la description de l'algorithme de [AČJY96, FS97].

Définition 43

- *Un ensemble* $S \subseteq E$ *muni d'une relation d'ordre* $\preceq$ *sur les éléments est fermé vers le haut si pour tout* $s \in S$ *et pour tout* $s' \in E$ *:*

$$s \preceq s' \Rightarrow s' \in S$$

- *On note* $\uparrow S$ *la fermeture vers le haut d'un ensemble* $S \subseteq E$ *muni d'une relation d'ordre* $\preceq$ *:*

$$\uparrow S = \{s' \in E \mid \exists s \in S,\ s \preceq s'\}$$

Nos ensembles sont des ensembles de marquages et la relation d'ordre est $\leq$ (l'ordre habituel sur les marquages). Nous précisons les définitions des ensembles de prédécesseurs :

Définition 44 *Étant donné un G-réseau* $N =< P, T, F, m_0 >$ *et un marquage* $m \in \mathbb{N}^{|P|}$,

- *Nous notons* $Pred(m)$ *l'ensemble des prédécesseurs (immédiats) de* m *:*

$$Pred(m) = \{m' \mid \exists t \in T,\ m' \xrightarrow{t} m\}$$

et nous étendons cette définition aux ensembles de marquages.

- *Nous notons* $Pred^*$ *la fermeture réflexive et transitive de* $Pred$. $Pred^*(m)$ *est l'ensemble de tous les prédécesseurs (pas forcément immédiats) de* m *:*

$$Pred^*(m) = \{m' \mid m' \xrightarrow{*} m\}$$

Définition 45 *Soit un G-réseau et* $m \in \mathbb{N}^{|P|}$. *Nous posons la récurrence :*

$$\begin{cases} K_0 \overset{def}{=} \uparrow\{m\} \\ K_i \overset{def}{=} K_{i-1} \cup Pred(K_{i-1})\ (i \geq 1) \end{cases}$$

Nous avons : $Pred^*(\uparrow\{m\}) = \cup_{i \in \mathbb{N}} K_i$.

Lemme 37 $K_n = K_{n+1} \Rightarrow \forall l \in \mathbb{N},\ K_n = K_{n+l}$

Preuve : étant donné un ensemble de marquages X, posons :

$$f(X) = X \cup Pred(X)$$

Nous avons :

$$K_n = K_{n+1} = K_n \cup Pred(K_n) = f(K_n)$$

avec $f^l(K_n) = K_{n+l} = K_n$. Nous en déduisons que $\forall l \in \mathbb{N},\ K_n = K_{n+l}$. □

Pour répondre au problème de couverture, il suffit de calculer $Pred^*(\uparrow\{m\})$ et tester si $m_0 \in Pred^*(\uparrow\{m\})$. Voyons deux lemmes préliminaires :

Lemme 38 *Soit* $N =< P, T, F, m_0 >$ *un G-réseau bien structuré et* S *un ensemble de marquages de* $\mathbb{N}^{|P|}$. *Si* S *est fermé vers le haut, alors* $Pred(S)$ *est fermé vers le haut.*

Preuve : prenons $m_1 \in Pred(S)$ et $m_2 \in \mathbb{N}^k$ tel que $m_1 \leq m_2$. Puisque $m_1 \in Pred(S)$, nous savons qu'il existe une transition $t \in T$ et $m'_1 \in S$ tels que $m_1 \xrightarrow{t} m'_1$. Puisque N est bien structuré, alors il existe un marquage $m'_2 \in \mathbb{N}^{|P|}$ tel que $m_2 \xrightarrow{t} m'_2$ et $m'_1 \leq m'_2$. Puisque S est fermé vers le haut, alors $m'_2 \in S$ et donc $m_2 \in Pred(S)$. Nous en concluons que $Pred(S)$ est fermé vers le haut. □

Lemme 39 *Toute suite croissante $S_0 \subseteq S_1 \subseteq S_2 \subseteq ..$ de sous-ensembles de vecteurs d'entiers fermés vers le haut se stabilise inévitablement (c'est-à-dire que $\exists k \in \mathbb{N}$ tel que $\forall i \in \mathbb{N}, i > k \Rightarrow S_k = S_i$).*

Preuve : supposons qu'il existe une telle suite qui ne se stabilise pas. Alors, on peut extraire une suite infinie strictement croissante d'ensembles $S_{n_0} \subset S_{n_1} \subset S_{n_2} \subset \cdots$ à partir de laquelle on peut extraire une suite infinie de vecteurs $v_0 \neq v_1 \neq v_2 \neq \cdots$ où $v_i \in S_{n_i}$ et $\forall k \in \mathbb{N},\ i < k,\ v_k \notin S_{n_i}$. Mais puisque la relation $\leq$ est un bel ordre, il existe $i, j \in \mathbb{N}$ tels que $i < j$ et $v_i \leq v_j$; et puisque S_{n_i} est fermé vers le haut, nous en concluons que $v_j \in S_{n_i}$, ce qui amène à une contradiction. □

Par les lemmes 38 et 39, on obtient :

Corollaire 40 *Soit un G-réseau bien structuré N et un marquage $m \in \mathbb{N}^{|P|}$, il existe un $n \in \mathbb{N}$ tel que $K_n = K_{n+l} = Pred^*(\uparrow\{m\})$, $\forall l \in \mathbb{N}$.*

Ainsi, pour un G-réseau bien structuré, nous savons qu'un nombre fini d'itérations (jusqu'à stabilisation) suffirait à calculer $Pred^*(\uparrow\{m\})$. Mais en pratique, il faudrait représenter des ensembles infinis. Alors comment serait-il possible de calculer effectivement $Pred^*(\uparrow\{m\})$? La réponse vient du fait que ces ensembles sont fermés vers le haut. On peut donc les représenter exactement à l'aide d'une *base* finie.

Définition 46 *Une base d'un ensemble S fermé vers le haut muni d'une relation $\preceq$ est un ensemble fini d'éléments $\{x_1, .., x_n\}$ tel que pour tout $s \in S$, il existe un s_i de la base tel que $s_i \preceq s$.*

Définition 47 *Étant donné un ensemble de marquages S, $Min(S)$ est l'ensemble des éléments minimaux de S (pour l'ordre habituel $\leq$ sur les vecteurs $\mathbb{N}^k$).*

Lemme 41 *$Min(S)$ est fini.*

Preuve : supposons le contraire; puisque $\leq$ est un bel ordre, alors il existe deux marquages $m_1, m_2 \in Min(S)$ tels que $m_1 \leq m_2$, ce qui amène à une contradiction. □

L'ensemble des éléments minimaux d'un ensemble S fermé vers le haut est une base de S et constitue une représentation unique et finie de S : $\uparrow Min(S) = S$.

Théorème 42 *Étant donné un G-réseau N bien structuré et un vecteur d'entiers m, si pour tout m' la fonction $Min(Pred(\uparrow\{m'\}))$ est calculable, alors on peut calculer l'ensemble $Pred^*(\uparrow\{m\})$ et décider de la couverture.*

Preuve : considérons d'abord l'algorithme qui calcule la suite BP (Base de Prédécesseurs) suivante :

$$\begin{cases} BP_0 = Min(\uparrow\{m\}) \\ BP_i = Min(BP_{i-1} \cup_{m' \in BP_{i-1}} Min(Pred(\uparrow\{m'\}))) \ (i \geq 1) \end{cases}$$

et qui stoppe lorsque la suite se stabilise. Montrons par induction que $\forall i \in \mathbb{N}$, $BP_i = Min(K_i)$. C'est vrai pour BP_0. Supposons que ce soit vrai pour tout $i \leq n-1$ et montrons que $BP_n = Min(K_n)$.

$$\begin{array}{lcl}
BP_n & = & Min(BP_{n-1} \cup_{m' \in BP_{n-1}} Min(Pred(\uparrow\{m'\}))) \\
& = & Min(Min(K_{n-1}) \cup Min(\cup_{m' \in BP_{n-1}} Min(Pred(\uparrow\{m'\})))) \\
& = & Min(Min(K_{n-1}) \cup Min(\cup_{m' \in BP_{n-1}} Pred(\uparrow\{m'\}))) \\
& = & Min(Min(K_{n-1}) \cup Min(Pred(\cup_{m' \in BP_{n-1}} \uparrow\{m'\}))) \\
& = & Min(Min(K_{n-1}) \cup Min(Pred(\uparrow\cup_{m' \in BP_{n-1}} \{m'\}))) \\
& = & Min(Min(K_{n-1}) \cup Min(Pred(\uparrow BP_{n-1}))) \\
& = & Min(Min(K_{n-1}) \cup Min(Pred(\uparrow Min(K_{n-1})))) \\
& = & Min(Min(K_{n-1}) \cup Min(Pred(K_{n-1}))) \\
& = & Min(K_{n-1} \cup Pred(K_{n-1})) \\
& = & Min(K_n)
\end{array}$$

Puisque pour tout n nous avons $BP_n = Min(K_n)$ et par le corollaire 40, nous savons qu'il existe un k tel que $BP_k = BP_{k+1}$ et donc que l'algorithme stoppera. Cet algorithme calcule BP_k qui est une base de $Pred^*(\uparrow\{m\})$.

Ensuite, pour décider de la couverture pour m, il suffit de vérifier s'il existe $m \in BP_k$ tel que $m \leq m_0$ (ce qui est possible puisque BP_k est fini).

Nous en concluons que la couverture est décidable pour les G-réseaux bien structurés, si pour tout m', $Min(Pred \uparrow\{m'\})$ est calculable. □

4.4.2 Application sur les G-Post + Reset

Nous voyons dans un premier temps comment calculer $Min(Pred(\uparrow\{m\}))$ pour les réseaux de Petri avec Reset.

Lemme 43 *Étant donné un réseau de Petri avec Reset $N =< P, T, F, m_0 >$ et un marquage $m \in \mathbb{N}^{|P|}$, il existe un algorithme pour calculer $Min(Pred(\uparrow\{m\}))$.*

Preuve : pour chaque transition t, cherchons (s'il existe) un prédécesseur couvrant (immédiat) de m par t, c'est-à-dire un marquage $m' \in \mathbb{N}^{|P|}$ tel que :

$$\exists m'' \in \mathbb{N}^{|P|},\ m' \xrightarrow{t} m'' \text{ et } m \leq m''$$

Trois cas se présentent :

1. **Cas 1 :** considérons le cas où $F(t, p_i) \geq m(p_i)$. Quel que soit m' tel que $m'(p_i) = 0$ et m'' tels que $m' \xrightarrow{t} m''$, forcément $m(p_i) \leq m''(p_i)$.
2. **Cas 2 :** considérons le cas où $F(t, p_i) < m(p_i)$ et il existe un arc de Reset sur p_i par t. Quels que soient m', m'' tels que $m' \xrightarrow{t} m''$, nous avons $m''(p_i) = F(t, p_i) < m(p_i)$ et $m(p_i)$ ne peut pas être couvert. Si ce cas se présente, alors on ne peut pas calculer de prédécesseur couvrant par t, ni pour m, ni pour tout marquage plus grand que m.

3. **Cas 3 :** considérons enfin le cas où $F(t,p_i) < m(p_i)$ et il n'existe pas d'arc de Reset sur p_i par t. Ici, $m(p_i) - F(t,p_i) + F(p_i,t)$ est positif. S'il existe un m' tel que $m'(p_i) = m(p_i) - F(t,p_i) + F(p_i,t)$ et un m'' tels que $m' \xrightarrow{t} m''$, alors forcément $m''(p_i) = m'(p_i) - F(p_i,t) + F(t,p_i) = m(p_i)$. Par ailleurs, $m(p_i) - F(t,p_i) + F(p_i,t)$ est la plus petite valeur possible pour $m'(p_i)$.

Si le cas 2 ne se présente pas, alors considérons le marquage m' tel que :

$$\forall p_i \in P,\ m'(p_i) = \begin{cases} 0 & \text{si } p_i \text{ est dans le cas 1} \\ m(p_i) - F(t,p_i) + F(p_i,t) & \text{si } p_i \text{ est dans le cas 3} \end{cases}$$

m' est le plus petit de tous les prédécesseurs couvrants de m par t.

L'union de tous les m' (au plus un par transition) forme une base pour $Pred(\uparrow\{m\})$, dont on ne garde que les éléments minimaux.

Toutes les opérations décrites sont calculables, nous en concluons qu'il existe un algorithme pour calculer $Min(Pred(\uparrow\{m\}))$. □

Par le théorème 42 et le lemme 43, on obtient le théorème suivant :

Théorème 44 *La couverture est décidable pour les réseaux de Petri avec Reset.*

Voyons comment l'on peut étendre ce résultat aux G-Post + Reset. Pour calculer une base des prédécesseurs dans le cadre des G-Post + Reset, on pourrait donner des règles comme on l'a fait pour les Reset. Le système que l'on obtiendrait serait cependant plus complexe : pour une transition, il faut considérer la résolution d'un système d'inéquations de polynômes. Les solutions, si elles existent, forment un ensemble fermé vers le haut, et il faut en déterminer une base finie. Puisque les équations ont une forme spécifique, et en particulier elles impliquent des polynômes à coefficients entiers et requièrent des solutions entières, il serait probablement possible de déterminer directement si le système admet des solutions.

Mais nous allons plutôt adopter une autre stratégie: pour décider de la couverture pour les G-Post, il suffit d'utiliser le dépliage (section 3.1) d'un réseau G-Post + Reset en un réseau de Petri avec Reset. La transformation préserve la couverture et il ne reste plus qu'à appliquer l'algorithme de chaînage arrière pour les réseaux de Petri avec Reset. On obtient le théorème suivant :

Théorème 45 *La couverture est décidable pour les réseaux G-Post + Reset.*

Nous pouvons en conclure que la couverture est décidable aussi pour les sous-classes des G-Post + Reset :

Corollaire 46 *La couverture est décidable pour les G-Post + Transfert et pour les réseaux de Petri avec arcs de Reset ou avec arcs de Transfert.*

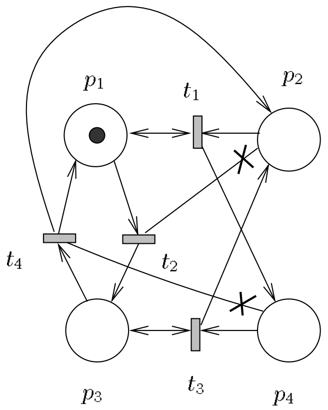
Mentionnons deux applications de la couverture. La couverture permet de décider du problème *d'accessibilité d'un état de contrôle* [AČJY96]. La couverture permet aussi de décider si un réseau G-Post + Reset est *déterministe*: pour savoir s'il existe un marquage accessible m tel que t_1 et t_2 sont toutes les deux franchissables à partir de m, il suffit de décider si le marquage m' tel que :

$$\forall p \in P,\ m'(p) = \begin{cases} 0 & \text{si } F(p,t_1) \notin \mathbb{N} \text{ et } F(p,t_2) \notin \mathbb{N} \\ F(p,t_1) & \text{si } F(p,t_1) \in \mathbb{N} \text{ et } F(p,t_2) \notin \mathbb{N} \\ F(p,t_2) & \text{si } F(p,t_1) \notin \mathbb{N} \text{ et } F(p,t_2) \in \mathbb{N} \\ \mathtt{max}(F(p,t_1), F(p,t_2)) & \text{si } F(p,t_1) \in \mathbb{N} \text{ et } F(p,t_2) \in \mathbb{N} \end{cases}$$

est couvert par le réseau. Le marquage m' est le plus petit marquage composé de l'union des conditions d'entrée de t_1 et t_2.

4.4.3 Application sur un réseau de Petri avec Reset

Considérons le réseau de Petri avec arcs de Reset suivant :



Nous voulons décider si ce réseau couvre le marquage $(1,2,0,0)$. Nous calculons la suite des ensembles BP_i. Nous obtenons le déroulement suivant :

Calcul de BP_0 :

$P_0 = Min(\uparrow\{m\}) = \{(1,2,0,0)\}$

Calcul de BP_1 (base minimale de K_1) :

$BP_1 = Min(BP_0 \cup Min(Pred(\uparrow\{(1,2,0,0)\})))$. Il nous faut calculer une base de l'ensemble des marquages qui couvrent $(1,2,0,0)$ en une transition. Il y a quatre transitions possibles :

- t_1 : $(1,3,0,0) \xrightarrow{t_1} (1,2,0,1) \geq (1,2,0,0)$
- t_2 : puisque t_2 met p_2 à zéro, il n'est certainement pas possible de couvrir $(1,2,0,0)$ par t_2.
- t_3 : $(1,1,1,1) \xrightarrow{t_3} (1,2,1,0) \geq (1,2,0,0)$
- t_4 : $(0,1,1,0) \xrightarrow{t_4} (1,2,0,0)$

Nous obtenons :

$$BP_1 = Min(\{(1,2,0,0),\ (1,3,0,0),\ (1,1,1,1),\ (0,1,1,0)\})$$

Et donc :

$$BP_1 = \{(1,2,0,0),\ (0,1,1,0)\}$$

Calcul de BP_2 :

$BP_2 = Min(BP_1 \cup Min(Pred(\uparrow\{(1,2,0,0)\})) \cup Min(Pred(\uparrow\{(0,1,1,0)\})))$. Calculons une base pour $Pred(\uparrow\{(0,1,1,0)\})$.

- t_1 : $(1,2,1,0) \xrightarrow{t_1} (1,1,1,1) \geq (0,1,1,0)$
- t_2 : puisque t_2 met p_2 à zéro, il n'est pas possible de couvrir $(0,1,1,0)$ par t_2.
- t_3 : $(0,0,1,1) \xrightarrow{t_3} (0,1,1,0)$
- t_4 : $(0,0,2,0) \xrightarrow{t_4} (1,1,1,0) \geq (0,1,1,0)$

Nous obtenons $BP_2 = \{(1,2,0,0),\ (0,1,1,0),\ (0,0,1,1),\ (0,0,2,0)\}$.

Calcul de BP_3 :

$BP_3 = \{(1,2,0,0),\ (0,1,1,0),\ (0,0,1,1),\ (0,0,2,0),\ (1,0,0,1),\ (1,0,1,0)\}$.

Calcul de BP_4 :

$BP_4 = \{(1,0,0,0),\ (0,0,1,1),\ (0,0,2,0), (0,1,1,0)\}$

Calcul de BP_5 :

$BP_5 = \{(1,0,0,0),\ (0,0,1,0)\}$

Calcul de $BP_6 = BP_5$. On atteint la stabilisation.

Nous pouvons répondre `OUI` à la question de départ : puisque $m_0 \geq (1,0,0,0)$, alors le marquage $(1,2,0,0)$ est couvert dans N.

Remarque : *on pourra optimiser l'algorithme en arrêtant avant la stabilisation, si l'on trouve un BP_i qui contient un marquage couvert par m_0.*

4.5 Richesse de la propriété d'accessibilité

Cette section est dédiée à l'accessibilité et contient plusieurs constructions qui mettent en évidence la richesse d'expression de la propriété d'accessibilité.

Nous commencerons par montrer que l'accessibilité est polynomialement équivalente au blocage de façon multivoque, pour les G-Post + Reset. Ainsi, ces deux problèmes étant très semblables, nous les regrouperons dans notre tableau récapitulatif. Par la suite, nous montrerons comment l'accessibilité peut exprimer la terminaison (pour les G-Post+Reset), le caractère borné du réseau et le caractère structurellement borné (pour les G-Post+Transfert). En utilisant le dépliage de la section 3.1, nous en déduirons que le caractère structurellement borné est décidable pour les G-Post.

4.5.1 Accessibilité et accessibilité partielle

Afin de simplifier les réductions que nous présentons, nous allons travailler non pas avec l'accessibilité, mais avec le problème d'accessibilité partielle d'un marquage défini sur un sous-ensemble de P :

• Problème de l'accessibilité partielle

Entrée: Un G-réseau $N =< P, T, F, m >$,
$P' \subseteq P$ et un marquage $m \in \mathbb{N}^{|P|}$.

Sortie: $\begin{cases} \textit{oui} & \text{si } \exists m' \in Acc(N) \text{ tel que } \forall p \in P',\ m'(p) = m(p) \\ \textit{non} & \text{sinon} \end{cases}$

Ce problème est polynomialement équivalent à l'accessibilité pour les réseaux de Petri [Hac76], et la construction s'étend aux G-réseaux. Nous donnons brièvement l'idée de la réduction du problème d'accessibilité partielle au problème d'accessibilité.

Soient un G-réseau N, $P \subseteq P'$ et m partiellement défini sur P'. Considérons le marquage m' tel que :

$$\forall p \in P \begin{cases} m'(p) = m(p) & \text{si } p \in P' \\ m'(p) = 0 & \textit{sinon} \end{cases}$$

et le réseau N' qui est une variation de N, où l'exécution courante de N peut être interrompue à tout moment et les places de $P \setminus P'$ décrémentées autant que possible (pour cela, on ajoute une transition t_p par place $p \in P \setminus P'$ telle que $F(p, t_p) = 1$). Le marquage m est accessible partiellement sur P' dans N si et seulement si $m' \in Acc(N')$ (où m' compte quelques autres places de contrôle, hormis les places de P). Cette transformation est forte puisqu'elle n'ajoute que des arcs de type réseau de Petri.

4.5.2 Accessibilité et blocage

Dans cette section, nous allons montrer que les problèmes d'accessibilité et de blocage sont polynomialement équivalents.

De l'accessibilité vers le blocage

Nous réduisons l'accessibilité au blocage. Cheng et al. [CEP95] ont montré que pour les réseaux de Petri, l'accessibilité se réduit au blocage et nous généralisons ce résultat aux G-Post + Reset.

Proposition 47 *Pour les G-Post + Reset, Accessibilité $\leq_{poly}$ Blocage.*

Preuve : étant donné un G-Post+Reset $N =< P, T, F, m_0 >$ à k places, nous voulons décider si un marquage $m \in Acc(N)$. Nous construisons $N_m =< P_m, T_m, F_M, m'_o >$ tel que :

- $P_m = P \cup \{p_c, p_{test}\}$.
- $T_m = T \cup \{t_{leurre}, t_{test}\} \cup_{p \in P} \{t_p\}$.

- F_m est représentée à la figure 4.2. $F'(x,y) = F(x,y)$ si $(x,y) \in (P \times T) \cup ((T \times P)$. De plus F_m est complétée par :
  - $\forall p \in P,\ F(p, t_p) = F(t_p, p) = F(p_{test}, t_p) = F(t_p, p_{test}) = 1$.
  - $F(t_{leurre}, p_c) = F(p_c, t_{leurre}) = 1$ et $\forall t \in T,\ F(p_c, t) = F(t, p_c) = 1$.
  - $F(p_c, t_{test}) = F(t_{test}, p_{test}) = 1$ et $\forall p \in P,\ F(p, t_{test}) = m(p)$.

  Par défaut, les autres arcs sont étiquetés par 0.
- $m'_0 = (m_0(1), .., m_0(k), 1, 0)$ où les k premières places sont celles de P, suivies de p_c et de p_{test}.

La place p_c est une place de contrôle sur $T \cup \{t_{leurre}\}$.

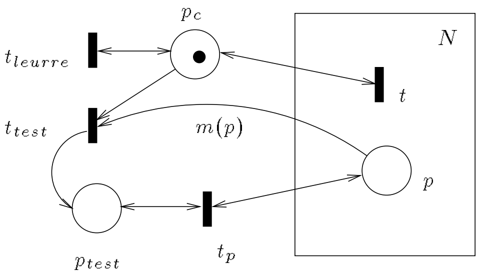


FIG. 4.2 – *Réduction de l'accessibilité vers le blocage.*

Correction : $m \in Acc(N)$ si et seulement si N_m peut atteindre un blocage.

($\Rightarrow$) Si $m \in Acc(N)$, alors $m' = (m(1), .., m(k), 1, 0) \in Acc(N_m)$ et nous avons $m' \xrightarrow{t_{test}}$, menant à $(0, .., 0, 0, 1)$ qui est un blocage pour N_m.

($\Leftarrow$) Si $m \notin Acc(N)$ alors dans N_m :

- Si p_c est marquée alors au moins t_{leurre} est franchissable.
- Si p_c est vide, alors t_{test} a été franchie, marquant p_{test}. Dès que p_{test} est marquée, elle le reste, les seules transitions franchissables sont celles de type t_p et le contenu des places de P ne varie plus. Puisque $m \notin Acc(N)$, il reste au moins un jeton dans une place $p \in P$ et t_p est franchissable.

N_m ne peut donc pas atteindre de blocage.

Complexité : la transformation est clairement polynomiale.

Nous en déduisons que pour les G-Post + Reset, Accessibilité $\leq_{poly}$ Blocage. □

Du blocage vers l'accessibilité

Nous voyons à présent l'autre sens de la réduction, du blocage vers l'accessibilité, qui nécessite une construction plus technique. Nous voulons décider si un G-Post + Reset $N =< P, T, F, m_0 >$ à k places, peut accéder à un blocage. Nous transformons ce problème en un problème d'accessibilité partielle.

Définition 48 *Soit N un G-Post + Reset.*

- *Soit $t \in T$, ${}^{\triangleright}t = \{p \in P \mid F(p,t) \in \mathbb{N}$ et $F(p,t) \geq 1\}$.*
- *$V_g(N) = \mathtt{max}_{t \in T}\ \{F(p,t) \mid p \in {}^{\triangleright}t\}$.*

${}^{\triangleright}t$ est une variation de ${}^{\bullet}t$ pour les G-Post + Reset; c'est l'ensemble des places d'entrée de t qui constituent une garde effective pour t (type réseau de Petri). Dans les G-Post + Reset, les pré-arcs sont soit de type Petri, soit de type Reset, mais seuls les Petri sont contraignants pour le franchissement des transitions. $V_g(N)$ (" g " pour "garde") est la valeur maximale qu'une transition peut retrancher d'une place, lors d'un franchissement.

Un blocage dans N est un marquage $m \in Acc(N)$ qui ne permet le franchissement d'aucune transition. Pour décrire un marquage mort, il n'est pas nécessaire de l'exhiber entièrement. Une description partielle sur un sous-ensemble $S \subseteq P$ suffit, si ce sous-ensemble satisfait l'expression booléenne suivante :

$$\wedge_{t \in T} \vee_{p \in (S \cap {}^{\triangleright}t)} [m(p) < F(p,t)] \tag{4.1}$$

Dans ce cas, on est sûr que m empêche le franchissement de toutes les transitions de T, parce qu'au moins une place par transition t ne contient pas assez de jetons pour franchir t. Remarquons qu'il suffit de chercher un m tel que $\forall p \in S, m(p) \leq V_g(N)$.

Nous allons construire un réseau $N' =< P', T', F', m'_0 >$ dont le comportement se décrit sur quatre niveaux de contrôle. Au premier niveau, N' devine $S \subseteq P$ et un marquage partiel $m_S \in \mathbb{N}^{|P|}$ sur S. Au second niveau, N' vérifie que m_S est mort pour N. Si c'est le cas, N' peut passer au niveau suivant. Au troisième niveau, N' simule N et, de façon non-déterministe, stoppe sur un marquage m_N. Au quatrième niveau, N' compare m_S et m_N sur S. S'ils sont semblables, alors m_N est un blocage pour N. De plus, $\forall p \in S,\ m_S(p) - m_N(p) = 0$ et ceci est caractérisable par la propriété d'accessibilité.

Structure générale du réseau N' :

Nous illustrons la construction à l'aide de la figure 4.3 sur laquelle nous mettons en évidence plusieurs modules (dans les rectangles). Les opérations accomplies sur les modules dépendent du niveau de contrôle courant du réseau. Chaque niveau de contrôle est géré par une place de contrôle. Nous utilisons des lignes pointillées pour mettre en évidence la relation entre une place de contrôle et un module.

Soit $N =< P, T, F, m_0 >$ un G-Post + Reset à k places. Nous utilisons l'abréviation V_g pour $V_g(N)$. Nous construisons un réseau $N' =< P', T', F', m'_0 >$ comme suit :

- $P' = P\ \cup_{p \in P} \{Oui_p,\ Non_p,\ p',\ \bar{p}'\} \cup \{p_{sat},\ p_{c1},\ p_{c2},\ p_{c3},\ p_{c4}\} \cup P_{Valider}$
  où $P_{Valider}$ est un ensemble de t places appartenant au module de validation qui est décrit plus loin.

- $T' = T \cup_{p \in P} \{choix_p,\ val_p,\ comp_p,\ decr_p\} \cup \{t_{1\to 2},\ t_{2\to 3}, t_{3\to 4}\} \cup T_{Valider}$
  où $T_{Valider}$ est l'ensemble des transitions du module de validation.

- $F'(x,y) = F(x,y)$ si $(x,y) \in (P \times T) \cup (T \times P)$. Le reste de la fonction F' est donnée par la figure 4.3 et par le détail du module de validation.

- $m'_0 = (m_0(1), .., m_0(k), 0, .., 0, 1, .., 1, 0, .., 0, V_g, .., V_g, 0, .., 0, 0, 1, 0, 0, 0)$ où les places sont ordonnées comme suit : k places de P, k places Oui_p, k places Non_p, k places p', k places $\bar{p}'$, t places de $P_{Valider}$, p_{sat}, p_{c1}, p_{c2}, p_{c3}, p_{c4}.

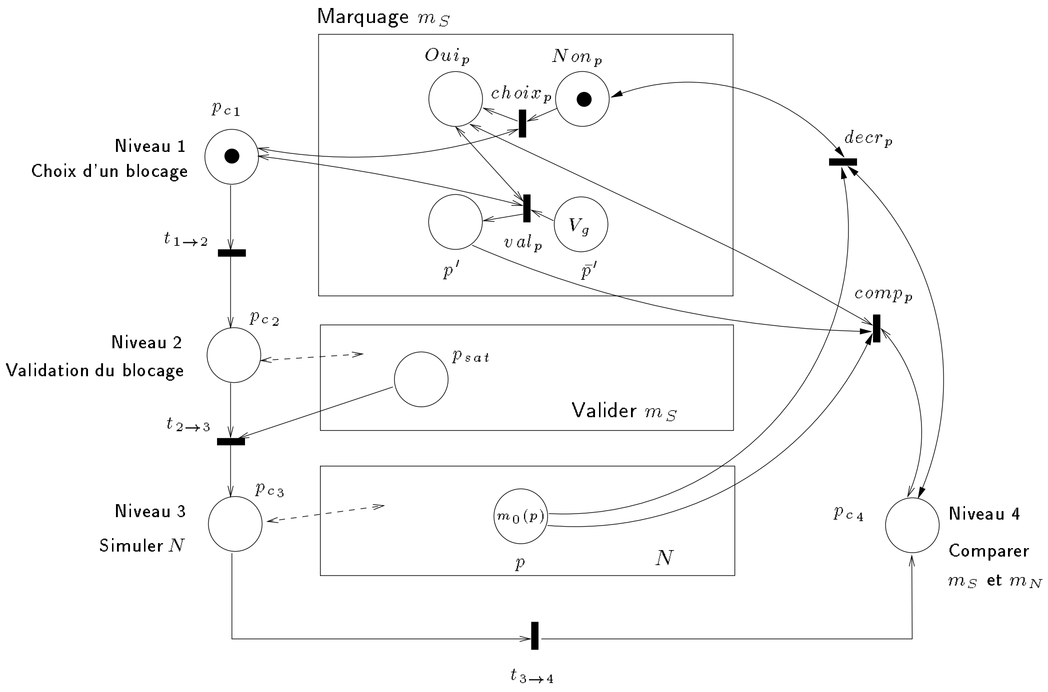


FIG. 4.3 – *Réduction du blocage vers l'accessibilité.*

Voici la description du comportement de N' aux quatre niveaux de contrôle :

- *Niveau 1* : N' choisit de façon non-déterministe un sous-ensemble $S \subseteq P$. Pour que p appartienne à S, il faut franchir $choix_p$ qui retire le jeton de Non_p et le met dans Oui_p. N' choisit ensuite m_S, un marquage bloquant pour N, et partiellement défini sur S seulement. Pour cela, à toute place $p \in P$, on associe deux nouvelles places : p', initialisée à 0, et sa complémentaire $\bar{p}'$, initialisée à V_g (p' et $\bar{p}'$ sont complémentaires pour $T' \setminus \{comp_p, decr_p \mid p \in P\}$). Il est suffisant de deviner un marquage m_S dont aucune des valuations ne dépasse V_g. Si $p \in S$, alors le choix de $m_S(p)$ dépend du nombre de fois que val_p est franchie : val_p retire un jeton de p' et le met dans $\bar{p}'$. À la fin du processus, p' contient $m_S(p)$ jetons et $\bar{p}'$ contient $V_g - m_S(p)$ jetons.

- *Niveau 2* : le réseau valide le marquage partiel m_S, qui apparaît concrètement dans les places p' de N', comme étant effectivement un blocage pour N. Le module de validation est décrit plus loin. Au terme de la validation, si m_S est mort pour N, alors la place p_{sat} est marquée. C'est la condition nécessaire pour que le réseau accède au niveau suivant.

