Le problème de couverture pour les réseaux de Petri: résultats classiques et développements récents

Pierre GANTY — **Gilles** GEERAERTS — **Jean-François** RASKIN — **Laurent** VAN BEGIN

Département d'Informatique – Université Libre de Bruxelles – CPI 212, Campus de la Plaine, Boulevard du Triomphe, B-1050 Bruxelles, Belgique

RÉSUMÉ. *Les réseaux de Petri sont une classe bien étudiée de modèles propres à représenter les comportements de systèmes informatiques* concurrents. *Dans ce cadre, le* problème de couverture *(étant donné un marquage* $\mathbf{m}$*, existe-t-il un marquage accessible qui assigne à chaque place du réseau au moins autant de jetons que* $\mathbf{m}$ *?) est de tout premier intérêt, car de nombreuses* propriétés de sûreté *(toutes les exécutions du système restent-elles dans un ensemble de* bons états *?) peuvent y être ramenées. De nombreux travaux ont récemment été consacrés à l'étude d'algorithmes efficaces pour résoudre le problème de couverture. Dans cet article, nous survolons plusieurs de ces travaux classiques ou récents: l'algorithme de Karp et Miller et un algorithme efficace en pratique, qui résolvent le problème de couverture en calculant un* ensemble de couverture *du réseau; l'approche « en arrière » d'Abdulla et al.; et l'approche « en avant » Expand, Enlarge and Check. Finalement, nous terminons l'article en présentant une* nouvelle *approche combinant les forces des méthodes « en avant » et « en arrière ».*

ABSTRACT. *Petri nets are a widespread class of models to represent behaviours of* concurrent systems. *An important problem on Petri nets is the coverability problem (does there exist a reachable marking that assigns to each place at least as many tokens as a given marking* $\mathbf{m}$ *?), mainly because many* safety properties *(does the system always stay inside a set of* good behaviours *?) can be reduced to it. Many recent works have been devoted to the study of efficient algorithms to decide the coverability problem. In this paper, we survey several of these classical or recent works: the Karp and Miller algorithm and an efficient algorithm, that both solve the coverability problem by computing the* coverability set *of the net; the "backward" approach of Abdulla et al.; and the Expand, Enlarge and Check "forward" approach. We close the paper by discussing a* new *method combining the strengths of the forward and backward methods.*

MOTS-CLÉS : *Réseaux de Petri, problème de couverture*

KEYWORDS: *Petri nets, coverability problem*

1. Introduction

Depuis leur introduction dans les années 60, les réseaux de Petri ont attiré l'attention de beaucoup de chercheurs en informatique. Parmi eux, nombreux se sont intéressés aux aspects pratiques liés à la modélisation de systèmes à l'aide des réseaux de Petri, à la sémantique de ces réseaux et leur relation avec des algèbres de processus, à leur analyse et leur vérification à l'aide d'algorithmes, *etc*. En effet, les réseaux de Petri sont un formalisme bien adapté à la modélisation de systèmes concurrents. Par exemple, les réseaux de Petri non bornés permettent en particulier de modéliser des systèmes concurrents paramétriques où le nombre de processus est non borné (German *et al.*, 1992; Van Begin, 2003).

Dans cet article, nous nous intéressons à la vérification algorithmique de réseaux de Petri non bornés. Plus précisément, nous nous intéressons au problème de *couverture* : étant donné un réseau de Petri, un marquage initial $\mathbf{m}_0$ et un marquage $\mathbf{m}$, le problème de couverture consiste à déterminer s'il existe dans le réseau une exécution qui démarre en $\mathbf{m}_0$ et permet d'atteindre un marquage $\mathbf{m}'$ tel que $\mathbf{m} \preccurlyeq \mathbf{m}'$, c'est-à-dire un marquage $\mathbf{m}'$ qui contient dans chacune des places du réseau au moins autant de jetons que le marquage $\mathbf{m}$. Ce problème est important car plusieurs problèmes de vérification, comme par exemple le problème d'*inclusion de traces finies*[1], peuvent être réduits à ce problème.

Dès 1969, Karp et Miller ont proposé un algorithme qui résout le problème de couverture (Karp *et al.*, 1969). Leur algorithme résout un problème plus général, car il calcule une sur-approximation des marquages accessibles dans un réseau de Petri à partir d'un marquage initial, appelée *ensemble recouvrant* (Finkel, 1990). Cette sur-approximation est suffisamment précise pour décider le problème de couverture. Nous rappelons brièvement cet algorithme dans cet article mais nous présentons également quatre autres algorithmes, plus récents, qui permettent de résoudre le problème de couverture.

Le premier algorithme, présenté dans la Section 4, est une alternative à l'algorithme de Karp et Miller pour le calcul de l'ensemble recouvrant. Contrairement à l'algorithme de Karp et Miller, cet algorithme ne construit pas un arbre mais calcule le plus petit point fixe d'une fonction monotone sur les ensembles de paires de marquages. Des propriétés de cette fonction sont utilisées pour limiter les calculs à des paires de marquages maximales pour un ordre bien choisi. Cette propriété permet d'obtenir un algorithme qui est, en général, beaucoup plus efficace que l'algorithme originel de Karp et Miller.

