Chapitre 4
Décidabilité et complexité de problèmes de réseaux de Petri

Serge HADDAD[1]

1. Introduction

Lors du chapitre précédent nous avons souligné que, d'une part, l'efficacité des méthodes basées sur le graphe d'accessibilité dépendait fortement de la taille de celui-ci et, d'autre part, que ces méthodes ne s'appliquaient qu'à des réseaux bornés. Aussi dans ce chapitre, nous nous attacherons à évaluer la complexité du graphe d'accessibilité et à proposer des méthodes de vérification applicables lorsque le graphe est infini.

Nous débuterons par une synthèse succincte des principales notions de décidabilité et de complexité qui seront reprises lors des autres sections. Nous rappellerons ensuite les résultats négatifs qui portent sur le graphe d'accessibilité à la fois en terme de décidabilité lorsque le graphe est infini et en terme de complexité dans le cas fini.

Puis nous évoquerons les méthodes usuelles capables de traiter le cas des graphes infinis. La première de ces méthodes, appelée construction du graphe de couverture, est une adaptation de la construction du graphe d'accessibilité. Dans cet algorithme, on substitue une valeur infinie (notée ω) à un marquage d'une place lorsqu'on détecte la possibilité d'atteindre des marquages supérieurs au marquage rencontré avec des valeurs aussi grandes que possible dans la place concernée. Ce graphe fini est une abstraction du graphe d'accessiblité qui permet de décider encore certaines propriétés générales. La deuxième méthode

[1]LAMSADE, Université Paris Dauphine, Place du Maréchal de Lattre de Tassigny, 75775 Paris cedex 16, France, (serge.haddad@lamsade.dauphine.fr)

repose sur le calcul d'une borne de la taille des plus courts chemins permettant de valider telle ou telle propriété. Il suffit alors de rechercher l'existence d'un chemin de longueur bornée pour vérifier si la propriété est satisfaite. Quant à la troisième méthode, partant d'un marquage cible, elle remonte le temps en effectuant des franchissements à rebours à partir d'ensembles de marquages pour essayer d'atteindre le marquage initial. Le mécanisme mis en oeuvre garantit la terminaison de l'algorithme. Nous illustrerons chacune de ces méthodes sur la propriété de couverture (i.e. l'accessibilité d'un marquage supérieur ou égal à un marquage donné).

Aucune des méthodes précédentes n'est capable de résoudre le problème central de savoir si, étant donnés deux marquages et un réseau, le deuxième marquage est accessible à partir du premier. Aussi nous décrirons de manière informelle l'algorithme de test d'accessibilité. L'importance de cet algorithme provient à la fois de la difficulté qu'il a fallu pour l'obtenir et du fait qu'il semble être un résultat limite de décidabilité dans les réseaux de Petri.

Les deux dernières sections seront consacrées au pouvoir d'expression du modèle des réseaux de Petri. Sachant que ce modèle est strictement moins expressif que le modèle des machines de Turing, il est tentant d'introduire de nouveaux mécanismes de contrôle et d'adapter les méthodes de vérification à ces modèles étendus. Nous présenterons trois extensions significatives des réseaux de Petri. Les réseaux à arcs inhibiteurs permettent d'interdire le franchissement d'une transition si le marquage d'une place est supérieur à une valeur. Dans un réseau auto-modifiant, une transition produit ou consomme un nombre de marques dépendant du marquage courant. Un réseau récursif comporte des transitions abstraites dont le franchissement conduit à un état constitué d'un arbre dynamique de réseaux marqués. Pour chacun de ces modèles, nous soulignerons l'impact de l'extension sur la vérification des propriétés.

L'une des façons standard de comparer des modèles en terme de comportement des systèmes est d'étudier les langages engendrés par les séquences de franchissement. Nous achèverons donc ce chapitre par une description des propriétés des langages des réseaux de Petri et une comparaison de cette famille de langages avec la hiérarchie traditionnelle des autres familles.

Le lecteur interessé trouvera dans [ESP 94] un panorama détaillé des résultats de décidabilité et de complexité dans les réseaux de Petri. On pourra aussi se reporter à [ESP 98] qui aborde des aspects complémentaires de ceux couverts par ce chapitre.

Les notations du chapitre précédent s'appliquent aussi à ce chapitre. De plus par souci de lisibilité nous rappelons ci-dessous les quelques résultats du chapitre précédent dont nous ferons un usage fréquent par la suite.

Lemme 1 (Lemme de monotonie) *Soit R un réseau de Petri,*

1. $\forall\, m_1 \leq m'_1\ m_1[\sigma\rangle m_2 \Rightarrow\ m'_1[\sigma\rangle m'_2$ *avec* $m_2 \leq m'_2$
2. *Si de plus pour une place* p, $m_1(p) < m'_1(p)$ *alors* $m_2(p) < m'_2(p)$

Lemme 2 (Lemme de Koening) *Soit A un arbre de degré fini (i.e. tout noeud admet un nombre fini de successeurs) et comportant un nombre de noeuds infini ; alors A admet une branche infinie.*

Lemme 3 (Lemme d'extraction) *Soit $m_0, m_1, \ldots$ une suite infinie de vecteurs de $\mathbb{N}^{\{1,\ldots,k\}}$, alors cette suite admet une sous-suite croissante.*

Proposition 4 (Caractérisation d'un réseau borné) *Soit R un réseau :*

(R, m_0) *est non borné* $\Leftrightarrow$

(R, m_0) *admet une séquence de franchissement* $m_0[\sigma_1\rangle m_1[\sigma_2\rangle m_2$ *avec* $m_1 < m_2$

2. Rappels de décidabilité et de complexité

Les rappels ci-dessous suffisants pour appréhender la suite du chapitre ne peuvent en aucun cas se substituer à un traitement plus complet de ces sujets. Le lecteur désirant d'approfondir ces questions pourra se reporter à des ouvrages spécialisés (e.g. [PAP 94]).

Définition 5 (Notion de problème) *Un problème est une fonction partielle f de A dans B. Les éléments de A et B disposent d'(au moins) une représentation finie susceptible d'être lue ou produite par une machine ou un programme.*

Dans le cas où B est le domaine booléen, on parle de problème de décision.

Informellement un élément a de A est une donnée possible du problème et $f(a)$ est le résultat du problème pour cette donnée. Voici deux exemples simples :

- La construction du graphe d'accessibilité est un problème où A est l'ensemble des réseaux de Petri et B est l'ensemble des graphes **finis**. f est la fonction **partielle** qui associe à un réseau **borné** son graphe d'accessibilité.
- La vivacité d'un réseau est un problème où A est l'ensemble des réseaux de Petri et B est le domaine booléen. f est la fonction cette fois-ci **totale** qui associe à un réseau la valeur de vérité de la vivacité pour le réseau.

En toute rigueur, il faudrait indiquer la représentation choisie pour les données et les résultats car, dans certains cas, ce choix a une influence sur la complexité du problème (e.g. représentation unaire des entiers). Nous ne le ferons pas en supposant que nous disposons d'une représentation raisonnable et relativement standard des objets étudiés.

Notre objectif est bien entendu de résoudre des problèmes à l'aide de programmes. Suivant la thèse de Church, nous ne préciserons pas le langage de programmation utilisé à partir du moment où il possède les constructions minimales : *si alors sinon, tant que, pour,...*

Définition 6 (Problème récursif) *Un problème f de A dans B est récursif s'il existe un programme qui prend en entrée un élément quelconque a de A et se termine en produisant (affichant, imprimant) $f(a)$ sous réserve que f soit définie pour a.*

Dans le cas où B est le domaine booléen, on parle de problème décidable.

Une fois qu'on a déterminé qu'un problème est récursif autrement dit qu'on a exhibé un algorithme pour le résoudre, il s'agit de mesurer l'efficacité de la résolution. Un élément pertubateur de cette mesure est la construction *tant que*. En effet même si on est en mesure d'affirmer que le test finira par être faux, il est parfois impossible de prédire le nombre de tours de boucle exécutés avant la terminaison du programme. Appelons *programme primitif*, un programme qui ne dispose que des trois constructeurs *pour* et *si alors sinon* et *la concaténation* et dont une variable est initialisée avec la taille du problème à résoudre et les autres variables sont initialisées à zéro.

Définition 7 (Problème primitif récursif) *Un problème f est primitif récursif s'il existe un programme primitif qui prend en entrée un élément quelconque a de A et se termine en produisant (affichant, imprimant) f(a) sous réserve que f soit définie pour a.*

De manière très informelle, on peut dire qu'un problème récursif non primitif est résoluble mais que la complexité de sa résolution nous demeurera inconnue. On vise donc à obtenir des programmes de résolution primitifs ou tout au moins susceptibles d'être transformés en programmes primitifs de telle sorte que leur complexité puisse être mesurée. Les deux critères de mesure de complexité les plus couramment adoptés sont la complexité en temps (nombre d'instructions effectuées) et en espace (taille de la place mémoire occupée). Dans ce dernier cas, la donnée est supposée résider sur une mémoire spéciale et n'entre pas en compte dans la mesure.

Définition 8 *Soit $Comp$ une fonction de $\mathbb{N}$ dans $\mathbb{R}$ et f un problème et prog un programme qui résout f. Notons $|a|$ la taille d'un objet a, alors :*

prog est borné en temps (resp. en espace) par $Comp$ si pour toute donnée a appartenant au domaine de f, prog exécute (resp. occupe) au plus $Comp(|a|)$ instructions (resp. cellules mémoires).

Cette mesure de complexité très précise dépend à la fois du choix du langage de programmation et de la taille des cellules mémoires. Aussi nous avons plutôt besoin de caractériser des classes de fonctions de complexité comme par exemple :

- $\mathcal{P}$, la classe des fonctions polynômes
- $\mathcal{EXP}$, la classe des fonctions de la forme 2^{Comp} avec $Comp \in \mathcal{P}$
- $\mathcal{LOG}$, la classe des fonctions de la forme $log(Comp)$ avec $Comp \in \mathcal{P}$

Nous arrivons ainsi à une notion plus robuste de mesure de complexité.

Définition 9 (Classes de complexité) *Soit $\mathcal{C}$ une classe de fonctions de $\mathbb{N}$ dans $\mathbb{R}$ et f un problème :*

- *f est dans $\mathcal{C}time$ ou plus simplement dans $\mathcal{C}$, s'il existe prog un programme, borné en temps par $Comp$, une fonction de $\mathcal{C}$ qui résout f.*
- *f est dans $\mathcal{C}space$, s'il existe prog un programme borné en espace par $Comp$, une fonction de $\mathcal{C}$, qui résout f.*

Pour de nombreux problèmes de décision, une manière de les résoudre consiste à faire des choix indéterministes (e.g. choix d'une transition franchissable) et de revenir en arrière si le choix fait ne permet pas d'obtenir **une réponse positive** à la question. Autrement dit, l'exécution peut être visualisée par un arbre où chaque branche représente une séquence d'exécution possible et où l'algorithme renvoie vrai si l'une des exécutions renvoie vrai. Si l'on disposait d'une machine qui à chaque choix indéterministe pouvait exécuter en parallèle les différentes alternatives, la complexité en temps serait égale au temps d'exécution de la branche la plus longue.

Dans ce cas, pour une classe $\mathcal{C}$ on appelle $\mathcal{NC}time$ ou plus simplement $\mathcal{NC}$, l'ensemble des problèmes décidables par un programme comportant des choix indéterministes et dont une fonction de C borne en temps toutes les séquences d'exécution. Cette notion se décline aussi pour la complexité en espace.

Remarques :

- Il est clair qu'un programme déterministe ne peut occuper au plus qu'un nombre de cellules proportionnel au nombre d'instructions exécutées. Donc $\mathcal{C}time \subset \mathcal{C}space$
- Un programme déterministe est un programme à choix indéterministes particulier. Donc $\mathcal{C}time \subset \mathcal{NC}time$

Nous résumons ci-dessous les relations entre les classes de complexité les plus usuelles et qu'on rencontrera dans cet ouvrage.

$$\mathcal{NLOG}space \subset \mathcal{P} \subset \mathcal{NP} \subset \mathcal{P}space \subset \mathcal{EXP} \subset \mathcal{NEXP} \subset \mathcal{EXP}space$$

Certaines inclusions sont strictes (e.g. $\mathcal{P} \neq \mathcal{EXP}$ et $\mathcal{P}space \neq \mathcal{EXP}space$) tandis que pour d'autres le problème reste ouvert (e.g. $\mathcal{P} \neq \mathcal{NP}$? et $\mathcal{NP} \neq \mathcal{P}space$?).

La complexité que nous avons décrite jusqu'à présent s'intéresse à borner supérieurement la difficulté d'un problème. Il est tout aussi utile de la borner inférieurement car, d'une part cela évite la recherche vaine d'hypothétiques algorithmes, et d'autre part cela permet de définir une notion d'optimalité pour certains algorithmes.

Pour formaliser le concept de borne inférieure, on définit tout d'abord la notion de réduction.