- *Niveau 3* : N' simule N qui peut évoluer tant que voulu jusqu'à atteindre un marquage que nous notons m_N. N' passe au niveau suivant.

- *Niveau 4* : le réseau N' compare m_S (dans les places p') et m_N. Si pour $p \in S$, $m_S(p) = m_N(p)$, alors il existe un blocage pour N. Pour toute place $p \in S$ (dont

Oui_p est marquée), N' peut décrémenter simultanément p et p' par la transition $comp_p$, jusqu'à ce que les deux places soient vides. Pour toute place $p \notin S$ (dont Non_p est marquée), N' peut décrémenter p par la transition $decr_p$, jusqu'à ce qu'elle soit vide. Si $\forall p \in S, m_S(p) = m_N(p)$ alors toutes les places de P et toutes les places de type p' peuvent être vidées (les places p' telles que $p \notin S$ sont restées vides depuis le début). Ceci s'exprime en terme d'accessibilité partielle.

Détail du module de validation

Nous détaillons le module $Valider$ qui vérifie que m_S est mort pour N. Le module apparaît à la figure 4.4. Nous notons m'_S le marquage qui représente concrètement m_S dans N' : m'_S utilise les places de type Oui_p, p' et $\bar{p}'$. Nous avons : $S = \{p \mid m'_S(Oui_p) = 1\}$ et $\forall p \in P,\ m_S(p) = m'_S(p')$. Pour vérifier que m'_S implémente bien un marquage m_S mort pour N, il faut résoudre l'équation suivante :

$$\wedge_{t\in T} \vee_{p\in(S\cap {}^{\triangleright}t)} [m'_S(p') < F(p,t)]$$

Pour cela, nous tirons profit du fait que p' et $\bar{p}'$ sont complémentaires pour les transitions actives aux niveaux 1 et 2. Tant que N' est au niveau 2 nous avons l'invariant suivant : $\forall p \in P,\ m'_S(p') + m'_S(\bar{p}') = V_g$. Nous pouvons transformer l'expression précédente en :

$$\wedge_{t\in T} \vee_{p\in(S\cap {}^{\triangleright}t)} [m'_S(\bar{p}') \geq V_g - F(p,t) + 1]$$

qui est implémentable par un réseau de Petri. Nous la raffinons en :

$$\wedge_{t\in T} \vee_{p\in {}^{\triangleright}t} [(m'_S(\bar{p}') \geq V_g - F(p,t) + 1) \wedge (m'_S(Oui_p) = 1)]$$

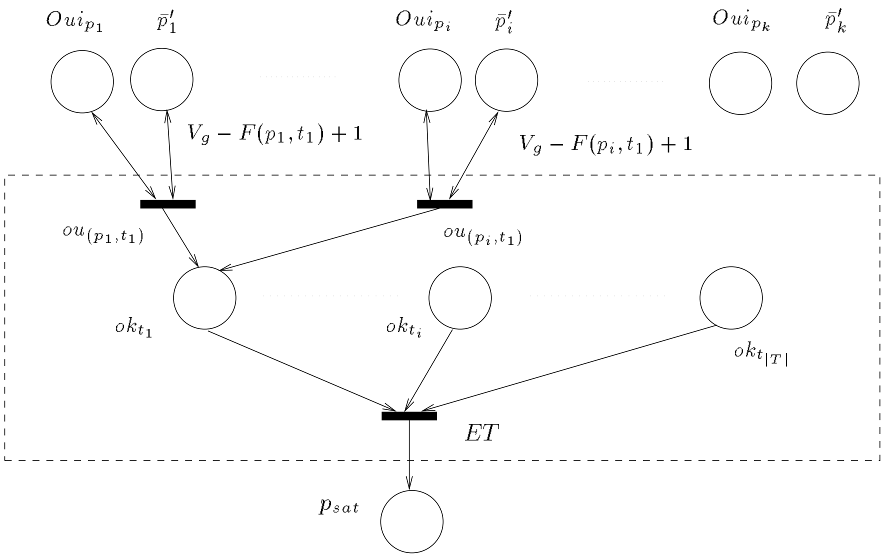


FIG. 4.4 – *Détail du module de validation*

Pour le module de validation, nous avons :

– $P_{Valider} = \{ok_t \mid t \in T\}$ (la place p_{sat} est déjà répertoriée dans P').

- $T_{Valider} = \{ET\} \cup_{t \in T, p \in {}^{\triangleright}t} \{ou_{(p,t)}\}$.

De plus, la fonction F' sur le module de validation est telle que :

- $\forall t \in T,\ p \in {}^{\triangleright}t : F(Oui_p, ou_{(p,t)}) = F(ou_{(p,t)}, Oui_p) = F(ou_{(p,t)}, ok_t) = 1$
- $\forall t \in T,\ p \in {}^{\triangleright}t : F(\bar{p}', ou_{(p,t)}) = F(ou_{(p,t)}, \bar{p}') = V_g - F(p,t) + 1$
- $\forall t \in T,\ F(ok_t, ET) = 1$
- $F(ET, p_{sat}) = 1$

Lorsque m_S comporte une place $p \in S$ qui empêche le franchissement d'une transition t, alors $ou_{(p,t)}$ est franchissable, et son franchissement marque ok_t. De plus, c'est le seul cas où ok_t peut être marquée.

Exemple : sur la figure 4.4 avons représenté le test opéré pour la transition t_1 qui deux places en entrée : p_1 et une place arbitraire p_i.

Lorsque toutes les places de type ok_t sont marquées, alors la transition ET est à son tour franchissable. Franchir ET marque p_{sat}. Le marquage m_S est alors validé comme mort pour N.

Nombre d'éléments : le module de validation n'implique que des transitions de type réseau de Petri. Pour l'implémenter, nous avons besoin de $O(|T|)$ places et $O(|T|.|P|)$ transitions. Le maximum des valuations apparaissant sur les arcs est $(V_g + 1)$, codé en $O(\log V_g)$ bits. La taille totale est en $O(|T|.|P|.|T|.\log V_g)$, ce qui est polynomial dans la taille du réseau de départ N.

Remarque : *par des constructions très semblables, on peut réduire en temps polynomial le problème de satisfaisabilité d'une expression booléenne (ou des variantes de ce problème) à des problèmes, par exemple, d'accessibilité ou de NON-vivacité pour des sous-classes des réseaux de Petri. On montre ainsi pour de petites classes de réseaux de Petri que ces problèmes sont* NP-*difficiles. Par exemple, dans le livre de* Desel *et* Esparza *[DE95], on montre que la NON-vivacité pour les réseaux à choix libres est* NP-*difficile.*

Proposition 48 *Pour les G-Post + Reset, Blocage $\leq_{poly}$ Accessibilité.*

Preuve : considérons un G-Post + Reset N et le réseau N' construit comme nous venons de le décrire.

Correction : nous prouvons que N peut atteindre un blocage si et seulement si N' peut atteindre un marquage m' tel que $m'(p_{c4}) = 1$ (m' est un marquage du niveau 4) et $\forall p \in P$, $m'(p) = m'(p') = 0$.

($\Rightarrow$) Supposons qu'il existe un blocage dans N et montrons que $m' \in Acc(N')$.

S'il existe un blocage dans N, alors au niveau 1 il existe une façon de choisir un $S \subseteq P$ et un m_S qui soit un blocage accessible par N. Le marquage m_S peut être validé comme mort pour N au niveau 2, marquant ainsi p_{sat} et permettant de passer au niveau 3. Là, il existe une façon d'accéder à $m_N = m_S$ dans le module correspondant à N. Enfin, il existe au niveau 4 une façon de vider dans N' simultanément pour $p \in S$ les places p de N et p', et de vider indépendamment les places p de N telles que $p \notin S$. Les places de type p' pour $p \notin S$ sont vides au début et n'ont pas été modifiées. Nous en concluons que $m' \in Acc(N')$.

($\Leftarrow$) Supposons qu'il n'existe pas de blocage dans N et montrons que $m' \notin Acc(N')$.

S'il n'existe pas de blocage dans N, alors il y a deux cas de figure pour N' :

1. Le marquage choisi au niveau 1 dans N' n'est pas mort pour N. La place p_{sat} ne sera pas marquée et le réseau ne pourra pas accéder au niveau 4, donc $m' \notin Acc(N')$.
2. Le marquage m_S choisi au niveau 1 dans N' est mort pour N, donc forcément $m_S \notin Acc(N)$. Donc, pour tout m_N obtenu au niveau 3, $\exists p \in S$ telle que $m_S(p) \neq m_N(p)$ et il sera impossible de vider p au niveau 4 simultanément avec la place p' associée à p. Nous en concluons que $m' \notin Acc(N')$.

Complexité : la taille de N est $\Theta(|P|^2.|T|.L(N).\log V(N))$ (où $L(N)$ est le nombre maximal de termes d'un polynôme et $V(N)$ la valuation maximale parmi les exposants, coefficients et valeurs du marquage initial). Voyons la taille de N'.

- *Places :* le module de validation contient $O(|T|)$ places. Par ailleurs, il y a les $|P|$ places pour P, $O(|P|)$ places pour deviner le marquage bloquant et gérer les différents contrôles. En tout, nous comptons $O(|P| + |T|)$ places.
- *Transitions :* le module de validation contient $O(|P|.|T|)$ transitions. Par ailleurs, il y a $O(|P|+|T|)$ transitions pour gérer les différentes opérations et exécuter le réseau N. En tout, nous comptons $O(|P|.|T|)$ transitions.
- *Valuation :* le nombre maximal de termes d'un polynôme pour N' reste $L(N)$ car nous n'ajoutons que des arcs de type réseaux de Petri, et la valuation maximale de N' reste $V(N)$.

Au total, la taille de N' est en $O((|P|+|T|)^2.(|P|.|T|).L(N).V(N))$ soit dans $O(|T|.|N|^2)$. La construction est directe et ne nécessite aucun calcul qui fasse éclater cette borne. Nous en déduisons que la transformation est polynomiale.

Puisque l'accessibilité partielle se réduit polynomialement à l'accessibilité, nous en concluons que pour les G-Post + Reset, Blocage $\leq_{poly}$ Accessibilité.

$\square$

La borne de complexité est évaluée très largement. Pour les réseaux de Petri, nous obtenons une complexité-temps en $O(max(|P|, |T|).|N|)$.

Remarque : *la transformation est forte.*

Des propositions 47 et 48, et parce que les transformations sont fortes, nous obtenons le théorème suivant :

Théorème 49

- *Pour les réseaux G-Post + Reset, l'accessibilité et le bloquage sont polynomialement équivalents.*

- *Pour les réseaux de Petri, l'accessibilité et le blocage sont polynomialement équivalents.*

4.5.3 Comparaison de l'accessibilité et des autres problèmes

En réduisant le blocage à l'accessibilité, nous venons de montrer que l'accessibilité est suffisamment riche pour exprimer, en temps polynomial, une propriété difficile. Dans cette section nous mettons de nouveau en évidence la richesse de l'accessibilité, en montrant comment lui réduire la couverture et la terminaison (dans le cas des G-Post+Reset), ainsi que le caractère borné du réseau et le caractère structurellement borné (dans le cas des G-Post+Transfert).

De la couverture vers l'accessibilité

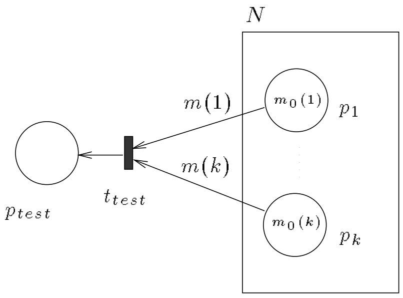
Nous montrons comment réduire la couverture à l'accessibilité. Cette première transformation est simple et reprend l'idée de la réduction de la couverture vers le caractère borné d'une place (section 4.2).

Théorème 50 *Pour les G-Post + Reset, Couverture $\leq_{poly}$ Accessibilité.*

Preuve : soit N un G-Post + Reset à k places. Nous voulons décider si m est couvert dans N. Nous construisons N_m à partir de N, en ajoutant une place p_{test} initialisée à 0, une transition t_{test} et en mettant à jour la fonction de flot pour t_{test} de la façon suivante :

$$F_m(t_{test}, p_{test}) = 1 \text{ et } \forall p \in P,\ F_m(p, t_{test}) = m(p)$$

Nous obtenons le réseau suivant :



Correction : m est couvert dans $N \Leftrightarrow t_{test}$ peut être franchie dans $N_m \Leftrightarrow p_{test}$ peut contenir 1 jeton.

Complexité : la transformation est clairement linéaire.

Nous en déduisons que pour les G-Post + Reset, Couverture $\leq_{poly}$ Accessibilité. □

Remarque : *cette transformation est forte.*

De la NON-terminaison vers l'accessibilité

Nous allons maintenant réduire la NON-terminaison à l'accessibilité. La construction est plus technique que pour la couverture.

Soit N un réseau G-Post + Reset à k places. Nous voulons décider de la terminaison de N. Rappelons que les G-Post + Reset sont bien structurés et donc que N ne termine pas si et seulement s'il existe une séquence $\sigma \in T^+$ telle que :

$$m_0 \xrightarrow{*} m \xrightarrow{\sigma} m', \text{ et } m \leq m'$$

Le réseau que nous construisons va deviner de façon non-déterministe les marquages m et m', puis les comparer en les soustrayant, place à place. Nous illustrons la construction du réseau $N_A =< P_A, T_A, F_A, m_{0A} >$ ($_A$ pour Accessibilité) à la figure 4.5.

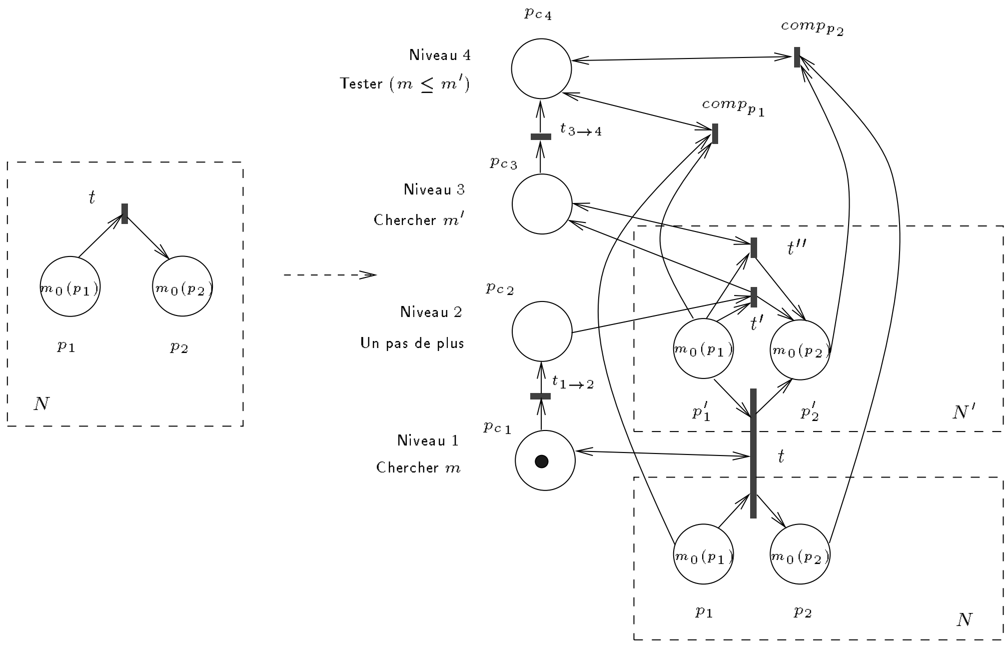


FIG. 4.5 – *Réduction de la terminaison vers l'accessibilité.*

Soit $N' =< P', T', F', m'_0 >$ une copie de N et soit T'' une copie de T. Nous définissons N_A à partir de N, de N' et de T'' comme suit :

- $P_A = P \cup P' \cup \{p_{c1}, p_{c2}, p_{c3}, p_{c4}\}$ avec $P = \{p_1, .., p_k\}$ et $P' = \{p'_1, .., p'_k\}$.
- $T_A = T \cup T' \cup T'' \cup_{p\in P} \{comp_p\} \cup \{p_{1\to2}, p_{3\to4}\}$ avec $T = \{t_1, .., t_{|T|}\}$, $T' = \{t'_1, .., t'_{|T|}\}$ et $T'' = \{t''_1, .., t''_{|T|}\}$.
- La fonction de transition F_A est représentée à la figure 4.5. Voyons sa définition plus formellement. Considérons d'abord les arcs qui ont trait à N et N' :

$$F_A(x, y) = \begin{cases} F(x, y) & \text{si } (x, y) \in (P \times T) \cup (T \times P) \\ F'(x, y) & \text{si } (x, y) \in (P' \times T') \cup (T' \times P') \end{cases}$$

Ensuite, T'' copie T' :

$$\forall j \in [1..|T|], \forall i \in [1..k], F_A(t''_j, p'_i) = F_A(t'_j, p'_i) \text{ et } F_A(p'_i, t''_j) = F_A(p'_i, t'_j)$$

De plus, nous lions P et P' par le jeu de transitions T :

$$\forall j \in [1..|T|], \forall i \in [1..k] \ : F_A(p'_i, t_j) = F_A(p'_i, t'_j) \text{ et } F_A(t_j, p'_i) = F(t'_j, p'_i)$$

Considérons les arcs qui ont trait à la soustraction place à place $m'(i) - m(i)$:

$$\forall i \in [1..k], F_A(p_i, comp_{p_i}) = F_A(p'_i, comp_{p_i}) = 1$$

Considérons à présent les arcs de contrôle : au niveau 1, p_{c1} est une place de contrôle pour T. La transition $t_{1\to 2}$ fait passer le contrôle au niveau 2. À ce niveau, nous avons $\forall t' \in T' \ : F_A(p_{c2}, t') = F_A(t', p_{c3}) = 1$. Toute transition de T' fait passer le réseau au niveau 3. À ce niveau, p_{c3} est une place de contrôle pour T''. La transition $t_{3\to 4}$ fait passer le contrôle au niveau 2. À ce niveau, p_{c4} est une place de contrôle pour toutes les transitions de type $comp_p$ ($\forall p \in P$).

Par défaut, les autres arcs sont étiquetés par 0.

- $m_{0A} = (m_0(1), .., m_0(k), m_0(1), .., m_0(k), 1, 0, 0, 0)$ où les k premières places sont celles de P, les k suivantes celles de P', puis p_{c1}, p_{c2}, p_{c3} et p_{c4}.

Notation : *nous utilisons N pour désigner le réseau N de départ et nous utilisons $N_A[N]$ pour désigner la partie N de N_A.*

Voici le comportement attendu de N_A qui va chercher m et m' tels que $m_0 \xrightarrow{*} m \xrightarrow{\sigma} m'$ avec $|\sigma| > 0$ et $m \leq m'$.

- Le *niveau 1* autorise les deux réseaux $N_A[N]$ et N' à franchir les transitions de T simultanément. Durant toute cette étape, $N_A[N]$ et N' contiennent successivement les mêmes marquages. De façon non déterministe, le réseau peut passer au niveau 2 en franchissant $t_{1\to 2}$.

- Au *niveau 2*, $N_A[N]$ est gelé et contient le marquage qui a été choisi pour m. Ici, seule une transition parmi l'ensemble T' sera franchie, modifiant N' et faisant passer N_A au niveau 3. Le niveau 2 valide la transition obligatoire qui doit exister entre m et m'.

- Le *niveau 3* laisse le réseau N' évoluer librement sur l'ensemble de transitions T''. De façon non déterministe, le réseau peut passer au niveau 4 en franchissant $t_{3\to 4}$. À ce moment, le marquage apparaissant dans N' est le marquage choisi comme étant m'.

- Le *niveau 4* compare m et m'. Si $m \leq m'$, alors $\forall i, 1 \leq i \leq k, m'(i) - m(i) \geq 0$ et il est possible de vider les places de $N_A[N]$ en franchissant autant de fois que nécessaire les transitions de type $comp_p$. Une transition $comp_p$ décrémente simultanément $p \in P$ et la place $p' \in P'$ qui lui correspond.

Le contrôle est donné d'abord au niveau 1, puis passe aux niveaux 2, 3 puis 4.

Proposition 51 *Pour les G-Post + Reset, NON-Terminaison $\leq_{poly}$ Accessibilité.*

Preuve : considérons un G-Post + Reset N et le réseau N_A construit comme nous venons de le décrire.

Correction : nous montrons que N ne termine pas si et seulement s'il existe un marquage $m_A \in Acc(N_A)$, tel que $m_A(p_{c4}) = 1$ (au niveau 4) et tel que $\forall p \in P, m_A(p) = 0$, c'est-à-dire que les places de $N_A[N]$ sont vides.

($\Rightarrow$) Supposons que N ne termine pas et montrons que $m_A \in Acc(N_A)$.

Si N ne termine pas, alors il existe une exécution $m_0 \xrightarrow{*} m \xrightarrow{\sigma} m'$ telle que $\sigma \in T^+$ et $m \leq m'$. Au niveau 1, N_A peut choisir m dans $N_A[N]$. Aux niveaux 2 et 3, N_A peut choisir m' dans N'. Enfin, au niveau 4, N_A peut comparer les deux marquages, les soustraire place à place et vider les places de $N_A[N]$. Nous en concluons donc que $m_A \in Acc(N_A)$.

($\Leftarrow$) Supposons que N termine et montrons que $m_A \notin Acc(N_A)$.

Tant que N_A est au niveau 1, 2 ou 3, le marquage m_A n'est pas atteint. Appelons m le marquage apparaissant dans $N_A[N]$ au moment où le réseau franchit $t_{1\to 2}$. Appelons m' le marquage apparaissant dans N' au moment où le réseau franchit $t_{3\to 4}$. Par construction de N_A, nous avons dans N : $m_0 \xrightarrow{*} m \xrightarrow{\sigma} m'$ avec $\sigma \in T^+$ (à cause du niveau 2 qui force un franchissement pour autoriser l'accès au niveau 3). Puisque N termine, $m \nleq m'$. Il y a deux cas : soit $m > m'$, soit m et m' sont incomparables. Dans les deux cas, il existe forcément au moins une position i telle que $m(i) > m'(i)$ et on ne pourra pas vider la place p_i de $N_A[N]$ au niveau 4. Nous en concluons que $m_A \notin Acc(N_A)$.

Complexité : voyons la taille de N_A. Nous comptons $O(|P|)$ places, $O(|T|)$ transitions et les valuations sur les arcs restent de la même taille que dans N. La taille de N_A est donc linéaire dans la taille de N. La construction est effectuée en temps linéaire.

Nous en concluons que pour les G-Post + Reset, NON-Terminaison $\leq_{poly}$ Accessibilité. □

Remarque : *cette transformation est forte.*

Du caractère NON-borné du réseau vers l'accessibilité

Nous réduisons le caractère NON-borné du réseau à l'accessibilité, pour G-Post + Transfert.

Soit N un réseau G-Post + Transfert à k places. Nous voulons décider du caractère NON-borné de N. Rappelons que puisque les G-Post + Transfert sont strictement structurés, N n'est pas borné si et seulement s'il existe deux marquages $m, m' \in \mathbb{N}^k$ tels que :

$$m_0 \xrightarrow{*} m \xrightarrow{*} m' \text{ et } m < m'$$

La construction que nous allons donner est une variante de la construction précédente pour la terminaison. Il faut cette fois vérifier que $m < m'$. Le réseau $N_A =< P_A, T_A, F_A, m_{0A} >$ que nous construisons apparaît à la figure 4.6.

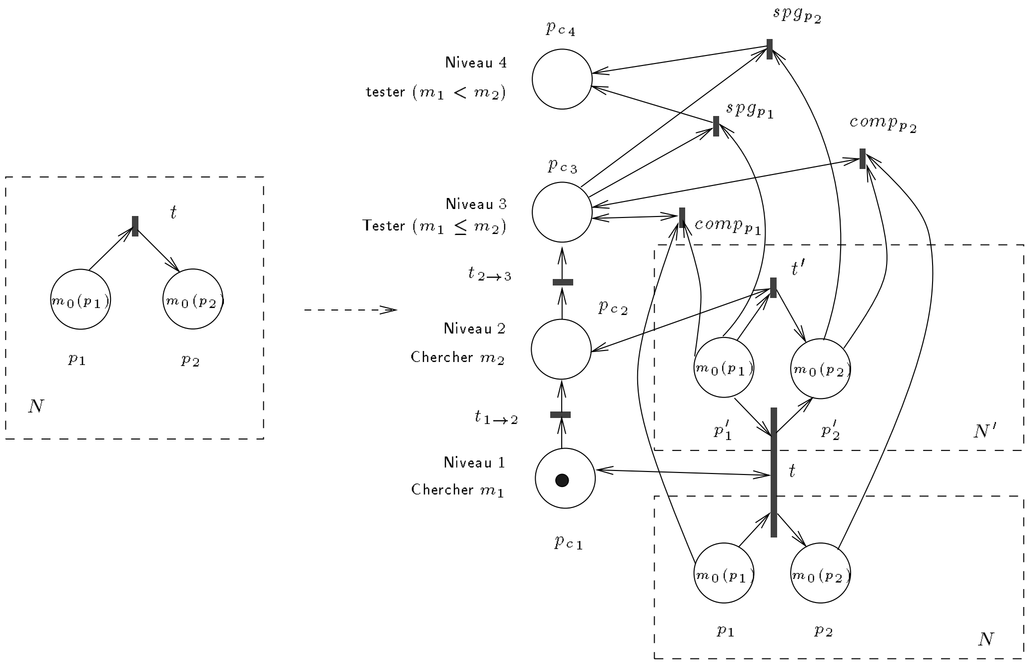


FIG. 4.6 – *Réduction du caractère NON-borné du réseau vers l'accessibilité.*

Soit $N' =< P', T', F', m'_0 >$ une copie de N. Nous définissons N_A à partir de N et de N', comme suit :

- $P_A = P \cup P' \cup \{p_{c1}, p_{c2}, p_{c3}, p_{c4}\}$.
- $T_A = T \cup T' \cup_{p \in P} \{comp_p, spg_p\} \cup \{t_{1\to 2},\ t_{2\to 3}\}$ (avec *comp* pour "comparer" et *spg* pour "strictement plus grand").
- La fonction F_A est représentée à la figure 4.6.
- $m_{0A} = (m_0(1), .., m_0(k), m_0(1), .., m_0(k), 1, 0, 0, 0)$ où les k premières places sont celles de P, les k suivantes celles de P' et les quatre autres p_{c1}, p_{c2}, p_{c3} puis p_{c4}.

N_A possède quatre niveaux de contrôle. Le comportement attendu de N_A est le suivant :

- Au *niveau 1*, N_A devine m dans $N_A[N]$.
- Au *niveau 2*, N_A devine m' dans N'.
- Au *niveau 3*, N_A vérifie que $m \leq m'$ en soustrayant m de m' place à place, par les transitions de type $comp_p$. Si $m \leq m'$ alors on peut vider les places de $N_A[N]$.
- Au *niveau 4*, N_A vérifie que $m < m'$ puisqu'on ne peut accéder au niveau 4 que par les transitions de type spg_p, en retirant un jeton de m' qui n'a pas participé à la comparaison au niveau 3.

Proposition 52
Pour les G-Post + Transfert, Caractère NON-Borné du Réseau $\leq_{poly}$ Accessibilité.

Preuve : considérons un G-Post + Transfert N et le réseau N_A construit comme nous venons de le décrire.

Correction : N n'est pas borné si et seulement si m_A tel que $m_A(p_{c4}) = 1$ (au niveau 4) et tel que $\forall p \in P$, $m_A(p) = 0$ (c'est-à-dire que les places de $N_A[N]$ sont vides) est accessible dans N_A. La preuve est une adaptation directe de la preuve pour la terminaison.

Complexité : la construction se fait en temps linéaire.

Nous en déduisons que pour les G-Post + Transfert, Caractère NON-Borné du Réseau $\leq_{poly}$ Accessibilité. □

Remarque : *la construction est forte.*

Du caractère NON-structurellement borné vers l'accessibilité

Nous montrons maintenant comment réduire le caractère NON-structurellement borné à l'accessibilité pour les réseaux G-Post + Transfert.

Soit N un réseau G-Post + Transfert à k places. Nous voulons décider du caractère NON-structurellement borné de N. Le réseau N n'est pas structurellement borné s'il existe un marquage $m_0 \in \mathbb{N}^k$ tel que $Acc(N, m_0)$ est infini. Nous réduisons ce problème à un problème d'accessibilité.

Nous construisons un réseau $N_A =< P_A, T_A, F_A, m_{0A} >$. L'idée générale est de laisser N_A deviner un marquage m_0 à partir duquel N n'est pas borné, puis de mettre en évidence l'exécution $m_0 \xrightarrow{*} m \xrightarrow{*} m'$ avec $m < m'$. Pour cela, nous reprenons la même construction que celle que nous venons de décrire pour le caractère borné du réseau et nous ajoutons un niveau 0. Ce niveau gère une étape préliminaire de recherche d'un marquage m_0. La construction apparaît à la figure 4.7. Définissons plus formellement N_A, à partir de N et de N', une copie de N :

- $P_A = P \cup P' \cup \{p_{c0}, p_{c1}, p_{c2}, p_{c3}, p_{c4}\}$.
- $T_A = T \cup T' \cup_{p \in P} \{init_p, comp_p, spg_p\} \cup \{t_{0\to 1},\ t_{1\to 2},\ t_{2\to 3}\}$
- La fonction F_A est représentée à la figure 4.7.
- $m_{0A} = (0, .., 0, 0, .., 0, 1, 0, 0, 0, 0)$ où les k premières places sont celles de P, les k suivantes celles de P' et les cinq autres $p_{c0}, p_{c1}, p_{c2}, p_{c3}$ puis p_{c4}.

Le niveau 0 choisit de façon non déterministe un marquage initial pour N. Dans N_A, ceci revient à choisir un même marquage initial pour $N_A[N]$ et pour N'. Pour cela, on ajoute k transitions de type $init_p$. La transition $init_{p_i}$ n'est franchissable qu'au niveau 0 et incrémente simultanément p_i dans $N_A[N]$ et p'_i dans N'.

Proposition 53 *Pour les G-Post + Transfert, Caractère NON-Structurellement Borné $\leq_{poly}$ Accessibilité.*

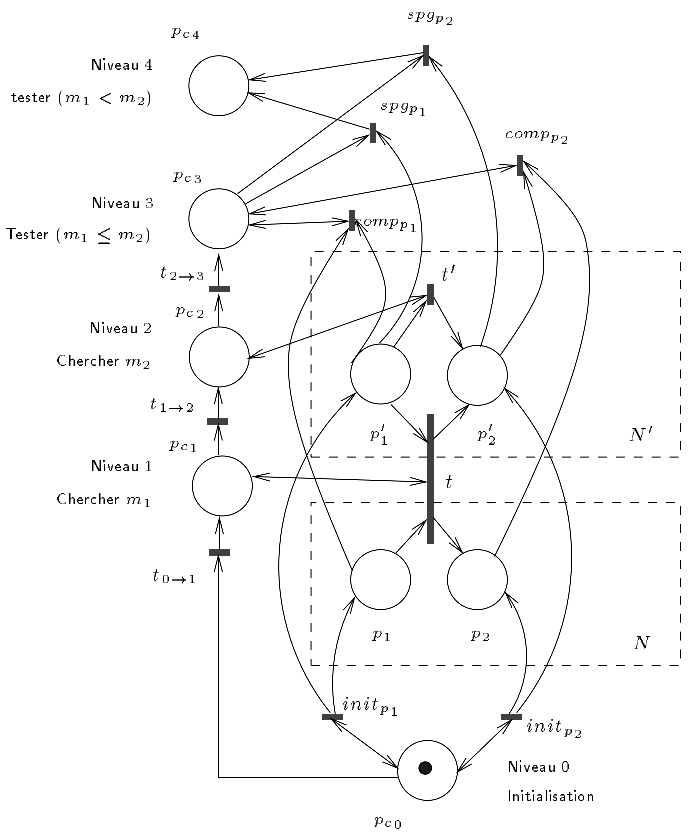


FIG. 4.7 – *Réduction du caractère NON-structurellement borné vers l'accessibilité.*

Preuve : considérons un G-Post + Transfert N et le réseau N_A construit comme nous venons de le décrire.

Correction : N n'est pas structurellement borné si et seulement si dans N_A, un marquage tel que $m_A(p_{c4}) = 1$ et tel que $\forall p \in P, m_A(p) = 0$ est accessible. La preuve est une adaptation directe des deux précédentes.

Complexité : la construction se fait en temps linéaire.

Nous en déduisons que pour les G-Post + Transfert, Caractère NON-Structurellement Borné $\leq_{poly}$ Accessibilité. □

Remarque : *la construction est forte.*

Cette construction peut être adaptée pour prouver la décidabilité du problème de caractère structurellement borné pour les G-Post, en utilisant le dépliage que nous avions décrit en section 3.1.