Contrairement à l'algorithme de Karp et Miller, les trois autres algorithmes, présentés dans les sections 5 à 7, ne calculent pas l'ensemble recouvrant mais calculent

1. Ce problème peut être défini comme suit. Étant donné une étiquette pour chaque transition du réseau de Petri, étant donné un automate fini qui définit un langage régulier sur ces étiquettes, déterminer si toutes les suites d'étiquettes qui correspondent à des exécutions du réseau de Petri sont incluses dans le langage régulier.

une information suffisante pour résoudre le problème de couverture. L'algorithme de la Section 5 est un algorithme qui calcule l'ensemble de tous les marquages pouvant accéder à un ensemble de marquages clos par le haut[2]. Cet algorithme « en arrière » a été proposé par Abdulla et al. (Abdulla *et al.*, 1996) et est applicable à une classe de systèmes appelés les *systèmes bien structurés* (Finkel *et al.*, 2001; Abdulla *et al.*, 1996)[3], cette classe de systèmes est très générale et contient les réseaux de Petri et leurs extensions monotones (voir p.ex. (Van Begin, 2003)).

Dans la Section 6, nous présentons un autre algorithme (Geeraerts, 2007; Geeraerts *et al.*, 2006) qui décide le problème de couverture. Cet algorithme est également applicable à la classe des systèmes bien structurés. Nous présentons toutefois ici une version spécialisée de cet algorithme pour la classe des réseaux de Petri. Contrairement à l'algorithme d'Abdulla *et al.*, cet algorithme a l'avantage important de réaliser une exploration « en avant » des marquages accessibles du réseau. Les approches basées sur une exploration « en avant » sont généralement plus efficaces en pratique que les approches « en arrière » (Henzinger *et al.*, 2003). Cet algorithme considère deux séquences d'approximations de l'ensemble des marquages accessibles du réseau de Petri : une première séquence représentant des *sous*-approximations de plus en plus précises et une deuxième séquence représentant des *sur*-approximations de plus en plus précises. Ces séquences sont utilisées respectivement pour détecter les instances positives (dans le cas des sous-approximations) et négatives (sur-approximations) du problème de couverture. Par ailleurs, ces séquences sont *complètes* pour ce problème dans le sens où, si la réponse au problème de couverture est positive (resp. négative), alors cet algorithme construira toujours en un temps fini une sous-approximation (sur-approximation) qui permet de l'établir.

Dans la Section 7, nous présentons un troisième algorithme qui décide le problème de couverture sans calculer l'ensemble recouvrant. Ce dernier algorithme est original et combine une exploration en avant et en arrière des marquages du réseau. Il considère également des approximations de plus en plus précises. Le calcul des approximations à l'étape i est fonction du manque de précision constaté à l'étape $i-1$. Cet algorithme peut-être vu comme une spécialisation d'un algorithme général de raffinements de domaines abstraits défini dans (Ganty, 2007; Cousot *et al.*, 2007).

Nous présentons ensuite, à la Section 8, plusieurs heuristiques qui permettent d'implémenter de façon efficace les algorithmes des sections 4 à 7. Ensuite, nous présentons à la Section 9 une comparaison expérimentale des algorithmes des sections 4 à 7. Nous terminons par la Section 10 qui passe en revue quelques travaux connexes.

Remerciements Les auteurs tiennent à exprimer leur gratitude envers les relecteurs anonymes, ainsi qu'envers le Prof. Raymond Devillers, pour leurs commentaires.

2. Un ensemble de marquages est clos par le haut si, quand il contient un marquage $\mathbf{m}$ il contient également tous les marquages $\mathbf{m}'$ qui assignent au moins autant de jetons que $\mathbf{m}$ dans chacune des places du réseau.
3. Une classe de systèmes similaire avait déjà été définie par Finkel en 1987 (Finkel, 1987; Finkel, 1990).

2. Préliminaires

Cette section rappelle les concepts principaux qui seront utilisés dans cet article.

2.1. *Réseaux de Petri*

Nous commençons par la définition de *réseau de Petri* (Reisig, 1986).

Définition 1 *Un réseau de Petri est un tuple $\mathcal{N} = \langle P, T \rangle$, tel que $P \cap T = \emptyset$ et où :*

– *P est un ensemble fini de* places,

– *T est un ensemble fini de* transitions. *Chaque transition t est un tuple $\langle I, O \rangle$, où :*

- *$I : P \mapsto \mathbb{N}$ est une fonction qui associe à chaque place un* poids d'entrée,
- *$O : P \mapsto \mathbb{N}$ est une fonction qui associe à chaque place un* poids de sortie.

Un exemple de réseau de Petri est donné à la Figure 1. Nous adoptons la convention graphique habituelle qui consiste à représenter un réseau par un graphe bipartite dans lequel les nœuds correspondant aux places sont des cercles et les nœuds correspondant aux transitions sont des rectangles noirs. On trace un arc de $p \in P$ à $t = \langle I, O \rangle \in T$ (resp. de $t = \langle I, O \rangle \in T$ à $p \in P$) pondéré par $I(p)$ ($O(p)$) ssi $I(p) \neq 0$ ($O(p) \neq 0$). Les pondérations valant 1 sont en général omises.

Afin de définir la sémantique des réseaux de Petri, nous devons définir la notion de *marquage* :

Définition 2 *Étant donné un ensemble de places P, un marquage $\mathbf{m}$ (sur les places de l'ensemble P) est une fonction $\mathbf{m} : P \mapsto \mathbb{N}$, qui associe $\mathbf{m}(p)$ jetons à chaque place $p \in P$.*

Étant donné un réseau $\mathcal{N} = \langle P, T \rangle$, on appelle *marquage de $\mathcal{N}$* un marquage dont le domaine est P. La convention graphique veut qu'un marquage $\mathbf{m}$ du réseau est représenté graphiquement en dessinant $\mathbf{m}(p)$ jetons noirs dans chaque place p. Le marquage $\mathbf{m}$ tel que $\mathbf{m}(p_2) = 1$ et $\mathbf{m}(p_1) = \mathbf{m}(p_3) = 0$ est représenté à la Figure 1.