Définition 10 (Réduction de problèmes) *Soit f un problème de A vers B et g un problème de A' vers B. f se réduit à g avec une complexité $\mathcal{C}$ s'il existe un programme prog,*

- *qui prend en entrée un objet a de A et fournit comme résultat un objet a' de A' sous réserve que f soit défini pour a,*
- *tel que $g(a') = f(a)$,*
- *borné par une fonction de $\mathcal{C}$ (en temps ou en espace selon la classe).*

Les deux principaux types de réduction sont en $\mathcal{P}time$ (suffisant à partir des classes $\mathcal{P}time$) et en $\mathcal{LOG}space$ (pour des complexités inférieures à $\mathcal{P}time$).

Définition 11 (Problèmes difficiles et complets) *Soit f un problème,*

- *f est $\mathcal{C}$-difficile si tout problème de $\mathcal{C}$ est réductible à f.*
- *f est $\mathcal{C}$-complet si f est $\mathcal{C}$-difficile et f est dans $\mathcal{C}$.*

Pour un problème $\mathcal{C}$-difficile, il est inutile de chercher un algorithme de complexité inférieure à $\mathcal{C}$ puisqu'à l'aide de la résolution de f on peut résoudre sans ajout de complexité significatif n'importe quel problème de $\mathcal{C}$. Un problème $\mathcal{C}$-complet fait quant à lui partie des problèmes "*les plus complexes de la classe $\mathcal{C}$*". Par exemple, l'accessibilité d'un noeud à partir d'un autre dans un graphe est un problème $\mathcal{NLOG}space$-complet et la satisfaction simultanée d'un ensemble de clauses propositionnelles disjonctives est un problème $\mathcal{NP}$-complet.

3. Résultats théoriques à propos du graphe d'accessibilité

Etant donné deux réseaux de Petri, construits sur le même ensemble de places, il est intéressant de comparer leur ensemble d'accessibilité. Par exemple, on souhaite comparer l'ensemble des états internes d'un système sûr à celui des états internes d'un système à certifier. Malheureusement nous avons ici les premiers résultats négatifs.

Proposition 12 (Comparaison d'ensembles d'accessibilité [HAC 75]) *Soient deux réseaux de Petri de même ensemble de places ; les problèmes de l'égalité ou de l'inclusion des ensembles d'accessibilité sont indécidables.*

En se restreignant aux réseaux bornés, ces questions deviennent naturellement décidables. Mais quelle est leur complexité ? Avant d'y répondre examinons la complexité de la construction du graphe d'accessibilité pour les réseaux bornés.

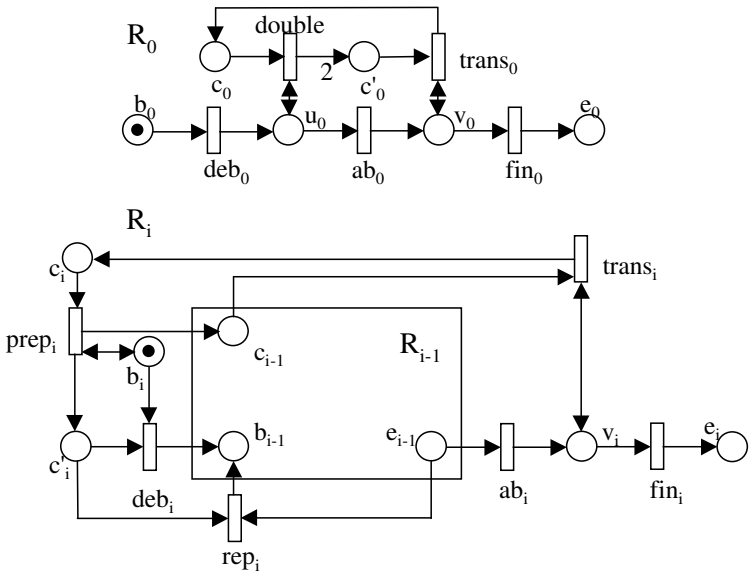

Figure 1: Une famille de réseaux bornés

Proposition 13 (Complexité du graphe d'accessibilité) *Soit un réseau de Petri* ***borné****; le problème de la construction du graphe (de l'ensemble) d'accessibilité n'est pas primitif récursif.*

Preuve
Nous démontrons qu'il existe une famille de réseaux marqués bornés indicée par $\mathbb{N}$ telle que la taille d'un réseau marqué soit proportionnelle à son indice et la taille des graphes d'accessibilité associés ne soit pas une fonction primitive récursive de l'indice.

Nous commençons par calculer un majorant $f_n(x)$ de la valeur absolue d'une variable après l'exécution d'un programme primitif comprenant n constructeurs et dont les variables sont initialisées à des valeurs inférieures ou égales en valeur absolue à x. Sans perte de généralité, nous nous limitons à $x \geq 2$. Plus précisément, le langage de programmation dispose d'une instruction de base, l'affectation d'une somme de variables $x = y(+/-)z$ et trois constructeurs : l'alternative *si alors sinon* avec la restriction que le test n'affecte pas la valeur des variables, *la concaténation* et la boucle *pour* dont le nombre de tours est déterminé par la valeur d'une variable. Montrons que les fonctions f_n définies ci-dessous conviennent :

- $f_0(x) = 2.x$
- $f_{i+1}(x) = f_i^{(x)}(x)$ où $f_i^{(x)}$ désigne la composition x fois répétée de f_i

Au moyen d'une récurrence immédiate on vérifie que les f_i sont strictement croissantes et que $f_{i+1}(x) > f_i(x)$.

Si un programme ne comporte pas de constructeurs, il a exactement une instruction de base et la valeur de la variable affectée ne peut excéder $2.x$. Nous supposons le majorant vérifié pour i constructeurs et nous étudions un programme doté de $i+1$ constructeurs. Si le constructeur le plus externe est l'alternative alors l'exécution est celle d'une des branches. Donc $f_i(x)$ majore le contenu des variables. Si le constructeur le plus externe est la concaténation alors l'exécution est celle d'un programme doté de i constructeurs prenant en entrée des variables calculées par un programme doté de i constructeurs dont les variables sont initialement inférieures ou égales à x en valeur absolue. Donc $f_i(f_i(x))$ majore le contenu des variables. Si le constructeur le plus externe est la boucle *pour* alors il ne peut y avoir plus de x tours de boucle et ce programme est équivalent à au plus $x-1$ concaténations de programmes dotés de i constructeurs. Donc $f_i^{(x)}(x)$ majore le contenu des variables.

En appliquant l'argument de diagonalisation usuel et en posant $f(x) = f_x(x)$, il vient que f n'est pas une fonction primitive récursive puisque ne pouvant être majorée par l'une des fonctions f_n.

Soit maintenant la famille de réseaux R_i décrits à la figure 1. Posons $m_i = i.\overrightarrow{c_i} + \overrightarrow{b_i}$. On prouve que (R, m_i) est borné et que c_i peut atteindre n'importe quelle valeur comprise entre 0 et $f(i)$. Puisque la taille d'un réseau marqué est proportionnelle à son indice, ceci démontre la proposition.

Pour parvenir à ce résultat et sans entrer dans les détails, on établit (pour $i > 1$) par récurrence les points suivants :

- R_i a un P-semi-flot $z_i = b_i + v_i + e_i + b_{i-1} + v_{i-1} + e_{i-1} + \cdots + b_0 + v_0 + u_0 + e_0$
- Pour un marquage initial m tel que $z_i^t.m = m(b_i) = 1$ les séquences maximales de R_i sont de la forme,
  $\sigma = prep_i^p.deb_i.\sigma_1.rep_i.\sigma_2.\ldots.rep_i.\sigma_q.ab_i.trans_i^r.fin_i$ où :
  - $p \leq m(c_i) + m(c'_i)$,
  - $q \leq p$,
  - σ_1 est une séquence de R_{i-1} débutant par un marquage m_1 vérifiant une condition similaire à m et telle que :
    $m_1(c_{i-1}) + m_1(c'_{i-1}) + \cdots + m_1(c_0) + m_1(c'_0) =$
    $m(c_{i-1}) + m(c'_{i-1}) + \cdots + m(c_0) + m(c'_0) + p$
  - Pour $1 < q' \leq q$ σ'_q est une séquence de R_{i-1} débutant par un marquage $m_{q'}$ vérifiant une condition similaire à m et telle que :
    $m_{q'}(c_{i-1}) + m_{q'}(c'_{i-1}) + \cdots + m_{q'}(c_0) + m_{q'}(c'_0) \leq$
    $f_{i-1}^{(q'-1)}(m(c_{i-1}) + m(c'_{i-1}) + \cdots + m(c_0) + m(c'_0) + p)$
  - $r \leq f_{i-1}^{(q)}(m(c_{i-1} + m(c'_{i-1}) + \cdots + m(c_0) + m(c'_0) + p)$
  - pour tout m' atteint à partir de m au cours de σ,

$m'(c_i) + m'(c'_i) + \cdots + m_q(c_0) + m_q(c'_0) \leq$
$f_i(m(c_i) + m(c'_i) + \cdots + m(c_0) + m(c'_0))$

– On exhibe une séquence σ particulière pour atteindre chaque valeur possible de $m'(c_i)$

$\Diamond$

Cette proposition signifie que toute méthode basée sur la construction du graphe d'accessibilité possède une complexité imprédictible. Ceci justifie l'intérêt des méthodes structurelles présentées au chapitre précédent. Pour en revenir au problème initial, aucune autre méthode que la construction du graphe d'accessibilité ne peut fournir des résultats significativement meilleurs.

Proposition 14 *[MAY 81] Soient deux réseaux de Petri* ***bornés*** *de même ensemble de places, les problèmes de l'égalité ou de l'inclusion des ensembles d'accessibilité ne sont pas primitifs récursifs.*

4. Analyse des réseaux de Petri non bornés

Les méthodes de vérification des réseaux de Petri non bornés doivent soit adapter soit se passer de la construction du graphe d'accessibilité. Puisque nous illustrerons trois de ces méthodes sur le problème de la couverture, rappelons sa définition.

Définition 15 (Couverture d'un marquage) *Soit (R, m_0) un réseau de Petri et m_c un marquage. Le réseau couvre m_c s'il existe $m \in A(R, m_0)$ tel que $m \geq m_c$.*

Ce problème présente au moins deux intérêts. D'abord, il a de nombreuses applications. Par exemple, si on veut tester que deux places sont en exclusion mutuelle, on cherchera à couvrir le marquage composé d'un jeton dans chaque place. D'autre part, parmi les algorithmes de décision des propriétés générales, ceux relatifs au problème de couverture sont plus simples conceptuellement et comme nous le verrons à la fin du chapitre s'appliquent plus facilement à certaines extensions des réseaux de Petri.

4.1. *Construction du graphe de couverture*

Imaginions que nous cherchions à produire le graphe d'accessibilité s'il est fini et sinon détecter que le réseau est non borné. Nous construisons un arbre dont chaque noeud est étiqueté par un marquage de la manière suivante :

1. la racine est étiquetée par le marquage initial,
2. chaque noeud *traité* a pour fils un ensemble de noeuds dont les étiquettes sont les marquages accessibles par le franchissement d'une transition à partir du marquage du noeud,
3. on ne planifie le traitement d'un noeud que si son marquage est différent de tous les marquages déjà présents au moment de sa création,
4. la construction est abandonnée si sur la branche qui conduit à un nouveau noeud se trouve un marquage strictement inférieur au marquage associé au nouveau noeud,
5. si la construction se termine sans abandon, on identifie les noeuds associés aux mêmes marquages ce qui aboutit au graphe d'accessibilité.

Les lemmes 2 et 3 assurent qu'un réseau non borné sera détecté par le point 4 et la proposition 4 garantit qu'on n'abandonne que la construction d'un réseau non borné.

Nous souhaitons modifier la procédure précédente pour tester la couverture d'un marquage par un réseau. Ceci nous conduira à la notion d'arbre de couverture $AC(R, m_0)$, proposée par Karp et Miller [KAR 69]. Pour ce faire, on introduit un nouvel ensemble en dotant $\mathbb{N}$ d'un plus grand élément noté ω : $\mathbb{N}_\omega = \mathbb{N} \cup \{\omega\}$ pour lequel on étend l'ordre $<$ et les opérations $+$ et $-$ de la façon suivante :
$\forall n \in \mathbb{N} : n < \omega,\ n + \omega = \omega + n = \omega,\ \omega - n = \omega$ ($n - \omega$ n'est pas défini)
On étend de la même manière $+, -, <$ à $(\mathbb{N}_\omega)^P$ appelé ensemble des ω-marquages. Pour un ω-marquage on appelle une place de marquage ω, une ω-composante et dans le cas contraire une composante finie.

Le résultat qu'on vise est l'équivalence suivante.

Proposition 16

(R, m_0) *couvre* $m_c \Leftrightarrow \exists$ *un noeud de* $AC(R, m_0)$ *étiqueté par* m_ω *t.q.* $m_c \leq m_\omega$

Si on reprend le cas d'une séquence répétitive croissante $m_0[\sigma_1\rangle m_1[\sigma_2\rangle m_2$ avec $m_1 < m_2$, il est clair qu'en répétant la séquence toutes les places sur lesquelles les marquages m_1 et m_2 sont différents croitront indéfiniment. Soit le ω-marquage $m_{2,\omega}$, obtenu en remplaçant dans m_2 le marquage de ces places par ω ; alors tous les marquages inférieurs à $m_{2,\omega}$ sont couverts par le réseau. Telle est l'idée sous-jacente à l'algorithme.