Proposition 54 *Pour les G-Post, le caractère structurellement borné est décidable.*

Preuve : soit $N =< P, T, F, m_0 >$ un réseau G-Post à k places.

Dans un premier temps, nous déplions N en un réseau de Petri N_D, par la construction donnée en section 3.1, sans tenir compte d'un marquage initial.

Soit $m \in \mathbb{N}^k$. Nous avons vu qu'un marquage $m_D \in \mathbb{N}^{|P_D|}$ tel que :

$$\forall p \in P_D,\ m_D(p) = \begin{cases} 1 & \text{si } p = p_{choix} \\ m(i) & \text{si } p = p_i \ (p_i \in N_D[P]) \\ m(i) & \text{si } p = M[p_{val}] \text{ avec } M \text{ de type } M_i \\ 0 & \text{sinon} \end{cases}$$

est un marquage initial pour N_D, afin de simuler N à partir de m.

Nous réduisons le caractère NON-structurellement borné de N à l'accessibilité, en construisant N_A à partir de N_D, comme décrit précédemment. Le niveau 0 choisit un marquage arbitraire pour N_D, mais sera modifié en vu de ne choisir qu'un marquage qui initialise correctement N_D pour simuler N. Ainsi, le remplissage de toutes $N_A[N_D[p_i]]$ et $N_A[p'_i]$ se fait simultanément avec toute $M[p_{val}]$ d'un module M de type M_i (dans $N_A[N_D]$ et dans N'_D). De plus, par le franchissement de $t_{0\to 1}$, les places $N_A[N_D[p_{choix}]]$ et $N'_D[p_{choix}]$ sont marquées. Lorsque N_A accède au niveau 1, $N_A[N_D]$ et N'_D sont correctement initialisés pour simuler N sur un même marquage arbitraire.

Soit $m_A \in \mathbb{N}^{|P_A|}$ tel que $m_A(p_{c4})$ et tel que $\forall p \in P_D, m_A(p) = 0$ (c'est-à-dire que toutes les places de $N_A[N_D]$ sont vides). L'équivalence suivante est directe : N n'est PAS structurellement borné si et seulement si m_A est accessible dans N_A.

Puisque N_A est un réseau de Petri et que l'accessibilité est décidable pour les réseaux de Petri, nous en déduisons que pour les réseaux G-Post, le problème du caractère structurellement borné est décidable.

□

Pour terminer cette section, mentionnons que l'idée des trois dernières réductions s'adapte aussi pour donner une réduction polynomiale du problème de l'*accessibilité répétée* vers le problème d'accessibilité (pour au moins la classe des réseaux de Petri). L'accessibilité répétée est une variante du problème d'accessibilité, permettant de vérifier des formules de logique temporelle. Ce problème est défini comme suit :

• Problème de l'accessibilité répétée

Entrée: $< N, P', m >$, un réseau de Petri N, un sous-ensemble de places P' et un marquage $m \in \mathbb{N}^{|P|}$.

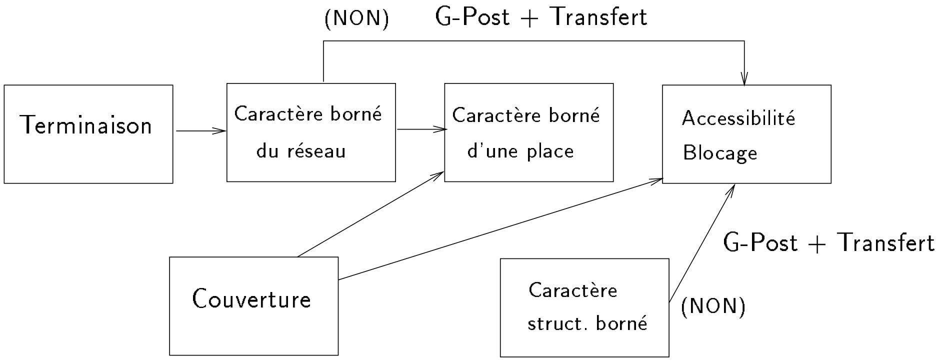
$$\textit{Sortie}: \begin{cases} \textit{oui} & \text{s'il existe une exécution au cours de laquelle} \\ & m \text{ est partiellement accessible sur } P' \text{ une infinité de fois} \\ \textit{non} & \text{sinon} \end{cases}$$

4.5.4 Récapitulation des relations entre propriétés

Nous récapitulons les relations des propriétés en terme de réductions multivoques polynomiales, pour les G-Post+Reset. Une flèche partant d'un problème A vers un problème B indique que $A \leq_{poly} B$. Toutes les réductions impliquent des transformations fortes (c'est-à-dire qu'elles préservent la classe de départ).

Sur la figure ci-dessous, on voit apparaître à deux reprises le terme *G-Post + Transfert*; ceci veut dire que la réduction n'est valable que pour cette classe de réseaux et en deçà. L'accessibilité et le blocage sont équivalents et apparaissent donc dans une même boîte. Enfin, lorsqu'une réduction concerne le complémentaire d'un problème, nous l'indiquons sur la flèche d'un NON entre parenthèses.

Voici la hiérarchie des problèmes selon les réductions multivoques polynomiales données jusqu'ici :



Parler de transformations *polynomiales* n'a de sens que un contexte de décidabilité : puisque nos réductions sont fortes, pour les réseaux de Petri de nombreuses propriétés, dont le caractère borné du réseau et le blocage, se réduisent polynomialement à l'accessibilité. Ainsi, toutes ces propriétés peuvent être vérifiées en n'utilisant qu'un seul outil résolvant l'accessibilité et accompagné de traducteurs pour les divers problèmes. Le résultat récent de Bouziane [Bou98b] qui donne un algorithme primitif-récursif pour l'accessibilité dans les réseaux de Petri renforce l'intérêt de ces réductions.

4.6 Problèmes de vivacité

Nous montrons que pour les G-Post + Reset, la vivacité et la t-vivacité sont polynomialement équivalentes.

De la t-vivacité vers la vivacité

Proposition 55 *Pour les G-Post + Reset, t-Vivacité $\leq_{poly}$ Vivacité.*

Preuve : étant donné un réseau G-Post + Reset $N =< P, T, F, m_0 >$ et une transition $t \in T$, nous voulons décider si t est vivante dans N. Nous transformons $< N, t >$ en un réseau $N' =< P', T', F', m_0' >$ tel que t est vivante dans N si et seulement si N' est vivant.

Nous illustrons la construction à la figure 4.8. Voici la définition de N' :

- $P' = P \cup \{p_{inf}\}$.

- $T' = T \cup \{t', t_{inf}\}$.
- $\forall (x, y) \in (T \times P) \cup (P \times T),\ F'(x, y) = F(x, y)$. De plus, la transition t' copie les places d'entrée de t :

$$\forall p \in {}^\bullet t,\ F'(p, t') = F(p, t)$$

  Le franchissement de t' marque la place p_{inf} : $F'(t', p_{inf}) = 1$.

  La transition t_{inf} peut inonder le réseau de jetons, dès que p_{inf} est marquée :

$$F'(p_{inf}, t_{inf}) = F'(t_{inf}, p_{inf}) = 1 \text{ et } \forall p \in P',\ F'(t_{inf}, p) = 1$$

- $m_0{}' = (m_0(1), .., m_0(k), 0)$ où les k premières places sont celles de P et la dernière est p_{inf}.

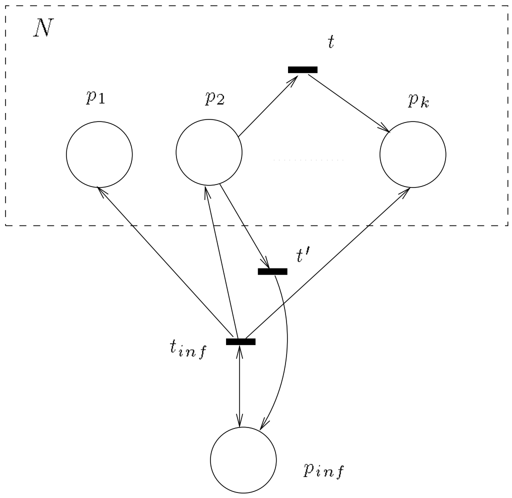


FIG. 4.8 – *Réduction de la t-vivacité vers la vivacité.*

Ainsi, partant de N on ajoute une nouvelle transition t' qui a les mêmes gardes que t et qui marque p_{inf} en sortie.

Correction : nous montrons que t est vivante dans N si et seulement si N' est vivant.

($\Rightarrow$) Supposons que t soit vivante dans N. Dans N', tant que t' n'est pas franchie, il existe une possibilité d'accéder à un marquage m tel que $\exists m',\ m \xrightarrow{t'} m'$. Franchir t' marque p_{inf} qui rend t_{inf} toujours franchissable. Or, t_{inf} peut ajouter à P autant de jetons que nécessaire pour franchir n'importe quelle transition de T. Donc, à partir de n'importe quel marquage dans $Acc(N', m')$, toute transition de N' peut devenir franchissable. Nous en concluons que N' est vivant.

($\Leftarrow$) Supposons que t ne soit pas vivante dans N, alors t n'est pas vivante dans N' et N' n'est donc pas vivant.

Complexité : la transformation est clairement polynomiale.

Nous en concluons que pour les G-Post + Reset, t-Vivacité $\leq_{poly}$ Vivacité.

□

De Vivacité vers t-Vivacité

Nous voyons l'autre sens de la réduction (simple) et nous réduisons vivacité à t-vivacité.

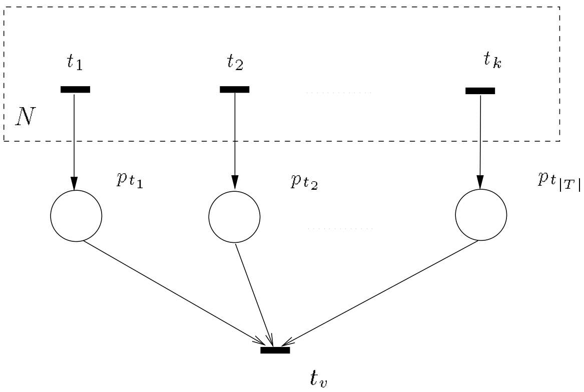


FIG. 4.9 – *Réduction de la vivacité vers la t-vivacité.*

Proposition 56 *Pour les Reset + G-Post, Vivacité $\leq_{poly}$ t-Vivacité.*

Preuve : soit un réseau G-Post+Reset $N =< P, T, F, m_0 >$, à k places, dont nous voulons décider s'il est vivant ou non. Nous construisons une instance $< N', t_v >$ telle que N est vivant si et seulement si t_v est vivante dans N'. La construction consiste à ajouter une place p_t pour toute transition $t \in T$. La place p_t sera incrémentée à chaque fois que t sera franchie. On ajoute ensuite une nouvelle transition t_v, franchissable si toutes les places p_t sont marquées. La construction est illustrée à la figure 4.9.

Correction : le réseau N est vivant si et seulement si la transition t_v dans N' est vivante. La preuve est directe.

Complexité : la transformation est clairement polynomiale.

Nous en concluons que pour les G-Post + Reset, Vivacité $\leq_{poly}$ t-Vivacité. □

On déduit le théorème suivant des propositions 55 et 56 :

Théorème 57 *Pour les Reset + G-Post, Vivacité et t-Vivacité sont polynomialement équivalentes.*

La vivacité et la t-vivacité sont polynomialement équivalentes de façon multivoque, et il est intéressant de noter que cette équivalence n'est pas toujours vérifiée pour les sous-classes de réseaux de Petri. En effet, pour les réseaux à *choix libres bornés*, la vivacité est polynomiale [ES92], alors que la t-vivacité est NP-difficile. Nous en concluons qu'une réduction polynomiale de la t-vivacité vers la vivacité ne peut PAS préserver à la fois le caractère borné et la topologie choix libres. En ce qui nous concerne, notre réduction ne préserve pas le caractère borné.

Comparaison de la vivacité et des autres problèmes

La vivacité est une propriété au moins aussi forte que la NON-accessibilité, le NON-blocage ou la t-vivacité. Il existe une réduction polynomiale de l'accessibilité vers la NON-vivacité pour les réseaux de Petri [Pet81]. La réduction part d'une variante du problème d'accessibilité partielle, qui consiste à demander si une place spécifique p peut être vidée au cours de l'exécution d'un réseau N. En voici une brève description :

La place p est transformée comme place de contrôle pour T (pour franchir $t \in T$, il faut que p contienne au moins un jeton). Une transition t_{inf} ayant p comme place d'entrée et p_{inf} comme place de sortie, est ajoutée. Tant que p contient un jeton, t_{inf} est franchissable. La place p_{inf} est initialement vide; dès qu'elle est marquée, p_{inf} peut inonder indéfiniment toutes les places de N (p_{inf} tient le même rôle que dans la réduction de la t-vivacité vers la vivacité que nous avons décrit précédemment). Si p est vidée, plus aucune transition n'est franchissable et le réseau tombe dans un blocage. Ainsi, il existe une façon de vider p dans N si et seulement si le réseau ainsi transformé est NON-vivant.

Une idée semblable est présentée par Cheng et al. en [CEP95], pour réduire le blocage à la NON-vivacité. Pour chaque transition $t \in T$, on ajoute une transition t_{copie} ayant les mêmes gardes que t et une place p_{inf} en sortie. Dès qu'une transition $t \in T$ est franchissable, la transition t_{copie} l'est aussi. Si la transition t_{copie} est franchie, alors le réseau est inondé de jetons. Ainsi, tant que le réseau n'accède pas à un blocage, il est potentiellement inondable. Le réseau N atteint donc un blocage si et seulement si le réseau ainsi transformé n'est pas vivant.

Les idées de [Pet81] et de [CEP95] s'étendent directement aux G-Post + Reset et nous obtenons le théorème suivant :

Théorème 58 *Pour les Reset + G-Post, NON-accessibilité* $\leq_{poly}$ *Vivacité et NON-Blocage* $\leq_{poly}$ *Vivacité.*

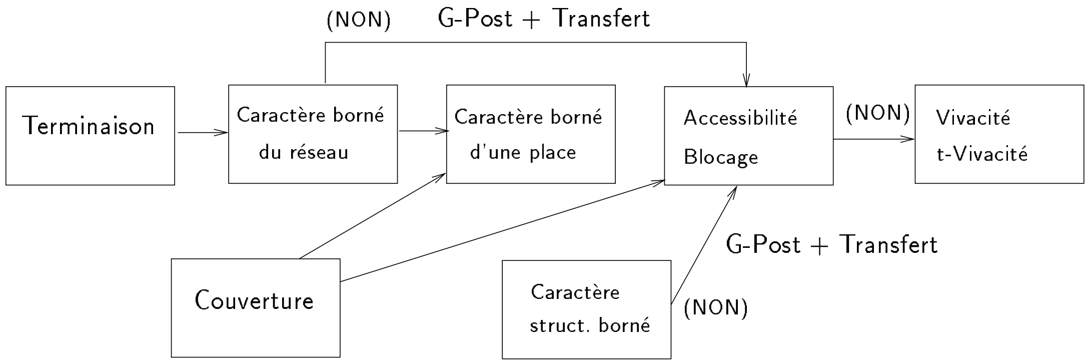
Remarque : *ces transformations sont fortes.*

Enfin, puique ces réductions sont fortes, nous obtenons le corollaire suivant :

Corollaire 59 *La vivacité et la t-vivacité sont indécidables pour toutes les sous-classes de G-Post + Reset pour lesquelles l'accessibilité est indécidable.*

4.7 Récapitulation des résultats

Au terme des études des comparaisons des problèmes à l'aide des réductions polynomiales, nous obtenons le diagramme suivant :



Voici le tableau récapitulatif des résultats de décidabilité mis à jour :

	Terminaison	Couverture	Réseau borné	Place bornée	Réseau structurellement borné	Accessibilité Blocage	Vivacité t-Vivacité
G-Réseaux *Self-Modifying*	**I**	**I**	**I**	**I**	**I**	**I**	**I**
G-Post + Reset	**D**	**D**	?	?	?	**I**	**I**
Reset	**D**	**D**	?	?	?	**I**	**I**
G-Post + Transfert	**D**	**D**	**D**	?	?	**I**	**I**
Transfert	**D**	**D**	**D**	?	?	**I**	**I**
G-Post *Post Self-Modifying*	**D**	**D**	**D**	**D**	**D**	**I**	**I**
Petri	**D**	**D**	**D**	**D**	**D**	**D**	**D**

Chapitre 5

Analyse des réseaux de Petri avec Reset

Nous avons vu dans les chapitres précédents plusieurs résultats de décidabilité et d'indécidabilité pour les réseaux de Petri dotés d'arcs de Reset. Pour cette classe, la terminaison et la couverture sont décidables, mais l'accessiblité ne l'est pas. Pour les problèmes de type "caractère borné" (du réseau, d'une place ou par structure), nous n'avons encore rien conclu. Ces problèmes ont résisté à une première investigation des techniques menées avec des systèmes de transitions bien structurés, ou dans le contexte des réseaux de Petri.

À la section 4.3 nous avons vu que la construction de l'arbre de couverture ne s'étend pas directement aux réseaux de Petri avec Reset. Dans ce chapitre, nous verrons qu'il est en fait impossible de construire de façon effective un arbre de couverture fini pour ces réseaux, car le problème du caractère borné d'un réseau avec Reset est indécidable. De plus, nous montrerons que trois arcs de Reset suffisent à rendre ce problème indécidable.

Résoudre le problème du caractère borné d'un réseau de Petri avec arcs de Reset n'a pas été immédiat. Nous avons commencé à réfléchir à ce problème en 1996, sur une idée d'Alain FINKEL. Bien sûr, ce problème est semi-décidable puisque si le réseau est borné, alors on peut construire son arbre d'accessibilité. Cependant, nous avons longtemps cherché entre *décidable* et *indécidable*, essayant d'adapter aux réseaux de Petri avec Reset l'algorithme de construction de l'arbre de couverture d'un réseau de Petri.

À la fin de l'année 1996, Philippe SCHNOEBELEN a mis en évidence un exemple de réseau avec deux arcs de Reset (section 5.1, figure 5.1) qui montre clairement que le problème du caractère borné est plus difficile que pour les réseaux de Petri. Dans ce réseau, la seule exécution produisant une infinité de jetons n'est pas formée d'une séquence itérée à l'infini. L'exécution est formée d'un motif de base qui est répété à l'infini, mais qui varie à chaque répétition pour devenir de plus en plus long, ce qui permet d'ajouter de plus en plus de jetons (nous reviendrons sur ce réseau plus loin).

Au début de l'année 1997, partant d'une idée de preuve de Philippe SCHNOEBELEN [Sch97], nous avons formulé et établi une preuve de l'indécidabilité de ce problème avec une réduction du *dixième Problème de Hilbert*. Cette *première* preuve est longue et technique, et s'énonce en quatre points : il faut d'abord décrire comment évaluer des polynômes;

puis comparer l'évaluation des polynômes; ensuite énumérer des vecteurs d'entiers et enfin agencer ces différentes constructions pour obtenir le réseau final. Le nombre d'arcs de Reset utilisés dans cette réduction dépend des polynômes et n'est donc pas borné.

En été 1997, alors que la première preuve d'indécidabilité était en rédaction, une "preuve" de *décidabilité* a été publiée dans [KCK+97]. L'idée partait d'une adaptation de l'algorithme de construction de l'arbre de couverture. Les auteurs ont pensé qu'ils pouvaient caractériser l'allure d'une séquence qui créerait une infinité de jetons, à la manière de ce qui est fait pour les réseaux de Petri. Mais le réseau de Philippe SCHNOEBELEN, ainsi que la preuve d'indécidabilité, contredisait cet énoncé.

Au début de l'année 1998, la réduction du dixième problème de Hilbert a été jointe en annexe lors d'une soumission à ICALP'98, mais n'apparaît pas dans la version publiée [DFS98], car trop longue. Dans ce chapitre, nous en rappellerons les idées principales.

Au début de l'année 1998, nous avons travaillé sur ce problème avec Petr JANČAR qui a trouvé une *seconde* preuve d'indécidabilité [Jan98], par une réduction du problème de la *terminaison d'une machine à deux compteurs*. Il y a deux avantages principaux à cette preuve. Tout d'abord, elle permet l'obtention d'une borne supérieure sur le nombre d'arcs de Reset suffisants (moins de dix) pour rendre le problème indécidable. Ensuite, elle montre la Σ_1-complétude du problème. Cependant, cette construction d'un réseau avec Reset qui simule une machine à compteurs reste aussi technique que la construction de la première preuve.

En juin 1998, au moment même de la rédaction de ce chapitre, nous avons trouvé une *troisième* preuve d'indécidabilité, simplifiant notablement la construction et amenant à une borne supérieure de *trois* arcs de Reset. Il s'agit d'une réduction du problème de *caractère borné d'une machine à deux compteurs*. Dans cette thèse, c'est cette preuve que nous donnerons en détail.

Les trois preuves d'indécidabilité sont liées, car elles utilisent les mêmes idées, tout en les raffinant. C'est la raison pour laquelle il était important de rappeler ce cheminement.

La troisième et dernière construction a aussi un autre intérêt. Elle permet d'étendre le résultat d'indécidabilité au caractère structurellement borné d'un réseau Reset, en réduisant le problème du caractère borné d'une machine à compteurs au problème du caractère structurellement borné. Mieux encore, une adaptation mineure donne une réduction du problème de *NON-terminaison d'une machine à compteurs* au problème du caractère structurellement borné. Ainsi, pour les réseaux de Petri avec Reset, ni le problème du caractère structurellement borné, ni son complémentaire, ne sont semi-décidables. Le caractère structurellement borné est donc strictement plus difficile que le caractère borné du réseau. Rappelons que pour les réseaux de Petri, c'est l'inverse, car le caractère structurellement borné est solvable en temps polynomial [MR80], alors que le caractère borné du réseau nécessite un espace exponentiel [Lip76].

À la section 5.1, nous commencerons par donner l'énoncé de [KCK+97] qui stipule que le caractère borné d'un réseau Reset est décidable, puis nous donnerons un contre-exemple à cet énoncé. À la section 5.2, nous donnerons les idées principales de la réduction du dixième Problème de Hilbert. À partir de la section 5.3, nous aborderons les constructions qui permettent de simuler des machines à compteurs à l'aide des réseaux Reset. Nous commencerons par définir les machines à compteurs en 5.3.1. À la section 5.3.2, nous réduirons le problème du caractère borné d'une machine à compteurs au problème du caractère borné d'un réseau avec arcs de Reset, montrant l'indécidabilité de ce dernier problème. Nous généraliserons ce résultat au caractère borné d'une place. Enfin, à la section 5.4, nous montrerons que pour les réseaux Reset, le problème du caractère structurellement borné est indécidable et qu'il est strictement plus difficile que le problème du caractère borné du réseau.

5.1 Une première intuition... fausse

Récemment, [KCK+97] ont donné une preuve pour le modèle des *Place-Chart Nets* (PCN), indiquant que le caractère borné du réseau est décidable. Ce fait est remarquable, car les PCN contiennent les réseaux de Petri avec Reset qui constituent une extension des réseaux de Petri. Le principe des PCN est de permettre la *préemption* de jetons, ce qui synonyme de *Reset*. Il s'agit d'ajouter au modèle de base de PETRI des transitions vidant des sous-ensembles de places. Quelques détails supplémentaires accompagnent le modèle qui est plus général que celui des réseaux de Petri avec Reset.

Le théorème énoncé dans [KCK+97] et concluant à la décidabilité du caractère borné d'un réseau PCN, se traduit comme suit dans le contexte des réseaux de Petri avec Reset :

Énoncé 1 *Un réseau de Petri avec Reset N est non borné si et seulement s'il existe une séquence $\sigma \in T^*$ telle que $m_0 \xrightarrow{*} m_1 \xrightarrow{\sigma} m_2$ et :*

1. *$m_1 < m_2$*
2. *$\exists p \in P,\ m_1(p) < m_2(p)$ et p n'est pas mise à zéro dans σ.*

Il est clair que la condition est suffisante pour qu'un réseau de Petri avec Reset soit non borné. Si l'on peut exhiber une séquence satisfaisant les deux conditions, alors on peut indéfiniment itérer σ à partir de m_1 et incrémenter la place p. Cependant, le sens nécessaire de cet énoncé est faux.

Nous présentons un réseau de Petri avec Reset qui n'est pas borné, mais dont aucune séquence ne correspond à la description donnée par l'énoncé. Nous reprenons l'exemple du réseau qui nous a servi à illustrer l'algorithme de chaînage arrière pour la couverture en section 4.4 et nous étudions le comportement de ce réseau.

Considérons le réseau $N_r =< P_r, T_r, F_r, m_{0r} >$ de la figure 5.1. Les places p_1 et p_3 sont des places de contrôle complémentaires : $\forall m \in Acc(N_r),\ m(p_1) + m(p_3) = 1$. Par contre, les places p_2 et p_4 sont non-bornées. La transition t_1 est franchissable lorsque p_1 est marquée et elle déverse un à un les jetons de p_2 dans p_4. La transition t_2 donne le contrôle à la place p_3 et vide la place p_2. La place p_3 autorise le franchissement de t_3 qui est la duale de t_1 : t_3 reverse dans p_2, un à un, les jetons de p_4. Enfin, la transition t_4 donne le contrôle à p_1, met la place p_4 à zéro et donne un jeton "bonus" à p_2. La figure 5.2 donne le graphe des premiers marquages accessibles à partir de $(1, 0, 0, 0)$.

Proposition 60 *Le réseau avec Reset N_r de la figure 5.1 n'est pas borné.*

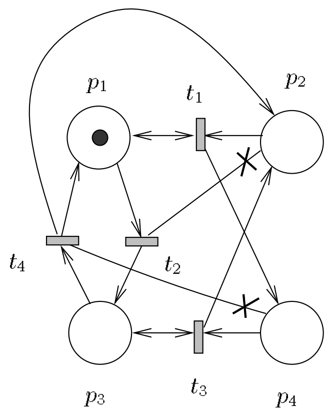


FIG. 5.1 – *Un réseau avec arcs de Reset non borné, mais pas par une séquence répétée itérativement.*

Preuve : nous montrons par induction que pour tout $n \in \mathbb{N}$, $(1, n, 0, 0)$ est accessible. $(1, 0, 0, 0)$ est le marquage initial. Supposons que pour tout $0 \leq i \leq n-1$, $(1, i, 0, 0)$ soit accessible. Montrons que $(1, n, 0, 0)$ est accessible. Par hypothèse d'induction, $(1, n-1, 0, 0)$ est accessible. À partir de $(1, n-1, 0, 0)$, la séquence $t_1^{n-1}t_2t_3^{n-1}t_4$ est franchissable, nous donnant :

$$(1, n-1, 0, 0) \overset{t_1^{n-1}}{\to} (1, 0, 0, n-1) \overset{t_2}{\to} (0, 0, 1, n-1) \overset{t_3^{n-1}}{\to} (0, n-1, 1, 0) \overset{t_4}{\to} (1, n, 0, 0)$$

Nous en concluons que $\forall n \in \mathbb{N}$, $(1, n, 0, 0) \in Acc(N_r)$, et donc que le réseau n'est pas borné.

□

La séquence que nous venons de mettre en évidence par la proposition 60 est la suivante :

$$t_2t_4 \;\; t_1t_2t_3t_4 \;\; t_1^2t_2t_3^2t_4 \cdots t_1^it_2t_3^it_4 \;\; t_1^{i+1}t_2t_3^{i+1}t_4 \cdots$$

Cette séquence apparaît sur la colonne principale de la figure 5.2. Elle a pour *motif* de base $t_1t_2^it_3t_4^i$ où i est égal à 0 la première fois, puis est incrémenté de 1 à chaque répétition du motif. Au cours de cette séquence, si l'on considère n'importe quels m_1, m_2 et σ tels que $(1, 0, 0, 0) \overset{*}{\to} m_1 \overset{\sigma}{\to} m_2$ avec $m_1 < m_2$, nous constatons que si $m_1(p) < m_2(p)$ alors σ met p à zéro (t_2 met p_2 à zéro et t_4 met p_4 à zéro).

Pour toute autre séquence que celle-là, un "déversement" de p_2 vers p_4 (ou de p_4 vers p_2) est interrompu avant la fin. La place p_2 (ou p_4) est mise à zéro, alors qu'elle n'est pas vide; le réseau perd des jetons et se retrouve dans un marquage déjà visité. Nous retrouvons ce comportement à la figure 5.2, par les effets de retour des arcs qui partent vers le haut.

Nous montrons à présent que bien que ce réseau soit non borné, il n'existe pas de séquence correspondant à l'énoncé 1. Nous précédons cette proposition d'un lemme qui décrit l'allure des marquages accessibles.

Lemme 61 $Acc(N_r) \subseteq (\{0\} \times \mathbb{N} \times \{1\} \times \mathbb{N}) \cup (\{1\} \times \mathbb{N} \times \{0\} \times \mathbb{N})$

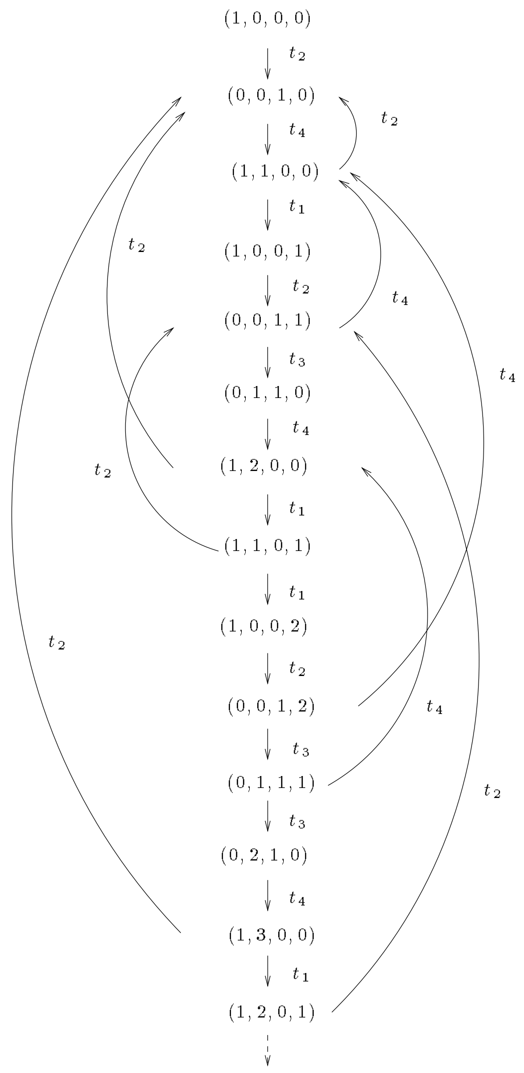


FIG. 5.2 – *Début du graphe des marquages accessibles.*

Preuve : par induction sur la longueur des séquences de transitions. Le marquage initial est $(1,0,0,0)$. Supposons que pour toute $\sigma \in T_r^*$, $|\sigma| = n-1$, il existe un $m \in (\{0\} \times \mathbb{N} \times \{1\} \times \mathbb{N}) \cup (\{1\} \times \mathbb{N} \times \{0\} \times \mathbb{N})$ tel que $(1,0,0,0) \xrightarrow{\sigma} m$. Considérons une séquence $\sigma' \in T_r^*$, $|\sigma| = n-1$ et supposons qu'il existe une transition $t \in T_r$ et deux marquages m et m' tels que $(1,0,0,0) \xrightarrow{\sigma'} m \xrightarrow{t} m'$. Par hypothèse d'induction, nous avons deux possibilités :

- Si $m(p_1) = 1$ et $m(p_3) = 0$, alors : soit $t = t_1$ et donc $m'(p_1) = 1$ et $m'(p_3) = 0$; soit $t = t_2$ et donc $m'(p_1) = 0$ et $m'(p_3) = 1$.
- Si $m(p_1) = 0$ et $m(p_3) = 1$, alors : soit $t = t_3$ et donc $m'(p_1) = 0$ et $m'(p_3) = 1$; soit $t = t_4$ et donc $m'(p_1) = 1$ et $m'(p_3) = 0$.

Nous en concluons que $Acc(N_r) \subseteq (\{0\} \times \mathbb{N} \times \{1\} \times \mathbb{N}) \cup (\{1\} \times \mathbb{N} \times \{0\} \times \mathbb{N})$. □

Proposition 62 *Pour toute exécution* $(1,0,0,0) \xrightarrow{*} m_1 \xrightarrow{\sigma} m_2$ *de* N_r *telle que* $m_1 < m_2$, *si* $m_1(p) < m_2(p)$ *alors* p *est mise à zéro par* σ.