Dans la suite de l'article, on suppose généralement que les places $p_1, p_2, \ldots, p_k$ du réseau sont ordonnées. Dans ce cas, un marquage $\mathbf{m}$ peut être vu comme le vecteur $\langle \mathbf{m}(p_1), \mathbf{m}(p_2), \ldots, \mathbf{m}(p_k) \rangle$.

Un *réseau de Petri initialisé* est un réseau de Petri auquel on a associé un marquage initial $\mathbf{m}_0$. Formellement, un réseau de Petri initialisé est un tuple $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$ où $\langle P, T \rangle$ est un réseau de Petri, et $\mathbf{m}_0$ est un marquage. Dans la suite, nous ne considérerons que des réseaux de Petri initialisés, que nous appellerons simplement *réseaux de Petri*.

Étant donné un marquage $\mathbf{m}$ et une transition $t = \langle I, O \rangle$ d'un réseau de Petri $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$, on dit que t est *tirable* en $\mathbf{m}$ si, pour tout $p \in P$: $\mathbf{m}(p) \geq I(p)$.

Dans ce cas, et dans ce cas seulement, t peut être *tirée*, ce qui transforme le marquage $\mathbf{m}$ en $\mathbf{m}'$, tel que, pour tout $p \in P$: $\mathbf{m}'(p) = \mathbf{m}(p) - I(p) + O(p)$. Ceci est noté $\mathbf{m} \xrightarrow{t} \mathbf{m}'$. Remarquons que les transitions des réseaux de Petri ont un *effet constant*, qui ne dépend pas du marquage à partir duquel les transitions sont tirées : $\mathbf{m}'(p) - \mathbf{m}(p) = O(p) - I(p)$ for all $p \in P$.

Nous adoptons les notations suivantes : $\mathbf{m} \rightarrow \mathbf{m}'$ signifie qu'il existe $t \in T$ telle que $\mathbf{m} \xrightarrow{t} \mathbf{m}'$; et $\xrightarrow{*}$ est la fermeture reflexo-transitive de $\rightarrow$, c'est-à-dire que $\mathbf{m} \xrightarrow{*} \mathbf{m}'$ si et seulement si $\mathbf{m} = \mathbf{m}'$ ou s'il existe une séquence de marquages $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_n$ (avec $n \geq 2$) telle que : $\mathbf{m} = \mathbf{m}_1 \rightarrow \mathbf{m}_2 \rightarrow \cdots \rightarrow \mathbf{m}_n = \mathbf{m}'$. Nous pouvons maintenant introduire les définitions suivantes qui seront utiles tout au long de l'article :

Définition 3 *Étant donné un marquage* $\mathbf{m}$ *et un réseau de Petri* $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$ *:*

– $\mathsf{Post}(\mathbf{m}) = \{\mathbf{m}' \mid \mathbf{m} \rightarrow \mathbf{m}'\}$ *est l'ensemble des successeurs en un pas de* $\mathbf{m}$*,*

– $\mathsf{Pre}(\mathbf{m}) = \{\mathbf{m}' \mid \mathbf{m}' \rightarrow \mathbf{m}\}$ *est l'ensemble des prédécesseurs en un pas de* $\mathbf{m}$*,*

– $\mathsf{Post}^*(\mathbf{m}) = \{\mathbf{m}' \mid \mathbf{m} \xrightarrow{*} \mathbf{m}'\}$ *est l'ensemble des successeurs de* $\mathbf{m}$ *en un nombre arbitraire de pas,*

– $\mathsf{Pre}^*(\mathbf{m}) = \{\mathbf{m}' \mid \mathbf{m}' \xrightarrow{*} \mathbf{m}\}$ *est l'ensemble des prédécesseurs de* $\mathbf{m}$ *en un nombre arbitraire de pas.*

Ces définitions sont étendues aux ensembles $S \subseteq \mathbb{N}^{|P|}$ *de marquages de la façon suivante :* $f(S) = \bigcup_{\mathbf{m} \in S} f(\mathbf{m})$ *où* $f \in \{\mathsf{Post}, \mathsf{Pre}, \mathsf{Post}^*, \mathsf{Pre}^*\}$.

Le réseau de Petri ci-contre a trois places, trois transitions, et son marquage initial est le marquage $\langle 0, 1, 0 \rangle$. Pour ce réseau, l'ensemble $\mathsf{Post}(\{\langle 0, 1, 0 \rangle\})$ est le singleton $\{\langle 2, 1, 0 \rangle\}$, $\mathsf{Pre}(\{\langle 2, 1, 0 \rangle\})$ est l'ensemble $\{\langle 0, 1, 0 \rangle, \langle 1, 0, 1 \rangle\}$, $\mathsf{Post}^*(\{\langle 0, 1, 0 \rangle\})$ est l'ensemble $\{\langle 2i, 1, 0 \rangle \mid i \in \mathbb{N}\} \cup \{\langle 2i, 0, 1 \rangle \mid i \in \mathbb{N}\}$, et $\mathsf{Pre}^*(\{\langle 2, 1, 0 \rangle\}) = \{\langle 0, 1, 0 \rangle, \langle 1, 0, 1 \rangle, \langle 2, 1, 0 \rangle\}$.