Algorithme de Karp et Miller
Chaque sommet de l'arbre de couverture est étiqueté par un ω-marquage et chaque arc est étiqueté par une transition. $AC(R, m_0)$ est initialisé avec une racine r étiquetée par m_0. On maintient dans $\mathcal{ATRAITER}$ (initialisé avec r) un sous-ensemble des feuilles de l'arbre courant qu'il reste à examiner.

Tant que $\mathcal{ATRAITER}$ est non vide **faire**
Extraire q étiqueté par m de $\mathcal{ATRAITER}$;
Pour chaque transition t sensibilisée en m **faire**
Créer q_t fils de q ;
Etiqueter l'arc reliant q à q_t avec t ;
$m_t := m + C(t)$;
$\Omega(m_t) := m_t$;
Pour chaque place $p \in P$ **faire**
Si q admet un ancêtre (y compris lui-même) a étiqueté par m_a,
avec $m_a < m_t$ et $m_a(p) < m_t(p)$ **Alors**
$\Omega(m_t)(p) = \omega$
Fin si
Fin pour
Etiqueter q_t avec $\Omega(m_t)$;
Si q n'admet pas un ancêtre (y compris lui-même)
ayant la même étiquette que $q(t)$ **Alors**
Insérer q dans $\mathcal{ATRAITER}$
Fin si
Fin pour
Fin tant que

Montrons d'abord que cette construction se termine toujours. Raisonnons par l'absurde. Si ce n'est pas le cas alors l'arbre de couverture est infini. D'après le lemme 2, cet arbre contient une branche infinie. Le lemme 3 reste valable pour $(\mathbb{N}_\omega)^P$. Aussi on peut extraire de cette branche une sous-suite infinie croissante et même strictement croissante car sinon la branche serait finie (cf la condition d'insertion dans $\mathcal{ATRAITER}$). D'après l'arithmétique de $\mathbb{N}_\omega$, les ω-composantes ne peuvent disparaitre sur une branche. Supposons que deux noeuds consécutifs de la sous-suite aient le même ensemble de ω-composantes. Cela signifie que pour le deuxième noeud, aucune ω-composante n'a été créée. Mais dans ce cas, $\Omega(m(t)) = m(t)$ ce qui est impossible puisqu'alors $m(t)$ est strictement supérieur au marquage du premier noeud et l'ajout d'une ω-composante aurait du survenir. Autrement dit, le nombre de ω-composantes est incrémenté à chaque nouveau marquage de la sous-suite. Ce qui nous conduit à une contradiction puisque ce nombre est borné par $|P|$.

Nous illustrons la construction de l'arbre de couverture sur le réseau de la figure 2. Ce réseau modélise (de façon très irréaliste) une journée d'un planteur de bananes. Initialement sur son champ (supposé de capacité infinie), le planteur cueille des bananes, puis il rentre avec ses bananes et se met à table pour en manger certaines. Une fois son repas terminé, il se lève et jette quelques unes des peaux de banane dans son jardin avant d'aller dormir. Nous avons indiqué entre parenthèses un étiquetage possible des transitions. Le langage de ce réseau étiqueté sera discuté à la fin du chapitre. Seules les deux premières

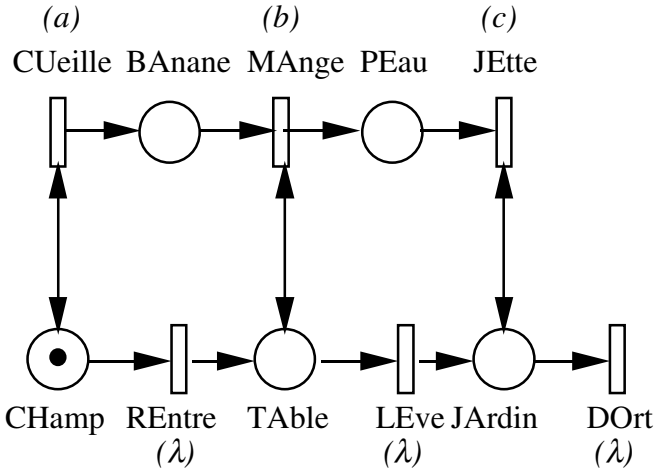

Figure 2: Une journée d'un planteur de bananes

lettres des transitions et places seront utilisées dans la suite.

L'arbre de couverture associée à cette modélisation est représenté sur la figure 3. Les noeuds munis d'une astérisque sont ceux dont l'ω-marquage associé est déjà présent sur la branche qui conduit au noeud. Sachant qu'il y a des feuilles sans astérisques, on conclut ici que le réseau n'est pas pseudo-vivant (attention, il ne s'agit que d'une condition suffisante de non pseudo-vivacité). Le mécanisme de création d'une ω-composante est mis en oeuvre dès la racine. Quand on franchit la transition CU, on atteint un marquage supérieur sur la place BA et identique sur les autres places ce qui conduit à la création d'une ω-composante sur BA. L'ω-marquage $\overrightarrow{TA} + \omega.\overrightarrow{BA} + \omega.\overrightarrow{PE}$ illustre un point significatif de la construction. On peut atteindre un marquage aussi grand que possible sur les places BA et PE **mais on ne peut les faire croître simultanément**. Il faut d'abord incrémenter le marquage de BA, puis le décrémenter lors de l'incrémentation de PE. Ceci explique les précautions à prendre lors de la preuve de la décidabilité de la couverture qui suit.

Preuve
(de la proposition 16)

Soit m_c un marquage couvert par le réseau. Il existe donc une séquence de franchissement $m_0[\sigma\rangle m$ avec $m \geq m_c$. Montrons par récurence que chaque marquage de la séquence est couvert par un ω-marquage de $AC(R, m_0)$. C'est le cas pour m_0. Supposons que m_i soit couvert par $m_{\omega,i}$ et que $m_i[t\rangle m_{i+1}$. Soit un noeud correspondant à une première occurence de $m_{\omega,i}$. A partir de ce noeud, on construira un fils correspondant au franchissement de t étiqueté par

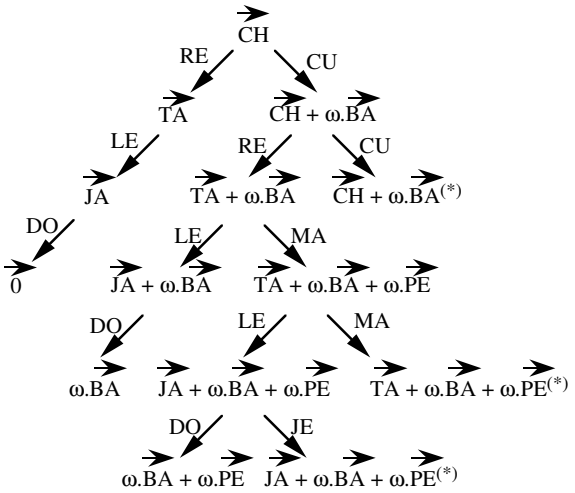

Figure 3: L'arbre de couverture du réseau *planteur*

$m_{\omega,i+1} \geq m_{\omega,i} + C(t) \geq m_i + C(t) = m_{i+1}$.

Soit maintenant un noeud q de $AC(R, m_0)$ étiqueté par m_ω, un ω-marquage. Nous allons montrer que tous les marquages inférieurs ou égaux à m_ω sont couverts par le réseau. Plus précisément, nous montrerons que pour tout $n \in \mathbb{N}$, il existe un marquage accessible m_n égal à m_ω sur ses composantes finies et supérieur ou égal à n sur les autres composantes.

Nous raisonnons par récurence sur le nombre de ω-composantes de m_ω. Si m_ω est un marquage ordinaire, alors la séquence de transitions qui étiquette la branche de r à q est une séquence de franchissement et m_ω est accessible.

Supposons que nous ayons vérifié la propriété pour tous les noeuds comportant au plus d ω-composantes, et que q soit le premier noeud sur sa branche dont le marquage comporte $d+1$ ω-composantes.

- notons $oldq$ le noeud précédent q, de ω-marquage associé $oldm$
- pour $n \in \mathbb{N}$, notons $\sigma_{0,n}$ la séquence de franchissement qui conduit de m_0 à un marquage m_n égal à $oldm$ sur ces composantes finies et supérieur ou égal à n sur les autres (séquence qui existe par hypothèse de récurrence)
- notons t la transition qui étiquette l'arc de $oldq$ à q
- notons $p_1, \ldots, p_k$ les places qui sont les nouvelles ω-composantes de m_ω
- notons $r_1, \ldots, r_k$ les noeuds ancêtres de q qui sont la cause de ces ω-composantes de ω-marquages associés $m_1, \ldots, m_k$
- notons σ_i la séquence de transitions qui étiquette la branche de r_i à q
- notons $\Delta \in \mathbb{N}$ un entier qui soit tel que le marquage composé de Δ jetons

dans chaque place permette de franchir les séquences σ_i et la transition t Prouvons que la séquence $\sigma = \sigma_{0,(k.n+1).\Delta+n}.t.(\sigma_1)^n.\ldots.(\sigma_k)^n$ permet d'établir la propriété recherchée. Trois types de places sont à considérer :

1. **en se restreignant aux composantes finies de** m_ω, d'une part, la séquence $\sigma_{0,(k.n+1).\Delta+n}.t$ conduit au marquage m_ω égal sur ces places aux marquages m_i. D'autre part les séquences σ_i sont répétitives stationnaires sur ces places et franchissables à partir de m_i.
2. **en se restreignant aux ω-composantes de** $oldm$, la séquence $\sigma_{0,(k.n+1).\Delta+n}$ conduit à un marquage où toutes ces places sont supérieures ou égales à $(k.n+1).\Delta+n$. Par définition de Δ, on peut alors franchir la séquence $t.(\sigma_1)^n.\ldots.(\sigma_k)^n$ atteignant un marquage de toutes ces places supérieur ou égal à n
3. **en se restreignant aux nouvelles ω-composantes de** m_ω, la séquence $\sigma_{0,k.n.\Delta+n+1}.t$ conduit à un marquage supérieur ou égal à chacun des m_i. Pour ces places, chacune des séquences σ_i est répétitive croissante (augmentant le marquage de p_i). Donc on peut franchir $(\sigma_1)^n.\ldots.(\sigma_k)^n$ pour conduire à un marquage de ces places supérieur ou égal à n.

Si q n'est pas le premier de sa branche dont le marquage a $d+1$ ω-composantes, soit q' de ω-marquage associé m'_ω le premier sur la branche à vérifier cette propriété et σ la séquence qui étiquette la branche de q' à q. Nous définissons Δ comme précédemment pour permettre le franchissement de σ et nous définissons $\sigma_{0,n}$ comme précédemment pour m'_ω. Nous laissons le soin au lecteur de vérifier que la séquence $\sigma_{0,n+\Delta}.\sigma$ convient. ♢

Cet arbre est l'un des composants de la preuve d'accessibilité (voir plus bas). Le principal inconvénient de cette construction est que la taille de l'arbre de couverture n'est pas primitive récursive. En effet, dans le cas de réseaux bornés, il contient au moins autant de noeuds que de marquages accessibles.

Le graphe de couverture

De nombreuses autres propriétés se décident par examen de l'arbre de couverture ou du *graphe de couverture* obtenu en identifiant les noeuds étiquetés par les mêmes marquages [VAL 85]. L'opération consistant à "quotienter" les graphes est une construction très utile que nous allons expliciter dans le cadre des systèmes de transitions. Nous appliquerons ensuite cette construction pour obtenir le graphe de couverture à partir de l'arbre de couverture.

Définition 17 (Systèmes de transitions) *Un système de transitions $\mathcal{G}$ est défini par un quadruplet de la forme $< S, s_0, \rightarrow, L >$ où :*
- S désigne l'ensemble d'états de $\mathcal{G}$,
- s_0, l'état initial de S,
- L, l'ensemble des labels de transitions et
- $\rightarrow$ la relation de transition ($\rightarrow \subset S \times L \times S$).

Ainsi le graphe d'accessibilité et l'arbre de couverture d'un réseau de Petri, peuvent être vus comme des systèmes de transitions dont les labels sont les transitions du réseau de Petri.

Définition 18 (Quotient d'un système de transitions) *Soit* $\mathcal{G} =< S, s_0, \rightarrow, L >$ *un système de transitions et* $\equiv$ *une relation d'équivalence sur* $S \times S$*, on note* $\mathcal{G}_{\equiv} =< S_{\equiv}, s_{0\ \equiv}, \rightarrow_{\equiv}, L >$*, le quotient de* $\mathcal{G}$ *par* $\equiv$*, défini ainsi :*

- $S_{\equiv}$ *représente l'ensemble quotient de* S *par* $\equiv$*,*
- $s_{0\ \equiv}$ *la classe d'équivalence de* s_0 *pour* $\equiv$*,*
- $\rightarrow_{\equiv}$ *le plus petit ensemble de* $S_{\equiv} \times L \times S_{\equiv}$ *vérifiant :*
  $[(s, l, s') \in \rightarrow] \Rightarrow [(s_{\equiv}, l, s'_{\equiv}) \in \rightarrow_{\equiv}]$

La construction ainsi décrite traite tous les labels de transition de façon identique. Nous verrons dans le deuxième tome de cet ouvrage une construction plus générale pour laquelle la définition du quotient inclut une partition de l'ensemble des labels de transitions en transitions observables et inobservables.