Preuve : considérons une exécution $m_{0r} \xrightarrow{*} m_1 \xrightarrow{\sigma} m_2$, avec $\sigma \in {T_r}^*$ et $m_1 < m_2$. Regardons l'effet de σ pour chaque place :

1. Supposons que $m_1(p_1) < m_2(p_1)$. Par le lemme 61, nous avons forcément $m_1(p_1) = 0$ et $m_1(p_1) = 1$. Dans ce cas, m_1 est de type $(0,i,1,j)$ et m_2 est de type $(1,i',0,j')$ et les deux marquages sont incomparables, ce qui amène à une contradiction.
2. Supposons que $m_1(p_3) < m_2(p_3)$. Pour les mêmes raisons qu'au cas précédent, on arrive à une contradiction.
3. Supposons que $m_1(p_2) < m_2(p_2)$. Nous avons deux cas de figure :
   (a) $m_1 = (1,i,0,j)$ et $m_2 = (1,i',0,j')$ avec $i,j,i',j' \in \mathbb{N}$ et $i < i'$. Puisque nous avons $m_1 < m_2$, forcément $1+i+j < 1+i'+j'$. La seule transition qui produit un jeton supplémentaire dans N_r est t_4. Les transitions t_1, t_2 et t_3 ne font que "bouger" les jetons. Il faut donc que t_4 soit franchie. Mais pour franchir t_4, il faut que p_3 soit marquée, et donc que t_2 soit franchie; or t_2 met p_2 à zéro.
   (b) $m_1 = (0,i,1,j)$ et $m_2 = (0,i',1,j')$ avec $i,j,i',j' \in \mathbb{N}$ et $i < i'$. Pour la même raison que précédemment, il faut que t_4 apparaisse dans σ, or t_4 met N_r dans un état de type $(1,i'',0,j'')$, qui est incomparable avec m_2. Pour que N_r retombe dans un état comparable à m_2, la seule solution est de franchir t_2, or t_2 met p_2 à zéro.
4. Supposons que $m_1(p_4) < m_2(p_4)$. Nous retombons sur un cas semblable au cas 3.

Ainsi, pour toute place $p \in P_r$, si $m_1(p) < m_2(p)$ alors p est mise à zéro par σ.

Nous en concluons que le lemme est vrai pour toute exécution de N_r. □

En conclusion, N_r est non borné et il n'est pas possible de détecter une séquence répondant aux deux conditions de l'énoncé 1. Contrairement aux réseaux de Petri, le caractère non borné d'un réseau avec Reset n'est pas forcément associé à l'existence d'une séquence itérée à l'infini. Dans notre exemple, il faut “pomper” à chaque cycle de type $t_1t_2^it_3t_4^i$ les transitions t_2 et t_4 pour produire de plus en plus de jetons.

L'énoncé 1 est donc faux, ce qui n'implique pas que le problème de caractère borné soit indécidable. Cependant il l'est, et nous le montrons dans les sections qui suivent.

5.2 Réduction du dixième Problème de Hilbert

Nous allons donner les idées principales de la première preuve d'indécidabilité pour le caractère borné des réseaux de Petri avec arcs de Reset. Il s'agit d'une réduction du dixième Problème de Hilbert ou Problème des équations diophantiennes. Mentionnons que Hack [Hac76], sur une idée de Rabin non publiée, utilise le dixième Problème de Hilbert pour prouver l'indécidabilité du problème de l'inclusion (et de l'équivalence) des ensembles d'accessibilité de deux réseaux.

Dans cette section, nous allons montrer comment construire un réseau avec arcs de Reset qui est borné si et seulement s'il existe une solution au dixième Problème de Hilbert.

• Dixième Problème de Hilbert

Entrée: $Q, R \in \mathbb{N}[v_1, .., v_k]$, $k \in \mathbb{N}$

Sortie: $\begin{cases} \textit{oui} & \text{s'il existe } v \in \mathbb{N}^k \text{ tel que } Q(v) = R(v) \\ \textit{non} & \text{sinon} \end{cases}$

Théorème 63 *[Dav77] Le dixième Problème de Hilbert est indécidable.*

Étant donnés deux polynômes $Q, R \in \mathbb{N}[v_1, .., v_k]$, nous construisons un réseau $N_{Dio(Q,R)}$ avec arcs de Reset qui est borné si et seulement s'il existe un $v \in \mathbb{N}^k$ tel que $Q(v) = R(v)$.

Le réseau $N_{Dio(Q,R)}$ “implémentera” l'algorithme qui considère un par un tous les vecteurs de $\mathbb{N}^k$. Si le vecteur courant v, stocké dans une variable x, est tel que $Q(v) = R(v)$, alors l'algorithme s'arrête; sinon, il passe au vecteur suivant. S'il n'existe pas de v tel que $Q(v) = R(v)$, alors l'algorithme ne s'arrêtera pas et engendrera (et testera) une infinité de marquages différents et, dans ce cas, la variable x ne sera pas bornée. Sinon, l'algorithme s'arrêtera au premier v (de la liste) tel que $Q(v) = R(v)$, et x sera donc bornée.

Dans $N_{Dio(Q,R)}$, nous représenterons x sur k places dont la $i^{\text{ème}}$ correspond à la valeur $v(i)$ de la variable v_i.

Le réseau $N_{Dio(Q,R)}$ “implémentera” cet algorithme dans un sens *faible*. Cela signifie que le comportement du réseau avec Reset contient celui de l'algorithme, appelé comportement nominal, ainsi que d'autres comportements qui seront ignorés.

La construction comprend quatre points: l'évaluation des polynômes, la comparaison des polynômes, la définition d'une énumération des vecteurs (comportant en particulier l'implémentation d'une fonction *successeur* par le réseau) et l'assemblage des différentes constructions pour donner le réseau final. Nous allons surtout détailler les deux premiers points.

5.2.1 Évaluation de polynômes dans $\mathbb{N}[v_1, .., v_k]$

L'évaluation d'un polynôme faite ici n'est pas de la même forme que celle de HACK [Hac75] (ou que celle de la section 3.1) où "évaluer $Q(v)$" signifie "ajouter $Q(v)$ jetons, ou moins, dans une place donnée". Ici, "évaluer $Q(v)$" signifiera "franchir une séquence de longueur $Q(v)$, ou moins".

De plus, la construction en elle-même ne sera pas basée sur le même principe que la construction de la section 3.1. Pour le dépliage, nous avions considéré le polynôme sous sa forme développée. Ici, nous allons construire récursivement les polynômes : polynôme constant, polynôme identité, addition de deux polynômes puis multiplication de deux polynômes. Nous ne pouvions pas utiliser une telle construction pour le dépliage, car elle ne permet pas de gérer correctement les enchaînements des calculs de polynômes.

Structure générale des réseaux

Les réseaux que nous construisons ont des places d'entrée, des places de sortie et des places intermédiaires. Les places d'entrée sont de deux types : une place *on* et un ensemble P_{in} de k places destinées à recevoir la valeur du vecteur donné en paramètre au polynôme. Les places de sortie sont aussi de deux types : une place *off* et un ensemble P_{out} de k places qui contiendront, à la fin du calcul, un vecteur. Les places d'entrée et de sortie sont l'interface du réseau. L'interface nous suffira pour détailler les comportements des réseaux et donner les preuves de correction. Nous montrons l'interface d'un réseau (en omettant les places intermédiaires) à la figure 5.3.

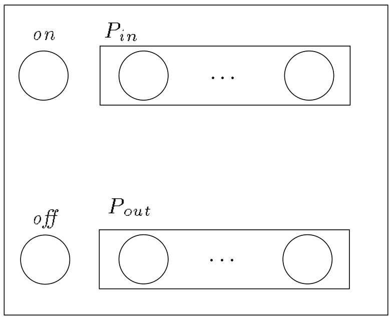


FIG. 5.3 – *Structure générale du réseau qui évalue les polynômes*

Au départ d'une évaluation, le réseau a un jeton dans la place *on* et un vecteur en entrée dans P_{in}. Toutes les autres places sont vides. À la fin du calcul, le réseau a un jeton dans la place *off* et un vecteur en sortie dans P_{out}. Toutes les autres places sont vides.

Si le réseau a adopté le comportement nominal, c'est-à-dire le comportement qui fait le bon calcul, alors le vecteur de sortie est égal au vecteur d'entrée. Sinon, le vecteur de sortie est strictement plus petit que le vecteur d'entrée. Dans ce cas, le réseau aura perdu des jetons. Le vecteur de sortie sert de "certificat de bon calcul".

Comportement d'un réseau qui évalue un polynôme

Décrivons le comportement du réseau $N_Q =< P_Q, T_Q, F_Q, m_{0Q} >$ qui évalue le polynôme Q. Le réseau N_Q est étiqueté en distinguant les transitions visibles et invisibles. Nous notons T_{vis} l'ensemble des transitions visibles et $T_{inv} = T_Q \setminus T_{vis}$ l'ensemble des transitions invisibles.

Notation :

- *Étant donnée une séquence $\sigma \in (T_{vis} \cup T_{inv})^*$, nous notons $|\sigma|_{T_{vis}}$ le nombre d'occurrences de transitions visibles dans σ.*

- *$\bar{0}$ représente le vecteur zéro.*

Évaluer le polynôme Q sur l'entrée v signifie "franchir une séquence σ dont la longueur visible est $Q(v)$", c'est-à-dire dont $|\sigma|_{T_{vis}} = Q(v)$. N_Q vérifie les cinq propriétés qui suivent.

Voici d'abord la description du comportement nominal dont l'existence constitue la première propriété :

$$\forall v \in \mathbb{N}^n,\ \exists \sigma \in {T_Q}^*\ tq\ (1, v, 0, \bar{0}, \bar{0}) \xrightarrow{\sigma} (0, \bar{0}, 1, v, \bar{0}) \text{ avec } |\sigma|_{T_{vis}} = Q(v), \tag{5.1}$$

où les places apparaissent de façon abrégée dans l'ordre suivant : *on*, P_{in}, *off*, P_{out} et les places intermédiaires. Ce comportement laisse le réseau avec un jeton dans *off*, le vecteur v dans P_{out} et toutes les autres places vides. En quelque sorte, le vecteur d'entrée est "transféré" en sortie, au cours d'un processus qui a "compté jusqu'à $Q(v)$".

Il n'est pas possible de contraindre un réseau avec Reset à effectuer exactement ce "bon" comportement. Le réseau N_Q peut avoir d'autres comportements non-nominaux. Cependant, ceux-ci satisfont les propriétés suivantes :

- dès que *off* possède au moins un jeton, N_Q accède à une *configuration de fin*, de la forme décrite ici :

$$(1, v, 0, \bar{0}, \bar{0}) \xrightarrow{*} (x_1, v', x_2, v'', m) \text{ avec } x_2 > 0 \Rightarrow \begin{cases} x_1 = 0 \\ v' = \bar{0} \\ x_2 = 1 \\ m = \bar{0} \end{cases} \tag{5.2}$$

- le comportement nominal est celui de la séquence qui possède la plus longue partie visible possible :

$$\forall \sigma \in {T_Q}^* \text{ telle que } (1, v, 0, \bar{0}, \bar{0}) \xrightarrow{\sigma}, |\sigma|_{T_{vis}} \leq Q(v) \tag{5.3}$$

- le réseau ne peut pas créer de jetons dans P_{out}, qui n'étaient pas dans P_{in} au départ. De plus, si le comportement n'est pas nominal, alors nécessairement des jetons ont été perdus.

$$(1, v, 0, \bar{0}, \bar{0}) \xrightarrow{\sigma} (0, \bar{0}, 1, v', \bar{0}) \Rightarrow \begin{cases} v' \leq v \\ |\sigma|_{T_{vis}} < Q(v) \Rightarrow v' < v \end{cases} \tag{5.4}$$

- enfin, lorsqu'il est correctement initialisé, le réseau est borné :

$$\forall v, (N_Q, (1, v, 0, \bar{0}, \bar{0})) \text{ est borné.} \tag{5.5}$$

Définition 49 *On dit d'un réseau qui possède la structure décrite ci-dessus et qui satisfait les propriétés (5.1) à (5.5), qu'il évalue (faiblement) le polynôme Q.*

Le réseau N_Q *évalue faiblement* le polynôme Q. Ceci veut dire que sur l'entrée v :

- Il existe une séquence dont $Q(v)$ transitions sont visibles, qui aboutit dans une configuration de fin avec v dans P_{out}.
- Toute séquence qui n'a pas $Q(v)$ transitions visibles en a forcément moins et si elle aboutit dans une configuration de fin, c'est avec $v' < v$ dans P_{out}. Dans ce cas, le réseau "a perdu" des jetons.

Nous construisons récursivement ces réseaux. Nous montrons d'abord comment évaluer le polynôme identité qui retourne la valeur d'une variable, puis le polynôme constant qui retourne la valeur 1. Ensuite, nous montrons comment construire l'addition, puis la multiplication de deux polynômes.

Construction pour le polynôme identité et le polynôme constant.

Proposition 64 *Pour $Q(v_1, \ldots, v_k) \stackrel{def}{=} v_i$, il existe un réseau de Petri avec Reset N_Q qui évalue (faiblement) Q.*

Preuve : le réseau N_Q apparaît à la figure 5.4 et nous lui associons $T_{vis} = \{t_i\}$. Vérifions que N_Q satisfait les cinq propriétés :

(5.1) Cherchons une séquence nominale. Prenons, par exemple, $\sigma = t_1^{v(1)} t_2^{v(2)} \cdots t_k^{v(k)} t_{off}$, nous avons :

$$(1, v, 0, \bar{0}) \xrightarrow{\sigma} (0, \bar{0}, 1, v) \text{ avec } |\sigma|_{T_{vis}} = v(i)$$

(5.2) Montrons que dès que *off* contient au moins un jeton, alors le réseau est dans une configuration de fin. La seule façon de marquer *off* est de franchir t_{off} qui vide *on* et met à zéro toutes les places de P_{in}. Après le franchissement de t_{off}, le marquage atteint est de la forme $(0, \bar{0}, 1, v')$, qui est un blocage pour N_Q.

(5.3) Montrons que toute séquence σ telle que $(1, v, 0, \bar{0}) \xrightarrow{\sigma}$ est telle que $|\sigma|_{T_{vis}} \leq Q(v) = v(i)$. La seule transition visible de N_Q est t_i ; or t_i est franchissable au plus $v(i)$ fois.

(5.4) Soit $\sigma \in T^*$ telle que $(1, v, 0, \bar{0}) \xrightarrow{\sigma} (0, \bar{0}, 1, v')$.

- Montrons que $v' \leq v$. Toute place $P_{out}[j]$ $(1 \leq j \leq k)$ reçoit un jeton par t_j ; or t_j est franchissable au plus $v(j)$ fois.
- Montrons que si $|\sigma|_{T_{vis}} < v(i)$, alors $v < v'$. Seule t_i est visible. Elle est donc franchie moins de $v(i)$ fois. Puisque $P_{out}[i]$ est incrémentée seulement par t_i, nous en déduisons que $v'(i) < v(i)$ et comme $v' \leq v$, nous en concluons que $v' < v$.

(5.5) Le réseau est borné puisqu'il ne possède pas de cycle.

Les cinq propriétés sont donc satisfaites et le réseau N_Q calcule donc (faiblement) le polynôme $Q(v_1, .., v_k) = v_i$.

□

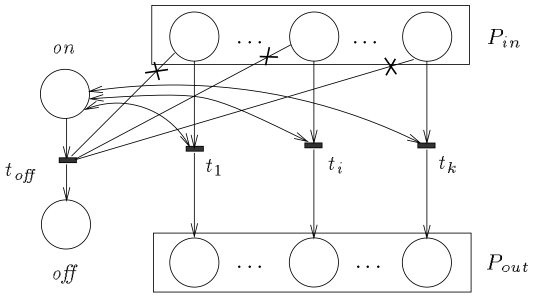


FIG. 5.4 – *Un réseau pour le polynôme identité et pour le polynôme constant*

Proposition 65 *Pour* $R(v_1, \ldots, v_k) \stackrel{def}{=} 1$*, il existe un réseau de Petri avec Reset* N_R *qui évalue (faiblement)* R*.*

Preuve : la preuve est similaire à celle du polynôme identité. Le réseau N_Q apparaît à la figure 5.4, avec cette fois $T_{vis} = \{t_{off}\}$ pour seule transition visible. Vérifions que N_Q répond aux cinq conditions.

(5.1) Prenons $\sigma = t_1^{v(1)} t_2^{v(2)} \cdots t_k^{v(k)} t_{off}$, nous avons :

$$(1, v, 0, \bar{0}) \xrightarrow{\sigma} (0, \bar{0}, 1, v) \text{ avec } |\sigma|_{T_{vis}} = 1$$

(5.2) Comme pour l'identité, dès que *off* est marquée, N_R atteint un marquage de la forme $(0, \bar{0}, 1, v')$.

(5.3) La seule transition visible est t_{off} ; or t_{off} n'est franchissable qu'une seule fois.

(5.4) Toute exécution $(1, v, 0, \bar{0}) \xrightarrow{\sigma} (0, \bar{0}, 1, v')$ est telle que $v' \leq v$. Pour terminer, si *off* est marquée, alors $|\sigma|_{T_{vis}} = 1$, car forcément t_{off} n'a été franchie qu'une fois.

(5.5) Le réseau est borné puisqu'il ne possède pas de cycle.

Les cinq propriétés sont donc satisfaites. N_R calcule donc (faiblement) le polynôme $R(v_1, \ldots, v_k) = 1$.

□

Construction pour la somme de deux polynômes.

Nous décrivons le réseau N_{Q+R} qui évalue la somme des polynômes Q et $R \in \mathbb{N}[v_1, .., v_k]$. N_{Q+R} est construit par une mise en parallèle des réseaux N_Q et N_R, qui évaluent respectivement Q et R dans le sens de la définition 49. En plus des calculs propres aux évaluations

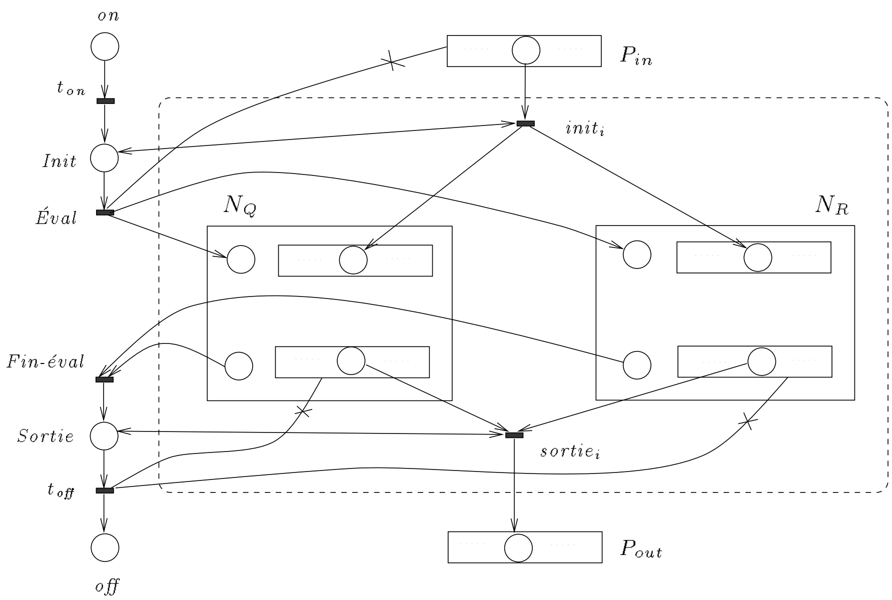


FIG. 5.5 – *Un réseau de Petri avec Reset pour $Q + R$*

de Q et R, nous ajoutons deux étapes de calcul pour N_{Q+R}. Une première étape est l'initialisation de N_Q et N_R; l'autre étape est la récupération des vecteurs en sortie de N_Q et N_R. Il y a donc en tout trois étapes pour N_{Q+R}. Ces étapes sont indépendantes et effectuées séquentiellement.

Notation : nous notons $N'[P]$ (resp. $N'[p]$) le sous-ensemble de places P (resp. la place p) d'un réseau N' qui est utilisé dans la construction couramment décrite. Par exemple, ici $N_Q[P_{in}]$ fait référence à l'ensemble P_{in} de N_Q qui est utilisé dans la construction du réseau N_{Q+R}.

Nous définissons le réseau $N_{Q+R} =< P, T, F, m_0 >$ à partir des réseaux N_Q, pour lequel $T_Q = T_{Q_{vis}} \cup T_{Q_{inv}}$, et N_R, pour lequel $T_R = T_{Rvis} \cup T_{Rinv}$:

- $P = \{on\} \cup P_{in} \cup \{off\} \cup P_{out} \cup P_Q \cup P_R \cup \{Init,\ Sortie\}$
- $T = \cup_{i=1}^{k}\{init_i,\ sortie_i\} \cup \{t_{on}, \textit{Éval}, \textit{Fin-éval}, t_{off}\} \cup T_Q \cup T_R$
- F est illustrée à la figure 5.5
- Soit $v \in \mathbb{N}^k$. Le marquage initial pour calculer $Q(v) + R(v)$ est $m_0 = (1, v, 0, \bar{0}, \bar{0})$ où la première place est *on*, les k suivantes sont celles de P_{in}, suivies de p_{off}, des k places de P_{out}, puis des places restantes.

De plus, nous avons pour N_{Q+R} : $T_{vis} = T_{Q_{vis}} \cup T_{Rvis}$ et $T_{inv} = T \setminus T_{vis}$.

Décrivons les trois étapes de calcul de N_{Q+R} pour l'évaluation $Q(v) + R(v)$:

1. *Initialisation :* toutes les transitions de cette étape sont invisibles. Au départ de l'évaluation de $Q(v)$, le réseau N_{Q+R} possède un jeton dans *on* et le vecteur v dans

P_{in}. Toutes les autres places sont vides, y compris les places des réseaux N_Q et N_R. La première étape consiste à initialiser N_Q et N_R pour les calculs de $Q(v)$ et $R(v)$, en franchissant les transitions de type $init_i$. La transition $init_i$ décrémente $P_{in}[i]$ et incrémente $N_Q[P_{in}[i]]$ et $N_R[P_{in}[i]]$. Si ce processus est effectué jusqu'à ce que les places de P_{in} soient vides, alors on retrouve v dans $N_Q[P_{in}]$ et dans $N_R[P_{in}]$. À tout moment, N_{Q+R} peut franchir la transition *Éval* pour amorcer l'étape suivante. Les places de P_{in} sont alors mises à zéro.

2. *Évaluation de $Q+R$:* les transitions visibles de N_Q et N_R restent visibles dans N_{Q+R}. Les réseaux N_Q et N_R évoluent en parallèle dans N_{Q+R} et les comportements de N_Q et de N_R sont entrelacés. Si les calculs sont effectués correctement sur l'entrée v, la longueur totale de la séquence visible sera égale à $Q(v) + R(v)$ (propriété 5.1).

3. *Récupération :* toutes les transitions de cette étape sont invisibles. Lorsque N_Q (resp. N_R) atteint une configuration de fin sans avoir effectué l'exécution nominale, le vecteur v' atteint dans $N_Q[P_{out}]$ (resp. dans $N_R[P_{out}]$) est tel que $v' < v$. Dans ce cas, il faut que l'erreur apparaisse en P_{out} (propriété 5.3). Ce sont les k transitions $sortie_i$ qui effectuent cette vérification d'erreur. L'idée consiste à chercher le minimum des jetons apparaissant dans $N_Q[P_{out}]$ et dans $N_R[P_{out}]$. La transition $sortie_i$ décrémente simultanément $N_Q[P_{out}[i]]$ et $N_R[P_{out}[i]]$ et incrémente $P_{out}[i]$. À tout moment, t_{off} peut être franchie, marquant *off* et mettant $N_Q[P_{out}]$ et $N_R[P_{out}]$ à zéro (pour la propriété 5.2).

De cette construction, nous en concluons la proposition suivante :

Proposition 66 *S'il existe deux réseaux de Petri avec Reset N_Q et N_R qui évaluent (faiblement) Q et R, alors il existe un réseau de Petri avec Reset N_{Q+R} qui évalue (faiblement) $Q + R$.*

Les arcs de Reset servent en fait à effacer les traces d'une mauvaise exécution, et c'est là toute leur force. Nous avons choisi de "nettoyer le réseau" par les Reset au terme de chaque étape. Nous aurions pu choisir de le faire en une fois, à la toute fin.

Construction pour la multiplication de deux polynômes.

Pour évaluer $Q(v) * R(v)$, on construit un réseau N_{Q*R} qui déclenchera $Q(v)$ fois le calcul du réseau N_R sur l'entrée $(1, v, 0, \bar{0}, \bar{0})$. La figure 5.6 illustre la construction.

Nous définissons le réseau $N_{Q*R} =< P, T, F, m_0 >$ à partir des réseaux N_Q, pour lequel $T_Q = T_{Q_{vis}} \cup T_{Q_{inv}}$, et N_R, pour lequel $T_R = T_{Rvis} \cup T_{Rinv}$:

- $P = \{on\} \cup P_{in} \cup \{off\} \cup P_{out} \cup P_Q \cup P_R \cup \{Init,\ Sortie,\ p_R\}$
- $T = \cup_{i=1}^{k}\{init_i,\ sortie_i,\ retour_i\} \cup \{t_{on},\ \textit{Éval},\ \textit{Éval-R},\ \textit{Fin-éval}, t_{off}\} \cup T_Q \cup T_R$
- F est illustrée par la figure 5.6
- Soit $v \in \mathbb{N}^k$. Le marquage initial pour calculer $Q(v) * R(v)$ est $m_0 = (1, v, 0, \bar{0}, \bar{0})$ où la première place est *on*, les k suivantes celles de P_{in}, suivies de p_{off}, des k places de P_{out} puis des places restantes.

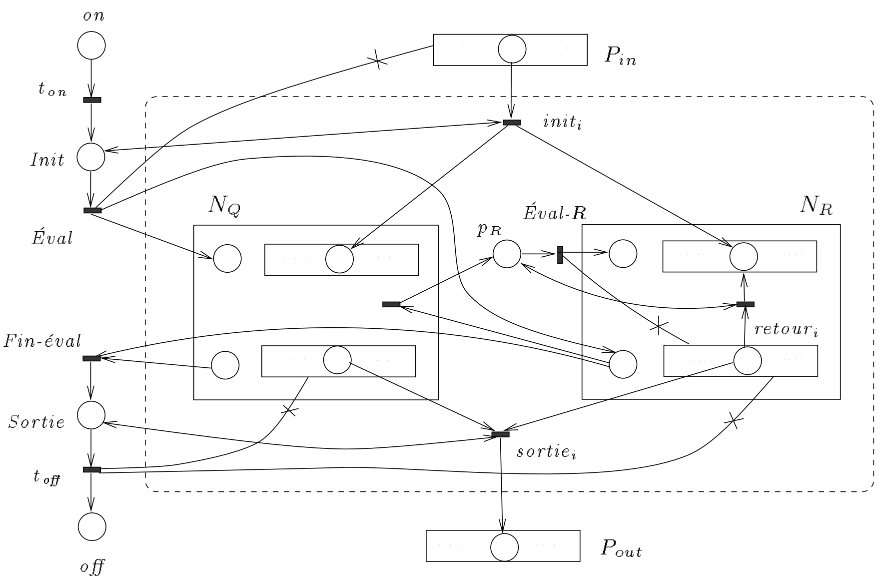


FIG. 5.6 – *Un réseau de Petri avec Reset pour $Q * R$*

De plus, nous avons pour N_{Q*R} : $T_{vis} = T_{Rvis}$ et $T_{inv} = T \setminus T_{vis}$.

Comme pour l'addition, il y a trois étapes de calcul. L'initialisation et la récupération du vecteur en sortie sont les mêmes que pour l'addition. La mise en place du calcul propre à la multiplication est un peu plus technique. Détaillons cette étape :

Évaluation de $Q * R$: le but est d'appeler $Q(v)$ fois l'évaluation de $R(v)$ par N_R. Dans N_{Q*R}, seules les transisions de N_R restent visibles. N_Q, au lieu de franchir une transition visible, "appelle" N_R sur l'entrée $(1, v, 0, \bar{0}, \bar{0})$. Pour cela, nous ajoutons la place $N_R[\mathit{off}]$ en condition d'entrée à toutes les transitions qui étaient visibles dans N_Q. Ceci veut dire que N_Q ne peut "appeler" une nouvelle évaluation de R tant que N_R n'a pas terminé son évaluation courante.

À chaque évaluation de $R(v)$, il faut que les places de $N_R[P_{in}]$ soient ré-initialisées correctement (avec v). Ce sont les k transitions $retour_i$ qui effectuent la ré-initialisation, en ré-injectant dans $N_R[P_{in}]$ le vecteur de $N_R[P_{out}]$ qui provient de la sortie de l'évaluation précédente. Lorsque la transition *Éval R* est franchie, l'évaluation propre de R débute. *Éval-R* met les places $N_R[P_{out}]$ à zéro.

Dans la phase de calcul, N_R peut faire deux sortes d'erreurs. La première erreur, celle que nous avons déjà rencontrée, apparaît lorsqu'une évaluation ne suit pas le parcours nominal : le vecteur de sortie est plus petit que le vecteur d'entrée (pour N_R aussi bien que pour N_Q).

La seconde erreur apparaît lors de la ré-initialisation de $N_R[P_{in}]$ par les transitions de type $retour_i$. Si les jetons ne sont pas correctement injectés, on retrouve en entrée moins de jetons qu'au tour précédent. Les jetons qui restent dans $N_R[P]$ sont éliminés lors du franchissement de *Éval-R*, ce qui assure qu'en sortie de N_{Q*R}, on retrouvera dans P_{out} un vecteur v' tel que $v' < v$ (propriété 5.3).

Lorsque $N_Q[p_{off}]$ est marquée, dans le meilleur des cas, N_Q a appelé $Q(v)$ fois l'exécution de N_R sur v et la séquence franchie est de longueur visible $Q(v) * R(v)$ (propriété 5.1). La transition *Fin-éval* n'est franchissable que lorsque N_Q et N_R sont dans une configuration de fin et fait passer N_{Q*R} à l'étape de récupération des valeurs en sortie de N_Q et N_R.

De cette construction, nous en concluons la proposition suivante :

Proposition 67 *S'il existe deux réseaux de Petri avec Reset N_Q et N_R qui évaluent (faiblement) Q et R, alors il existe un réseau de Petri avec Reset N_{Q*R} qui évalue (faiblement) $Q * R$.*

Ici, au contraire de la construction pour l'addition, les arcs de Reset sont nécessaires au calcul. Il servent à nettoyer le réseau après une ré-initialisation de N_R, pour assurer qu'il n'y a pas eu d'erreur. Si l'on omettait de faire ce ménage à chaque appel, N_{Q*R} pourrait retourner en sortie son vecteur de départ, sans pour autant avoir franchi une séquence de longueur visible égale à $Q(v) * R(v)$.

Les quatre dernières propositions nous amènent au corollaire suivant :

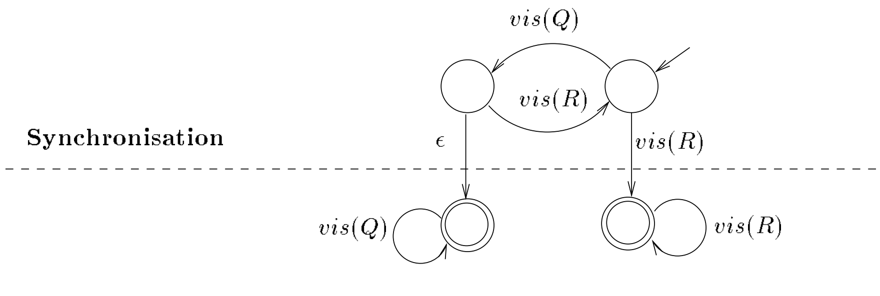
Corollaire 68 *Pour tout polynôme $Q \in \mathbb{N}[v_1, .., v_n]$, il existe un réseau qui évalue (faiblement) Q.*

5.2.2 Comparaison de deux polynômes dans $\mathbb{N}[v_1, .., v_k]$

Nous montrons comment construire des réseaux de Petri avec Reset qui comparent des polynômes. Ces réseaux ont la même interface que les réseaux qui évaluent les polynômes : *on*, P_{in}, *off* et P_{out} et des places intermédiaires.

Étant donnés deux polynômes Q et $R \in \mathbb{N}[v_1, .., v_k]$, et $v \in \mathbb{N}^k$, nous décrivons le comportement attendu du réseau $N_{Q \neq R}$ qui compare $Q(v)$ et $R(v)$. Au début de la comparaison, le marquage de $N_{Q \neq R}$ est $(1, v, 0, \bar{0}, \bar{0})$. À la fin, $N_{Q \neq R}$ ne pourra accéder au marquage $(0, \bar{0}, 1, v, \bar{0})$ que si et seulement si $Q(v) \neq R(v)$.