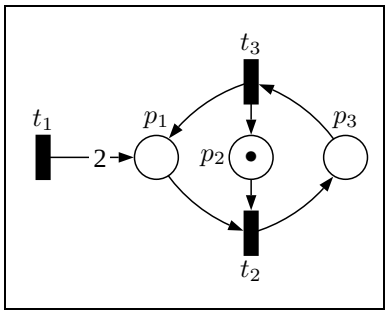


Figure 1. *Un exemple de réseau de Petri.*

2.2. *Ordre sur les marquages*

Dans la suite de cet article, nous utiliserons souvent la notion de *beau pré-ordre* :

Définition 4 *Un* beau pré-ordre $\sqsubseteq$ *sur un ensemble* E *est une relation réflexive et transitive (pré-ordre) sur* E *tel que, pour toute séquence infinie* $e_1, e_2, \ldots, e_i, \ldots$ *d'éléments de* E*, il existe* $1 \leq k < \ell$ *tels que* $e_k \sqsubseteq e_\ell$.

Dans cet article nous utiliserons des beaux pré-ordres qui sont aussi des ordres partiels (c'est-à-dire qu'ils sont aussi antisymétriques). Dans le cas des marquages des réseaux de Petri, on utilise l'ordre $\preccurlyeq$:

Définition 5 *Étant donné un ensemble de places* P*, l'ordre partiel* $\preccurlyeq \subseteq \mathbb{N}^{|P|} \times \mathbb{N}^{|P|}$ *est tel que pour toute paire de marquages* $\mathbf{m}_1$, $\mathbf{m}_2$ *on a que* $\mathbf{m}_1 \preccurlyeq \mathbf{m}_2$ *si et seulement si pour toute place* $p \in P$ *:* $\mathbf{m}_1(p) \leq \mathbf{m}_2(p)$.

Par la suite, la notation $\mathbf{m}_1 \prec \mathbf{m}_2$ sera utilisée dans le cas où $\mathbf{m}_1 \preccurlyeq \mathbf{m}_2$ et $\mathbf{m}_2 \not\preccurlyeq \mathbf{m}_1$. Il est bien établi que $\preccurlyeq$ est un beau pré-ordre (partiel) :

Lemme 1 ((Dickson, 1913)) $\preccurlyeq$ *est un beau pré-ordre.*

Maintenant que l'ordre sur les marquages $\preccurlyeq$ est défini, nous pouvons définir la propriété de monotonie stricte des réseaux de Petri :

Lemme 2 *Les réseaux de Petri sont* strictement monotones *pour l'ordre* $\preccurlyeq$*, c'est-à-dire que pour tout réseau de Petri* $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$*, pour toute transition* $t \in T$ *et pour tous marquages* $\mathbf{m}_1$, $\mathbf{m}_2$ *et* $\mathbf{m}_3$ *de* $\mathcal{N}$ *tels que* $\mathbf{m}_1 \prec \mathbf{m}_2$ *et* $\mathbf{m}_1 \xrightarrow{t} \mathbf{m}_3$*, il existe un marquage* $\mathbf{m}_4$ *de* $\mathcal{N}$ *tel que* $\mathbf{m}_2 \xrightarrow{t} \mathbf{m}_4$ *et* $\mathbf{m}_3 \prec \mathbf{m}_4$.

À partir de l'ordre $\preccurlyeq$, nous pouvons également définir deux classes d'ensembles de marquages qui seront particulièrement utiles dans la suite.

Définition 6 *Étant donné un ensemble* $G \subseteq \mathbb{N}^{|P|}$ *de marquages sur les places de* P*, sa* clôture par le haut *est l'ensemble* $\uparrow^{\preccurlyeq}(G) = \{\mathbf{m} \in \mathbb{N}^{|P|} \mid \exists \mathbf{m}' \in G : \mathbf{m}' \preccurlyeq \mathbf{m}\}$. *Symétriquement, la* clôture par le bas *de* G *est :* $\downarrow^{\preccurlyeq}(G) = \{\mathbf{m} \in \mathbb{N}^{|P|} \mid \exists \mathbf{m}' \in G : \mathbf{m} \preccurlyeq \mathbf{m}'\}$. *Un ensemble* E *de marquages est dit* clos par le haut *si et seulement si* $\uparrow^{\preccurlyeq}(E) = E$*. Il est* clos par le bas *si et seulement si* $\downarrow^{\preccurlyeq}(E) = E$.

Le lecteur trouvera des exemples de tels ensembles à la Figure 2. Quand l'ensemble à clore par le bas est un singleton $\{\mathbf{m}\}$, nous écrivons simplement $\downarrow^{\preccurlyeq}(\mathbf{m})$ au lieu de $\downarrow^{\preccurlyeq}(\{\mathbf{m}\})$. Nous procédons de même pour les ensembles à clore par le haut.

2.3. *Problème de couverture*

Les notions présentées ci-dessus nous permettent d'introduire le problème auquel nous nous intéressons dans cet article, appelé *problème de couverture* :

Les ensembles A et B sont deux ensembles clos par le bas infinis qui contiennent, respectivement, les marquages $\{(x,y) \in \mathbb{N}^2 \mid y \leq 1\}$ et $\{(x,y) \in \mathbb{N}^2 \mid x \leq 1\}$. L'ensemble C est un ensemble clos par le bas fini qui contient les marquages $\{(x,y) \in \mathbb{N}^2 \mid x \leq 2 \wedge y \leq 2\}$. L'ensemble D est l'ensemble clos par le haut $\{(x,y) \in \mathbb{N}^2 \mid x \geq 3 \wedge y \geq 2\}$.