Soient q et q' deux sommets de l'arbre de couverture $AC(R, m_0)$ d'ω-marquages associés m_ω et m'_ω.On définit la relation d'équivalence entre sommets par :

$$q \equiv q' \text{ ssi } m_\omega = m'_\omega$$

Définition 19 *(Graphe de couverture) Le graphe de couverture, noté* $GC(R, m_0)$*, est obtenu en quotientant*

l'arbre de couverture $AC(R, m_0)$ *par la relation d'équivalence* $\equiv$*.*

Le résultat suivant est immédiat.

Proposition 20 *Si* (R, m_0) *est borné alors son graphe de couverture et son graphe d'accessibilité coïncident : i.e.,* $GC(R, m_o) = G(R, m_o)$

Pour le réseau du planteur de bananes (figure 2), on obtient le graphe de couverture représenté figure 4.

Dans le cas d'un réseau non borné, le graphe de couverture représente un sur-ensemble du comportement réel associé au réseau. En ne considérant pas de marquages terminaux notons $LC(R, m_0)$ le langage associé au graphe de couverture, alors on a : $L(R, m_0) \subset LC(R, m_0)$.

Cette inclusion est le plus souvent stricte. Ainsi dans l'exemple, la séquence $CU.RE.MA.MA$ du graphe de couverture n'est pas une séquence de franchissement : le planteur ne peut manger plus de bananes qu'il n'en a cueilli.

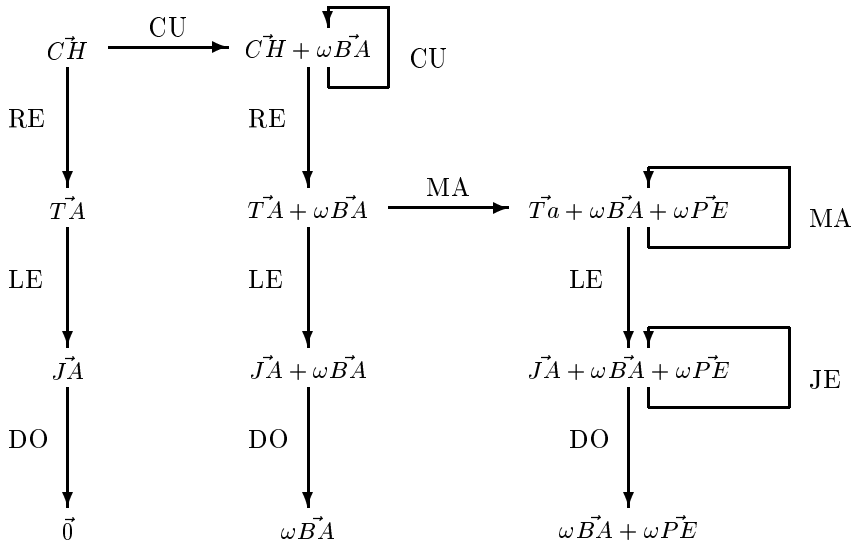

Figure 4: Graphe de couverture du réseau planteur

4.2. *Plus courtes séquences*

Le principe de cette méthode repose sur le résultat suivant : si un marquage peut être couvert par le réseau, alors il peut l'être par une séquence dont la longueur est bornée par une fonction (calculable) de la taille du réseau et du marquage à couvrir [RAC 78]. Notons que la borne **ne dépend pas du marquage initial**.

L'algorithme consiste alors à calculer la borne et faire une recherche non déterministe parmi les chemins de longueur inférieure à cette borne.

Nous allons donc nous concentrer sur l'obtention de la borne. Notons k le nombre de places du réseau, $P = \{p_1, \ldots, p_k\}$ et n la taille du problème de couverture (i.e. de sa représentation). Remarquons que $k \leq n$ et que toute valuation d'arc et le marquage m_c de toute place sont inférieurs ou égaux à 2^n (représentation binaire d'un entier).

Nous allons raisonner sur des pseudo-séquences de franchissement, c'est à dire telles que la condition de franchissabilité n'est que partiellement testée.

Définition 21 *Soit $s = m_1.t_1.m_2.t_2 \ldots t_{y-1}.m_y$ une suite alternée d'éléments de $\mathbb{Z}^k$ appelés pseudo-marquages et de transitions, soit i un indice de place compris entre 0 et k :*

– *s est une i-séquence de franchissement si*

$$\forall x \leq y-1,\ m_{x+1} = m_x + C(t_x)\ et\ \forall j \leq i,\ m_x(p_j) \geq Pré(p_j, t_x)\ et\ m_1(p_j) \geq 0$$

– *s est une i-séquence de franchissement r-bornée si* ***de plus*** *:*

$$\forall x \leq y,\ \forall j \leq i,\ m_x(p_j) < r$$

– *s est une i-séquence de franchissement couvrante si* ***de plus*** *:*

$$\forall j \leq i,\ m_y(p_j) \geq m_c(p_j)$$

Autrement dit, pour une i-séquence on ne fait le test de franchissabilité que pour les i premières places mais on consomme et on produit les marques de la même façon ce qui conduit à des valeurs négatives. De manière similaire, une i-séquence est r-bornée si tous les pseudo-marquages de la séquence sont bornés par r sur les i premières places. Enfin une i-séquence couvrante ne couvre que les i premières places. On dira que (R, m) i-couvre m_c s'il existe une i-séquence couvrante de marquage initial m.

Sans nous préoccuper de m_0, appelons $lg(i, m)$ la longueur de la plus courte i-séquence couvrante partant d'un pseudo-marquage m **comptée en nombre de pseudo-marquages**. Cette quantité n'est définie que pour les m tels que (R, m) i-couvre m_c. Appelons $f(i)$ le maximum de $lg(i, m)$ pour tous les m couvrants. A priori, $f(i)$ pourrait être infini. Nous remarquons que $f(k)$ est la borne que nous recherchons puisque nous avons alors affaire à de véritables séquences de franchissement couvrantes.

Puisque tout (R, m) 0-couvre m_c par la pseudo-séquence réduite à m, $f(0) = 1$. Notre objectif est de majorer $f(i+1)$ en fonction de $f(i)$.

Proposition 22

$$\forall\, 0 \leq i < k,\ f(i+1) \leq (2^n.f(i))^{i+1} + f(i)$$

Preuve
Soit m quelconque tel que (R, m) $(i+1)$-couvre m_c et soit s une plus courte $(i+1)$-séquence couvrante.

Cas n° 1 s est une $(i+1)$-séquence $2^n.f(i)$-bornée. Pour toute plus courte i-séquence, les pseudo-marquages restreints aux $i+1$ premières places doivent tous être différents sinon on pourrait raccourcir la séquence en supprimant une sous-séquence. Or le nombre de marquages restreints différents est égal $(2^n.f(i))^{i+1}$ ce qui nous donne la majoration recherchée.

Cas n° 2 s n'est pas une $(i+1)$-séquence $2^n.f(i)$-bornée. Donc $s = s_1.t.s_2$ avec s_1 qui est $2^n.f(i)$-bornée et s_2 qui commence par un pseudo-marquage m_2

dont l'une des $(i+1)$ premières composantes est au moins égale à $2^n.f(i)$. Sans perte de généralité, on peut supposer qu'il s'agit de p_{i+1}. Pour les mêmes raisons qu'au cas n° 1, la longueur de s_1 est $\leq (2^n.f(i))^{i+1}$.

(R, m_2) $i+1$-couvre m_c donc bien sûr il i-couvre aussi m_c. Ce qui signifie qu'il existe une séquence i-couvrante s'_2 de longueur au plus $f(i)$ qui part de m_2. Cette séquence a au plus $f(i)-1$ transitions. Le marquage de p_{i+1} ne peut donc décroitre de plus de $2^n.(f(i)-1)$ et donc le marquage final de p_{i+1} est $\geq 2^n \geq m_c(p_{i+1})$. s'_2 est donc une séquence $(i+1)$-couvrante et $s_1.t.s'_2$ a la longueur recherchée.

♢

Par un raisonnement simple sur la relation de récurence, on obtient que $f(k) \leq 2^{(3n)^k} \leq 2^{2^{c.n.log(n)}}$ pour une constante c indépendante de tous les paramètres du problème.

Pour rechercher de manière indéterministe un chemin, il suffit de conserver le dernier marquage du chemin en cours de construction et un compteur de la longueur. Le marquage ne peut excéder $2^{2^{c.n.log(n)}}.2^n$ jetons dans une place. Cet algorithme occupera donc une place mémoire $2^{d.n.log(n)}$ pour une autre constante d.

Par la procédure de Savitch [AHO 74], tout algorithme peut être rendu déterministe à condition de réserver une taille mémoire de l'ordre du carré de la taille mémoire de l'algorithme initial. On obtient donc un algorithme dans $\mathcal{EXP}space$.

Le caractère non borné d'un réseau peut-être résolu d'une façon analogue (mais plus élaborée) à partir de la caractérisation de la proposition 4. Sachant que ces deux problèmes sont $\mathcal{EXP}space$-difficiles [LIP 76], nous obtenons :

Théorème 23 (Complexité de la couverture et du caractère borné) *Le problème de la couverture et celui du caractère borné d'un réseau sont $\mathcal{EXP}space$-complets*

Dans [YEN 92], cette méthode est étendue à un langage de formules portant sur les séquences, pour lequel chaque fois qu'une formule est satisfaisable, elle l'est par une séquence dont on connait une borne de la longueur. Une application de cette méthode sera aussi évoquée au premier chapitre du deuxième tome.

4.3. *Analyse rétrograde*

Le principe de l'analyse rétrograde consiste à construire une représentation finie de l'ensemble des marquages initiaux qui valident la propriété de couver-

ture. Il a été employé d'abord dans [ARN 76] puis redécouvert par [ABD 96] et enfin généralisé dans [FIN 98].

Définition 24 *Soit R un réseau de Petri et m_c un marquage,*
- *$Couv(R, m_c)$ est l'ensemble des marquages m tels que (R, m) couvre m_c*
- *$\forall n \in \mathbb{N}, Couv_n(R, m_c)$ est l'ensemble des marquages m tels que (R, m) couvre m_c par une séquence de longueur au plus n*

Avant de décrire l'algorithme, nous rassemblons quelques faits élémentaires.
- $Couv(R, m_c) = \cup_{n \in \mathbb{N}} Couv_n(R, m_c)$
- $Couv_n(R, m_c) \subset Couv_{n+1}(R, m_c)$
- $Couv_n(R, m_c)$ est clos supérieurement : $m \in Couv_n(R, m_c)$ et $m' \geq m \Rightarrow m' \in Couv_n(R, m_c)$

Pour un ensemble de marquages E, notons $E^{\uparrow}$ l'ensemble des marquages supérieurs ou égaux à un marquage de E. Les ensembles clos supérieurement sont caractérisés par la propriété $E = E^{\uparrow}$.

Nous désirons obtenir $Couv(R, m_c)$ en calculant par itérations successives les $Couv_n(R, m_c)$. Ceci nécessite que cette suite d'ensembles se stabilise, c'est à dire qu'à partir d'un certain rang, tous les ensembles soient égaux et donc égaux à $Couv(R, m_c)$.

Lemme 25 *Soit $\{E_n\}_{n \in \mathbb{N}}$ une suite d'ensembles de marquages clos supérieurement et croissants ; cette suite se stabilise.*

Preuve
Dans le cas contraire, chaque fois qu'un ensemble n'est pas égal au précédent, ceci signifie qu'il y a un marquage de cet ensemble qui n'est supérieur ou égal à aucun des marquages des ensembles précédents. Nous sélectionnons l'un de ces éléments. En itérant le procédé, on obtient une suite infinie de marquages qui ne sont pas supérieurs ou égaux aux marquages précédents de la suite. Mais ceci contredit le lemme 3. ♢

Pour obtenir une représentation finie d'un ensemble clos supérieurement, il suffit de prendre les éléments minimaux de cet ensemble. En effet, puisqu'ils sont tous incomparables, ils ne peuvent être en nombre infini car de nouveau cela contredirait le lemme 3. Appelons $Min(E)$ l'ensemble des éléments minimaux de E. Puisque E est clos supérieurement, $E = Min(E)^{\uparrow}$ ce qui montre qu'un ensemble clos supérieurement est déterminé de manière univoque par ses élements minimaux.

$Couv_0(R, m_c)$ représente l'ensemble des marquages initiaux qui couvrent m_c par une séquence de longueur nulle. Autrement dit ce sont les marquages supérieurs ou égaux à m_c. Donc $Min(Couv_0(R, m_c)) = \{m_c\}$.

Pour parvenir à une méthode effective, nous devons montrer comment calculer $Min(Couv_{n+1}(R, m_c))$ à partir de $Min(Couv_n(R, m_c))$.

Soit $m' \in Couv_{n+1}(R, m_c)$. Par définition, m' permet de couvrir m_c par une séquence de longueur au plus n ou égale à $n+1$. Dans le premier cas, m' est supérieur ou égal à l'un des marquages de $Min(Couv_n(R, m_c))$. Dans le second cas, il permet de franchir une transition et d'obtenir un marquage supérieur ou égal à l'un des marquages de $Min(Couv_n(R, m_c))$.