Nous donnons l'idée générale de la construction de $N_{Q \neq R}$. Considérons les deux réseaux N_Q et N_R qui évaluent respectivement Q et R selon la définition 49. Nous allons synchroniser N_Q et N_R sur leurs transitions visibles, afin d'obtenir le comportement défini par l'automate suivant :



où $vis(Q)$ correspond au franchissement d'une transition visible par N_Q et $vis(R)$ correspond au franchissement d'une transition visible par N_R. Au départ, N_Q et N_R sont en phase de synchronisation et, tour à tour, franchissent une transition visible. En particulier, dès que N_R franchit sa transition visible, les deux réseaux ont franchi autant de transitions visibles l'un que l'autre.

De façon non déterministe, l'un des deux réseaux peut provoquer un changement de phase, certifiant qu'il franchit une transition visible de plus que l'autre. Cela correspond à l'atteinte d'un état final de l'automate. Dès ce moment, les réseaux sont désynchronisés, et seul le réseau qui a franchi la transition visible supplémentaire peut continuer son évaluation, alors que l'autre réseau est gelé.

Pour comparer $Q(v)$ et $R(v)$, l'exécution nominale est celle qui synchronise les deux réseaux pendant $min(Q(v), R(v))$ transitions visibles (chaque réseau franchit $min(Q(v), R(v))$ transitions visibles). Voyons ce qu'il advient dans ce cas :

1. Si $Q(v) \neq R(v)$, alors l'un des deux réseaux a terminé son évaluation et retournera en sortie le vecteur v. L'autre réseau, par contre, peut franchir une nouvelle transition visible, certifiant de la différence, provoquer le changement de phase, puis terminer son évaluation pour retourner le vecteur v.

2. Si $Q(v) = R(v)$, alors N_Q et N_R ont terminé leur évaluation, et aucun des deux réseaux ne peut franchir de transition visible supplémentaire. Le processus s'arrête dans un état de blocage.

Dans une exécution non nominale, les deux réseaux sont synchronisés sur un nombre insuffisant de transitions visibles. Cela survient lorsque l'un des réseaux provoque la phase de désynchronisation trop tôt. Supposons que ce soit N_Q. Dans ce cas, forcément, N_R n'achèvera pas son calcul et retournera un vecteur v' tel que $v' < v$, témoignant de l'erreur. N_Q, quant à lui, certifie de la différence et peut continuer son évaluation. Cependant, l'erreur engendrée par N_R sera répercutée lors de la phase de récupération des vecteurs de sortie de $N_{Q \neq R}$.

Définissons les propriétés du réseau $N_{Q \neq R}$. La comparaison est effectuée dans un sens faible. Le comportement nominal, c'est-à-dire le bon calcul, revêt deux formes :

$$\forall v \in \mathbb{N}^k \ : \ Q(v) \neq R(v) \ \Rightarrow \ (1, v, 0, \bar{0}, \bar{0}) \xrightarrow{*} (0, \bar{0}, 1, v, \bar{0}) \tag{5.6}$$

De plus, nous avons :

$$\forall v \in \mathbb{N}^k \ : Q(v) = R(v) \ \Rightarrow \ [(1, v, 0, \bar{0}, \bar{0}) \xrightarrow{*} (0, \bar{0}, 1, v', \bar{0}) \ \Rightarrow \ (v' < v)] \tag{5.7}$$

Ainsi, si $Q(v) \neq R(v)$, alors le réseau pourra "transférer" dans P_{out} le vecteur de v. Par contre, si $Q(v) = R(v)$, forcément tout vecteur v' apparaissant dans P_{out} est tel que $v' < v$.

Indépendamment du fait que les évaluations des polynômes soient égales ou non, $N_{Q \neq R}$ peut adopter d'autres comportements, mais les conditions suivantes restent vraies :

- dès que *off* est marquée, le marquage courant est une configuration de fin et retourne un vecteur majoré par v :

$$(1, v, 0, \bar{0}, \bar{0}) \xrightarrow{*} (x_1, v', x_2, v'', m) \text{ avec } x_2 > 0 \Rightarrow \begin{cases} x_1 = 0 \\ v' = \bar{0} \\ x_2 = 1 \\ v'' \leq v \\ m = \bar{0} \end{cases} \tag{5.8}$$

- enfin, il est permis d'avoir des comportements non nominaux amenant à un blocage. Cependant,

$$\forall v, (N_Q, (1, v, 0, \bar{0}, \bar{0})) \text{ est borné.} \tag{5.9}$$

Remarque : *il n'est plus question ici de transitions visibles ou de longueur de séquences de transitions. Une comparaison est "validée" par la valeur du vecteur de fin dans l'ensemble P_{out} de $N_{Q \neq R}$.*

Définition 50 *On dit d'un réseau qui possède la structure décrite plus haut et qui satisfait les propriétés 5.6 à 5.9, qu'il compare (faiblement) les polynômes Q et R.*

Construction du réseau qui compare deux polynômes

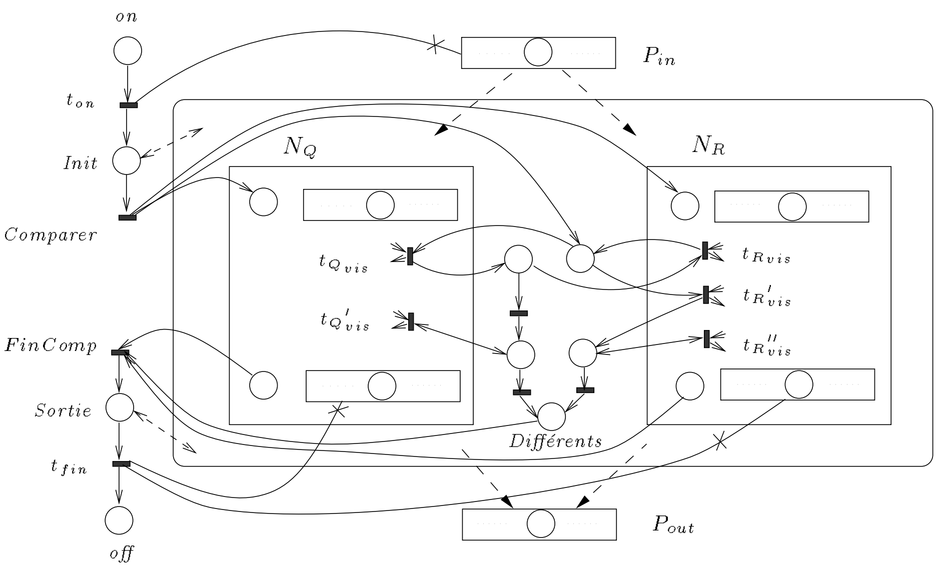


FIG. 5.7 – *Un réseau pour comparer Q et R*

Nous définissons le réseau $N_{Q \neq R} =< P, T, F, m_0 >$ à partir de $N_Q =< P_Q, T_Q, F_Q, m_{0Q} >$ pour lequel $T_Q = T_{Q_{vis}} \cup T_{Q_{inv}}$ et à partir de $N_R =< P_R, T_R, F_R, m_{0R} >$ pour lequel $T_R = T_{Rvis} \cup T_{Rinv}$. Soit $T_{Q'_{vis}}$ une copie de l'ensemble $T_{Q_{vis}}$ et soient $T_{R'_{vis}}$ et $T_{R''_{vis}}$ deux copies de l'ensemble T_R.

- $P = \{on\} \cup P_{in} \cup \{off\} \cup P_{out} \cup P_Q \cup P_R \cup \{Init, Sortie, Différents\} \cup P_S$ où P_S contient les quatre places utilisées pour la synchronisation.

- $T = \cup_{i=1}^{k}\{init_i,\ sortie_i\} \cup \{t_{on},\ Comparer,\ Fin\ comp, t_{fin}\} \cup T_Q \cup T_{Q'_{vis}} \cup T_R \cup T_{R'_{vis}} \cup T_{R''_{vis}} \cup T_S$ où T_S contient les trois transitions utilisées pour gérer la synchronisation.
- F est illustrée par la figure 5.7
- Soit $v \in \mathbb{N}^k$. Le marquage initial pour comparer $Q(v)$ et $R(v)$ est $m_0 = (1, v, 0, \bar{0}, \bar{0})$ où la première place est on, les k suivantes celles de P_{in}, suivies de p_{off}, des k places de P_{out}, puis des places restantes.

Nous représentons concrètement la construction du réseau $N_{Q \neq R}$ qui compare Q et R à la figure 5.7. Tout comme pour l'addition ou la multiplication, il y a trois étapes de calcul : initialisation de N_Q et N_R, comparaison de Q et R, récupération du vecteur de sortie. Afin d'alléger le dessin, nous n'avons représenté que schématiquement l'initialisation et la récupération, dont le "câblage" est le même que pour l'évaluation des polynômes.

Le contrôle de synchronisation est situé entre N_Q et N_R. Pour implémenter la comparaison, le jeu de transitions visibles de N_Q est dupliqué en $T_{Q_{vis}}$ et $T_{Q'_{vis}}$ et le jeu de transitions visibles de N_R est triplé en $T_{R_{vis}}$, $T_{R'_{vis}}$ et $T_{R''_{vis}}$. La place *Différents* garantit que l'un des réseaux a franchi un nombre plus grand de transitions visibles que l'autre. Pour accéder à l'étape de récupération, cette garantie est nécessaire.

De cette construction, nous concluons la proposition suivante :

Proposition 69 *Étant donnés deux polynômes Q et R, il existe un réseau de Petri avec Reset qui les compare (faiblement).*

5.2.3 Enumération de vecteurs d'entiers

Soit $k \in \mathbb{N}$. Nous montrons comment construire un réseau de Petri avec Reset qui implémente la fonction *successeur* sur une liste $L_k = v_0, v_1, v_2, v_3, \ldots,$ de vecteurs dans $\mathbb{N}^k$. L_k n'est pas une énumération exhaustive des vecteurs de $\mathbb{N}^k$. Définissons quels vecteurs sont éléments de L_k.

Définition 51 *Un vecteur $v \in \mathbb{N}^k$ est "propre" si $\forall i,\ v(1) \geq v(i)$.*

Exemple : $(7, 5, 3, 7)$ est propre puisque 7 est plus grand ou égal à 5 ou 3 ou 7. $(7, 8, 1, 1)$ n'est pas propre puisque 7 est plus petit que 8.

Définition 52 *Soit $k \in \mathbb{N}$, la liste L_k est une énumération de tous les vecteurs propres de $\mathbb{N}^k$ à partir de $\bar{0}$ par ordre lexicographique.*

Notation : *$\leq_{lex}$ représente la relation d'ordre lexicographique.*

En fait, L_k est une énumération de tous les vecteurs propres bornés par 0, suivis des vecteurs propres bornés par 1, puis par 2, puis par 3, etc.

Exemple : $L_3 = (0, 0, 0), (1, 0, 0), (1, 0, 1), (1, 1, 0), (1, 1, 1), (2, 0, 0), (2, 0, 1), (2, 0, 2), (2, 1, 0), (2, 1, 1), (2, 1, 2), (2, 2, 0), (2, 2, 1), (2, 2, 2), (3, 0, 0), (3, 0, 1), (3, 0, 2), (3, 0, 3), \cdots$

L_k satisfait la propriété suivante : tout vecteur $v \in \mathbb{N}^{k-1}$ est une projection de $k-1$ composantes d'un vecteur de L_k :

Lemme 70 *$\forall v \in \mathbb{N}^{k-1}$, $\exists v' \in L_k$ tel que $\forall j$, $1 \leq j < k, v'(j+1) = v(j)$ (k entier).*

Preuve : soit $v \in \mathbb{N}^{k-1}$. Posons $x = max\{v(i) \,|\, 1 \leq i < k\}$ et considérons $v' \in \mathbb{N}^k$ tel que $\forall j$, $1 \leq j < k$, $v'(j+1) = v(j)$ et $v'(1) = x$. Le vecteur v' est propre, donc $v' \in L_k$, et satisfait l'énoncé du lemme 70. □

Nous appelons la première composante d'un vecteur la *base*. Trouver le successeur revient à additionner 1 au vecteur, modulo (*base* +1). Techniquement, la fonction successeur $Succ : \mathbb{N}^k \to \mathbb{N}^k$, est calculée par l'algorithme suivant :

fonction du calcul du successeur :

Début
  Si $v(1) = v(i)$ pour tout i, $2 \leq i \leq k$ **alors**
    $pos := 1$
  Sinon
    $pos := max\{i \,|\, v(i) \neq v(1)\}$;
  $v(pos) := v(pos) + 1$;
  $v(i) := 0$ pour tout i, $pos < i \leq k$;
Fin

Exemple : $Succ((8,5,8,8)) = (8,6,0,0)$, $Succ((8,8,8,1)) = (8,8,8,2)$, $Succ((8,4,1,8)) = (8,4,2,0)$, $Succ((8,8,8,8)) = (9,0,0,0)$.

Ainsi, si toutes les composantes de v ont la même valeur, $Succ(v)$ est le vecteur qui incrémente la base et met à zéro toutes les places à droite de la base. Sinon, pour calculer le successeur, on cherche la première composante qui, partant de la droite, a une valeur différente de la base. On incrémente cette composante et on met à zéro toutes les composantes à sa droite.

Comportement du réseau qui calcule la fonction successeur

Décrivons le comportement du réseau N_L qui calcule la fonction successeur de vecteurs propres à k composantes. Tout comme pour l'évaluation des polynômes et la comparaison, N_L possède pour places d'entrée *on* et P_{in} et pour places de sorties *off* et P_{out}.

Le comportement nominal est le suivant :

$$\forall v \in \mathbb{N}^k, v \text{ propre}, (1, v, 0, \bar{0}, \bar{0}) \xrightarrow{*} (0, \bar{0}, 1, succ(v), \bar{0}) \tag{5.10}$$

La définition suivante nous servira pour la seconde propriété.

Définition 53 *Soit $v \in \mathbb{N}^k$, $\hat{v} = \mathtt{max}\ \{w \in \mathbb{N}^k \,|\, w \text{ propre et } w \leq v\}$*

Pour un vecteur v donné, $\hat{v}$ est le plus grand vecteur propre couvert par v. Si v est propre, alors $v = \hat{v}$; sinon $v > \hat{v}$.

Dans N_L, il existe des comportements non nominaux; cependant, si *off* est marquée, N_L accède à une configuration de fin particulière. $\forall v \in \mathbb{N}^k$:

$$(1, v, 0, \bar{0}, \bar{0}) \xrightarrow{*} (x_1, v', x_2, v'', m) \text{ avec } x_2 > 0 \Rightarrow \begin{cases} x_1 = 0 \\ v' = \bar{0} \\ x_2 = 1 \\ v'' \text{ propre et} \\ v'' \leq_{lex} Succ(\hat{v}) \\ m = \bar{0} \end{cases} \tag{5.11}$$

Ceci veut dire que quel que soit le vecteur v, le vecteur v'' obtenu dans une configuration de fin est au mieux égal à $Succ(\hat{v})$ dans L_k (sinon il est prédécesseur).

Enfin, N_L satisfait une dernière propriété:

$$\forall v \in \mathbb{N}^k, (N_L, (1, v, 0, \bar{0}, \bar{0})) \text{ est borné.} \tag{5.12}$$

Un réseau avec arcs de Reset ne peut calculer de façon exacte la fonction successeur, mais il peut la calculer faiblement.

Définition 54 *On dit d'un réseau qui possède la structure décrite plus haut et qui satisfait les propriétés 5.10, 5.11 et 5.12, qu'il calcule (faiblement) la fonction successeur.*

Le réseau N_L est une "implémentation" de la fonction décrite en début de section. Le principe est toujours le même: sur une entrée v propre, si le réseau n'exécute pas la séquence nominale, alors des jetons sont éliminés et le vecteur v'' en sortie sera tel que: $v'' < Succ(v)$; or la relation $<$ sur les vecteurs d'entiers implique la relation lexicographique $\leq_{lex}$ et v'' précède donc $Succ(v)$ dans L_k.

Nous ne détaillerons pas la construction. Nous énonçons seulement la proposition suivante:

Proposition 71 *Étant donné un entier k, il existe un réseau de Petri avec Reset qui calcule (faiblement) la fonction successeur des vecteurs dans $\mathbb{N}^k$.*

5.2.4 Assemblage final

Soient deux polynômes Q et $R \in \mathbb{N}[v_1, .., v_k]$. À partir des réseaux $N_{Q \neq R}$ et N_L, nous pouvons construire le réseau final $N_{Dio(Q,R)}$ qui implémente l'algorithme de la recherche d'une solution au dixième Problème de Hilbert (défini en section 5.2).

Soit le réseau $N_{Q \neq R}$ qui compare Q et R, initialisé avec $(1, \bar{0}, 0, \bar{0}, \bar{0})$ et soit le réseau N_L qui calcule la fonction $Succ$ pour la liste L_k. Les deux réseaux sont fusionnés sur leurs places de la façon suivante: les places $N_{Q \neq R}[P_{out}]$ sont fusionnées aux places $N_L[P_{in}]$ et sont appelées P_{in} dans $N_{Dio(Q,R)}$. Les places $N_{Q \neq R}[P_{in}]$ sont fusionnées aux places $N_L[P_{out}]$ et sont appelées P_{out} dans $N_{Dio(Q,R)}$. La place $N_{Q \neq R}[on]$ est fusionnée avec $N_L[off]$ et elle est appelée *on* dans $N_{Dio(Q,R)}$. Enfin, $N_L[on]$ est fusionnée avec $N_{Q \neq R}[off]$ et elle est appelée *off* dans $N_{Dio(Q,R)}$. Nous illustrons (schématiquement) la construction à la figure 5.8.

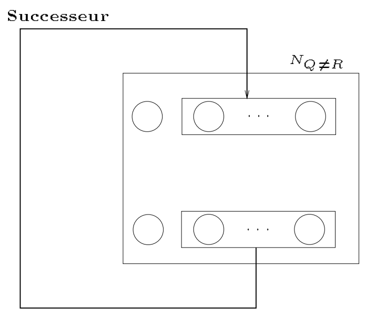


FIG. 5.8 – *Un réseau pour trouver une solution aux dixième Problème de Hilbert*

Pour un vecteur $v \in \mathbb{N}^k$, dès que la comparaison par $N_{Q\neq R}$ est achevée, $N_{Dio(Q,R)}$ réinitialise ses places P_{in} (de $N_{Q\neq R}$) pour la prochaine comparaison, par le calcul dans P_{in} du successeur du vecteur de P_{out}.

$N_{Dio(Q,R)}$ possède les trois propriétés suivantes :

$\forall v \in \mathbb{N}^k$, v propre, si $Q(v) \neq R(v)$ alors :

$$(1, v, 0, \bar{0}, \bar{0}) \xrightarrow{*} (0, \bar{0}, 1, v, \bar{0}) \xrightarrow{*} (1, Succ(v), 0, \bar{0}, \bar{0}) \tag{5.13}$$

ce qui veut dire que partant de v (propre), $N_{Dio(Q,R)}$ peut simuler une boucle de l'algorithme et tester $Succ(v)$, à condition que $Q(v) \neq R(v)$.

De plus $\forall v$, si $Q(v) = R(v)$ alors :

$$(1, v, 0, \bar{0}, \bar{0}) \xrightarrow{*} (x_1, v', x_2, v'', m) \text{ avec } x_1 > 0 \Rightarrow \begin{cases} x_1 = 1 \\ v' \text{ propre et } v' \leq_{lex} v \\ x_2 = 0 \\ v'' = \bar{0} \\ m = \bar{0} \end{cases} \tag{5.14}$$

ce qui veut dire que si $Q(v) = R(v)$, alors $N_{Dio(Q,R)}$ ne pourra tester aucun successeur de v, mais "plafonnera" à v.

Ainsi, pour le réseau $N_{Dio(Q,R)}$ nous avons le comportement suivant :

$$\forall v \in L : (1, v, 0, \bar{0}, \bar{0}) \in Acc(N_{Dio(Q,R)}, (1, \bar{0}, 0, \bar{0}, \bar{0})) \text{ si et seulement si } Q(v) \neq R(v)$$

La dernière propriété est la suivante :

$$N_{Dio(Q,R)}[P_{in}] \text{ borné } \Leftrightarrow N_{Dio(Q,R)} \text{ borné .} \tag{5.15}$$

Définition 55 *On dit d'un réseau qui possède la structure décrite plus haut et qui satisfait les propriétés 5.13 et 5.15, qu'il cherche s'il existe une solution dans L_k au problème des équations diophantiennes pour Q et $R \in \mathbb{N}[v_1, .., v_k]$.*

Proposition 72 *Le réseau $N_{Dio(Q,R)}$ satisfait la définition 55.*

Voici notre proposition principale :

Proposition 73 *Soient Q, $R \in \mathbb{N}[v_1, .., v_k]$, $N_{Dio(Q,R)}$ est borné si et seulement s'il existe un $v \in L_k$ tel que $Q(v) = R(v)$.*

Nous sommes en mesure à présent de réduire le problème des équations diophantiennes au problème de caractère borné pour les réseaux de Petri avec Reset. La construction que nous venons de présenter ne teste pas tous les vecteurs d'entiers, mais seulement une liste de vecteurs propres.

Proposition 74 *Soient Q et $R \in \mathbb{N}[v_1, .., v_k]$. Le dixième Problème de Hilbert sur Q et R se réduit au problème du caractère borné pour les réseaux de Petri avec Reset.*

Preuve : nous définissons Q' et $R' \in \mathbb{N}[v_0, .., v_k]$ à partir de Q et R en ajoutant une variable v_0 supplémentaire, dont le coefficient est zéro.

Ainsi, $\forall v_i \in \mathbb{N}$, $Q(v_1, .., v_k) = Q'(v_0, .., v_k)$ et $R(v_1, .., v_k) = R'(v_0, .., v_k)$. Nous en déduisons que pour tout $v' \in L_{k+1}$, il existe un $v \in \mathbb{N}^k$ tel que $Q(v) = Q'(v')$ et $R(v) = R'(v')$.

Considérons la liste L_{k+1}. Par le lemme 70, pour tout vecteur $v \in \mathbb{N}^k$, il existe un $v' \in L_{k+1}$ tel que $\forall j$, $0 \leq j \leq k, v'(j+1) = v(j)$. Nous en déduisons que pour tout $v \in \mathbb{N}^k$, il existe un $v' \in L_k$ tel que $Q(v) = Q'(v')$ et $R(v) = R'(v')$.

Nous avons donc les équivalences suivantes : $\exists v \in \mathbb{N}^k$ tel que $Q(v) = R(v) \Leftrightarrow \exists v' \in \mathrm{L}_{k+1}$ tel que $Q'(v') = R'(v') \Leftrightarrow N_{Dio(Q',R')}$ est borné.

Nous en déduisons que le dixième Problème de Hilbert sur Q et R se réduit au problème du caractère borné pour les réseaux de Petri avec Reset.

$\square$

Et puisque le dixième Problème de Hilbert est indécidable :

Corollaire 75 *Le caractère borné du réseau pour les réseaux de Petri avec Reset est indécidable.*

5.3 Réseaux de Petri avec Reset et machines à compteurs.

Plusieurs problèmes ont été montrés indécidables pour les réseaux de Petri et leurs extensions par diverses simulations de machines à compteurs, dont l'accessibilité pour les *Post Self-Modifying nets* [Val78b] ou pour les réseaux avec Reset [AK77], ou encore l'équivalence des langages (pour les réseaux étiquetés) et plusieurs autres problèmes d'équivalences pour les réseaux de Petri [Jan95].

Dans cette section, nous allons montrer comment les réseaux de Petri avec Reset peuvent simuler, dans un sens faible, les machines à compteurs, de telle sorte que nous obtenions l'équivalence suivante : une machine à deux compteurs M est bornée si et seulement si un réseau Reset associé N_M est borné. Trois arcs de Reset suffisant à notre construction, nous conclurons que pour les réseaux de Petri avec trois arcs de Reset, le caractère borné du réseau est indécidable.

Nous commencerons par rappeler la définition d'une machine à compteurs, puis nous définirons le problème (indécidable) de caractère borné d'une machine à compteurs. Nous montrerons ensuite que ce problème se réduit au problème du caractère borné du réseau, pour les réseaux de Petri avec Reset.

5.3.1 Machines à compteurs

Une machine déterministe à deux compteurs [Min67] exécute un programme déterministe décrit par une suite finie d'instructions étiquetées de façon unique (pour étiqueter $n+1$ instructions, on utilisera e_0, $e_1, \cdots, e_n$). Les instructions portent sur deux compteurs (entiers naturels) c_1 et c_2 à capacité a priori infinie.

Il y a quatre types d'instructions :

- *Instruction de départ :* cette instruction n'apparaît qu'une fois et elle est étiquetée par e_0 :

  $$e_0 \ : \textit{goto } e_i$$

  où e_i est une étiquette autre que e_0. Nous utilisons cette instruction à des fins de simplicité.

- *Instruction halte :* cette instruction termine le programme et n'apparaît qu'une fois. Nous l'étiquetons par e_n :

  $$e_n \ : \textit{Halte}$$

- *Instruction de type test-à-zéro de c :*

  e_i : *Si* $c = 0$ alors
  goto e_j
  Sinon
  { $c \leftarrow c - 1$;
  goto e_k };

  où e_i, e_j et e_k sont des étiquettes ($e_i, e_j, e_k \neq e_0$ et $e_i \neq e_n$) et où $c \in \{c_1, c_2\}$.

- *Instruction de type incrémentation de c :*

  e_i : $c \leftarrow c + 1$;
  goto e_j;

  où e_i et e_j sont des étiquettes ($e_i, e_j \neq e_0$ et $e_i \neq e_n$) et où $c \in \{c_1, c_2\}$.

À tout moment lors de l'exécution du programme, la configuration courante de la machine est caractérisée par trois paramètres : l'étiquette de la prochaine instruction à exécuter, la valeur de c_1 et la valeur de c_2.

Notation :

- *Nous notons une configuration de la machine par* $<e, x_1, x_2>$ *où* e *est l'étiquette de l'instruction courante,* x_1 *est la valeur de* c_1 *et* x_2 *est la valeur de* c_2*.*

- *On utilisera les notations suivantes pour décrire les actions opérées sur les compteurs : Inc_{c_1}, Inc_{c_2}, $Déc_{c_1}$, $Déc_{c_2}$, Taz_{c_1} ou Taz_{c_2} (incrémentation, décrémentation et test-à-zéro de c_1 ou de c_2).*
- *On utilisera la notation $< e, x_1, x_2 > \xrightarrow{a} < e', x'_1, x'_2 >$ pour décrire le passage de la configuration $< e, x_1, x_2 >$ à la configuration $< e', x'_1, x'_2 >$ par l'action a.*

Sans perte de généralité, nous considérons que les compteurs c_1 et c_2 sont initialisés à zéro. La configuration initiale de M est donc $< e_0, 0, 0 >$. La machine changera de configuration et les compteurs changeront de valeur, au fur et à mesure de l'exécution des instructions.

Remarque : *les instructions de type Test-à-zéro ou Incrémentation sont toujours exécutables. Lorsque la machine atteint l'instruction Halte, elle s'arrête.*

Définition 56 *Soit M une machine à compteurs.*

- *M est bornée par k (k entier), si toute configuration $< e, x_1, x_2 >$ accessible par M est telle que $x_1 + x_2 \leq k$.*
- *M est bornée s'il existe un $k \in \mathbb{N}$ tel que M est bornée par k.*

Nous allons considérer le problème suivant :

• Problème du caractère borné d'une machine à compteurs

Entrée : M une machine déterministe à deux compteurs initialisés à zéro

Sortie : $\begin{cases} \textit{oui} & \text{si } M \text{ est bornée} \\ \textit{non} & \text{sinon} \end{cases}$

Ces machines sont équivalentes aux machines de Turing et on en déduit le théorème suivant :

Théorème 76 *Le problème du caractère borné d'une machine déterministe à deux compteurs initialisés à zéro est indécidable.*

5.3.2 Simulation d'une machine à compteur par un réseau Reset

Étant donnée une machine déterministe M à deux compteurs initialisés à zéro, nous construisons un réseau de Petri avec Reset N_M qui est borné si et seulement si la machine M est bornée.

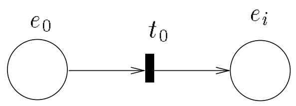
Commençons par donner le principe général. Nous allons construire un réseau N_M qui va simuler M, en se bornant d'abord à une capacité de 0, c'est-à-dire en interdisant que la somme des deux compteurs de M dépasse zéro. Si la somme des compteurs atteint 0 (ce qui est bien sûr vrai au départ), alors N_M va pouvoir commencer une nouvelle simulation de M, à partir de l'état initial, avec une capacité de 1 cette fois. Si la somme des compteurs atteint la borne 1, alors le réseau va pouvoir commencer une nouvelle simulation avec une capacité de 2, et ainsi de suite.

Si M n'est pas bornée, alors le réseau N_M va pouvoir enchaîner une infinité de simulations avec une capacité incrémentée de 1 à chaque essai et, dans ce cas, N_M ne sera pas borné. Si par contre M est bornée, alors N_M ne pourra pas tester de capacité plus grande que la borne maximale (+1) utilisée par M et, dans ce cas, N_M sera borné.

Construction du réseau N_M

Considérons une machine M à $n+1$ instructions étiquetées par $e_0, \cdots, e_n$. Soient INC l'ensemble des étiquettes des instructions de type *incrémentation* et TAZ l'ensemble des étiquettes des instructions de type *test-à-zéro*. Voici la définition du réseau avec Reset associé $N_M = < P_M, T_M, F_M, m_{0M} >$:

- $P_M = \{c_1, c_2, C, p_{init}\} \cup_{i=0}^{n} \{e_i\}$. Les places c_1 et c_2 correspondent respectivement aux compteurs c_1 et c_2. La place C est la place de *capacité* qui contiendra la capacité couramment considérée par le réseau. La place p_{init} interviendra lors du processus d'incrémentation de la capacité, au début d'une nouvelle simulation. Enfin, toute place $e_i \in P_M$ correspond à l'étiquette e_i d'une instruction de la machine.
- $T_M = \{r_{c_1}, r_{c_2}, t_{init}, t_0\} \cup_{i=1}^{n-1}\{Inc_i \mid e_i \in INC\} \cup_{i=1}^{n-1}\{Taz_i, Dec_i \mid e_i \in TAZ\} \cup_{i=1}^{n}\{t_{ci}\}$
- Décrivons la fonction F_M à l'aide de schémas.
  - La transition t_0 correspond à l'instruction de départ :



$$e_0 : goto \;\; e_i; \; (e_i \neq e_0)$$

  - Les transitions Taz_i et Dec_i $(1 \leq i \leq n-1)$ implémentent l'instruction de test-à-zéro. La construction est illustrée à gauche de la figure 5.9 (elle est donnée sans perte de généralité pour le compteur c_1). Le franchissement de Taz_i met la place c_1 à zéro.

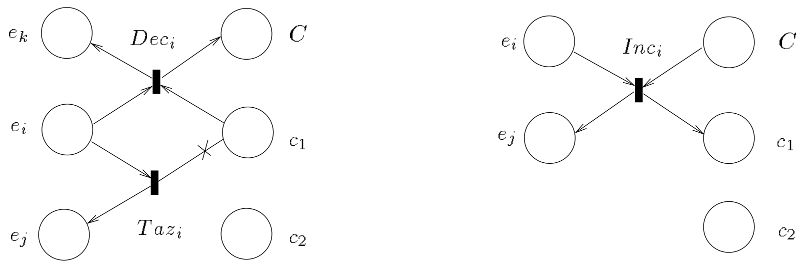


FIG. 5.9 – *Instructions de test-à-zéro et d'incrémentation de c_1.*

  - La transition Inc_i $(1 \leq i \leq n-1)$ implémente l'instruction d'incrémentation. La construction est illustrée à droite de la figure 5.9 (elle est donnée sans perte de généralité pour le compteur c_1).
  - Enfin, le reste de la description de F est propre à la construction et correspond à l'incrémentation de la place de capacité. Référons-nous à la figure 5.10. Toute transition t_{ci} $(1 \leq i \leq n)$ amorce un processus de mise à jour de la capacité courante de N_M. Toute t_{ci} est franchissable lorsque e_i est marquée, met C à zéro, décrémente e_i et incrémente p_{init}. Lorsque p_{init} est marquée, r_{c_1}, r_{c_2} et t_{init} sont franchissables. La transition t_{init} met c_1 et c_2 à zéro.