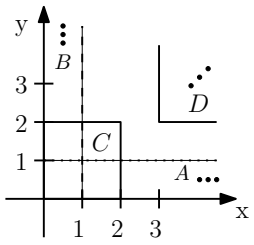


Figure 2. *Des ensembles clos par le haut et par le bas (pour $\preccurlyeq$) dans $\mathbb{N}^2$.*

Problème de couverture

Données Un réseau de Petri $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$ et un ensemble U de marquages de $\mathcal{N}$ qui est clos par le haut.

Question Est-ce que $\mathsf{Post}^*(\mathbf{m}_0) \cap U \neq \emptyset$?

Le problème de couverture consiste donc à déterminer si un certain ensemble U clos par le haut est accessible dans un réseau de Petri $\mathcal{N}$. La complexité au pire cas de ce problème est exponentielle en espace (Rackoff, 1978).

Exemple 1 *Considérons le réseau $\mathcal{N}$ de la Figure 1. Une instance du problème de couverture consiste à demander si $U = \{\mathbf{m} \mid \mathbf{m}(p_1) \geq 0 \wedge \mathbf{m}(p_2) \geq 0 \wedge \mathbf{m}(p_3) \geq 2\}$ est accessible dans $\mathcal{N}$ pour le marquage initial $\mathbf{m}_1 = \langle 0,1,0 \rangle$. Cette instance est négative. Une instance positive est constituée par le réseau $\mathcal{N}$, équippé du marquage initial $\mathbf{m}_2 = \langle 0,1,1 \rangle$ et de l'ensemble U.*

Notons que l'ensemble $\mathsf{Post}^*(\mathbf{m}_0)$ n'est pas représentable de manière effective. En effet, on ne peut pas construire une représentation de cet ensemble dans un formalisme pour lequel l'équivalence est décidable. Autrement, on pourrait décider l'équivalence des accessibles de deux réseaux de Petri, ce qui est impossible (Hack, 1976). Pour cette raison, aucun algorithme pour résoudre le problème de couverture ne se base sur la construction de cet ensemble. Cependant, une sur-approximation de cet ensemble, appelée *ensemble recouvrant*, a été intensivement étudiée. Cette définition est due à Alain Finkel (Finkel, 1987; Finkel, 1990) :

Définition 7 *Étant donné un réseau de Petri $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$, son ensemble recouvrant* $\mathsf{Cover}(\mathcal{N})$ *est donné par* $\downarrow^{\preccurlyeq}(\mathsf{Post}^*(\mathbf{m}_0))$.

L'intérêt principal de l'ensemble recouvrant est donné par la proposition suivante :

Proposition 1 *Pour tout réseau de Petri $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$, et tout ensemble U clos par le haut de marquages de $\mathcal{N}$,* $\mathsf{Post}^*(\mathbf{m}_0) \cap U \neq \emptyset$ *ssi* $\mathsf{Cover}(\mathcal{N}) \cap U \neq \emptyset$.

Le second intérêt de l'ensemble recouvrant est qu'il est possible d'en construire une *représentation finie* que l'on peut *manipuler algorithmiquement*, comme expliqué

dans la fin de cette section. Avant cela, expliquons comment représenter et manipuler les ensembles clos par le haut et par le bas.

Remarquons que, si l'ensemble recouvrant permet de résoudre le problème de couverture, sa construction est plus difficile que le problème de couverture, car elle est non-primitive récursive (Valk *et al.*, 1981).

2.4. *Représentation et manipulation d'ensembles clos par le haut et par le bas*

Les ensembles de marquages clos par le haut et par le bas sont en général infinis. Pour les représenter et les manipuler en pratique il nous faut donc une représentation effective, c'est-à-dire une représentation finie qui supporte les opérations nécessaires sur ces ensembles.

Remarquons que, dans cet article, les propriétés de correction des algorithmes qui manipulent des *ensembles clos par le haut* **ne** dépendent **pas** de la représentation choisie pour ces ensembles. C'est pourquoi, dans cet article, nous manipulons directement les ensembles clos par le haut en considérant implicitement que ceux-ci sont représentés à l'aide d'une structure de données effective. Un exemple de représentation effective est l'ensemble des éléments minimaux des ensembles clos par le haut :

Définition 8 *Étant donné un ensemble $S \subseteq \mathbb{N}^k$, l'ensemble des* éléments minimaux *de S, noté $\mathsf{Min}^{\preccurlyeq}(S)$, est donné par $\{\mathbf{m} \in S \mid \nexists \mathbf{m}' \in S : \mathbf{m}' \prec \mathbf{m}\}$.*

L'utilisation de $\mathsf{Min}^{\preccurlyeq}(U)$ pour représenter l'ensemble clos par le haut U est justifiée par le lemme suivant.

Lemme 3 *Pour tout ensemble clos par le haut $U \subseteq \mathbb{N}^k$, $\mathsf{Min}^{\preccurlyeq}(U)$ est un ensemble unique, fini, et tel que $\uparrow^{\preccurlyeq}\Big(\mathsf{Min}^{\preccurlyeq}(U)\Big) = U$.*

Afin de montrer que cette représentation est bien *effective*, nous rappelons brièvement les algorithmes qui implémentent les opérations et les tests sur les ensembles clos par le haut dont nous aurons besoin dans la suite.