Soit m, l'un des marquages de $Min(Couv_n(R, m_c))$. Un marquage obtenu par franchissement de t qui doit être supérieur ou égal à m est nécessairement supérieur ou égal à $Sup(Post(t), m)$ et le marquage antérieur au franchissement est supérieur ou égal à $Sup(Post(t), m) - C(t)$. En synthétisant :

$$Min(Couv_{n+1}(R, m_c)) =$$
$$Min(Min(Couv_n(R, m_c)) \cup_{t \in T} \cup_{m \in Min(Couv_n(R, m_c))} Sup(Post(t), m) - C(t))$$

Chaque étape multipliant au plus par $|T|+1$ le nombre d'éléments minimaux et connaissant grâce à la méthode précédente une borne sur le nombre d'étapes avant la stabilisation, on en déduit que la méthode est primitive récursive.

A priori, la méthode des plus courtes séquences semble plus efficace. Cependant l'intérêt de l'analyse rétrograde sera illustré lors de la présentation des extensions de réseaux de Petri.

5. Le problème de l'accessibilité

Le problème de l'accessibilité consiste à décider si, étant donné un réseau R et deux marquages m_i et m_f, il existe une séquence de franchissement : $m_i[\sigma\rangle m_f$. L'algorithme de décision de ce problème a été établi de manière indépendante par E. Mayr [MAY 84] et S.R. Kosaraju [KOS 82]. Il n'est pas question ici de reproduire cette preuve dont certains aspects sont particulièrement techniques et à laquelle un livre est dévolu [REU 89]. Nous décrirons le schéma de la preuve en développant uniquement son aspect central c'est à dire la condition suffisante d'accessibilité.

Tout d'abord, le problème de l'accessibilité est résolu pour un modèle plus général que celui des réseaux de Petri appelé chaine de systèmes d'addition de vecteurs à états (*CSAVE*) et ceci pour des raisons liées uniquement à la preuve. Celle-ci se décompose comme suit :

- Une *taille* est définie pour le problème d'accessibilité. Cette taille est un élément d'un ensemble bien fondé (c'est à dire tel **qu'il n'existe pas de suite infinie de tailles strictement décroissantes**).
- On établit une condition suffisante décidable d'accessibilité à l'aide entre autres **du graphe de couverture**.

- Si la condition n'est pas remplie, on construit un ensemble fini (éventuellement vide) de problèmes **de plus petite taille** tels qu'il existe une solution au problème initial si et seulement si il existe une solution à l'un des problèmes réduits.
- Partant du problème initial pour racine, ceci nous donne un arbre de problèmes en prenant comme fils d'un problème, les problèmes réduits. Si cet arbre était infini, il aurait une branche infinie ce qui est impossible en raison du premier point.
- Le problème initial a une solution si et seulement si l'un des feuilles a une solution (par la condition suffisante). Dans la pratique, on s'arrêtera au premier problème qui a une solution.

Cette procédure appelle quelques commentaires. Puisqu'elle s'appuie sur le graphe de couverture, elle n'est pas primitive récursive. D'autre part, on sait seulement que le problème de l'accessibilité est $EXPspace$-difficile [LIP 76]. Autrement dit, à l'heure actuelle, une question ouverte est de déterminer si ce problème est primitif récursif. La réduction du problème en problèmes (plus petits) de même nature est la cause du choix du modèle *CSAVE*. Comme nous le verrons plus loin, dans les réseaux de Petri lorsque la condition suffisante n'est pas remplie, on réduit le problème à des problèmes de nature différente.

Le problème de décision d'un grand nombre de propriétés se réduit de manière directe ou indirecte au problème de l'accessibilité. On prouve ainsi que la vivacité [HAC 75], la pseudo-vivacité et la propriété pour un marquage d'être un état d'accueil sont décidables [FRU 86, FRU 89].

Nous débutons la présentation de la condition suffisante par la procédure de décision d'une condition nécessaire, partie intégrante de la condition suffisante à venir.

5.1. *Une condition nécessaire d'accessibilité*

S'il existe une séquence σ telle que $m_i[\sigma\rangle m_f$ alors, d'après l'équation de changement d'état, on a : $C.\overrightarrow{\sigma} = m_f - m_i$. Montrons comment décider l'existence d'une solution $x \in \mathbb{N}^T$ pour l'équation $C.x = b$ avec $b \in \mathbb{Z}^P$. Dans le cas où $|T| = 1$, l'équation se résout trivialement. Aussi nous supposerons que $|T| > 1$.

Nous appliquons d'abord le calcul des T-flots. Si aucun T-flot n'est trouvé, alors les vecteurs colonnes $\{C(t)\}_{t\in T}$ forment une famille linéairement indépendante et il existe au plus une solution dans $\mathbb{Q}^T$ à ce problème. L'existence et le calcul de cette solution s'effectue par la variante principale de l'élimination de Gauss. Si une solution est trouvée, il suffit de vérifier qu'elle appartient à $\mathbb{N}^T$.

Dans l'autre cas, appelons a le T-flot trouvé ($C.a = \overrightarrow{0}$), et notons $T' = \{t/a(t) > 0\}$. Nous pouvons supposer sans perte de généralité que T' n'est pas vide. Supposons qu'il existe une solution x à l'équation telle que $\forall\, t \in T',\ x(t) > a(t)$ alors de manière évidente $x - a$ est aussi une solution. En itérant ce procédé, on obtient y une solution vérifiant $\exists\, t \in T'$ t.q. $y(t) \leq a(t)$. On peut donc remplacer notre équation par une famille d'équations, une par couple $t \in T'$ et $0 \leq i \leq a(t)$ où on remplace l'inconnue $x(t)$ par i. L'existence d'une solution pour l'une quelconque des équations est équivalente à l'existence d'une solution pour l'équation initiale. Chacune des équations a une inconnue de moins. Ainsi par itération, on arrive (dans le pire des cas), à des équations à une inconnue dont la résolution est triviale.

5.2. *Une condition suffisante d'accessibilité*

Nous introduisons ici le réseau inverse $\widetilde{R}$ qui remonte le temps du réseau R. Les places et les transitions de ce réseau sont identiques et les matrices de conditions sont définies par : $\forall\, t \in T,\ \widetilde{Pré}(t) = Post(t)$ et $\widetilde{Post}(t) = Pré(t)$

Il est élémentaire de vérifier que :

$\forall\, \sigma \in T^*\ m[\sigma\rangle_R m' \Leftrightarrow m'[\widetilde{\sigma}\rangle_{\widetilde{R}} m$, $\widetilde{C} = -C$ et $\overrightarrow{\widetilde{\sigma}} = \overrightarrow{\sigma}$

Nous cherchons à transformer la condition précédente en condition suffisante d'accessibilité. Reprenant le vocabulaire de la section 4.2, nous appellerons pseudo-marquage un élément de $\mathbb{Z}^P$. Etant donnés deux pseudo-marquages m, m' et une séquence de transitions σ, nous appellerons pseudo-séquence de franchissement (notée $m(\sigma\rangle m'$), une séquence qui vérifie l'équation de changement d'état $m' = m + C.\overrightarrow{\sigma}$.

Nous établissons d'abord un résultat préliminaire qui garantit qu'une pseudo-séquence de franchissement est en réalité une séquence.

Lemme 26 *Soit R un réseau de Petri et $m_0(\sigma\rangle m_1(\sigma\rangle m_2 \ldots m_{k-1}(\sigma\rangle m_k$ une pseudo-séquence de franchissement alors :*

$m_0(\sigma\rangle m_1$ et $m_{k-1}(\sigma\rangle m_k$ sont des séquences de franchissement
$\Leftrightarrow$ La séquence complète est une séquence de franchissement

Preuve
Seule l'implication de gauche à droite est à démontrer. Nous partitionnons P en deux sous-ensembles $P' = \{p \in P/\overrightarrow{p}^t.C.\overrightarrow{\sigma} \geq 0\}$ et $P" = \{p \in P/\overrightarrow{p}^t.C.\overrightarrow{\sigma} < 0\}$. La séquence σ restreinte à P' est répétitive pour le réseau R, donc puisque $m_0[\sigma\rangle_R m_1$ alors $m_0[\sigma^k\rangle_{R,P'} m_k$.

En appliquant les résultats élémentaires qui lient R et $\widetilde{R}$, la séquence $\widetilde{\sigma}$ restreinte à P" est répétitive croissante pour le réseau $\widetilde{R}$. Donc puisque $m_k[\widetilde{\sigma}\rangle_{\widetilde{R}} m_{k-1}$ alors $m_k[\widetilde{\sigma}^k\rangle_{\widetilde{R},P"} m_0$ ce qui est équivalent à $m_0[\sigma^k\rangle_{R,P"} m_k$.

Sachant que $P = P' \cup P"$, nous obtenons : $m_0[\sigma^k\rangle_R m_k$. $\Diamond$

Pour transformer la condition nécessaire en condition suffisante, il faudrait s'affranchir des conditions de franchissabilité. Très approximativement, l'idée générale consiste à faire croître le marquage initial par une séquence de franchissement (σ_1^k dans la preuve) de telle sorte que la pseudo-séquence de franchissement (σ_2) devienne une séquence, puis de faire décroître le marquage par une séquence (σ_4^k). Cependant avant ce dernier franchissement, il faut préalablement annuler les effets conjugués de la première et de la dernière séquence par une séquence intermédiaire (σ_3^k).

Proposition 27 (Condition suffisante d'accessibilité) *Soit R un réseau de Petri, m_i et m_f deux marquages, si :*

1. *R est consistant*
2. *$\exists\, v \in \mathbb{N}^T$ t.q. $C.v = m_f - m_i$*
3. *$\sum_{p\in P} \omega.\overrightarrow{p}$ appartient aux arbres de couverture $AC(R, m_i)$ et $AC(\widetilde{R}, m_f)$*

Alors m_f est accessible depuis m_i dans le réseau R.

Preuve
Nous allons construire par étapes la séquence d'accessibilité.

Tout d'abord la condition 3 se traduit par l'existence d'une séquence σ_1 répétitive croissante (pour toutes les places) depuis m_i dans R et d'une séquence $\widetilde{\sigma_4}$ répétitive croissante (pour toutes les places) depuis m_f dans $\widetilde{R}$.

Puisque R est consistant il existe un vecteur positif w de support T tel que $C.w = 0$. Puisque son support est T, il existe un entier n suffisamment grand tel que $w' = n.w - \overrightarrow{\sigma_1} - \overrightarrow{\sigma_4} \geq \overrightarrow{0}$. Appelons σ_3 une séquence quelconque qui vérifie $\overrightarrow{\sigma_3} = w'$.

Finalement, appelons σ_2 une séquence quelconque vérifiant $\overrightarrow{\sigma_2} = v$. Remarquons que σ_2 est une pseudo-séquence de franchissement de m_i à m_f. Soit $k \geq 2$ un entier tel que le marquage de toutes les places avec k jetons rende franchissable :

- la séquence $\sigma_2.\sigma_3$ dans R
- la séquence $\widetilde{\sigma_3}$ dans $\widetilde{R}$

Nous allons prouver que $m_i[\sigma_1^k\rangle m_1[\sigma_2\rangle m_2[\sigma_3\rangle m_3[\sigma_3^{k-2}\rangle m_4[\sigma_3\rangle m_5[\sigma_4^k\rangle m_f$ est bien une séquence de franchissement (les marquages intermédiaires sont introduits pour faciliter le raisonnement).

Calculons d'abord l'incidence de cette séquence :

$$C.(k.\overrightarrow{\sigma_1} + \overrightarrow{\sigma_2} + k.\overrightarrow{\sigma_3} + k.\overrightarrow{\sigma_4}) = C.(v + k.(w' + \overrightarrow{\sigma_1} + \overrightarrow{\sigma_4}))$$

$$= C.(v + k.n.w) = C.v = m_f - m_i$$

Nous avons donc affaire à une pseudo-séquence de franchissement. Montrons que les conditions de franchissabilité sont réunies (nous utilisons implicitement à plusieurs reprises les relations entre R et $\widetilde{R}$) :

- par définition de σ_1 et σ_4, nous avons $m_i[\sigma_1^k\rangle m_1$ et $m_5[\sigma_4^k\rangle m_f$,
- par le choix de k, nous avons $m_1[\sigma_2\rangle m_2[\sigma_3\rangle m_3$ et $m_4[\sigma_3\rangle m_5$,
- ce qui implique en vertu du lemme précédent $m_2[\sigma_3\rangle m_3[\sigma_3^{k-2}\rangle m_4[\sigma_3\rangle m_5$.

$\Diamond$

Examinons, comment poursuivre la procédure de décision si l'un des points de la condition suffisante n'est pas remplie. S'il s'agit du point 2 (i.e la condition nécessaire) alors m_f n'est pas accessible. S'il s'agit du point 3, cela signifie qu'au cours de l'hypothétique séquence de franchissement, le marquage de l'une des places reste bornée par sa plus grande valeur finie apparaissant sur les deux arbres de couverture (en fait le minimum de ses plus grandes valeurs finies sur chaque arbre). On remplace donc le problème initial par $|P|$ problèmes d'accessibilité avec une place devant rester bornée par une valeur connue.

Pour indiquer le traitement de cas où le réseau n'est pas consistant, nous introduisons des sous-ensembles particuliers de vecteurs à coefficients entiers positifs.