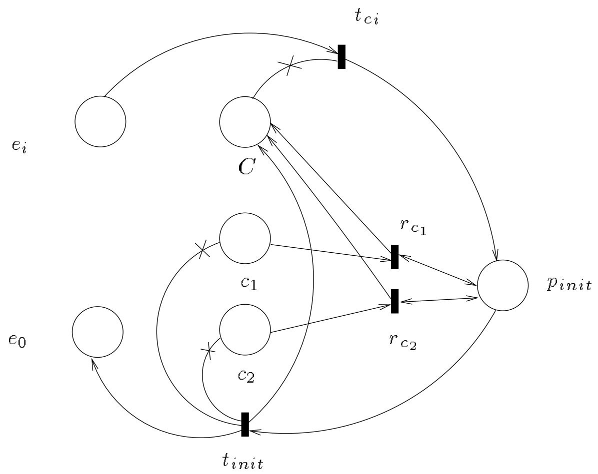


FIG. 5.10 – *Mise à jour de la capacité.*

- $m_{0M}(p) = \begin{cases} 1 & si\ p = e_0 \\ 0 & \text{sinon} \end{cases}$

Remarque : *les seules places de N_M susceptibles d'être non bornées sont c_1, c_2 et C. Un "jeton de contrôle" circule dans les autres places :*

$$pour\ tout\ m \in Acc(N_M),\ \ nous\ avons\ \Sigma_{p \in P_M \setminus \{c_1,c_2,C\}} m(p) = 1$$

Notation : *nous noterons de façon abrégée par (p, x_1, x_2, k), un marquage $m \in \mathbb{N}^{|P_M|}$ tel que $m(c_1) = x_1$, $m(c_2) = x_2$ et $m(C) = k$, et avec le jeton de contrôle dans la place $p \in P_M \setminus \{c_1, c_2, C\}$.*

Exemple : le marquage de départ m_{0M} est $(e_0, 0, 0, 0)$.

Comportement du réseau N_M

Si la machine M n'est pas bornée, alors le réseau pourra effectuer l'exécution infinie suivante :

$$(e_0, 0, 0, 0) \xrightarrow{*} (e_0, 0, 0, 1) \xrightarrow{*} (e_0, 0, 0, 2) \cdots (e_0, 0, 0, i) \xrightarrow{*} (e_0, 0, 0, i+1) \cdots$$

où chaque $(e_0, 0, 0, i)$ correspond au commencement d'une simulation de M avec capacité i, c'est-à-dire avec i pour borne sur la somme des valeurs des deux compteurs.

Quand une transition de type Inc_i ajoute un jeton dans c_1 (ou dans c_2), elle retire un jeton de C. De façon duale, quand une transition de type Dec_i retire un jeton de c_1 (ou de c_2), elle ajoute un jeton dans C. La seule façon d'augmenter le nombre de jetons dans le réseau est de franchir la transition t_{init} qui ajoute un nouveau jeton dans C. Toutes les autres transitions, soit "bougent" les jetons d'une place à l'autre, soit éliminent des jetons par les arcs de Reset. Ainsi, à partir d'un marquage $(e_0, 0, 0, k)$, avec $k \in \mathbb{N}$, tant que t_{init} n'est pas franchie, le nombre maximal de jetons présents dans le réseau ne pourra pas dépasser $k+1$.

Nous détaillons maintenant le comportement des réseaux implémentant les deux types d'instructions au cours d'une simulation à capacité k (qui a commencé à partir du marquage $(e_0, 0, 0, k)$).

Décrivons les comportements nominaux de N_M, c'est-à-dire ceux qui "imitent" correctement M (sans perte de généralité sur le compteur c_1) :

- *Test-à-zéro (figure 5.9) :*

  $$e_i : \textit{Si } c_1 = 0 \textit{ alors goto } e_j \text{ Sinon } \{c_1 \leftarrow c_1 - 1; \textit{ goto } e_k\};$$

  avec $e_i, e_j, e_k \neq e_0$ et $e_i \neq e_n$.

  - Lorsque dans N_M la place e_i est marquée, si la place c_1 est marquée, alors la transition $Déc_i$ est franchissable. $Déc_i$ retire un jeton de c_1 pour l'ajouter dans C et fait passer le contrôle de e_i à e_j.
  - Si la place c_1 n'est pas marquée, la seule transition franchissable est Taz_i, qui met c_1 à zéro (déjà vide) et fait passer le contrôle de e_i à e_k.

- *Incrémentation (figure 5.9) :*

  $$e_i : c_1 \leftarrow c_1 + 1; \textit{ goto } e_j; \quad (\text{avec } e_i, e_j \neq e_0 \text{ et } e_i \neq e_n)$$

  Lorsque dans N_M la place e_i est marquée, si la place C est marquée, alors Inc_i est franchissable. La transition Inc_i retire un jeton de C pour l'ajouter dans c_1, et fait passer le contrôle de e_i vers e_j.

- *Mise à jour de la capacité (figure 5.10) :* si la place C est vide (ce qui correspond au cas où les compteurs utilisent la capacité maximale) et que e_i est marquée, alors N_M peut franchir t_{ci} qui met C (déjà vide) à zéro et fait passer le contrôle à p_{init}.

  Lorsque p_{init} est marquée, r_{c_1} et r_{c_2} sont franchissables. Elles ont pour but de "renflouer" C en lui ré-injectant les jetons de c_1 et c_2. Si le renflouement est effectué jusqu'au bout, alors C retrouve la capacité qu'elle avait au début de la simulation courante, c'est-à-dire k ici. Ensuite, t_{init} peut être franchie, mettant c_1 et c_2 (déjà vides) à zéro, redonnant le contrôle à e_0 (qui correspond à l'étiquette de l'instruction de départ de M) et ajoutant un nouveau jeton dans la capacité C. Le réseau accède alors au marquage $(e_0, 0, 0, k+1)$ qui correspond à une nouvelle simulation, avec une capacité augmentée de 1.

Les comportements que nous venons de décrire sont caractérisés par le fait que les mises à zéro sont toujours effectuées sur des places déjà vides. Voyons à présent ce qu'il se passe en cas d'erreur. L'erreur est caractérisée par des mises à zéro effectuées sur des places non vides. Nous reprenons les types d'instructions (toujours pour une simulation à capacité k).

- *Test-à-zéro (figure 5.9) :* si les places e_i et c_1 sont marquées (permettant de franchir $Déc_i$) et que Taz_i est franchie, alors la mise à zéro de c_1 va éliminer au moins un jeton du réseau. Après le franchissement de Taz_i, la somme des jetons de c_1, c_2 et C sera strictement inférieure à k.

- *Mise à jour de la capacité (figure 5.10) :* si le réseau franchit t_{c_i}, alors que C est marquée, alors la mise à zéro de C élimine au moins un jeton du réseau. Après le franchissement de t_{ci}, la somme des jetons de c_1, c_2 et C sera strictement inférieure à k.

- *Renflouement (figure 5.9) :* il y a une autre erreur possible, propre au réseau N_M. Si le renflouement de C par les transitions r_{c_1} et r_{c_2} n'est pas effectué jusqu'au bout, la transition t_{init}, en mettant c_1 et c_2 à zéro, diminue aussi le nombre de jetons dans le réseau.

La seule transition susceptible d'ajouter des jetons supplémentaires dans N_M est t_{init} qui commence une nouvelle exécution de M. Partant du marquage $(e_0, 0, 0, k)$, si le réseau fait une erreur en simulant M, alors la prochaine simulation ne pourra commencer que par $(e_0, 0, 0, k')$ avec $k' \leq k$. En quelque sorte, le réseau "revient en arrière" sur un marquage déjà visité.

5.3.3 Indécidabilité du caractère borné du réseau pour les réseaux Reset

Nous allons montrer que M est bornée si et seulement si N_M est borné. Voyons un premier sens de la réduction pour lequel nous utilisons le lemme suivant :

Lemme 77 *Soit k entier, si un segment initial de l'exécution de M est donné par la suite de configurations $<e_{i_j}, x_{1j}, x_{2j}>$, $1 \leq j \leq l$, où $x_{1j} + x_{2j} \leq k$ pour tout j, alors :*

$$(e_0, 0, 0, k) \xrightarrow{*} (e_{i_j}, x_{1j}, x_{2j}, k - x_{1j} - x_{2j}) \text{ pour tout } j$$

Preuve : nous donnons une preuve par induction sur la longueur l d'un segment borné par k.

Pour $l = 0$, le lemme est vérifié car au départ M est dans la configuration $<e_0, 0, 0>$ et N_M est dans l'état initial $(e_0, 0, 0, k)$. Supposons le lemme vrai pour tout segment borné par k de longueur $n - 1$. Montrons qu'il est vrai pour un segment de longueur n.

Soit $<e_i, x_1, x_2>$ la $(n-1)^{\text{ème}}$ configuration d'un segment initial d'un calcul de M, borné par k. Par hypothèse d'induction, nous savons qu'il existe un $(e_i, x_1, x_2, k - x_1 - x_2) \in Acc(N_M, (e_0, 0, 0, k))$. Voyons quelle est la $n^{\text{ème}}$ instruction exécutée par M.

- Cas d'un test-à-zéro de c_1 (le cas du test-à-zéro de c_2 est similaire) :

  $$e_i : \textit{Si } c_1 = 0 \textit{ alors goto } e_j \text{ Sinon } \{c_1 \leftarrow c_1 - 1; \textit{ goto } e_k\};$$

  avec $e_i, e_j, e_k \neq e_0$ et $e_i \neq e_n$. Deux sous-cas sont possibles.

  1. Si $c_1 = 0$, alors dans M nous avons :

     $$<e_i, x_1, x_2> \xrightarrow{Taz_{c_1}} <e_j, x_1, x_2>$$

     et dans N_M nous avons :

     $$(e_i, x_1, x_2, k - x_1 - x_2) \xrightarrow{Taz_i} (e_j, x_1, x_2, k - x_1 - x_2)$$

     Nous vérifions donc bien que $(e_j, x_1, x_2, k - x_1 - x_2) \in Acc(N_M, (e_0, 0, 0, k))$.

2. Si $c_1 \neq 0$, alors dans M nous avons :

$$< e_i, x_1, x_2 > \stackrel{Dec_{c_1}}{\longrightarrow} < e_k, x_1 - 1, x_2 >$$

et dans N_M nous avons :

$$(e_i, x_1, x_2, k - x_1 - x_2) \stackrel{Dec_{c_i}}{\rightarrow} (e_k, x_1 - 1, x_2, k - x_1 - x_2 + 1)$$

Nous vérifions donc bien que $(e_k, x_1 - 1, x_2, k - (x_1 - 1) - x_2) \in Acc(N_M, (e_0, 0, 0, k))$.

- Cas d'une incrémentation de c_1 (le cas de l'incrémentation de c_2 est similaire) :

$$e_i : c_1 \leftarrow c_1 + 1;\ goto\ e_j;$$

avec $e_i, e_j \neq e_0$ et $e_i \neq e_n$. Dans M nous avons :

$$< e_i, x_1, x_2 > \stackrel{Inc_{c_1}}{\longrightarrow} < e_j, x_1 + 1, x_2 >$$

avec $(x_1 + x_2) < k$, sinon la nouvelle configuration ne serait pas bornée par k. Dans N_M nous avons :

$$(e_i, x_1, x_2, k - x_1 - x_2) \stackrel{Inc_{c_i}}{\rightarrow} (e_j, x_1 + 1, x_2, k - x_1 - x_2 - 1)$$

Nous vérifions donc bien que $(e_j, x_1 + 1, x_2, k - (x_1 + 1) - x_2) \in Acc(N_M, (e_0, 0, 0, k))$.

Nous en concluons que pour toute configuration $< e_i, x_1, x_2 >$ d'un segment initial de M borné par k, il existe un marquage $(e_i, x_1, x_2, k - x_1 - x_2) \in Acc(N_M, (e_0, 0, 0, k))$. □

Proposition 78 *Si M n'est pas bornée alors N_M n'est pas borné.*

Preuve : nous allons montrer que si M est non bornée, alors pour tout k, le marquage $(e_0, 0, 0, k)$ est accessible dans N_M. Nous donnons une preuve par induction sur k.

Le cas $k = 0$ est tout de suite vérifié puisque $(e_0, 0, 0, 0)$ est la configuration initiale de N_M. Supposons que pour tout $i < k$, $(e_0, 0, 0, i) \in Acc(N_M)$. Montrons à présent que $(e_0, 0, 0, k) \in Acc(N_M)$.

Soit $< e_j, x_1, x_2 >$ la première configuration accessible par M telle que $x_1 + x_2 = k$. Cette configuration existe puisque M est non borné et elle est forcément atteinte après une instruction de type incrémentation (que l'on suppose sans perte de généralité sur c_1). Dans M, nous avons :

$$< e_0, 0, 0 > \stackrel{*}{\rightarrow} < e_i, x_1 - 1, x_2 > \stackrel{Inc_{c_1}}{\longrightarrow} < e_j, x_1, x_2 >$$

Par hypothèse d'induction, puisque $x_1 - 1 + x_2 < k$ alors $(e_0, 0, 0, k - 1) \in Acc(N_M)$. Par le lemme 77 nous savons que :

$$(e_0, 0, 0, k - 1) \stackrel{*}{\rightarrow} (e_i, x_1 - 1, x_2, 0)$$

De là, N_M peut franchir la suite de transitions $\sigma = t_{ci} {r_{c_1}}^{x_1 - 1} {r_{c_2}}^{x_2}$, qui permet d'effectuer :

$$(e_i, x_1 - 1, x_2, 0) \xrightarrow{\sigma} (p_{init}, 0, 0, x_1 - 1 + x_2) \xrightarrow{t_{init}} (e_0, 0, 0, x_1 + x_2)$$

Nous vérifions donc bien que $(e_0, 0, 0, k) \in Acc(N_M)$.

Nous en concluons que pour tout k, il est possible d'accéder dans N_M à un marquage contenant au moins k jetons et que donc, N_M n'est pas borné.

□

Nous allons prouver la proposition inverse, c'est-à-dire que si la machine M est bornée, alors N_M l'est aussi. Nous précédons cette proposition de plusieurs lemmes. Remarquons d'abord qu'à chaque fois que le réseau N_M crée un nouveau jeton, il accède à un marquage de la forme $(e_0, 0, 0, k)$.

Lemme 79 *Soient M une machine et N_M son réseau associé, $t \in T_M$ et $m_1, m_2 \in Acc(N_M)$ tels que $m_1 \xrightarrow{t} m_2$. Si m_1 contient $k > 0$ jetons et m_2 contient $k+1$ jetons, alors $m_2 = (e_0, 0, 0, k)$.*

Preuve : pour passer de m_1 à m_2, il faut forcément franchir t_{init} qui est la seule transition qui peut créer un jeton. Franchir t_{init} met c_1 et c_2 à zéro, ajoute un jeton dans C et fait passer le jeton de contrôle dans e_0. Forcément, puisque m_2 contient $k+1$ jetons, $m_2 = (e_0, 0, 0, k)$. □

Corollaire 80 *Une exécution dans N_M amenant à un marquage m contenant $k+1$, $k \geq 0$, jetons est forcément de la forme :*

$$m_{0M} \xrightarrow{*} (e_0, 0, 0, 1) \xrightarrow{*} (e_0, 0, 0, 2) \cdots (e_0, 0, 0, k-1) \xrightarrow{*} (e_0, 0, 0, k) \xrightarrow{*} m$$

Caractérisons à présent l'allure d'une séquence particulière permettant d'accéder à $(e_0, 0, 0, k)$.

Lemme 81 *Soient un réseau N_M et $k > 0$. Si $(e_0, 0, 0, k) \in Acc(N_M)$ alors $\exists \sigma_1, \sigma_2 \in T_M^*$ telles que :*

$$m_{0M} \xrightarrow{\sigma_1} (e_0, 0, 0, k-1) \xrightarrow{\sigma_2} (e_0, 0, 0, k)$$

et telles que tous les marquages précédant $(e_0, 0, 0, k)$ à partir de $(e_0, 0, 0, k-1)$ contiennent k jetons.

Preuve : supposons que $(e_0, 0, 0, k) \in Acc(N_M)$ et considérons une plus petite séquence $\sigma \in T_M^*$ telle que $m_{0M} \xrightarrow{\sigma} (e_0, 0, 0, k)$. Par le corollaire 80, le marquage $(e_0, 0, 0, k-1)$ apparaît dans l'exécution de σ et $\exists\, \sigma_1,\ \sigma_2 \in T_M^*$ telles que :

$$\sigma = \sigma_1 \sigma_2 \text{ et } m_{0M} \xrightarrow{\sigma_1} (e_0, 0, 0, k-1) \xrightarrow{\sigma_2} (e_0, 0, 0, k)$$

Considérons la partition de σ qui minimise la longueur de σ_2 (on prend le "dernier" $(e_0, 0, 0, k-1)$ de la suite de marquages obtenus par le franchissement de σ).

Supposons par contradiction qu'il existe un m' entre $(e_0, 0, 0, k-1)$ et $(e_0, 0, 0, k)$ qui ne contienne pas k jetons. Il y a deux cas :

- m' contient $k+1$ jetons ou plus. Dans ce cas, par le corollaire 80, le marquage $(e_0, 0, 0, k)$ précède m'. Ceci mène à une contradiction, car si c'était le cas, nous aurions :
$$m_{0M} \xrightarrow{\sigma'} (e_0, 0, 0, k) \xrightarrow{\sigma''} m' \xrightarrow{\sigma'''} (e_0, 0, 0, k)$$
avec $\sigma = \sigma'\sigma''\sigma'''$, $|\sigma'''| > 1$, et σ ne serait pas une plus petite séquence menant à $(e_0, 0, 0, k)$.

- m' contient moins de k jetons. Forcément, entre m' et $(e_0, 0, 0, k)$, il y a un marquage m'' qui contient k jetons, sinon on ne pourrait pas accéder à $(e_0, 0, 0, k)$ qui contient $k+1$ jetons. Mais par le lemme 79, le premier marquage après m' contenant k jetons est $(e_0, 0, 0, k-1)$. Ceci mène à une contradiction, car si c'était le cas nous aurions :
$$m_{0M} \xrightarrow{\sigma_1} (e_0, 0, 0, k-1) \xrightarrow{\sigma'} m' \xrightarrow{\sigma''} (e_0, 0, 0, k-1) \xrightarrow{\sigma'''} (e_0, 0, 0, k)$$
avec $\sigma_2 = \sigma'\sigma''\sigma'''$, $|\sigma'| > 1$, et la séquence σ_2 ne serait pas la plus petite séquence telle que $(e_0, 0, 0, k-1) \xrightarrow{*} (e_0, 0, 0, k)$

Nous en concluons que si $(e_0, 0, 0, k) \in Acc(N_M)$, alors il existe σ_1, σ_2 telles que
$$m_{0M} \xrightarrow{\sigma_1} (e_0, 0, 0, k-1) \xrightarrow{\sigma_2} (e_0, 0, 0, k)$$
et telles que tout marquage m' accessible au cours de σ_2 (et précédant $(e_0, 0, 0, k)$) contient k jetons (incluant le jeton de contrôle). $\square$

Nous montrons qu'à partir d'un marquage $(e_0, 0, 0, k)$, tant que le réseau N_M visite des marquages contenant globalement $k+1$ jetons, alors il imite correctement la machine M.

Lemme 82 *Soient M une machine à deux compteurs et N_M son réseau associé, et soit $k \in \mathbb{N}$. Posons $Etiq = \cup_{i=0}^{n}\{e_i\}$, l'ensemble des places de N_M qui correspondent aux étiquettes de M. Considérons $\sigma \in {T_M}^*$, $\sigma = t_{i_1}t_{i_2}t_{i_3}\cdots t_{i_n}$, telle que*
$$(e_0, 0, 0, k) \xrightarrow{t_{i_1}} m_1 \xrightarrow{t_{i_2}} m_2 \xrightarrow{t_{i_3}} \cdots \xrightarrow{t_{i_n}} m_n$$
avec $m_n = (p, x_1, x_2, x_3)$. Si $\forall i \in [1..n]$, $m_i(c_1) + m_i(c_2) + m_i(C) = k$ et si $p \in Etiq$ alors $<p, x_1, x_2>$ est une configuration accessible par M.

Preuve : nous prouvons ce lemme par induction sur la longueur de σ.

Le cas où $|\sigma| = 0$ est vérifié car le marquage initial de N_M est $(e_0, 0, 0, k)$ et la configuration initiale de M est $<e_0, 0, 0>$.

Supposons que le lemme soit vrai pour toute séquence $\sigma \in {T_M}^*$ de longueur $n-1$. Montrons que le lemme est vrai pour une séquence de longueur n.

Considérons une séquence $\sigma \in {T_M}^*$, $|\sigma| = n$, $\sigma = \sigma' t$ avec $t \in T_M$, telle que
$$(e_0, 0, 0, k) \xrightarrow{\sigma'} (p, x_1, x_2, x_3) \xrightarrow{t} (p', x'_1, x'_2, x'_3)$$
dont tout marquage m intermédiaire soit tel que $m(c_1) + m(c_2) + m(C) = k$. Considérons les types possible de t :

- $t = r_{c_1}$ ou $t = r_{c_2}$ ou t est de type t_{ci}, alors $p' = p_{init}$, mais $p' \notin Etiq$.

- t est de type *Inc*$_i$ et incrémente c_1 (le cas où c_2 est incrémenté est similaire), alors $p = e_i$, $p' = e_j$ avec $e_i, e_j \in Etiq$. Nous avons dans N_M :

$$(e_0, 0, 0, k) \xrightarrow{\sigma'} (e_i, x_1, x_2, x_3) \xrightarrow{Inc_i} (e_j, x_1 + 1, x_2, x_3 - 1)$$

Par hypothèse d'induction, $< e_i, x_1, x_2 >$ est une configuration accessible par M. Par construction de N_M, nous avons dans M l'instruction :

$$e_i \ : c_1 \leftarrow c_1 + 1; goto\ e_j;$$

avec $e_i, e_j \neq e_0$ et $e_i \neq e_n$, et donc pour M nous avons :

$$< e_i, x_1, x_2 > \xrightarrow{Inc_{c_1}} < e_j, x_1 + 1, x_2 >$$

Nous vérifions donc bien que $< e_j, x_1 + 1, x_2 >$ est accessible par M.

- t est de type *Taz*$_i$ et met c_1 à zéro (le cas où c_2 est mis à zéro est similaire), alors $p = e_i$, $p' = e_j$ avec $e_i, e_j \in Etiq$. Nous avons dans N_M :

$$(e_0, 0, 0, k) \xrightarrow{\sigma'} (e_i, x_1, x_2, x_3) \xrightarrow{Taz_i} (e_j, 0, x_2, x_3)$$

mais puisque, par hypothèse, $x_1 + x_2 + x_3 = x_2 + x_3 = k$, alors forcément $x_1 = 0$. Par hypothèse d'induction, $< e_i, x_1, x_2 >=< e_i, 0, x_2 >$ est une configuration accessible par M. Par construction de N_M, nous avons dans M l'instruction :

$$e_i : Si\ c_1 = 0\ alors\ goto\ e_j\ \ Sinon\ \{c_1 \leftarrow c_1 - 1;\ goto\ e_k\};$$

avec $e_i, e_j, e_k \neq e_0$ et $e_i \neq e_n$. Puisque $c_1 = 0$, pour M nous avons :

$$< e_i, 0, x_2 > \xrightarrow{Taz_{c_1}} < e_j, 0, x_2 >$$

Nous vérifions donc bien que $< e_j, 0, x_2 >$ est accessible par M.

- t est de type *Dec*$_i$ et décrémente c_1 (le cas où c_2 est décrémenté est similaire), alors $p = e_i$, $p' = e_k$ avec $e_i, e_k \in Etiq$ et nous avons dans N_M :

$$(e_0, 0, 0, k) \xrightarrow{\sigma'} (e_i, x_1, x_2, x_3) \xrightarrow{Dec_i} (e_k, x_1 - 1, x_2, x_3 + 1)$$

avec $x_1 > 0$ (sinon *Dec*$_i$ ne serait pas franchissable). Par hypothèse d'induction, $< e_i, x_1, x_2 >$ est une configuration accessible par M. Par construction de N_M, nous avons dans M l'instruction :

$$e_i : Si\ c_1 = 0\ alors\ goto\ e_j\ \ Sinon\ \{c_1 \leftarrow c_1 - 1;\ goto\ e_k\};$$

avec $e_i, e_j, e_k \neq e_0$ et $e_i \neq e_n$, et comme $c_1 = x_1 > 0$, nous avons pour M :

$$< e_i, x_1, x_2 > \xrightarrow{Dec_{c_1}} < e_k, x_1 - 1, x_2 >$$

Nous vérifions donc bien que $< e_k, x_1 - 1, x_2 >$ est accessible par M.

- $t = t_0$ et correspond à l'instruction de départ de M. Dans ce cas $p = e_0$, $p' = e_i$ avec $e_i \in Etiq$, et nous avons dans N_M :

$$(e_0, 0, 0, k) \xrightarrow{\sigma'} (e_0, x_1, x_2, x_3) \xrightarrow{t_0} (e_i, x_1, x_2, x_3)$$

  Par construction de N_M, la seule transition qui peut marquer e_0 est t_{init}. Cette transition a donc été franchie juste avant t_0, mettant c_1 et c_2 à zéro. Nous avions donc dans N_M :

$$(e_0, 0, 0, k) \xrightarrow{\sigma'} (e_0, 0, 0, x_3) \xrightarrow{t_0} (e_i, 0, 0, x_3)$$

  et par hypothèse d'induction $< e_0, 0, 0 >$ est une configuration accessible par M. Par construction de N_M nous avons dans M l'instruction :

$$e_0 \ : \textit{goto } e_i \text{ (avec } e_i \neq e_0)$$

  Nous vérifions donc bien que $< e_i, 0, 0 >$ est accessible dans M.

- $t = t_{init}$ et c_1 et c_2 sont mises à zéro. Nous avons $p' = e_0$ et $p = p_{init}$:

$$(e_0, 0, 0, k) \xrightarrow{\sigma'} (p_{init}, x_1, x_2, x_3) \xrightarrow{t_{init}} (e_0, 0, 0, x_3 + 1)$$

  où $< e_0, 0, 0 >$ est la configuration initiale de M, et donc est accessible par M.

Dans tous les cas, le lemme est vérifié pour la séquence σ de longueur n. Nous en concluons que le lemme est vrai pour toute séquence de ${T_M}^*$. □

Nous prouvons maintenant que si $(e_0, 0, 0, k)$ est accessible dans N_M, alors il existe une configuration accessible dans M dont la somme des deux compteurs est égale à $k-1$. Nous prouvons ici que la machine M peut imiter N_M.

Lemme 83 *Soient M une machine et N_M son réseau associé. Si $(e_0, 0, 0, k) \in Acc(N_M)$ ($k > 0$) alors il existe une configuration $< e, v_1, v_2 >$ accessible dans M avec $v_1 + v_2 = k-1$.*

Preuve : puisque $(e_0, 0, 0, k) \in Acc(N_M)$, appliquons le lemme 81 et considérons les séquences σ_1, $\sigma_2 \in T_M^*$ telles que :

$$m_{0M} \xrightarrow{\sigma_1} (e_0, 0, 0, k-1) \xrightarrow{\sigma_2} (e_0, 0, 0, k)$$

et telles que tout m' entre $(e_0, 0, 0, k-1)$ et $(e_0, 0, 0, k)$ contient k jetons. Nous allons "remonter" σ_2 afin de montrer quels sont les prédécesseurs de $(e_0, 0, 0, k)$. La dernière transition de σ_2 est forcément t_{init} puisque c'est la seule qui puisse ajouter un jeton. Nous avons dans N_M :

$$m_{0M} \xrightarrow{\sigma_1} (e_0, 0, 0, k-1) \xrightarrow{\sigma'} (p_{init}, x_1, x_2, k-1) \xrightarrow{t_{init}} (e_0, 0, 0, k)$$

avec forcément $x_1 = x_2 = 0$, sinon $(p_{init}, x_1, x_2, k-1)$ contiendrait plus de k jetons. Pour marquer p_{init}, il faut forcément qu'une des n transitions t_{ci} ait été franchie. Nous avons la situation suivante :

$$m_{0M} \xrightarrow{\sigma_1} (e_0, 0, 0, k-1) \xrightarrow{*} (e_i, x_3, x_4, x_5) \xrightarrow{t_{ci}} (p_{init}, x_3, x_4, 0) \xrightarrow{\sigma_5} (p_{init}, 0, 0, k-1) \xrightarrow{t_{init}} (e_0, 0, 0, k)$$

avec $\sigma_r \in \{r_{c_1}, r_{c_2}\}^*$. Forcément, $x_3 + x_4 = k - 1$ sinon $(p_{init}, x_3, x_4, 0)$ ne contiendrait pas k jetons. On en déduit que $x_5 = 0$.

Posons $v_1 = x_3$ et $v_2 = x_4$. Nous avons une séquence $(e_0, 0, 0, k-1) \xrightarrow{*} (e_i, v_1, v_2, 0)$, avec tous les marquages intermédiaires contenant k jetons. Par le lemme 82, nous en déduisons que $< e_i, v_1, v_2 >$ avec $v_1 + v_2 = k - 1$ est une configuration accessible par M. □

Voici notre proposition principale.

Proposition 84 *Si M est bornée, alors N_M est borné.*

Preuve : nous prouvons que si N_M est non borné, alors M est non bornée.

Si N_M est non borné, alors pour tout n, il existe un marquage $m \in Acc(N_M)$ contenant $n + 1$ jetons (c'est vrai car le réseau n'ajoute qu'un seul jeton à la fois et ne pourra donc "sauter" des valeurs). Par le lemme 79, le marquage $(e_0, 0, 0, n) \in Acc(N_M)$ et par le lemme 83, il existe une configuration $< e, x_1, x_2 >$ accessible par M telle $x_1 + x_2 = n - 1$.

Nous en concluons que pour tout n, il existe une configuration de la machine $< e, x_1, x_2 >$ telle que $x_1 + x_2 = n - 1$ et donc que M n'est pas bornée. □

Nous concluons des propositions 78 et 84 le théorème suivant :

Théorème 85 *Pour les réseaux de Petri avec Reset, le caractère borné du réseau est indécidable.*

Enfin, puisque seules trois places sont mises à zéro, par une construction simple (que nous ne détaillerons pas ici), on peut étendre le résultat en donnant une borne de trois sur le nombre d'arcs suffisant à rendre ce problème indécidable :

Théorème 86 *Pour les réseaux de Petri avec* **trois** *arcs de Reset, le caractère borné du réseau est indécidable.*

Terminons cette section avec un résultat récemment annoncé par Petr JANČAR :

Théorème 87 *[Jan98] Pour les réseaux de Petri avec* **deux** *arcs de Reset, le caractère borné du réseau est décidable.*

Nous en concluons que *trois* est une borne optimale sur le nombre d'arcs de Reset suffisant à rendre ce problème indécidable.

5.3.4 Indécidabilité du caractère borné d'une place pour les réseaux Reset et Transfert

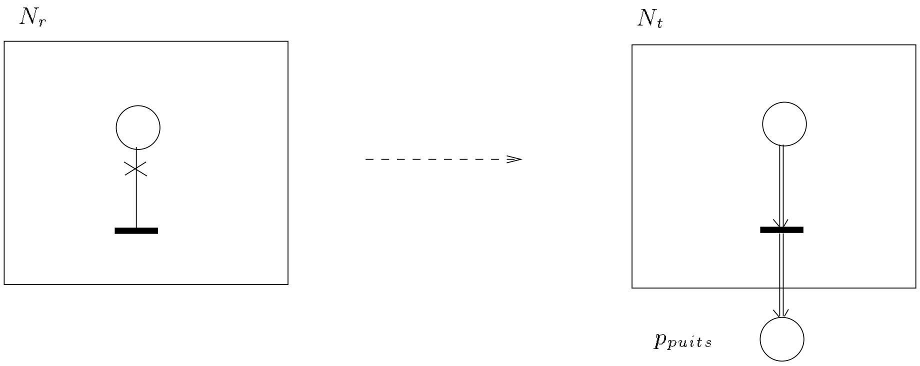
Sachant que caractère borné d'un réseau avec arcs de Reset est indécidable, on obtient directement l'indécidabilité du caractère borné d'une place.

Corollaire 88 *Le caractère borné d'une place est indécidable pour les réseaux de Petri avec arcs de Reset et les G-Post + Reset.*

Preuve : un réseau de Petri avec Reset est borné si et seulement si toutes ses places sont bornées. Décider du caractère borné d'une place pour les réseaux de Petri avec Reset (et donc d'un G-Post + Reset) impliquerait la décision du caractère borné du réseau pour les Reset. □

Corollaire 89 *Le caractère borné d'une place est indécidable pour les réseaux de Petri avec arcs de Transfert et les G-Post + Transfert.*

Preuve : il suffit de transformer les arcs de Reset d'un réseau Reset N_r en arcs de Transfert se déversant dans une place "puits" :



Appelons N_t le réseau résultant. Nous avons N_r borné si et seulement si toute place $p \in P_t \setminus \{p_{puits}\}$ est bornée dans N_t. Décider du caractère borné d'une place pour les Transfert (et donc pour les G-Post + Transfert) impliquerait la décision du caractère borné du réseau pour les Reset. □

5.3.5 Tableau récapitulatif

Voici le tableau récapitulatif mis à jour.