L'opérateur Pre**.** Étant donné un réseau de Petri $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$ et $U \subseteq \mathbb{N}^{|P|}$ clos par le haut, l'ensemble $\mathsf{Pre}(U)$ est également clos par le haut (Abdulla *et al.*, 1996). Pour calculer $\mathsf{Min}^{\preccurlyeq}(\mathsf{Pre}(U))$ à partir de $\mathsf{Min}^{\preccurlyeq}(U)$, nous avons besoin des définitions suivantes. Étant donné un marquage $\mathbf{m}$ et une transition $t = \langle I, O \rangle \in T$, le marquage $\mathsf{Minpre}_t(\mathbf{m})$ est tel que pour toute place p : $\mathsf{Minpre}_t(\mathbf{m})(p) = \max\{\mathbf{m}(p) - O(p) + I(p), I(p)\}$. L'ensemble $\mathsf{Minpre}(\mathbf{m})$ est défini comme étant $\bigcup_{t \in T}\{\mathsf{Minpre}_t(\mathbf{m})\}$. L'opérateur Minpre est étendu aux ensembles finis de marquages M de la façon suivante : $\mathsf{Minpre}(M) = \mathsf{Min}^{\preccurlyeq}\left(\bigcup_{\mathbf{m} \in M} \mathsf{Minpre}(\mathbf{m})\right)$. Il est alors facile de voir que (Abdulla *et al.*, 1996) : $\mathsf{Minpre}(\mathsf{Min}^{\preccurlyeq}(U)) = \mathsf{Min}^{\preccurlyeq}(\mathsf{Pre}(U))$.

L'union. L'union de deux ensembles clos par le haut est également close par le haut. De plus, étant donné deux ensembles clos par le haut de marquages U_1 et

U_2 représentés respectivement par $\mathsf{Min}^{\preccurlyeq}(U_1)$ et $\mathsf{Min}^{\preccurlyeq}(U_2)$: $\mathsf{Min}^{\preccurlyeq}(U_1 \cup U_2) = \mathsf{Min}^{\preccurlyeq}\Big(\mathsf{Min}^{\preccurlyeq}(U_1) \cup \mathsf{Min}^{\preccurlyeq}(U_2)\Big)$.

Le test d'appartenance et le test d'inclusion.

Étant donné deux ensembles clos par le haut U_1,U_2 et un marquage $\mathbf{m}$:

$$\mathbf{m} \in U_1 \text{ ssi } \exists \mathbf{m}' \in \mathsf{Min}^{\preccurlyeq}(U_1) : \mathbf{m}' \preccurlyeq \mathbf{m}$$

et

$$U_1 \subseteq U_2 \text{ ssi } \forall \mathbf{m}_1 \in \mathsf{Min}^{\preccurlyeq}(U_1)\, \exists \mathbf{m}_2 \in \mathsf{Min}^{\preccurlyeq}(U_2) : \mathbf{m}_2 \preccurlyeq \mathbf{m}_1 \ .$$

Contrairement aux algorithmes manipulant des ensembles clos par le haut, les algorithmes qui manipulent des ensembles *clos par le bas* que nous présentons dans la suite, feront *directement référence* à une *structure de données* particulière, appelée *ω-marquages*, pour représenter ces ensembles[4]. Les ω-marquages constituent une généralisation des marquages définie comme suit :

Définition 9 *Étant donné un réseau de Petri $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$, un ω-marquage de $\mathcal{N}$ est une fonction $\mathbf{m} : P \mapsto \mathbb{N} \cup \{\omega\}$.*

Comme dans le cas des marquages, nous représenterons souvent un ω-marquage par le vecteur correspondant. Intuitivement, la valeur ω représente « un nombre quelconque de jetons ». Pour formaliser cette idée, nous étendons $\leq$ à $\mathbb{N} \cup \{\omega\}$ de la façon suivante : pour toute constante $c \in \mathbb{N} : c < \omega$. Ceci permet de généraliser les définitions de $\preccurlyeq$ et de la clôture par le bas aux ω-marquages. Étant donné deux ω-marquages $\mathbf{m}_1, \mathbf{m}_2$ sur l'ensemble de places P, $\mathbf{m}_1 \preccurlyeq \mathbf{m}_2$ ssi $\forall p \in P : \mathbf{m}_1(p) \leq \mathbf{m}_2(p)$. Étant donné un ensemble G d'ω-marquages, $\downarrow^{\preccurlyeq}(G) = \{\mathbf{m} \in \mathbb{N}^{|P|} \mid \exists \mathbf{m}_1 \in G \colon \mathbf{m} \preccurlyeq \mathbf{m}_1\}$.

L'idée sous-jacente aux ω-marquages est similaire à celle des ensembles minimaux pour représenter les ensembles clos par le haut de manière effective.

Lemme 4 ((Geeraerts *et al.*, 2006)) *Pour tout ensemble clos par le bas E de marquages, il existe un ensemble fini G_E d'ω-marquages tel que $\downarrow^{\preccurlyeq}(G_E) = E$.*

Un ensemble G_E dans le Lemme 4 est appelé un *ensemble de couverture (de E)*. Dans la suite, nous manipulerons régulièrement un ensemble de couverture de l'*ensemble recouvrant* d'un réseau de Petri $\mathcal{N}$. Par abus de langages, nous appellerons cet ensemble « un ensemble de couverture de $\mathcal{N}$ ». Un ensemble de couverture de $\mathcal{N}$ est donc un ensemble de couverture de $\mathsf{Cover}(\mathcal{N})$.