Définition 28 (Ensembles semi-linéaires) *Un ensemble linéaire de vecteurs entiers positifs E est défini par un vecteur u et une famille de vecteurs $V = \{v_1, \ldots, v_m\}$:*

$$E = \{w / \exists\, \lambda_1, \ldots, \lambda_m \ \in \mathbb{N}, t.q.\ w = u + \textstyle\sum_{i=1}^{m} \lambda_i.v_i\}$$

Un ensemble semi-linéaire E de vecteurs entiers est une union finie d'ensembles linéaires.

Les ensembles semi-linéaires sont des représentations finies d'ensembles infinis qui possèdent de nombreuses propriétés intéressantes. On peut calculer l'union et l'intersection de deux ensembles ainsi que le complémentaire d'un ensemble semi-linéaire ; tous ces ensembles étant eux-mêmes semi-linéaires. On peut aussi décider si un vecteur appartient à un ensemble semi-linéaire. Si, par exemple, l'ensemble d'accessibilité était toujours un ensemble semi-linéaire calculable de $\mathbb{N}^P$ alors le problème de l'accessibilité serait automatiquement résolu. Malheureusement, les ensembles d'accessibilité de certains réseaux ne sont pas semi-linéaires [HOP 79] et on peut décider si c'est le cas [HAU 90].

Par contre l'ensemble E des solutions de $C.x = m_f - m_i$ est un ensemble semi-linéaire dont la représentation est calculable (voir par exemple [REU 89]) et plus précisément égal à $\{u + \sum \lambda_k.v_k / u$ est une solution minimale de $C.x = m_f - m_i$, $\lambda_k \in \mathbb{N}$ et $\{v_k\}$ est l'ensemble des solutions minimales de $C.x = \overrightarrow{0}\}$; toutes ces solutions minimales étant nécessairement en nombre fini d'après le lemme 3. En conséquence, le réseau n'est pas consistant si et seulement il existe t tel que $v(t) = 0$ pour toute solution de $C.x = \overrightarrow{0}$. Autrement dit, toute séquence d'accessibilité aura pour nombre d'occurences de t, la valeur $u(t)$ d'une des solutions minimales de $C.x = m_f - m_i$. On remplace donc le problème d'accessibilité par un ensemble de problèmes pour lesquels la séquence d'accessibilité comprend un nombre fixé d'occurences d'une certaine transition.

Intuitivement dans les deux transformations, le caractère *infini* du problème a été réduit puisque dans un cas le marquage d'une place est contraint à demeurer borné par une valeur fixée et dans l'autre cas le nombre d'occurences d'une transition est fixé. La formalisation de cet argument conduit naturellement au modèle des *CSAVE*.

6. Extensions de réseaux de Petri

6.1. *Les réseaux à arcs inhibiteurs*

Le pouvoir d'expression des réseaux de Petri est proche d'un langage de programmation travaillant sur des entiers. Il manque cependant aux réseaux, le test d'égalité entre le marquage d'une place et une valeur fixe pour parvenir à en faire un véritable langage de programmation. C'est à cette fin qu'ont été introduits les réseaux à arcs inhibiteurs. Dans ce modèle, les matrices d'incidence sont complétées par une matrice d'inhibition qui impose que le marquage d'une place soit strictement inférieur à une valeur donnée.

Définition 29 (Réseau de Petri à arcs inhibiteurs) *Un réseau de Petri à arcs inhibiteurs est défini par un tuple $R = \langle P, T, Pré, Post, Inh\rangle$ où :*

- *P est un ensemble fini de places, T est un ensemble fini de transitions.*
- *$Pré$ et $Post$ sont les matrices d'incidence arrière et avant définies dans $\mathbb{N}^{P\times T}$.*
- *Inh est la matrice d'inhibition définie dans $(\mathbb{N}_\omega \setminus 0)^{P\times T}$*

Définition 30 (Franchissement dans les réseaux à arcs inhibiteurs) *Soit m un marquage d'un réseau de Petri à arcs inhibiteurs et t une transition :*

- *t est franchissable à partir de m si et seulement si*

$$m \geq Pré(t) \text{ et } m < Inh(t)$$

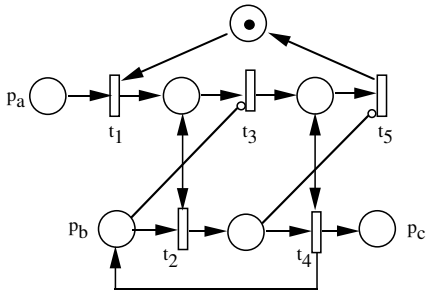

Figure 5: Produit de deux nombres à l'aide d'un réseau à arcs inhibiteurs

– *Le franchissement de t à partir de m conduit au marquage m' défini par :*
$$m' = m + C(t)$$

Seule la condition de franchissabilité est modifiée. Un arc inhibiteur est représenté par un trait terminé par un petit cercle. Les valuations différentes de 1 étiquettent les arcs et un arc valué ω n'est pas représenté (car il ne contraint pas le réseau).

La figure 5 explique la différence fondamentale entre les réseaux à arcs inhibiteurs et les réseaux ordinaires. Si on ajoute au réseau représenté a jetons dans p_a et b jetons dans p_b, alors l'unique séquence maximale $(t_1.t_2^b.t_3.t_4^b.t_5)^a$ conduit à un marquage où p_c contient $a.b$ jetons. Si l'on supprime les arcs inhibiteurs, il y a plusieurs séquences maximales, toutes conduisant à un nombre de jetons dans p_c inférieur ou égal à $a.b$ et l'une d'elles vérifiant l'égalité. Autrement dit, un réseau de Petri est capable de calculer *faiblement* (au sens décrit précédemment) une fonction arithmétique croissante calculable tandis qu'un réseau à arcs inhibiteurs peut calculer *exactement* toute fonction arithmétique calculable. Le réseau de la figure 2 illustre aussi la différence de pouvoir d'expression des deux modèles. Dans ce réseau (**et dans tout réseau de Petri** modélisant les actions du planteur), il est impossible d'obliger le planteur à manger toutes ses bananes avant d'aller au jardin alors que l'ajout d'un seul arc inhibiteur suffit à obtenir ce comportement.

Ce pouvoir d'expression étendu conduit inévitablement à l'indécidabilité de toutes les propriétés intéressantes (accessibilité, vivacité, caractère borné, couverture, terminaison,...). En effet, l'arrêt d'un programme est un problème indécidable et il n'est pas difficile de réduire ce problème au problème de décision d'une de ces propriétés même en présence uniquement de deux arcs inhibiteurs.

Néanmoins, il a été démontré que l'accessibilité reste décidable en présence d'un seul arc inhibiteur ou d'une structure d'arcs inhibiteurs ordonnée par les places ($Inh(p_i,t) \neq \omega$ et $j < i \Rightarrow Inh(p_j,t) \neq \omega$), (tous les arcs en question étant valués par 1) [REI 95].

6.2. *Les réseaux auto-modifiants*

Une autre extension intéressante est de rendre le nombre de jetons à consommer et à produire dépendant du marqauge courant. Dans le modèle initial des réseaux auto-modifiants, la valuation d'un arc s'exprime comme une combinaison linéaire du marquage des places plus une constante [VAL 78]. Dans le modèle généralisé appelé G-réseaux, la valuation est un polynôme à coefficients positifs sur les places [DUF 98b].

Notation
- $\mathbb{N}[P]$ désigne l'ensemble des polynômes à coefficients entiers positifs dont les variables sont les places.
- Soit m un marquage et Q un polynôme sur les places, alors $Q[m]$ désigne la valeur du polynôme lorsque chaque variable p vaut $m(p)$.

Définition 31 (G-réseau)
Un G-réseau est défini par un tuple $R = \langle P, T, Pré, Post\rangle$ où :
- *P est un ensemble fini de places, T est un ensemble fini de transitions.*
- *$Pré$ et $Post$ sont les matrices d'incidence arrière et avant définies dans $\mathbb{N}[P]^{P\times T}$.*

Définition 32 (Franchissement dans les G-réseaux) *Soit m un marquage d'un G-réseau, t une transition :*
- *t est franchissable à partir de m si et seulement si :*
$$m \geq Pré(t)[m]$$
- *Le franchissement de t à partir de m conduit au marquage m' défini par :*
$$m' = m - Pré(t)[m] + Post(t)[m]$$

Sans restriction sévère, les G-réseaux simulent facilement les arcs inhibiteurs et par conséquent les propriétés principales sont toutes indécidables. La manière la plus facile de simuler un arc inhibiteur de p vers t est de poser $Pré(p,t) = 2.p$ puisque $m(p) \geq 2.m(p) \Leftrightarrow m(p) = 0$.

Cette simulation donne une idée des restrictions à apporter pour rendre décidables certaines propriétés. Nous nous limitons à trois restrictions et renvoyons le lecteur à la thèse de C. Dufourd [DUF 98a] pour une étude approfondie de la hiérarchie des restrictions.

- Les *G-Post-réseaux* sont obtenus en exigeant que les valuations des préconditions soient des entiers. D'après la règle de franchissement, les deux assertions du lemme 1 de monotonie restent vérifiées. Ce qui a pour conséquence de rendre valide la construction de l'arbre de couverture (moyennant quelques précautions dans la démonstration). On peut donc décider pour ces réseaux, la couverture, le caractère borné du réseau, le caractère borné d'une place et la terminaison.

- Les *G-post-réseaux avec reset* sont obtenus en autorisant la valuation d'un arc p vers t à être soit un entier, soit le polynôme p. Dans ce dernier cas, le franchissement de t vide la place p. Pour ces réseaux, seule la première assertion du lemme de monotonie reste vérifiée :
$$\forall\, m_1 \leq m_1'\ m_1[\sigma\rangle m_2 \Rightarrow\ m_1'[\sigma\rangle m_2' \text{ avec } m_2 \leq m_2'$$
On peut donc encore décider de la terminaison par une construction qui détecte les séquences répétitives. La couverture est aussi décidable mais cette fois-ci **grâce à l'analyse rétrograde de la section 4.3** qui ne repose que sur cette partie du lemme de monotonie. Un résultat important et difficile est l'indécidabilité du caractère borné de ce type de réseau.
- Les *G-post-réseaux avec transfert* sont des restrictions des réseaux précédents où la présence d'un arc de p vers t valué par p impose la présence d'un arc de t vers une place p' valué par $p + Q$ où $Q \in \mathbb{N}^{[P]}$. Autrement dit, lorsqu'on vide le contenu d'une place, il est transféré dans une autre place (avec éventuellement d'autres jetons). Cette fois-ci, la construction de l'arbre de couverture permet de décider à l'apparition du premier ω que le réseau est non borné. Il est alors amusant de constater que pour ces réseaux le caractère borné d'une place est indécidable. En effet, on réduit le problème du caractère borné d'un réseau avec reset au problème du caractère borné d'un sous-ensemble de places d'un réseau avec transfert par une transformation élémentaire. On ajoute au réseau avec reset une place *puits* et pour chaque transition t, un arc t vers *puits* valué par $\sum_{p \in P'} p$ où P' est l'ensemble des places ayant un arc de reset vers t. Il est alors immédiat que le réseau avec reset est borné *ssi* toutes les places de $P \setminus \{puits\}$ sont bornées dans le nouveau réseau.

Nous terminons en démontrant que l'accessibilité reste indécidable moyennant la présence :
- d'arcs en sortie de transition valués par la place de destination que l'on appelle arcs doubles (pour une raison évidente)
- d'arcs de reset

Proposition 33 *Le problème de l'accessibilité dans les réseaux de Petri à arcs inhibiteurs est réductible :*
- *au problème de l'accessibilité dans les G-réseaux avec uniquement des arcs ordinaires et doubles*
- *au problème de l'accessibilité dans les G-réseaux avec uniquement des arcs ordinaires et de reset.*

Preuve
Pour chaque place p du réseau à arcs inhibiteurs, nous ajoutons une place p^+ qui a les mêmes incidences que p. Soit m un marquage du premier réseau, alors m^+ dans le second réseau est défini par $m^+(p) = m^+(p^+) = m(p)$. Les arcs inhibiteurs du premier réseau sont transformés comme indiqué dans la figure 6

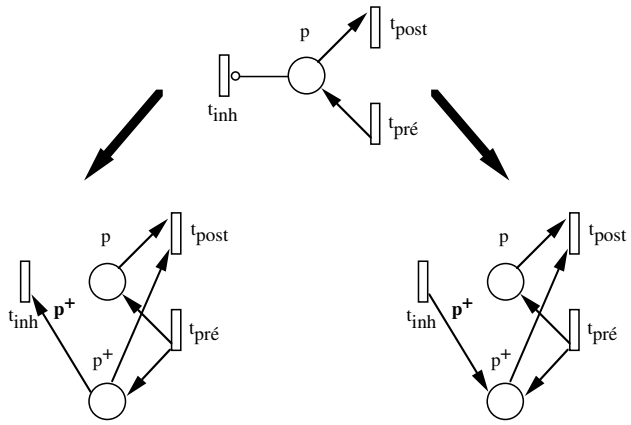

Figure 6: Contrôle du bon fonctionnement d'un arc inhibiteur

à l'aide soit d'un arc double, soit d'un arc avec reset. Nous laissons le soin au lecteur d'établir que quelque soit la construction :

$$m' \text{ est accessible depuis } m \text{ dans le premier réseau}$$
$$\Leftrightarrow m'^{+} \text{ est accessible depuis } m^{+} \text{ dans le second réseau}$$

Indication : p^+ contient le même nombre de jetons que p si et seulement si aucune transition avec un arc inhibiteur issu de p dans le réseau initial n'est franchie dans le réseau transformé alors que p est marqué. ♢

6.3. *Les réseaux récursifs*

Un réseau récursif [HAD 99b] a la même structure qu'un réseau de Petri ordinaire, à la différence près que dans les réseaux récursifs, les transitions sont partitionnées en deux catégories : transitions abstraites et transitions élémentaires. De plus, un marquage *originel* est associé à chaque transition abstraite et un ensemble semi-linéaire de marquages terminaux est défini. La sémantique d'un tel réseau peut être informellement expliquée comme suit. Dans un réseau de Petri ordinaire, un processus *joue* avec les jetons, en franchissant une transition et mettant à jour le marquage courant. Dans un réseau récursif, il y a un arbre dynamique de processus correspondant à une relation de paternité, chacun jouant son propre jeu de jetons. Un pas d'un réseau récursif est alors un pas d'exécution de n'importe lequel des processus. Si le processus franchit une transition élémentaire, il met à jour son marquage courant en utilisant la règle de franchissement élémentaire. Si le processus franchit une transition abstraite, il consomme les jetons des pré-conditions de la transition et engendre un nouveau fils qui démarre son jeu de jetons avec le marquage originel associé à la

transition. Si le processus a atteint un marquage terminal, il peut se terminer en tuant toute sa descendance et en produisant dans le marquage de son père, les jetons des post-conditions de la transition dont le franchissement lui a donné vie. S'il s'agit du processus racine, on obtient un arbre vide. Nous formalisons ce fonctionnement dans les définitions qui suivent.