	Terminaison	Couverture	Réseau borné	Place bornée	Réseau structurellement borné	Accessibilité Blocage	Vivacité t-Vivacité
G-Réseaux *Self-Modifying*	**I**	**I**	**I**	**I**	**I**	**I**	**I**
G-Post + Reset	**D**	**D**	**I**	**I**	?	**I**	**I**
Reset	**D**	**D**	**I**	**I**	?	**I**	**I**
G-Post + Transfert	**D**	**D**	**D**	**I**	?	**I**	**I**
Transfert	**D**	**D**	**D**	**I**	?	**I**	**I**
G-Post *Post Self-Modifying*	**D**	**D**	**D**	**D**	**D**	**I**	**I**
Petri	**D**	**D**	**D**	**D**	**D**	**D**	**D**

Il reste maintenant à traiter du caractère structurellement borné des réseaux avec Reset et Transfert.

5.4 Indécidabilité du caractère structurellement borné pour les réseaux Reset

Nous allons montrer que le problème du caractère borné d'une machine à deux compteurs se réduit au problème de caractère structurellement borné pour les réseaux de Petri avec Reset. Nous montrerons ensuite que le problème de terminaison d'une machine à deux compteurs se réduit au *complémentaire* du problème de caractère structurellement borné des réseaux de Petri avec Reset.

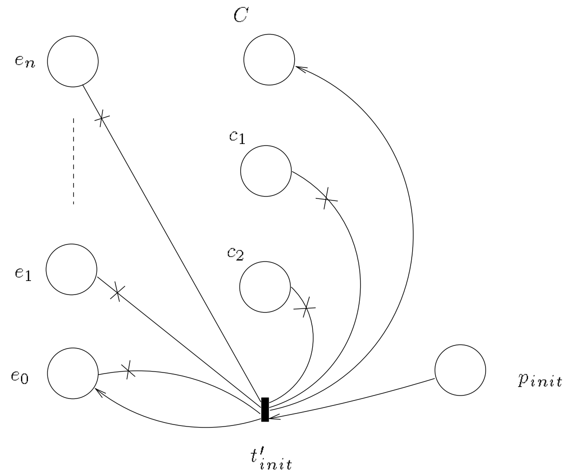
Le problème de caractère borné ainsi que le problème de terminaison d'une machine à compteurs sont tous les deux indécidables, mais semi-décidables. Nous en conclurons que ni le problème de caractère structurellement borné, ni son complémentaire, ne sont semi-décidables. Pour les réseaux de Petri avec Reset, le caractère structurellement borné est donc strictement plus difficile que le problème du caractère borné d'un réseau, qui est lui semi-décidable.

5.4.1 Réduction du problème du caractère borné d'une machine à compteurs

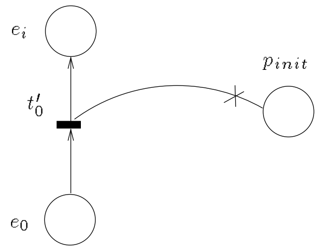
Étant donnée une machine M à deux compteurs (c_1 et c_2) et à $n+1$ instructions, nous allons construire un réseau de Petri avec arcs de Reset N'_M qui est structurellement borné si et seulement si M est bornée.

Nous adaptons la construction de $N_M =< P_M, T_M, F_M, m_{0M} >$ qui a été présentée à la section 5.3.2 et apparaît aux figures 5.9 et 5.10. Nous définissons N'_M tel que $P'_M = P_M$. Ensuite, l'ensemble T'_M est presque identique à T_M, mais nous renommons t_0 par t'_0 et t_{init} par t'_{init}.

La fonction de flot F'_M est une copie de F_M avec deux adaptations qui concernent t'_{init} et t'_0. Puisque le marquage initial de N'_M est quelconque, le but des adaptations est de contraindre N'_M à atteindre "le plus vite possible" un marquage qui ne contienne qu'un jeton de contrôle pour l'ensemble des places $T'_M \setminus \{c_1, c_2, C\}$. Pour cela, la transition t'_{init} de N'_M met à zéro toutes les places de P_M, sauf p_{init} et C. Nous ne pouvons pas aussi mettre p_{init} à zéro, car le modèle ne le permet pas.



Ainsi, dès que t'_{init} est franchie, N'_M est "presque" dans un marquage de type $(e_0, 0, 0, k)$ avec $k \in \mathbb{N}$, sauf que p_{init} peut encore contenir des jetons. Pour remédier à cela, la transition t'_0, correspondant à l'instruction de départ de M, met la place p_{init} à zéro.



Les adaptations que nous avons faites sont telles qu'à partir d'un marquage de type $(e_0, 0, 0, k)$, les réseaux N_M et N'_M ont le même comportement :

$$\forall k \in \mathbb{N},\ Acc(N_M, (e_0, 0, 0, k)) = Acc(N'_M, (e_0, 0, 0, k))$$

Voici nos deux propositions principales :

Proposition 90 *Si M n'est pas bornée alors N'_M n'est pas structurellement borné.*

Preuve : supposons que M ne soit pas bornée, alors N'_M n'est pas borné sur le marquage initial $(e_0, 0, 0, 0)$. La preuve est identique à celle de la proposition 78. Nous en déduisons que N'_M n'est pas structurellement borné.

$\square$

Proposition 91 *Si M est bornée alors N'_M est structurellement borné.*

Preuve : nous montrons que si N'_M n'est pas structurellement borné, alors M n'est pas bornée.

Supposons qu'il existe un $m_0 \in \mathbb{N}^{|P_M|}$ tel que (N'_M, m_0) ne soit pas fini. Alors forcément, il existe une séquence franchissable à partir de m_0 contenant une infinité de franchissements de t'_{init}, puisque t'_{init} est la seule transition de N'_M qui puisse ajouter des jetons.

Considérons le premier franchissement de t'_{init}. Soit $\sigma \in (T'_M \setminus \{t'_{init}\})^*$ et $m_1,\ m_2 \in \mathbb{N}^{|P'_M|}$ tels que :

$$m_0 \xrightarrow{\sigma} m_1 \xrightarrow{t'_{init}} m_2$$

Le marquage m_2 est "presque" de type $(e_0, 0, 0, m_1(C)+1)$, sauf que $m_2(p_{init}) = m_1(p_{init}) - 1$ et donc p_{init} n'est pas forcément vide. À partir de m_2, N'_M ne peut franchir qu'une séquence de type ${t'}^s_{init}\, t'_0$ avec $1 \leq s \leq m_2(p_{init})$:

$$m_0 \xrightarrow{\sigma} m_1 \xrightarrow{t'_{init}} m_2 \xrightarrow{{t'}^s_{init}} m_3 \xrightarrow{t'_0} (e_i, 0, 0, m_1(C) + s + 1)$$

où e_i est l'étiquette apparaissant dans l'instruction e_0 : *goto* e_i $(e_i \neq e_0)$. Ici m_3 est "presque" de type $(e_0, 0, 0, m_1(C) + s + 1)$, sauf que $m_3(p_{init}) = m_2(p_{init}) - s$. Dès que N'_M franchit t'_0, la place p_{init} est mise à zéro et N'_M se retrouve dans le "bon" marquage

$(e_i, 0, 0, m_1(C) + s + 1)$, avec un seul jeton de contrôle dans $T'_M \setminus \{c_1, c_2, C\}$.

Posons $m_1(C) + s + 1 = k$. Puisque (N'_M, m_0) n'est pas borné, nous avons forcément l'exécution suivante dans N'_M :

$m_0 \xrightarrow{*} (e_i, 0, 0, k) \xrightarrow{*} (e_0, 0, 0, k+1) \xrightarrow{*} (e_0, 0, 0, k+2) \xrightarrow{*} (e_0, 0, 0, k+3) \cdots step* \cdots$
$(e_0, 0, 0, k+i) \xrightarrow{*} (e_0, 0, 0, k+i+1) \cdots$

Le lemme 81 de la section précédente se généralise, permettant de montrer que pour tout $i \in \mathbb{N}$, il existe une séquence $\sigma \in {T'_M}^*$ telle que :

$$(e_0, 0, 0, k+i) \xrightarrow{\sigma} (e_0, 0, 0, k+i+1)$$

et telle que tout marquage intermédiaire entre $(e_0, 0, 0, k+i)$ et $(e_0, 0, 0, k+i+1)$ contient $k+i+1$ jetons.

Par l'application du lemme 83, on déduit que pour tout $i \in \mathbb{N}$, il existe une configuration $< e, x_1, x_2 >$ accessible par la machine M telle que $x_1 + x_2 = k + i - 1$.

On en déduit que la machine M n'est pas bornée.

□

Remarque : *le nombre d'arcs Reset utilisés cette fois dépend du nombre d'instructions de la machine M et n'est donc pas a priori borné par une constante.*

Nous déduisons des propositions 90 et 91 qu'étant donnée une machine à compteurs M, on peut construire un réseau de Petri avec arcs de Reset N'_M tel que N'_M est structurellement borné si et seulement si M est bornée. Puisque le caractère NON-borné d'une machine à compteurs n'est pas semi-décidable, on en déduit les théorèmes suivants :

Théorème 92 *Pour les réseaux de Petri avec arcs de Reset,*

- *le problème du caractère NON-structurellement borné n'est pas semi-décidable.*
- *le problème du caractère structurellement borné est indécidable.*

5.4.2 Réduction du problème de la NON-terminaison d'une machine à compteurs

Nous allons réduire le problème de la NON-terminaison d'une machine à deux compteurs au problème de caractère structurellement borné pour les réseaux avec arcs de Reset. Nous construisons un réseau avec arcs de Reset qui n'est PAS structurellement borné si et seulement si une machine termine (c'est-à-dire que la machine atteint l'instruction *Halte*).

Comme précédemment, nous supposons que les compteurs c_1 et c_2 sont initialisés à zéro. De plus, sans perte de généralité, nous supposons que si la machine atteint l'instruction *Halte*, alors les compteurs sont à zéro.

• **Problème de la terminaison d'une machine déterministe à deux compteurs**

Entrée: M une machine déterministe à deux compteurs initialisés à zéro

Sortie: $\begin{cases} \textit{oui} & \text{si } < e_n, 0, 0 > \text{ est accessible par } M \\ \textit{non} & \text{sinon} \end{cases}$

Remarque : *une machine à compteurs peut être bornée, sans toutefois terminer. Par contre, si la machine termine, alors elle est bornée.*

Nous allons construire un réseau N_M qui simulera M avec, comme dans les constructions précédentes, la place de capacité C. À la différence des deux constructions précédentes, la capacité du réseau ne sera jamais augmentée (il n'y a plus de transitions de type t_{c_i} et plus de transitions t_{init}, r_{c_1}, r_{c_2} ou t'_{init}). Voyons ce qu'il advient dans N_M pour une simulation avec une capacité de k.

Si M est bornée par k ET termine, alors N_M pourra franchir une transition t_{test} qui incrémente une place p_{test}, puis fait passer le jeton de contrôle à e_0 amenant au début d'une nouvelle simulation; N_M pourra ainsi boucler et incrémenter la place p_{test} une infinité de fois et p_{test} sera alors non bornée. Par contre, si la machine n'est pas bornée par k OU ne termine pas, alors la place p_{test} ne pourra être incrémentée qu'un nombre fini de fois (fonction de k) et les autres places étant bornées, N_M est borné.

En résumé, M termine si et seulement s'il existe un k tel que N_M n'est PAS borné sur une simulation avec une capacité de k.

Construction du réseau

Considérons une machine M à $n+1$ instructions étiquetées par $e_0, .., e_n$. L'instruction de départ est étiquetée par e_0 et l'instruction *Halte* est étiquetée par e_n. Soient INC l'ensemble des étiquettes des instructions de type *incrémentation* et TAZ l'ensemble des étiquettes des instructions de type *test-à-zéro*.

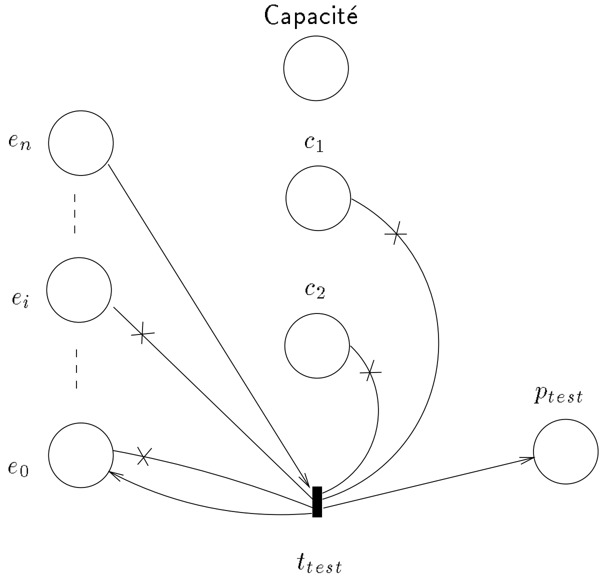
Voici la définition du réseau de Petri avec arcs de Reset N_M associé à M :

- $P_M = \{c_1, c_2, C, p_{test}\} \cup_{i=0}^{n} \{e_i\}$.
- $T_M = \{t_{test},\ t_0\} \cup_{i=1}^{n-1} \{Inc_i \mid e_i \in INC\} \cup_{i=1}^{n-1} \{Taz_i, Dec_i \mid e_i \in TAZ\}$
- Nous décrivons la fonction F_M à l'aide de schémas.
  - Instruction de fin :

    e_n : *Halte*

    Lorsque la place e_n est marquée, la transition t_{test} est franchissable. t_{test} met à zéro c_1 et c_2 ainsi que toutes les places $e_i (0 \leq i < n)$, incrémente p_{test} et fait passer le jeton de contrôle de e_n vers e_0, pour commencer une nouvelle simulation.

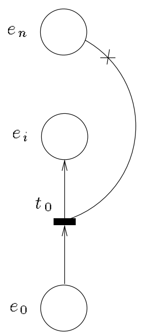
En voici l'illustration :



– Instruction de départ :

$$e_0 : goto \;\; e_i; \; (e_i \neq e_0)$$

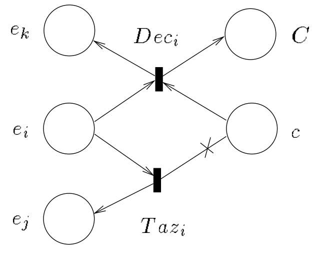
Lorsque e_0 est marquée, la transition t_0 est franchissable. t_0 fait passer le jeton de contrôle de e_0 vers e_i et met la place e_n à zéro. En voici l'illustration :



– Instruction de type test-à-zéro du compteur $c \in \{c_1, c_2\}$:

$$e_i : \textit{Si } c = 0 \textit{ alors goto } e_j \textit{ Sinon } \{c \leftarrow c - 1; \textit{ goto } e_k\};$$

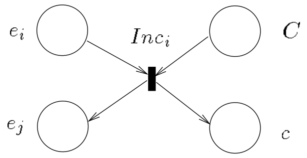
avec $e_i, e_j, e_k \neq e_0$ et $e_i \neq e_n$. En voici l'illustration (il s'agit de la même construction que pour les deux autres réductions) :



– Instruction de type incrémentation du compteur $c \in \{c_1, c_2\}$:

$$e_i : c \leftarrow c + 1;\ goto\ e_j;$$

avec $e_i, e_j \neq e_0$ et $e_i \neq e_n$. En voici l'illustration (il s'agit de la même construction que pour les deux autres réductions) :



Notation : *nous noterons de façon abrégée par* (p, x_1, x_2, k, x_3)*, un marquage* $m \in \mathbb{N}^{|P_M|}$ *tel que* $m(c_1) = x_1$*,* $m(c_2) = x_2$*,* $m(C) = k$*,* $m(p_{test}) = x_3$ *avec le jeton de contrôle étant dans la place* $p \in P_M \setminus \{c_1, c_2, C\}$*.*

Réduction du problème de terminaison d'une machine a compteurs

Nous allons prouver d'abord que si M termine alors N_M n'est pas structurellement borné. Nous commençons par montrer que si la machine est bornée, alors il existe un marquage à partir duquel N_M peut simuler M.

Lemme 93 *Soit une machine* M *bornée par un entier* K *et le réseau* N_M *associé à* M*. Pour toute configuration* $<e, x_1, x_2>$ *accessible par* M*, il existe un marquage* $(e, x_1, x_2, k, 0) \in Acc(N_M, (e_0, 0, 0, K, 0))$ *tel que* $x_1 + x_2 + k = K$*.*

Preuve : nous donnons une preuve par induction sur la longueur d'un préfixe de l'exécution σ_M de M (rappelons que M est déterministe). La configuration de départ de la machine est $<e_0, 0, 0>$ et le marquage initial de N_M est $(e_0, 0, 0, K, 0)$. Le "cas zéro" est donc vérifié. Nous allons aussi vérifier le lemme pour le préfixe de σ_M de longueur 1. Soit $<e_i, 0, 0>$ provenant de l'exécution de l'instruction de départ :

$$<e_0, 0, 0> \stackrel{e_i}{\rightarrow} <e_i, 0, 0>$$

Dans N_M, nous avons :

$$m_{0M} = (e_0, 0, 0, K, 0) \stackrel{t_0}{\rightarrow} (e_i, 0, 0, K, 0)$$

Les cas de base sont donc vérifiés.

Supposons que le lemme soit vérifié pour le préfixe de σ_M de longueur n (s'il existe); montrons que le lemme est vérifié pour le préfixe de longeur $n + 1$ (s'il existe).

Supposons qu'il existe un préfixe de σ_M de longueur $n + 1$ et soit $< e_i, x_1, x_2 >$ la $n^{\text{ème}}$ configuration atteinte par M. Par hypothèse d'induction, nous savons qu'il existe un $(e_i, x_1, x_2, k, 0) \in Acc(N_M)$ avec $x_1 + x_2 + k = K$. Voyons quelle est la $(n + 1)^{\text{ème}}$ instruction exécutée par M.

- Si l'instruction est un test-à-zéro de c_1 (le cas du test-à-zéro de c_2 est similaire) :

$$e_i : \textit{Si } c_1 = 0 \textit{ alors goto } e_j \text{ Sinon } \{ \; c_1 \leftarrow c_1 - 1; \textit{ goto } e_k\};$$

  avec $e_i, e_j, e_k \neq e_0$ et $e_i \neq e_n$, alors deux cas sont possibles.

  1. Si $c_1 = x_1 = 0$, M effectue :

$$< e_i, x_1, x_2 > \stackrel{Taz_{c_1}}{\longrightarrow} < e_j, x_1, x_2 >$$

     et dans N_M nous avons :

$$(e_0, 0, 0, K, 0) \stackrel{*}{\rightarrow} (e_i, x_1, x_2, k, 0) \stackrel{Taz_i}{\rightarrow} (e_j, x_1, x_2, k, 0)$$

     donc $(e_j, x_1, x_2, k, 0) \in Acc(N_M, (e_0, 0, 0, K, 0))$ et $x_1 + x_2 + k = K$.
  2. Si $c_1 \neq 0$, M effectue :

$$< e_i, x_1, x_2 > \stackrel{Dec_{c_1}}{\longrightarrow} < e_k, x_1 - 1, x_2 >$$

     et dans N_M nous avons :

$$(e_0, 0, 0, K, 0) \stackrel{*}{\rightarrow} (e_i, x_1, x_2, k, 0) \stackrel{Dec_i}{\rightarrow} (e_k, x_1 - 1, x_2, k + 1, 0)$$

     donc $(e_k, x_1 - 1, x_2, k + 1, 0) \in Acc(N_M, (e_0, 0, 0, K, 0))$ et $x_1 - 1 + x_2 + k + 1 = K$.

- Si l'instruction est une incrémentation de c_1 (le cas de l'incrémentation de c_2 est similaire) :

$$e_i : c_1 \leftarrow c_1 + 1; \textit{ goto } e_j$$

  avec $e_i, e_j \neq e_0$ et $e_i \neq e_n$, alors M effectue :

$$< e_i, x_1, x_2 > \stackrel{Inc_{c_1}}{\longrightarrow} < e_j, x_1 + 1, x_2 >$$

  Dans $(e_i, x_1, x_2, k, 0)$, forcément $k > 0$, sinon nous aurions $x_1 + x_2 = K$ et M ne pourrait pas accéder à $< e_j, x_1 + 1, x_2 >$ car, par hypothèse, elle est bornée par K. Ainsi, dans N_M nous avons :

$$(e_0, 0, 0, K, 0) \stackrel{*}{\rightarrow} (e_i, x_1, x_2, k, 0) \stackrel{Inc_i}{\rightarrow} (e_j, x_1 + 1, x_2, k - 1, 0)$$

  donc $(e_j, x_1 + 1, x_2, k - 1, 0) \in Acc(N_M, (e_0, 0, 0, K, 0))$ et $x_1 + 1 + x_2 + k - 1 = K$.

- Si l'instruction est *Halte* la machine ne fait rien.

Ainsi pour toute configuration $< e_i, x_1, x_2 >$ accessible par M, il existe un marquage $(e_i, x_1, x_2, k, 0) \in Acc(N_M, (e_0, 0, 0, K, 0))$ tel que $x_1 + x_2 + k = K$. □

Proposition 94 *Si M termine alors N_M n'est pas structurellement borné.*

Preuve : supposons que M termine, alors la configuration $<e_n, 0, 0>$ est accessible par M. De plus, il existe un $K \in \mathbb{N}$ tel que pour toute configuration $<e_i, x_1, x_2>$ atteinte par M, $x_1 + x_2 \leq K$. Par le lemme 93, nous savons que dans N_M, il existe $\sigma \in {T_M}^*$ telle que :

$$(e_0, 0, 0, K, 0) \xrightarrow{\sigma} (e_n, 0, 0, K, 0)$$

De $(e_n, 0, 0, K, 0)$, la transition t_{test} est franchissable, menant à :

$$(e_0, 0, 0, K, 0) \xrightarrow{\sigma} (e_n, 0, 0, K, 0) \xrightarrow{t_{test}} (e_0, 0, 0, K, 1)$$

et puisque les réseaux avec Reset sont bien structurés, et que $(e_0, 0, 0, K, 0) \leq (e_0, 0, 0, K, 1)$, la séquence σt_{test} peut être itérée une infinité de fois à partir de $(e_0, 0, 0, K, 1)$, ajoutant à chaque fois un jeton dans p_{test}. Ainsi, p_{test} n'est pas bornée et donc $Acc(N_M, (e_0, 0, 0, K, 0))$ est infini.

Nous en concluons que le réseau N_M n'est pas structurellement borné. □

Nous prouvons à présent l'autre sens de la réduction. Nous montrons d'abord qu'à partir d'une configuration de type $(e_0, 0, 0, k, x)$, k et x entiers, tant que la somme des jetons dans c_1, c_2 et C est égale à k, N_M simule correctement la machine M. Ce lemme est similaire au lemme 82 de la section 5.3.3.

Notation : *nous utiliserons un tiret "$-$" pour indiquer une composante dont la valeur n'est pas significative pour nous.*

Lemme 95 *Soient M une machine à deux compteurs et N_M son réseau associé, et soit $K \in \mathbb{N}$. Posons $Etiq = \cup_{i=0}^{n}\{e_i\}$. Considérons $\sigma \in {T_M}^*$, $\sigma = t_{i_1} t_{i_2} t_{i_3} \cdots t_{i_n}$, telle que*

$$(e_0, 0, 0, K, -) \xrightarrow{t_{i_1}} m_1 \xrightarrow{t_{i_2}} m_2 \xrightarrow{t_{i_3}} \cdots \xrightarrow{t_{i_n}} m_n$$

avec $m_n = (p, x_1, x_2, k, -)$. Si $\forall i \in [1..n]$, $m_i(c_1) + m_i(c_2) + m_i(C) = K$ et si $p \in Etiq$ alors $<p, x_1, x_2>$ est une configuration accessible par M.

Preuve : nous prouvons ce lemme par induction sur la longueur de σ.

Le cas où $|\sigma| = 0$ est vérifié car le marquage initial de N_M est $(e_0, 0, 0, K, -)$ et la configuration initiale de M est $<e_0, 0, 0>$.

Supposons que le lemme soit vrai pour toute séquence $\sigma \in {T_M}^*$ de longueur $n - 1$. Montrons que le lemme est vrai pour une séquence de longueur n.

Considérons une séquence $\sigma \in {T_M}^*$, $|\sigma| = n$, $\sigma = \sigma' t$ avec $|\sigma'| = n - 1$ et $t \in T_M$, telle que

$$(e_0, 0, 0, K, -) \xrightarrow{\sigma'} (p, x_1, x_2, k, -) \xrightarrow{t} (p', x_1', x_2', k', -)$$

dont tout marquage m intermédiaire soit tel que $m(c_1) + m(c_2) + m(C) = K$. Regardons quel peut être le type de t :

- t est de type Inc_i et incrémente c_1 (le cas où c_2 est incrémenté est similaire), alors $p = e_i$, $p' = e_j$ avec $e_i, e_j \in Etiq$ et nous avons :

$$(e_0, 0, 0, K, -) \xrightarrow{\sigma'} (e_i, x_1, x_2, k, -) \xrightarrow{Inc_i} (e_j, x_1+1, x_2, k-1, -)$$

Par hypothèse d'induction, $<e_i, x_1, x_2>$ est une configuration accessible par M. Par construction de N_M, nous avons dans M l'instruction :

$$e_i \ : c_1 \leftarrow c_1 + 1; \textit{goto } e_j; (\text{avec } e_i, e_j \neq e_0 \text{ et } e_i \neq e_n)$$

et donc :

$$<e_i, x_1, x_2> \xrightarrow{Inc_{c_1}} <e_j, x_1+1, x_2>$$

Nous en concluons que $<e_j, x_1+1, x_2>$ est accessible par M.

- t est de type Taz_i et met c_1 à zéro (le cas où c_2 est mis à zéro est similaire), alors $p = e_i$, $p' = e_j$, avec $e_i, e_j \in Etiq$, et nous avons :

$$(e_0, 0, 0, K, -) \xrightarrow{\sigma'} (e_i, x_1, x_2, k, -) \xrightarrow{Taz_i} (e_j, 0, x_2, k, -)$$

mais puisque par hypothèse $x_1 + x_2 + k = x_2 + k = K$, alors forcément $x_1 = 0$. Par hypothèse d'induction, $<e_i, 0, x_2>$ est une configuration accessible par M. Par construction de N_M, nous avons dans M l'instruction :

$$e_i : \textit{Si } c_1 = 0 \textit{ alors goto } e_j \textit{ Sinon } \{c_1 \leftarrow c_1 - 1; \textit{ goto } e_k\};$$

(avec $e_i, e_j, e_k \neq e_0$ et $e_i \neq e_n$) et comme $c_1 = 0$, l'action effectuée par M est :

$$<e_i, 0, x_2> \xrightarrow{Taz_{c_1}} <e_j, 0, x_2>$$

Nous en concluons que $<e_j, 0, x_2>$ est accessible par M.

- t est de type $Déc_i$ et décrémente c_1 (le cas où c_2 est décrémenté est similaire), alors $p = e_i$, $p' = e_k$ avec $e_i, e_k \in Etiq$ et nous avons :

$$(e_0, 0, 0, K, -) \xrightarrow{\sigma'} (e_i, x_1, x_2, k, -) \xrightarrow{Dec_i} (e_k, x_1 - 1, x_2, k+1, -)$$

avec $x_1 > 0$ (sinon $Déc_i$ ne serait pas franchissable). Par hypothèse d'induction, $<e_i, x_1, x_2>$ est une configuration accessible par M. Par construction de N_M, nous avons dans M l'instruction :

$$e_i : \textit{Si } c_1 = 0 \textit{ alors goto } e_j \textit{ Sinon } \{c_1 \leftarrow c_1 - 1; \textit{ goto } e_k\};$$

(avec $e_i, e_j, e_k \neq e_0$ et $e_i \neq e_n$) et comme $c_1 = x_1 > 0$, nous avons dans M :

$$<e_i, x_1, x_2> \xrightarrow{Dec_{c_1}} <e_k, x_1 - 1, x_2>$$

Nous en concluons que $<e_k, x_1 - 1, x_2>$ est accessible par M.

- $t = t_0$ et correspond à l'instruction de départ de M. Dans ce cas, $p = e_0$, $p' = e_i$ ($e_i \in Etiq$), et nous avons :

$$(e_0, 0, 0, K, -) \xrightarrow{\sigma'} (e_0, x_1, x_2, k, -) \xrightarrow{t_0} (e_i, x_1, x_2, k, -)$$

  Par construction de N_M, la seule transition qui peut marquer e_0 est t_{test}. Cette transition a donc été franchie juste avant t_0 mettant c_1 et c_2 à zéro. Nous avons :

$$(e_0, 0, 0, K, 0) \xrightarrow{\sigma'} (e_0, 0, 0, k, -) \xrightarrow{t_0} (e_i, 0, 0, k, -)$$

  et $<e_0, 0, 0>$ est une configuration accessible par M. Par construction de N_M, nous avons dans M l'instruction :

$$e_0 \ : goto\ e_i; \text{ (avec } e_i \neq e_0)$$

  et donc $<e_i, 0, 0>$ est accessible dans M.

- $t = t_{test}$, et c_1 et c_2 sont mises à zéro. Nous avons $p' = e_0$ et $p = e_n$:

$$(e_0, 0, 0, K, -) \xrightarrow{\sigma'} (e_n, x_1, x_2, k, -) \xrightarrow{t_{test}} (e_0, 0, 0, k, -)$$

  où $<e_0, 0, 0>$ est la configuration initiale de M, et donc est accessible par M.

Dans tous les cas, le lemme est vérifié pour la séquence σ de longueur n et nous en concluons que le lemme est vrai pour toute séquence de $T_M{}^*$. □

Le lemme suivant énonce que le nombre total de jetons dans $P_M \setminus \{p_{test}\}$ ne peut pas augmenter.

Lemme 96 *Soient une machine M et son réseau N_M associé, et soit $P' = P_M \setminus \{p_{test}\}$. Pour tout $m_0 \in \mathbb{N}^{|P_M|}$ et $m \in Acc(N_M, m_0)$,*

$$\Sigma_{p \in P'} m(p) \leq \Sigma_{p \in P'} m_0(p)$$

Preuve : sur le sous-ensemble de places P', toutes les transitions soit "bougent" les jetons d'une place à l'autre, soit éliminent des jetons par les arcs de Reset. □

Nous allons maintenant montrer la forme particulière d'une exécution qui rend un réseau N_M non borné.

Lemme 97 *Soient une machine M et son réseau N_M associé. Si N_M est non borné à partir de $m_0 \in \mathbb{N}^{|P_M|}$, alors il existe $K \in \mathbb{N}$ tel que :*

$$m_0 \xrightarrow{*} (e_0, 0, 0, K, -) \xrightarrow{*} (e_n, 0, 0, K, -)$$

Preuve : pour produire une infinité de jetons, il faut forcément que t_{test} soit franchie une infinité de fois, puisque t_{test} est la seule transition qui puisse créer des jetons.

Dès que t_{test} est franchie, N_M accède à un marquage m qui est "presque" de type $(e_0, 0, 0, k, -)$, sauf que e_n peut encore contenir des jetons. À partir de m, la seule séquence franchissable est $t_{test}^s t_0$, avec $0 \leq s \leq m(e_n)$. Dès que t_0 est franchie, N_M accède au marquage $(e_i, 0, 0, m(C), -)$, où e_i est l'étiquette de l'instruction de départ de M. Au prochain franchissement de t_{test}, N_M accédera à un marquage propre.

Il existe forcément une suite infinie d'entiers $v_{i>0}$ telle que l'exécution qui rend N_M non bornée est de la forme :

$$m_0 \xrightarrow{*} (e_0, 0, 0, v_1, -) \xrightarrow{*} (e_0, 0, 0, v_i, -) \xrightarrow{*} (e_0, 0, 0, v_{i+1}, -) \cdots$$

où chaque $(e_0, 0, 0, v_i, -)$ est un marquage accessible par un franchissement de t_{test}. Par le lemme 96, la suite des $v_{i>0}$ est non croissante et donc forcément se stabilise puisque qu'une place ne contient pas moins de 0 jeton. Soit K la limite des v_i, nous déduisons qu'il existe x_1, x_2 tels que :

$$m_0 \xrightarrow{*} (e_0, 0, 0, K, -) \xrightarrow{\sigma} (e_n, x_1, x_2, K, -) \xrightarrow{t_{test}} (e_n, 0, 0, K, -)$$

avec $\sigma \in (T_M \setminus \{t_{test}\})^*$. Par le lemme 96, nous avons $K \leq (x_1 + x_2 + K)$ et nous en déduisons que $x_1 = x_2 = 0$. Nous vérifions donc bien que :

$$m_0 \xrightarrow{*} (e_0, 0, 0, K, -) \xrightarrow{*} (e_n, 0, 0, K, -)$$

□

Proposition 98 *Si M ne termine pas, alors N_M est structurellement borné.*

Preuve : nous prouvons que si N_M n'est pas structurellement borné alors M termine.