4. Remarquons qu'il est possible de représenter un ensemble clos par le bas à l'aide de son complément, qui est clos par le haut. Néanmoins, la représentation sur base des ω-marquages est plus compacte et permet d'obtenir des algorithmes plus simples (Ganty *et al.*, 2006).

Exemple 2 *Les ensembles clos par le bas A, B et C de la Figure 2 peuvent être représentés, respectivement, par les ensembles de couverture $\{\langle\omega, 1\rangle\}$, $\{\langle 1, \omega\rangle\}$ et $\{\langle 2, 2\rangle\}$. L'ensemble $\mathsf{Min}^{\preccurlyeq}(D) = \{\langle 3, 2\rangle\}$ représente de façon univoque l'ensemble clos par le haut D.*

Afin de pouvoir manipuler les ω-marquages, nous devons étendre certaines opérations arithmétiques sur $\mathbb{N} \cup \{\omega\}$. Pour ce faire, nous fixons $\omega \bullet c = \omega$ pour tout $c \in \mathbb{N}$ et $\bullet \in \{+, -\}$. Étant donné un réseau de Petri $\langle P, T, \mathbf{m}_0\rangle$ et un ω-marquage $\mathbf{m}$, on dit que $t = \langle I, O\rangle \in T$ est tirable en $\mathbf{m}$ si pour toute place $p \in P : \mathbf{m}(p) \geq I(p)$. Dans ce cas, t peut être tirée et transforme l'ω-marquage $\mathbf{m}$ en l'ω-marquage $\mathbf{m}'$ tel que pour toute place $p \in P : \mathbf{m}'(p) = \mathbf{m}(p) - I(p) + O(p)$. Les notations $\mathbf{m} \xrightarrow{t} \mathbf{m}'$, $\mathbf{m} \rightarrow \mathbf{m}'$ ainsi que les définitions des ensembles $\mathsf{Post}(\mathbf{m})$ et $\mathsf{Post}^*(\mathbf{m})$ sont étendues de façon directe aux ω-marquages.

Comme dans le cas des ensembles clos par le haut, il est possible de calculer certaines opérations ensemblistes sur les ensembles clos par le bas en manipulant directement leurs représentations en termes d'ω-marquages. En particulier, les opérations qui seront utiles par la suite sont :

L'inclusion. Étant donné deux ensembles D_1 et D_2 clos par le bas représentés par les ensembles M_1 et M_2 d'ω-marquages, on peut déterminer si $D_1 \subseteq D_2$ de la façon suivante : $D_1 \subseteq D_2$ ssi $\forall \mathbf{m}_1 \in M_1\, \exists \mathbf{m}_2 \in M_2 : \mathbf{m}_1 \preccurlyeq \mathbf{m}_2$.

L'intersection avec un ensemble clos par le haut. Étant donné un ensemble D clos par le bas, représenté par M_D, et un ensemble U clos par le haut, représenté par $\mathsf{Min}^{\preccurlyeq}(U)$, il est possible de déterminer s'il existe une intersection non-vide entre D et $U : D \cap U \neq \emptyset$ ssi $\exists \mathbf{m}_D \in M_D\, \exists \mathbf{m}_U \in \mathsf{Min}^{\preccurlyeq}(U) : \mathbf{m}_U \preccurlyeq \mathbf{m}_D$.

3. L'algorithme de Karp et Miller

L'algorithme de Karp et Miller est un algorithme classique pour calculer un ensemble de couverture d'un réseau de Petri. Cet algorithme a été introduit par Karp et Miller en 1969 (Karp *et al.*, 1969) (voir aussi l'ouvrage de Reisig (Reisig, 1986) pour une présentation plus classique). Nous en donnons ici une brève présentation.

Les deux ingrédients principaux de l'algorithme de Karp et Miller sont : la construction d'un arbre d'accessibilité, d'une part, et une fonction d'accélération qui permet de calculer et représenter de façon finie un ensemble infini de marquages résultant d'un ensemble infini d'exécutions, d'autre part.

L'algorithme de Karp et Miller construit un arbre étiqueté $\langle N, B, racine, \Lambda\rangle$ dont les nœuds sont étiquetés par des ω-marquages. N est l'ensemble des nœuds de l'arbre ; $racine \in N$ est la racine de l'arbre ; $B \subseteq N \times N$ est l'ensemble des arêtes de l'arbre, et B^+ et B^* dénotent, respectivement, la fermeture transition et réflexo-transitive de B ; et $\Lambda : N \mapsto (\mathbb{N} \cup \{\omega\})^{|P|}$ est la fonction d'étiquetage des nœuds de l'arbre. Cette fonction est étendue aux ensembles $S \subseteq N : \Lambda(S) = \{\Lambda(s) \mid s \in S\}$.