Définition 34 (Réseau de Petri récursif) *Un réseau de Petri récursif est défini par un tuple* $R = \langle P, T, Pré, Post, \Omega, \Upsilon \rangle$ *où :*

- P *est un ensemble fini de places,* T *est un ensemble fini de transitions.*
- *Une transition de* T *est soit élémentaire soit abstraite. Les sous-ensembles des transitions élémentaires et abstraites sont respectivement notés par :* T_{el} *et* T_{ab}*.*
- $Pré$ *et* $Post$ *sont les matrices d'incidence arrière et avant définies dans* $\mathbb{N}^{P \times T}$*.*
- Ω *est une fonction qui associe à chaque transition abstraite un marquage ordinaire (i.e. un élément de* $\mathbb{N}^P$*) appelé le marquage originel de* t*.*
- Υ *est un ensemble semi-linéaire effectif de marquages terminaux*

Une représentation effective d'un ensemble semi-linéaire est une représentation qui peut (par un algorithme) être transformée sous la forme de la définition 28. Par exemple, une (in)équation linéaire sur les marquages est une représentation effective.

Définition 35 (Marquage étendu) *Un* marquage étendu *tr d'un réseau récursif R est un arbre étiqueté $tr = \langle V, M, E, A \rangle$ où :*

- V *est l'ensemble des noeuds,* M *est une fonction de* $V \to \mathbb{N}^P$*,*
- $E \subseteq V \times V$ *est l'ensemble des arcs et* A *est une fonction de* $E \to T_{ab}$*.*

Un réseau récursif marqué (R, tr_0) *est un réseau récursif associé à un marquage étendu initial.*

Soit v un noeud d'un marquage étendu, $pred(v)$ désigne son père dans l'arbre (défini si v n'est pas la racine) et $Succ(v)$ l'ensemble de ses successeurs directs et indirects (v inclus). Un *pas élémentaire* d'un réseau récursif est, soit le franchissement d'une transition, soit la suppression d'un sous-arbre (désigné comme un *pas de terminaison* et noté par τ).

Définition 36 *Une transition t est* franchissable *dans un noeud v d'un marquage étendu tr (noté par $tr[t, v\rangle$) si* $M(v) \geq Pré(t)$ *et un* pas de terminaison *est* franchissable *dans v (noté par $tr[\tau, v\rangle$) si* $M(v) \in \Upsilon$

Définition 37 *Le* franchissement *d'un pas élémentaire franchissable t dans un noeud v d'un marquage étendu tr conduit au marquage tr' défini selon le type de t.*

- $t \in T_{el}$

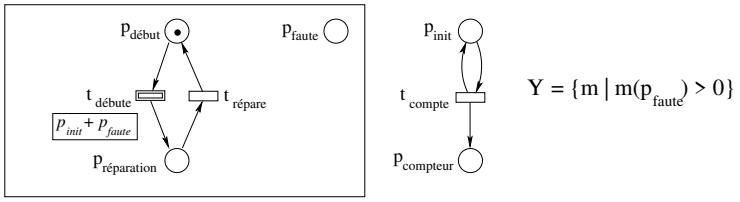

Figure 7: Un système élémentaire tolérant aux pannes

- $V' = V$, $E' = E$, $\forall e \in E, A'(e) = A(e)$, $\forall v' \in V \backslash \{v\}$, $M'(v') = M(v')$
  - $M'(v) = M(v) - Pré(t) + Post(t)$
- $t \in T_{ab}$, *(v' est un nouvel identifiant absent dans V)*
  - $V' = V \cup \{v'\}$, $E' = E \cup \{(v, v')\}$, $\forall e \in E, A'(e) = A(e)$, $A'((v, v')) = t$
  - $\forall v" \in V \setminus \{v\}, M'(v") = M(v")$, $M'(v) = M(v) - Pré(p)$
  - $M(v') = \Omega(t)$
- $t = \tau$
  - $V' = V \setminus Succ(v)$, $E' = E \cap (V' \times V')$, $\forall e \in E', A'(e) = A(e)$
  - $\forall v' \in V' \setminus \{pred(v)\}, M'(v') = M(v')$
  - $M'(pred(v)) = M(pred(v)) + Post(A(pred(v), v))$

  Si v est la racine de l'arbre, alors le franchissement de τ conduit à l'arbre vide noté $\perp$.

A première vue, il semble qu'associer le même réseau à toutes les transitions abstraites soit restrictif et artificiel. En réalité, il est facile de simuler un réseau avec une activation d'un réseau propre à chaque transition abstraite. Utiliser un seul réseau simplifie énormément les notations et facilite les preuves. La plupart des conditions usuelles s'expriment à l'aide d'ensemble semi-linéaires calculables à partir du réseau. Par exemple, on peut décrire l'ensemble des marquages morts, la franchissabilité d'une transition, des contraintes sur le marquage de certaines places,... Nous allons maintenant illustrer le pouvoir d'expression de ce modèle à l'aide d'une modélisation simple. D'autres exemples significatifs sont décrits dans [HAD 00]. Une transitions abstraite est figurée par un rectangle à double bordure accompagné dans un cadre de son marquage originel.

En vue d'analyser des systèmes tolérants aux fautes, l'ingénieur débute avec une description d'un système nominal puis introduit les comportements fautifs ainsi que les mécanismes de reprise sur pannes. Le système nominal représenté enregistre périodiquement une mesure de l'environnement (transition élémentaire t_{compte}). Le nombre de mesures est stocké dans la place $p_{compteur}$. Le système complet est obtenu en ajoutant la partie gauche de la figure 7. Le comportement de ce réseau récursif est le suivant. Initialement et immédiatement après l'occurence d'une panne, le marquage étendu est réduit à un simple

noeud. Un jeton dans la place $p_{réparation}$ indique que le système est en cours de réparation tandis qu'un jeton dans la place p_{init} signifie que le système est prêt. Quand la transition abstraite $t_{débute}$ est franchie, le comportement correct du système est exécuté par le nouveau processus. La terminaison de ce processus représente une occurence de panne. Comme la place p_{faute} est toujours marquée dans ce deuxième noeud et au vu de la définition de Υ, l'occurence d'une faute est possible à tout instant. Avec l'ajout de quelques places et la modification de Υ, nous pourrions modéliser des schémas de faute plus complexes (e.g. des fautes conditionnées par l'exécution de logiciel).

L'état du réseau est, soit un état avec un simple noeud, soit un état avec un noeud et une feuille. Cependant le nombre de marquages accessibles dans cette feuille est infini. Ce qui signifie que l'état de panne peut être atteint par une infinité d'états. Cette modélisation est **impossible** à l'aide d'un réseau de Petri car un état ne peut être atteint que par au plus $|T|$ transitions. Les réseaux auto-modifiants offrent aussi cette possibilité mais tel n'est pas le cas des réseaux à arcs inhibiteurs.

Contrairement aux autres extensions, les deux propriétés importantes décidables des réseaux de Petri le sont aussi pour les réseaux récursifs : l'accessibilité et le caractère borné [HAD 99a].

7. Langages de réseaux de Petri

L'introduction d'extensions de réseaux a pour but d'augmenter le pouvoir d'expression du réseau tout en préservant la décidabilité d'un certain nombre de propriétés. Les familles de langages engendrés par les réseaux sont justement l'un des moyens de cerner ce pouvoir d'expression. Initialement, les langages formels ont été étudiés du point de vue des grammaires qui les engendraient [HOP 69]. Rappelons brièvement qu'une grammaire repose sur des symboles non terminaux (dont un symbole initial) et terminaux (les lettres de l'alphabet). Une grammaire est constituée de règles de transformation $\{S_1 \ldots . S_m \rightarrow T_1 \ldots . T_n\}$. Pour composer un mot du langage, on démarre avec le symbole initial et on applique l'une quelconque des règles de transformation à une partie du mot courant jusqu'à ce qu'on obtienne un mot composé uniquement de symboles terminaux.

Selon la structure des grammaires, on définit des familles de langage et on étudie des problèmes tels que :

- l'appartenance d'un mot au langage,
- l'existence d'au moins un mot du langage,
- la clôture d'une famille de langages par des opérations comme l'union, l'intersection et le passage au complémentaire.

Chacun de ces problèmes a une interprétation en termes de comportement de systèmes modélisés par exemple par des réseaux de Petri. L'appartenance d'un mot au langage, permet de tester si des séquences de comportement attendues sont réalisables. L'existence d'au moins un mot du langage permet avec un choix judicieux de marquages terminaux de tester l'existence d'(au moins) une séquence fautive. La clôture d'une famille par une opération offre la possibilité au concepteur de construire des systèmes de manière modulaire à partir de spécifications définies à l'aide de ces opérations. Par exemple, l'intersection de langages traduit le plus souvent une composition synchronisée de sous-systèmes.

Classiquement, on distingue quatre familles de langages strictement incluses les unes dans les autres. Les langages réguliers sont engendrés par des grammaires dont les règles sont de la forme ($S \to \lambda$, $S \to a.T$, $S \to a$) avec S,T non terminaux et a terminal. Les langages algébriques sont engendrés par des grammaires dont les règles sont de la forme ($S \to T_1.\ldots.T_n$) avec n éventuellement nul. Les langages à contexte sont engendrés par des grammaires dont les règles sont de la forme ($S_{init} \to \lambda$, $S_1.\ldots.S_m \to T_1.\ldots.T_n$) avec $n \geq m$ et où S_{init} est le symbole initial. Enfin les langages de type 0 n'ont pas de restriction sur les règles de grammaire.

Exemple 1 *Grammaire régulière et algébrique*
La grammaire régulière ci-dessous dénote le comportement d'un processus engageant une action jusqu'à ce qu'elle soit couronnée de succès :
$S \to essai.T$, $T \to échec.S$, $T \to succès$
Le langage associé L est défini par :
$L = \{essai.(échec.essai)^n.succès\}_{n \in \mathbb{N}}$ ce que l'on note de manière abrégée
$L = essai.(échec.essai)^*.succès$

La grammaire algébrique ci-dessous engendre le langage L' des mots palindromes sur l'alphabet $\Sigma = \{a, b\}$, $L' = \{\sigma \in \Sigma^* \mid \widetilde{\sigma} = \sigma\}$
$S \to \lambda$, $S \to a$, $S \to b$, $S \to a.S.a$, $S \to b.S.b$

On peut décider de l'appartenance d'un mot pour les trois premières familles de langage et ce problème est indécidable pour les langages de type 0. On peut décider de l'existence d'au moins un mot pour les langages réguliers et algébriques mais ce problème devient indécidable pour les langages à contexte. Enfin les langages réguliers sont clos par les opérations usuelles alors que, par exemple, l'intersection de deux langages algébriques n'est pas nécessairement un langage algébrique [AUT 87].

La théorie des langages de réseaux de Petri consiste à étudier le même type de problèmes et à situer les langages de réseaux de Petri par rapport aux familles traditionnelles [PET 81]. Par souci de lisibilité, nous rappelons ici la définition d'un langage de réseau de Petri. Puis nous indiquons les principaux résultats obtenus.