Supposons qu'il existe un $m_0 \in \mathbb{N}^{|P_M|}$ tel que (N_M, m_0) n'est pas borné. Par le lemme 97, il existe $K \in \mathbb{N}$ tel que :

$$m_0 \xrightarrow{*} (e_0, 0, 0, K, -) \xrightarrow{*} (e_n, 0, 0, K, -)$$

Par le lemme 96, tout marquage m entre $(e_0, 0, 0, K, -)$ et $(e_n, 0, 0, K, -)$ est tel que $m(c_1) + m(c_2) + m(K) = K$. Enfin, nous pouvons appliquer le lemme 95 et conclure que $< e_n, 0, 0 >$ est une configuration accessible par M et donc que M termine. □

Nous déduisons des propositions 94 et 98 qu'étant donné une machine à compteurs M, on peut construire un réseau de Petri avec arcs de Reset N_M tel que N_M n'est PAS structurellement borné si et seulement si M termine. Par le théorème 92, et puisque la NON-terminaison d'une machine à compteurs n'est pas semi-décidable, on en déduit les théorèmes suivants :

Théorème 99 *Pour les réseaux de Petri avec arcs de Reset,*

- *le problème du caractère structurellement borné n'est pas semi-décidable.*
- *le problème du caractère structurellement borné est strictement plus difficile que le problème de caractère borné du réseau.*

De plus, la construction que nous venons de réaliser s'adapte au problème du caractère structurellement borné pour les réseaux avec Transfert.

Théorème 100 *Le problème de caractère structurellement borné pour les réseaux de Petri avec Transfert est indécidable.*

Preuve : soit M une machine à compteurs et N_M son réseau associé. Tout jeton éliminé par un arc de Reset est éliminé de l'ensemble $P_M \setminus \{p_{test}\}$ et n'est jamais remplacé par la suite. À partir d'un marquage m_0, on ne pourra donc éliminer qu'au plus $m_0(c_1) + m_0(c_2) + m_0(C)$ jetons.

Considérons le réseau de Petri N'_M avec arcs de Transfert obtenu en remplaçant les arcs de Reset de N_M par des arcs de Transfert vers une "place-puits" supplémentaire. La place-puits est forcément bornée par $m_0(c_1) + m_0(c_2) + m_0(C)$. De plus, le reste du réseau conserve le même comportement que N_M.

On en déduit que N'_M est structurellement borné si et seulement si M ne termine PAS et donc que le problème de caractère borné pour les réseaux de Petri avec Transfert est indécidable.

□

Par contre, au contraire des réseaux avec Reset, puisque le caractère borné d'un réseau avec Transfert est décidable, nous obtenons le théorème suivant :

Théorème 101 *Pour les réseaux de Petri avec arcs de Transfert, le caractère NON-structurellement borné est semi-décidable.*

Preuve : soit N_t un réseau avec arcs de Transfert. Si N_t n'est pas structurellement borné, alors il existe un marquage $m \in \mathbb{N}^{|P_t|}$ tel que $Acc(N_t, m)$ est infini. Puisque l'on peut décider du caractère borné d'un réseau avec arcs de Transfert, pour trouver m, il suffit d'énumérer tous les marquages dans $\mathbb{N}^{|P_t|}$ et de les tester un à un. On en déduit que le problème de caractère NON-structurellement borné est semi-décidable. □

Remarquons enfin que la construction que nous venons de réaliser s'adapte facilement pour montrer l'indécidabilité du problème de l'*accessibilité répétée d'un état de contrôle* pour les Reset et les Transfert. On peut en effet construire un réseau qui :

- devine d'abord de façon non déterministe la capacité nécessaire pour que la machine termine son exécution (si elle termine);
- laisse ensuite le réseau simuler la machine, tel que décrit précédemment (avec boucle lorsque l'état e_n est atteint).

Dans ce réseau, l'état de contrôle e_n est accessible une infinité de fois au cours d'une exécution si et seulement si la machine termine.

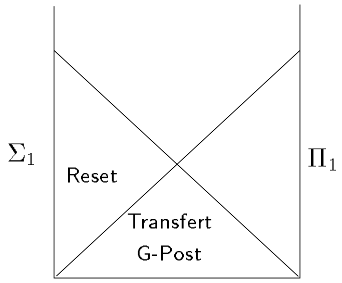
Corollaire 102 *Pour les réseaux de Petri avec arcs de Reset ou de Transfert, l'accessibilité répétée d'un état de contrôle est indécidable.*

5.5 Conclusion

Au cours de notre étude, nous avons pu séparer les trois problèmes de type caractère borné et voici un résumé de leur place dans la hiérarchie arithmétique.

Caractère borné du réseau

Ce problème est indécidable pour les réseaux de Petri avec trois arcs de Reset. Ce résultat a été difficile à obtenir et va à l'encontre de l'intuition. Par contre, pour les Transfert + G-Post, ce problème est décidable et sépare donc la classe des Reset de celle des Transfert. Nous synthétisons la situation pour les G-Post, les Transfert et les Reset. Pour une classe donnée, le problème de caractère borné du réseau se place là où apparaît le nom de cette classe :

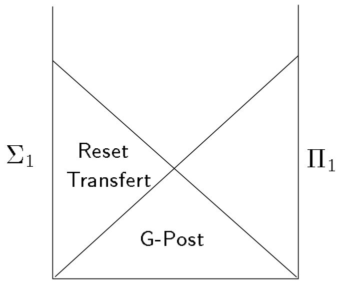


Caractère borné du réseau

Caractère borné d'une place

Une fois que l'on sait que le caractère borné du réseau est indécidable, il est facile d'établir l'indécidabilité du caractère borné d'une place pour les réseaux avec Reset et pour les réseaux avec Transfert. La classe des Transfert sépare donc le problème de caractère borné du réseau (décidable) du problème de caractère borné d'une place (indécidable). Ce résultat est intéressant, car ces deux propriétés sont souvent traitées comme une seule dans la littérature.

Mais où placer dans la hiérarchie arithmétique le caractère borné d'une place pour les Transfert et les Reset? La réponse vient du fait que le problème de couverture est décidable pour les deux classes. Pour décider du caractère borné d'une place p, un semi-algorithme consistera à lister tous les marquages avec la valeur 0 dans toutes les places, sauf p. La place p prendra successivement les valeurs de $\mathbb{N}$. Pour chaque marquage, on peut décider s'il est couvert ou non. Si la place est bornée, alors on arrivera forcément à un marquage non couvert et on pourra répondre `OUI` à la question. Voici la situation de ce problème :



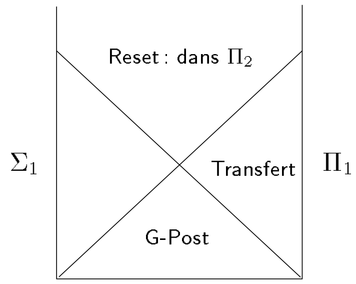
Caractère borné d'une place

Caractère structurellement borné

L'indécidabilité de ce problème pour les réseaux avec Reset et pour les réseaux avec Transfert a été le résultat le plus inattendu. Pour les réseaux avec Reset, ce problème se place plus haut qu'au premier niveau de la hiérarchie arithmétique. Cependant, il est dans Π_2 car un réseau N est structurellement borné si et seulement si :

$$\forall m \in \mathbb{N}^{|P|}, (N, m) \text{ borné}$$

or, pour les réseaux avec arcs de Reset, le problème du caractère borné de (N, m) est dans Σ_1. Par contre, pour les réseaux avec Transfert, le caractère borné du réseau est décidable, donc le problème de caractère NON-structurellement borné est semi-décidable. Enfin, nous avons vu que pour les G-Post, ce problème était décidable. Voici la situation de ce problème :



Caractère structurellement borné

Tableau récapitulatif

Voici le tableau final des résultats de calculabilité :

	Terminaison	Couverture	Réseau borné	Place bornée	Réseau structurellement borné	Accessibilité Blocage	Vivacité t-Vivacité
G-Réseaux *Self-Modifying*	**I**	**I**	**I**	**I**	**I**	**I**	**I**
G-Post + Reset	**D**	**D**	**I**	**I**	**I**	**I**	**I**
Reset	**D**	**D**	**I**	**I**	**I**	**I**	**I**
G-Post + Transfert	**D**	**D**	**D**	**I**	**I**	**I**	**I**
Transfert	**D**	**D**	**D**	**I**	**I**	**I**	**I**
G-Post *Post Self-Modifying*	**D**	**D**	**D**	**D**	**D**	**I**	**I**
Petri	**D**	**D**	**D**	**D**	**D**	**D**	**D**

Chapitre 6

Conclusions et perspectives

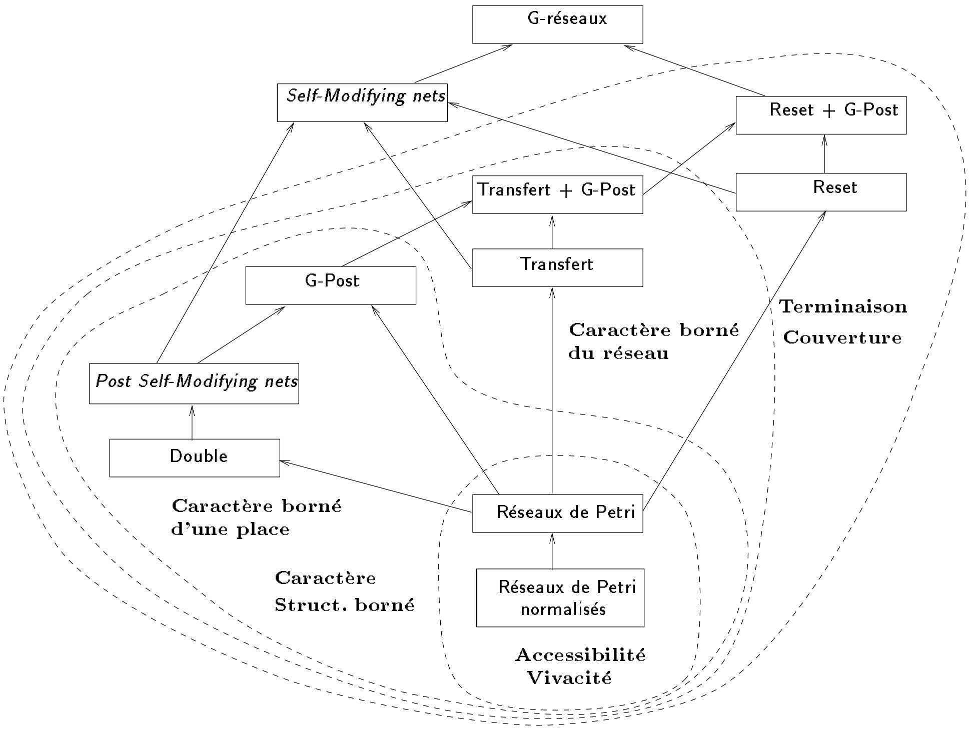
Dans cette thèse, nous avons défini les G-réseaux qui constituent une extension des réseaux de Petri et qui englobent les *Self-Modifying nets* de VALK. Nous avons mis en évidence et étudié une hiérarchie de neuf sous-classes de G-réseaux pertinentes, car elle mettent en lumière une hiérarchie fine de décidabilité.

Nous avons en particulier considéré des classes basées sur les réseaux de Petri comportant les arcs étendus suivants :

- les arcs de *Reset*, qui permettent de mettre une place à zéro,
- les arcs *G-Post*, qui permettent d'ajouter à une place l'évaluation d'un polynôme (à coefficients positifs) fonction du marquage courant du réseau (et qui étendent les arcs *Post Self-Modifying*),
- les arcs *Transfert*, qui permettent de transférer le contenu d'une place vers une autre. Les arcs de Transfert peuvent être implémentés par une combinaison des deux types d'arcs précédents.

Nous avons étudié sur ces classes neuf problèmes fondamentaux, et parfois avec une variante quand la distinction est importante : la *terminaison*, la *couverture*, le *caractère borné du réseau*, le *caractère borné d'une place*, le *caractère structurellement borné*, l'*accessibilité*, le *blocage*, la *vivacité* et la *t-vivacité*.

Nous avons déterminé pour chaque classe si chacun des problèmes était décidable, ce qui nous a permis de regrouper (ou de séparer) les classes. Le diagramme qui suit résume les résultats (nous avons regroupé pour chaque problème les sous-classes pour lesquelles il est décidable).



Voici quelques commentaires sur ce diagramme :

- Nous avons mis en évidence plusieurs classes qui étendent les réseaux de Petri tout en étant partiellement analysables. En particulier, la couverture est décidable pour les G-Post + Reset qui constituent une large classe de réseaux de Petri étendus. La couverture permet aussi de résoudre pour les G-Post le problème d'*accessibilité d'un état de contrôle* ou le problème du *déterminisme* du réseau. Ensuite, le caractère borné du réseau est décidable pour les G-Post + Transfert et le caractère borné d'une place est décidable pour les G-Post.

- Nous avons montré que le caractère borné du réseau est indécidable pour les réseaux de Petri avec arcs de Reset. Ce résultat va contre l'intuition et a été difficile à obtenir. De plus, nous avons montré que *trois* arcs suffisaient à rendre le problème indécidable. Cette borne est optimale par un résultat récent de JANČAR [Jan98], qui montre la décidabilité du caractère borné du réseau pour *deux* arcs de Reset.

- Nous avons séparé les trois problèmes de caractère borné en montrant que :

  – Pour les réseaux de Petri avec arcs de Transfert, le caractère borné du réseau est *décidable*, alors que le caractère borné d'une place est *indécidable*.

  – Pour les réseaux de Petri avec arcs de Reset, les problèmes de caractère borné du réseau et d'une place sont tous les deux dans Σ_1, alors que le caractère structurellement borné n'est ni dans Σ_1 ni dans Π_1.

- Nous avons séparé les classes de réseaux. Remarquons surtout que le caractère borné du réseau sépare les réseaux avec Transfert des réseaux avec Reset, et que le caractère borné d'une place sépare les réseaux G-Post des réseaux avec Transfert.

D'autres résultats découlent du dépliage des G-Post vers les réseaux de Petri (section 3.1) :

– le dépliage est polynomial en temps dans le cas particulier des *Post Self-Modifying nets*, et permet d'obtenir, pour cette classe, une borne supérieure de complexité exponentielle (en espace) pour vérifier la couverture ou le caractère borné d'un réseau.

– Le dépliage permet de vérifier plusieurs propriétés pour les G-Post, dont le caractère borné du réseau ou d'une place en utilisant, sans modification, les outils d'analyse existant pour les réseaux de Petri.

Enfin, au cours de notre étude, nous avons exhibé plusieurs autres constructions, dont une normalisation polynomiale des réseaux avec arcs de Reset ou de Transfert, et des réductions de plusieurs problèmes vers le problème d'accessibilité. Toutes ces constructions sont autant d'exemples de modélisation.

Problèmes ouverts

De nombreuses questions sont apparues au cours de notre étude. Nous avons pu résoudre la plupart, mais certaines restent encore ouvertes. Nous en citons ici quelques unes :

– Quelle est la complexité pour décider de la couverture pour les G-Post+Reset ?

– Quelle est la complexité pour décider du caractère borné pour les G-Post + Transfert ?

– Existe-t-il un dépliage polynomial des G-Post vers les réseaux de Petri ?

– Comment séparer la terminaison de la couverture ? (soulignons que nous n'avons pas réussi à comparer ces problèmes par les réductions polynomiales.)

– Comment séparer l'accessibilité de la vivacité ? Existe-t-il une réduction polynomiale de la vivacité vers l'accessibilité ?

– Suite au résultat de Bouziane [Bou98b] qui montre que l'accessibilité est primitive-récursive, on peut aussi se poser cette question : existe-t-il une réduction polynomiale ou exponentielle de l'accessibilité vers le caractère borné du réseau ou d'une place ?

– Quelle est la borne du nombre d'arcs suffisant à rendre indécidable le caractère structurellement borné pour les réseaux de Petri avec Reset ?

Perspectives

L'étude des extensions des réseaux de Petri peut prendre diverses directions. Nous en citons ici quelques unes :

– Maintenant que l'on a dressé un panorama des problèmes sur les G-réseaux, l'objectif consistera à exhiber des restrictions de classes de G-réseaux qui soient plus analysables que les classes originales, mais tout de même plus grandes que les réseaux de Petri ou incomparables. Prenons par exemple la classe des réseaux à *choix* *libres*, ou encore *sans conflits*, avec arcs de Reset; on pourra se demander si cette classe permet d'obtenir plus de résultats de décidabilité ou de meilleures bornes de complexité que la classe entière des réseaux avec Reset.

- Il sera aussi intéressant d'étendre l'étude des sous-classes de G-réseaux à d'autres problèmes.

- Ciardo [Cia94] a étudié les classes de langages associées aux réseaux avec arcs Reset ou arcs de Transfert. Il en ressort qu'il est difficile de prouver qu'une classe de réseau ne peut pas reconnaître un langage. En fait, on connaît peu de résultats pour les classes étendues (pour les réseaux de Petri, voir [Jan86]) et il serait intéressant de donner de nouvelles techniques de preuves dans ce domaine.

Bibliographie

[AČJY96] P. A. Abdulla, K. Čerāns, B. Jonsson, and T. Yih-Kuen. General decidability theorems for infinite-state systems. In *Proc. 11th IEEE Symp. Logic in Computer Science (LICS'96), New Brunswick, NJ, USA, July 1996*, pages 313–321, 1996.

[AD90] R. Alur and D. Dill. Automata for modeling real-time systems. In *Proc. 17th Int. Coll. Automata, Languages, and Programming (ICALP'90), Warwick, GB, July 1990*, volume 443 of *LNCS*, pages 322–335. Springer-Verlag, 1990.

[AK77] T. Araki and T. Kasami. Some decision problems related to the reachability problem for Petri nets. *Theoretical Computer Science*, 3(1):85–104, 1977.

[AL76] A. Arnol and M. Latteux. Vector addition systems and semi-Dyck language. Research Report 78, Laboratoire de Calcul de l'Université des Sciences et Techniques de LILLE, December 1976.

[AL78] A. Arnol and M. Latteux. Récursivité et cônes rationnels fermés par intersection. *Estratto da Calcolo*, XV(IV), Ottobre-Dicembre 1978.

[Arn92] A. Arnold. *Systèmes de transitions finis et sémantique des processus communicants.* Masson, Paris, 1992.

[BD97] E. Badouel and Ph. Darondeau. Stratified Petri nets. In *Proceedings of the FCT'97*, volume 1279 of *LNCS*. Springer-Verlag, 1997.

[Bil91] J. Billington. *Extensions to coloured Petri nets and their applications to protocols.* PhD thesis, University of Cambridge, UK, May 1991. Available as Tech. Report No.222.

[BJP90] G. Berthelot, C. Johnen, and L. Petrucci. Papetri: Environment for the analysis of Petri nets. In *Proc. 2nd Int. Workshop Computer-Aided Verification (CAV'90), New Brunswick, NJ, USA, June 1990*, volume 531 of *LNCS*. Springer-Verlag, 1990.

[BO98] E. Badouel and J. Olivier. Reconfigurable nets, a class of high level Petri nets supporting dynamic changes. In *Proceedings workshop Workflow Management : Net-based concepts, Models, Techniques and Tools*, 1998.

[Boc78] G. von Bochmann. Finite state description of communication protocols. *Computer Networks and ISDN Systems*, 2:361–372, 1978.

[Bou96] Z. Bouziane. *Algorithmes primitifs récursifs et problèmes* EXPSPACE*-complets pour les réseaux de Petri cycliques*. PhD thesis, LSV, École Normale Supérieure de Cachan, France, November 1996.

[Bou98a] Z. Bouziane. Communication personnelle, 1998.

[Bou98b] Z. Bouziane. A primitive recursive algorithm for the general Petri net. In *Proc. 39th IEEE Symp. Foundations of Computer Science (FOCS'98), Palo Alto, Ca, USA, Nov. 1998*, 1998.

[CEP95] A. Chen, J. Esparza, and J. Palsberg. Complexity results for 1-safe nets. *Theoretical Computer Science*, 147:117–136, 1995.

[Cia94] G. Ciardo. Petri nets with marking-dependent arc cardinality: Properties and analysis. In *Proc. 15th Int. Conf. Applications and Theory of Petri Nets, Zaragoza, Spain, June 1994*, volume 815 of *LNCS*, pages 179–198. Springer-Verlag, 1994.

[CLM76] E. Cardoza, R. Lipton, and A. R. Meyer. Exponential space complete problems for Petri nets and commutative subgroups. In *Proc. 8th ACM Symp. Theory of Computing (STOC'76), Hershey, PA, USA, May 1976*, pages 50–54, 1976.

[Dav77] M. Davis. Unsolvable problems. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 567–594. North-Holland, 1977.

[DC] Design-CPN. `http://www.daimi.aau.dk/designCPN/`. Développé à Århus (Danemark).

[DE95] J. Desel and J. Esparza. *Free Choice Petri Nets*. Cambridge University Press, 1995.

[DF97] C. Dufourd and A. Finkel. Polynomial-time many-one reductions for Petri nets. In *Proc. 17th Conf. Found. of Software Technology and Theor. Comp. Sci. (FST&TCS'97), Kharagpur, India, Dec. 1997*, volume 1346 of *LNCS*, pages 312–326. Springer-Verlag, 1997.

[DF98] C. Dufourd and A. Finkel. A polynomial λ-bisimilar normalization for Petri nets. *Theoretical Computer Science*, 1998. *À paraître comme note.*

[DFS98] C. Dufourd, A. Finkel, and Ph. Schnoebelen. Reset nets between decidability and undecidability. In *Proc. 25th Int. Coll. Automata, Languages, and Programming (ICALP'98), Ålborg, Denmark, July 1998*, volume 1443 of *LNCS*, pages 103–115. Springer-Verlag, 1998.

[EN94] J. Esparza and M. Nielsen. Decidability issues on Petri nets – a survey. *Bulletin of the EATCS*, 52:254–262, 1994.

[ES92] J. Esparza and M. Silva. A polynomial-time algorithm to decide liveness of bounded free choice nets. *Theoretical Computer Science*, 102:185–205, 1992.

[Fin87] A. Finkel. Well structured transition systems. Research Report 365, Lab. de Recherche en Informatique (LRI), Univ. Paris-Sud, Orsay, August 1987.

[Fin90] A. Finkel. Reduction and covering of infinite reachability trees. *Information and Computation*, 89(2):144–179, 1990.

[Fin93] A. Finkel. The minimal coverability graph algorithm. In *Advances in Petri Nets 1993*, volume 674 of *LNCS*, pages 210–243. Springer-Verlag, 1993.

[FMP96] A. Finkel, P. McKenzie, and C. Picaronny. Minimal coverability trees for high level nets. Technical report, École Normale Supérieure de Cachan, 1996. Mars 96 – 3.

[FS97] A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! Submitted to Theor. Comp. Sci., October 1997.

[GL81] H. Genrich and K. Lautenbach. System modelling with high-level Petri nets. *Theoretical Computer Science*, 13:109–136, 1981.

[GRT+95] B. Grahlmann, S. Römer, T. Thielke, B. Graves, M. Damm, R. Riemann, L. Jenner, S. Melzer, and A. Gronewold. Pep: Programming environment based on Petri nets. Tech. Report in Hildesheimer Informatik-Berichte no. 14/95, Univ. Hildesheim, 1995.

[GW89] R. J. van Glabbeek and W. P. Weijland. Branching time and abstraction in bisimulation semantics (extended abstract). Research Report CS-R8911, CWI, April 1989.

[Hac75] M. Hack. Petri net languages. CSG, Memo 124, Project MAC, MIT, 1975.

[Hac76] M. Hack. Decidability questions for Petri nets. PhD Thesis MIT/LCS/TR-161, Massachusetts Institute of Technology, Lab. for Computer Science, June 1976.

[Hen96] T. A. Henzinger. The theory of hybrid automata. In *Proc. 11th IEEE Symp. Logic in Computer Science (LICS'96), New Brunswick, NJ, USA, July 1996*, pages 278–292, 1996.

[Hig52] G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc. (3)*, 2(7):326–336, 1952.

[HU79] J. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[HyT] HyTech. `http://www.eacs.berkeley.edu/~tah/HyTech`. Développé par T. A. Henzinger, P.-H. Ho et H. Wong-Toi.

[Jan86] M. Jantzen. Language theory of Petri nets. In *Petri Nets: Central Models and Their Properties, Bad Honnef, LNCS 254*, pages 397–412. Springer-Verlag, September 1986.

[Jan95] P. Jančar. Undecidability of bisimilarity for Petri nets and some related problems. *Theoretical Computer Science*, 148(2):281–301, 1995.

[Jan98] P. Jančar. Communication personnelle, 1998.

[Jen92] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Vol. 1, Basic Concepts*. EATCS Monographs on Theoretical Computer Science. Springer, 1992.

[Jen95] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Vol. 2, Analysis Methods.* EATCS Monographs on Theoretical Computer Science. Springer, 1995.

[Jen97] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Vol. 3, Practical Use.* EATCS Monographs on Theoretical Computer Science. Springer, 1997.

[KCK+97] M. Kishinevsky, J. Cortadella, A. Kondratyev, L. Lavagno, A. Taubin, and A. Yakovlev. Coupling asynchrony and interrupts: Place chart nets. In *Proc. 18th Int. Conf. Application and Theory of Petri Nets, Toulouse, France, June 1997*, volume 1248 of *LNCS*, pages 328–347. Springer-Verlag, 1997.

[KM69] R. M. Karp and R. E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195, 1969.

[KM82] T. Kasai and R.E. Miller. Homomorphism between models of parallel computation. *Journal of Computer Science*, 25, 1982.

[Kos82] S. R. Kosaraju. Decidability of reachability in vector addition systems. In *Proc. 14th ACM Symp. Theory of Computing (STOC'82), San Francisco, CA, May 1982*, pages 267–281, 1982.

[Kru72] J. B. Kruskal. The theory of well-quasi-ordering: A frequently discovered concept. *J. Combinatorial Theory, Series A*, 13(3):297–305, 1972.

[Lam92] J. L. Lambert. A structure to decide reachability in Petri nets. *Theoretical Computer Science*, 99:79–104, 1992.

[LC94] C. Lakos and S. Christensen. A general approach to arc extensions for coloured Petri nets. In *Proc. 15th Int. Conf. Applications and Theory of Petri Nets, Zaragoza, Spain, June 1994*, volume 815 of *LNCS*, pages 338–357. Springer-Verlag, 1994.

[Lip76] R. J. Lipton. The reachability problem requires exponential space. Technical Report 62, Department of Computer Science, Yale University, January 1976.

[May84] E. W. Mayr. An algorithm for the general Petri net reachability problem. *SIAM Journal on Computing*, 13(3):441–460, 1984.

[Min67] M. Minsky. *Computation : Finite and Infinite Machines.* Prentice Hall Int., 1967.

[MR80] G. Memmi and G. Roucairol. Linear algebra in net theory. In *Proc. Net Theory and Applications, LNCS 84*, pages 213–223. Springer-Verlag, 1980.

[Pel92] E. Pelz. Normalization of place/transition-systems preserves net behavior. *Theoretical Informatics and Applications, RAIRO*, 26(1), 1992.

[Pet62] C. A. Petri. *Kommunikation mit Automaten.* PhD thesis, Univ. Bonn, 1962. Schriften des Instituts für Instrumentelle Mathematik.

[Pet81] J. L. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice Hall Int., 1981.

[Rac78] C. Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6(2):223–231, 1978.

[Rei95] K. Reinhardt. Reachability in Petri nets with inhibitor arcs, November 1995. Manuscrit non publié.
Voir `www-fs.informatik.uni-tuebingen.de/~reinhard`.

[Sch85] S. Schwer. *Décidabilité de l'algébricité des langages associés aux réseaux de Petri.* PhD thesis, Université Paris VII, 1985. Thèse 3ième cycle.

[Sch97] Ph. Schnoebelen. Communication personnelle, 1997.

[Sif79] J. Sifakis. *Le contrôle des systèmes asynchrones: concepts, propriétés, analyse statique.* PhD thesis, U.S.M. Grenoble, 1979. Thèse 3ième cycle.

[Smi96] E. Smith. A survey on high-level Petri-net theory. *EATCS Bull.*, 59:267–293, June 1996.

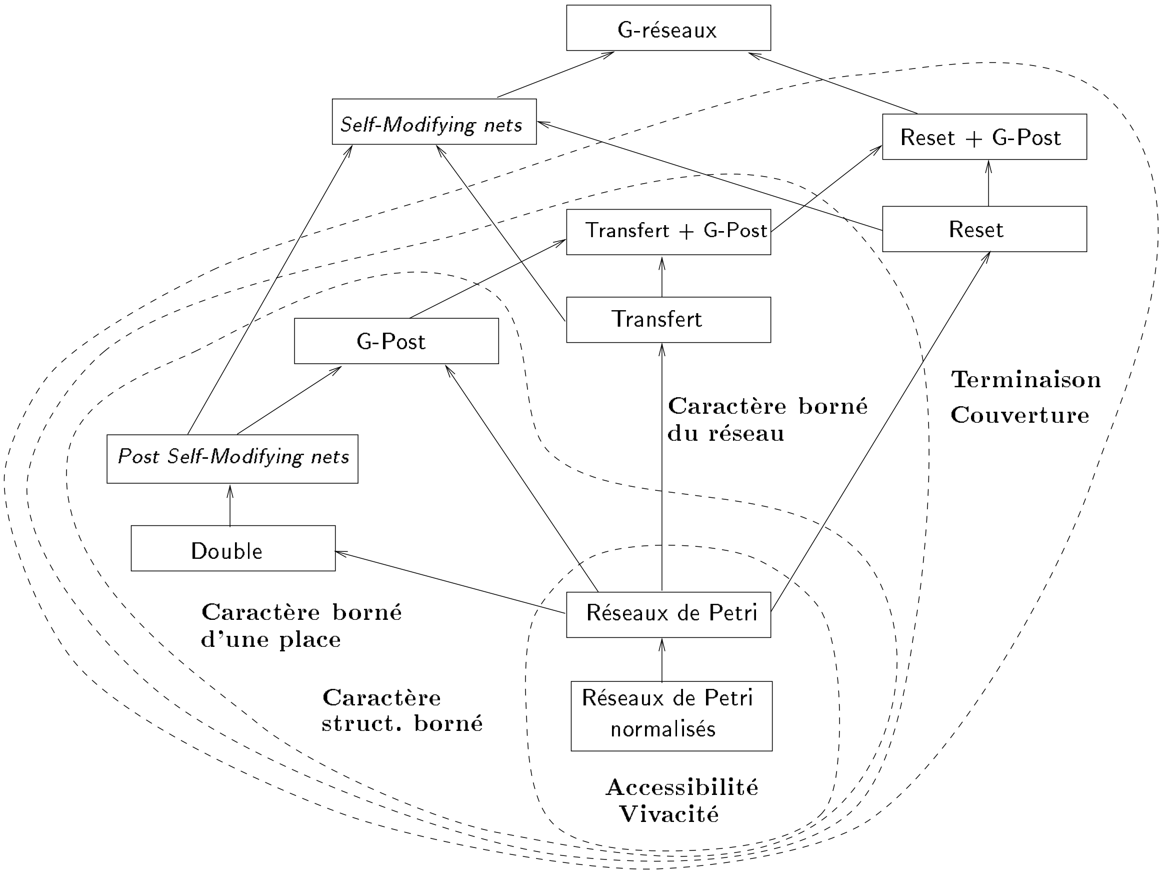
[Spi] Spin. `http://netlib.bell-labs.com/netlib/spin/whatispin.html`. Développé par G. Holzmann.

[Upp] Uppaal. `http://www.docs.uu.se/docs/rtmv/uppaal`. Développé à Ålborg (Danemark) et Uppsala (Suède).

[Val78a] R. Valk. On the computational power of extended Petri nets. In *Proc. 7th Int. Symp. Math. Found. Comp. Sci. (MFCS'78), Zakopane, Poland, Sep. 1978*, volume 64 of *LNCS*, pages 526–535. Springer-Verlag, 1978.

[Val78b] R. Valk. Self-modifying nets, a natural extension of Petri nets. In *Proc. 5th Int. Coll. Automata, Languages, and Programming (ICALP'78), Udine, Italy, Jul. 1978*, volume 62 of *LNCS*, pages 464–476. Springer-Verlag, 1978.

[Val81] R. Valk. Generalizations of Petri nets. In *Proc. of the* 10^{th} *Symposium on Mathematical Fondations of Computer Science*, volume 118 of *LNCS*, pages 140–155. Springer-Verlag, 1981.



Les classes de décidabilité

- Les arcs de *Reset* mettent une place à zéro;
- les arcs de *Transfert* transfèrent le contenu d'une place vers une autre place;
- les arcs *G-Post* ajoutent $P(m)$ jetons à une place, où P est un polynôme à coefficients non négatifs et m est le marquage courant du réseau.