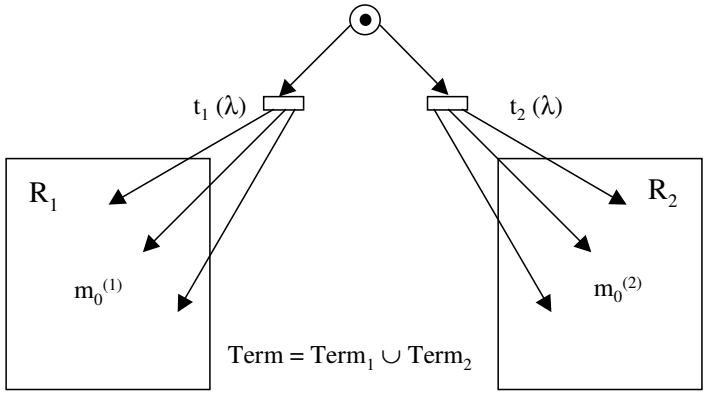

Figure 8: Construction d'un réseau pour l'union des langages

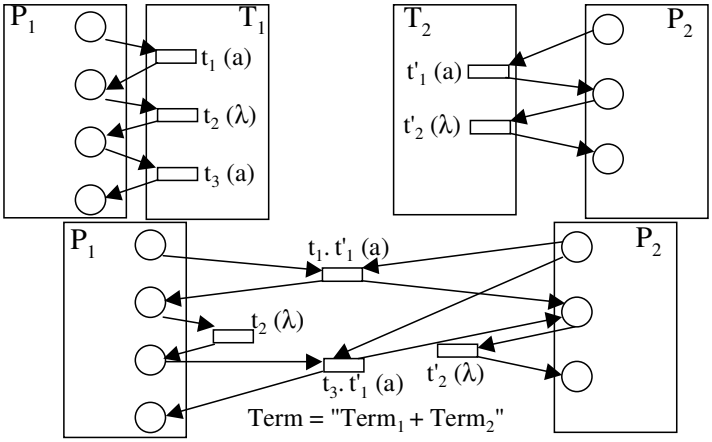

Figure 9: Construction d'un réseau pour l'intersection des langages

Définition 38 (Langage d'un réseau) *Soit (R, m_0) un réseau de Petri, soit Σ un alphabet et l une fonction d'étiquetage de T dans $\Sigma \cup \lambda$ (le mot vide). La fonction d'étiquetage s'étend aux séquences par $l(\lambda) = \lambda$ et $l(\sigma.t) = l(\sigma).l(t)$. Soit $Term$ un ensemble fini de marquages terminaux. Le langage du réseau noté $L(R, m_0, l, Term)$ est défini par :*

$$L(R, m_0, l, Term) = \{w \in \Sigma^* / \exists\, \sigma \in T^*, \exists\, m_f \in Term,\ m_0[\sigma\rangle m_f \text{ et } w = l(\sigma)\}$$

Proposition 39 (Clôture de la famille des langages de RdP) *Les langages de réseaux de Petri sont clos par union et intersection.*

Preuve
La construction du réseau qui reconnait l'union des langages est représentée sur la figure 8. On reproduit les réseaux (supposés disjoints) sans marquage. On ajoute aux deux réseaux une place initialement marquée entrée de deux nouvelles transitions étiquetées par le mot vide franchissables initialement et dont les sorties sont respectivement les marquages initiaux des deux réseaux. L'ensemble des marquages terminaux est l'union des ensembles des marquages terminaux des deux réseaux. Ainsi ce réseau choisit de manière indéterministe d'"activer" l'un des deux réseaux qui produira l'un des mots de son langage.

La construction du réseau qui reconnait l'intersection des langages est représentée sur la figure 9. On reproduit les places (supposés disjointes) des deux réseaux avec leur marquage initial. Pour chaque couple de transitions (une par réseau) étiquetées par la même lettre, on crée une transition de même étiquette dont les incidences sont les sommes des incidences de chacune des transitions. Les transitions étiquetées par le mot vide sont reproduites sans changement. Un marquage terminal est la somme d'un marquage terminal du premier réseau et d'un marquage terminal du deuxième réseau. Ainsi un mot est reconnu simultanément dans les deux réseaux, car chaque lettre est produite par le franchissement simultané des deux transitions du couple. Les transitions étiquetées par le mot vide doivent pouvoir être franchies indépendamment afin d'engendrer toutes les sous-séquences originelles de chacun des réseaux qui produisent le mot vide. ♢

Proposition 40 (Analyse d'un langage de RdP) *L'appartenance d'un mot au langage d'un réseau de Petri et l'existence d'au moins un mot du langage d'un réseau de Petri sont décidables.*

Preuve
L'existence d'un mot au langage se réduit à vérifier si l'un des marquages terminaux est accessible. Pour vérifier si un mot appartient au langage d'un réseau de Petri, il suffit de construire un deuxième réseau qui reconnait uniquement ce mot, puis le réseau qui reconnait l'intersection des deux réseaux et vérifier que son langage n'est pas vide. ♢

Proposition 41 (Situation de la famille des langages de RdP) *Les langages des réseaux de Petri contiennent les langages réguliers et sont incomparables avec les langages algébriques [JAN 79].*

Preuve
Pour simuler une grammaire régulière par un réseau de Petri, il suffit d'associer à chaque symbole non terminal une place et à chaque règle de production

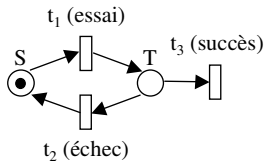

Figure 10: Simulation d'une grammaire régulière

une transition étiquetée par le symbole terminal et dont la précondition est le symbole du membre gauche de la règle et la postcondition est symbole non terminal du membre droit de la règle s'il existe. La place du symbole initial contient un jeton et le marquage terminal est le marquage nul. Une traduction de la grammaire donnée dans l'exemple 1 est montrée sur la figure 10.

Pour ce qui est de l'incomparabilité avec les langages algébriques, d'une part le langage du réseau de la figure 2 (le planteur de bananes) dont le marquage final est le marquage nul ($\{a^n.b^n.c^n \mid n \geq 0\}$) n'est pas un langage algébrique (en vertu du lemme d'Ogden [AUT 87]). D'autre part, on démontre que le langage des palindromes n'est pas un langage de réseau de Petri. ♢

Proposition 42 (Situation d'un langage) *Etant donné un langage de réseau de Petri,*

- *Si la fonction d'étiquetage est l'identité et tout marquage est un marquage terminal, on peut décider de son appartenance aux langages réguliers [VAL 81] et aux langages algébriques [SCH 92].*
- *Si de plus un ensemble de marquages terminaux est spécifié, alors l'appartenance aux langages réguliers reste décidable [LAM 92].*

Du point de vue de la position des réseaux de Petri par rapport à la hiérarchie usuelle des langages, le modèle des réseaux de Petri récursifs constitue une sorte d'unification des modèles des réseaux de Petri et des grammaires algébriques. En effet, la famille des langages des réseaux récursifs contient strictement l'union des langages de réseaux de Petri et des langages algébriques et à l'instar de ces familles on peut décider l'appartenance d'un mot au langage et l'existence d'au moins un mot du langage d'un réseau récursif. Cependant contrairement aux deux familles précédentes, l'intersection d'un langage de réseau récursif avec un langage régulier n'est pas nécessairement un langage de réseau récursif. Ce dernier point a des conséquences qui seront discutées au premier chapitre du deuxième tome dans le cadre de la vérification de propriétés spécifiques.

Références

[ABD 96] P.A. Abdulla, K. Čerāns, Jonsson B. et Yih-Kuen T. General decidability theorems for infinite-state systems. In *Proc.* 11^{th} *IEEE Symp. Logic In Computer Science (LICS'96)*, LNCS, pages 313–321, New Brunswick, NJ, USA, Juillet 1996.

[AHO 74] Alfred V. Aho, John E. Hopcroft et J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, 1974.

[ARN 76] A. Arnold et M. Latteux. Vector addition systems and semi-Dick language. Rapport de Recherche 78, Laboratoire de Calcul, Université des Sciences et Techniques de Lille, Décembre 1976.

[AUT 87] J.M. Autebert. *Langages algébriques.* Etudes et recherches en informatique. Masson, 1987.

[DUF 98a] C. Dufourd. Réseaux de Petri avec reset/transfert : Décidabilité et indécidabilité. Thèse d'Université de l'ENS de Cachan. Laboratoire Spécification et Vérification, Octobre 1998.

[DUF 98b] C. Dufourd, A. Finkel et P. Schnoebelen. Reset nets between decidability and undecidability. *(ICALP'98) L.N.C.S*, 1443 :103–115, Juillet 1998.

[ESP 94] J. Esparza et M. Nielsen. Decidability issues for Petri nets - A survey. *Bulletin of the EATCS*, 52 :245–262, 1994.

[ESP 98] J. Esparza. Decidability and complexity of Petri net problems - an introduction. In *Lectures on Petri Nets I : Basic Models*, volume 1491 of *LNCS*, pages 374–428. Springer Verlag, 1998.

[FIN 98] A. Finkel et P. Schnoebelen. Fundamental structures in well-structured infinite transition. In *Proc.* 3^{rd} *Latin American Theoretical Informatics Symposium (LATIN'98)*, volume 1380 of *LNCS*, pages 102–118, Campinas, Brésil, Avril 1998.

[FRU 86] D. Frutos Escrig. Decidability of home states in place transition systems. Rapport interne, Dpto. Informatica y Automatica., Univ. Complutense de Madrid, 1986.

[FRU 89] D. Frutos Escrig et C. Johnen. Decidability of home space property. Rapport LRI-503, Laboratoire de Recherche en Informatique, Univ. de Paris-Sud, Orsay, 1989.

[HAC 75] M. Hack. *Decidability Questions for Petri Nets.* PhD thesis, M.I.T., Cambridge, MA, Décembre 1975. publié comme rapport technique 161, Lab. for Computer Science, Juin 1976.

[HAD 99a] S. Haddad et D. Poitrenaud. Decidability and undecidability results for recursive Petri nets. Rapport de Recherche 019, LIP6, Paris VI University, Paris, France, Septembre 1999.

[HAD 99b] S. Haddad et D. Poitrenaud. Theoretical aspects of recursive Petri nets. In *Proc.* 20^th *Int. Conf. on Applications and Theory of Petri nets*, volume 1639 of *Lecture Notes in Computer Science*, pages 228–247, Williamsburg, VA, USA, Juin 1999. Springer Verlag.

[HAD 00] S. Haddad et D. Poitrenaud. Modelling and analyzing systems with recursive Petri nets. In *Proc. of the Workshop on Discrete Event Systems - Analysis and Control*, pages 449–458, Gand, Belgique, August 2000. Kluwer Academics Publishers.

[HAU 90] D. Hauschildt. Semilinearity of the reachability set is decidable for Petri nets. Rapport FBI-HH-B-146/90, Université de Hambourg, 1990.

[HOP 69] J. E. Hopcroft et J. D. Ullman. *Formal Languages and their Relation to Automata*. Addison-Wesley, Reading, 1969.

[HOP 79] J. Hopcroft et J-J. Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 5 :135–159, 1979.

[JAN 79] M. Jantzen. On the hierarchy of Petri net languages. *R.A.I.R.O. Informatique Theorique*, 13 :19–30, 1979.

[KAR 69] R.M. Karp et R.E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2) :147–195, 1969.

[KOS 82] S.R. Kosaraju. Decidability of reachability in vector addition systems. In *Proc.* 14^{th} *ACM Symp. Theory of Computing (STOC'82)*, pages 267–281, San Franciscp, CA, Mai 1982.

[LAM 92] J.L. Lambert. A structure to decide reachability in Petri nets. *Theoretical Computer Science*, 99 :79–104, 1992.

[LIP 76] R. Lipton. The reachability problem requires exponential space. Technical Report 62, Department of Computer Science, Yale University, January 1976.

[MAY 81] E. Mayr et A. Meyer. The complexity of the finite containment problem for Petri nets. *Journ. Assoc. Comput. Mach.*, 28 :561–576, 1981.

[MAY 84] E.W. Mayr. An algorithm for the general Petri net reachability problem. *SIAM Journal of Computing*, 13 :441–460, 1984.

[PAP 94] C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, Mass., 1994.

[PET 81] J. L. Peterson. *Petri Net Theory and the Modelling of Systems*. Prentice Hall, 1981.

[RAC 78] C. Rackoff. The covering and boudedness problems for vector addition systems. *Theoretical Computer Science*, 6(2) :223–231, 1978.

[REI 95] K. Reinhardt. Reachability in Petri nets with inhibitor arcs. Manuscript non publié. accessible via www-fs.informatik.uni-tuebingen.de/∼reinhard, 1995.

[REU 89] C. Reutenauer. *Aspects mathématiques des réseaux de Petri*. Etudes et recherches en informatique. Masson, 1989.

[SCH 92] S. Schwer. The context-freeness of the languages associated with vector addition systems is decidable. *Theoretical Computer Science*, 98 :199–247, 1992.

[VAL 78] R. Valk. Self-modifying nets, a natural extension of Petri nets. *(ICALP'78) L.N.C.S*, 62 :464–476, Juillet 1978.

[VAL 81] R. Valk et G. Vidal-Naquet. Petri nets and regular languages. *Journal of Computer and System Sciences*, 3(23) :299–325, 1981.

[VAL 85] R. Valk et M. Jantzen. The residue of vectors sets with applications to decidability problems in Petri nets. *Acta Informatica*, 21 :643–674, 1985.

[YEN 92] H-C. Yen. A unified approach for deciding the existence of certain Petri net paths. *Information and Computation*, 96 :119–137, 1992.