La *racine* de l'arbre construit par l'algorithme de Karp et Miller est étiquetée par le marquage initial $\mathbf{m}_0$ du réseau considéré. Les *fils* d'un nœud n de l'arbre sont construits de la façon suivante. S'il existe, sur la branche menant de la racine à n, un autre nœud n' tel que $\Lambda(n) = \Lambda(n')$, n n'a pas de fils. Sinon, on calcule d'abord l'ensemble $\mathsf{Post}(\Lambda(n))$ et pour chaque $\mathbf{m} \in \mathsf{Post}(\Lambda(n))$, l'algorithme construit un marquage $\mathbf{m}_\omega$ par le biais d'une *fonction d'accélération*. Celle-ci initialise $\mathbf{m}_\omega$ à $\mathbf{m}$. Ensuite, elle remplace certaines coordonnées de $\mathbf{m}_\omega$ par ω : on met à ω toutes les coordonnées p pour lesquelles on peut trouver, sur la branche allant de la racine à n, un nœud $\overline{n}$ tel que $\Lambda(\overline{n}) \prec \mathbf{m}$ et $\Lambda(\overline{n})(p) < \mathbf{m}(p)$. Finalement, on crée dans l'arbre un nouveau nœud étiqueté par $\mathbf{m}_\omega$ comme successeur à n.

Un exemple d'arbre de Karp et Miller est donné à la Figure 3 : il s'agit de l'arbre de Karp et Miller du réseau de la Figure 1.

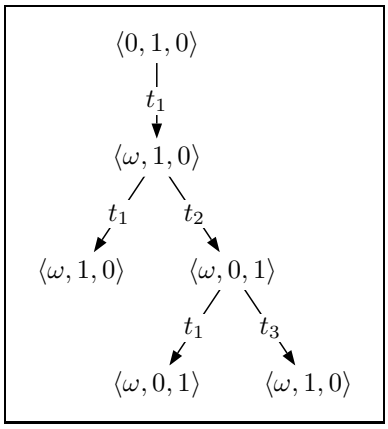


Figure 3. *L'arbre de Karp et Miller du réseau de Petri de la Figure 1. On a étiqueté les branches par le nom de la transition correspondante.*

La fonction d'accélération décrite ci-dessus est fournie à l'Algorithme 1. Elle reçoit le marquage $\mathbf{m}$ à accélérer et l'ensemble S des nœuds présents sur la branche menant à n. Elle renvoie le marquage $\mathbf{m}_\omega$ résultat de l'accélération. L'algorithme de Karp et Miller est, quant à lui, donné à l'Algorithme 2. Il suit la description que nous avons donnée ci-dessus. La terminaison et la correction de cet algorithme ont été établies par Karp et Miller :

Théorème 1 ((Karp *et al.*, 1969)) *Pour tout réseau de Petri $\mathcal{N}$, l'Algorithme 2 a une exécution finie. À la terminaison, $\{\Lambda(n) \mid n \in N\}$ forme un ensemble de couverture de $\mathcal{N}$, c'est-à-dire : $\downarrow^{\preccurlyeq}(\Lambda(N)) = \mathsf{Cover}(\mathcal{N})$.*

La correction de la fonction d'accélération peut être justifiée par la propriété de monotonie stricte des réseaux de Petri. En effet, considérons un nœud n dont on calcule les descendants. Pour ce faire, on considère $\mathbf{m} \in \mathsf{Post}(\Lambda(n))$ que la fonction

Données : Un ensemble fini S d'ω-marquages et un ω-marquage $\mathbf{m}$
Résultat : Un ω-marquage, résultat de l'accélération de $\mathbf{m}$
Acceleration $(S, \mathbf{m})$
$\mathbf{m}_\omega := \mathbf{m}$;
pour chaque $\mathbf{m}' \in S$ *t.q.* $\mathbf{m}' \prec \mathbf{m}$ **faire**
  pour chaque *place* p *t.q.* $\mathbf{m}'(p) < \mathbf{m}(p)$ **faire**
    $\mathbf{m}_\omega(p) := \omega$;
retourner $\mathbf{m}_\omega$;

Algorithme 1 : La fonction d'accélération de Karp et Miller.

Données : Un réseau de Petri $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$
Résultat : Un sous-ensemble D fini de $(\mathbb{N} \cup \omega)^{|P|}$ tel que $\downarrow^{\preccurlyeq}(D) = \mathsf{Cover}(\mathcal{N})$.
$\mathcal{T} := \langle N, B, n_0, \Lambda \rangle$ tel que $N = \{n_0\}$, $B = \emptyset$ et $\Lambda(n_0) = \mathbf{m}_0$;
$en\_attente := \{n_0\}$;
tant que $en\_attente \neq \emptyset$ **faire**
  Soit n un nœud de $en\_attente$;
  $en\_attente := en\_attente \setminus \{n\}$;
  si $\nexists \overline{n} \colon B^+(\overline{n}, n) \wedge \Lambda(\overline{n}) = \Lambda(n)$ **alors**
    pour chaque $\mathbf{m} \in \mathsf{Post}(\Lambda(n))$ **faire**
      $S := \{\Lambda(n') \mid \exists n' \in N \colon B^*(n', n)\}$;
      Soit n' un nouveau nœud tel que $\Lambda(n') = \mathsf{Acceleration}(S, \mathbf{m})$;
      $N := N \cup \{n'\}$;
      $B := B \cup \{(n, n')\}$;
      $en\_attente := en\_attente \cup \{n'\}$;
retourner $\Lambda(N)$;

Algorithme 2 : L'algorithme de Karp et Miller pour calculer un ensemble de couverture.

d'accélération va transformer en $\mathbf{m}_\omega$ en ajoutant ω dans toutes les places $p \in P_a$, où $P_a = \{p \in P \mid \mathbf{m}_\omega(p) = \omega \neq \mathbf{m}(p)\}$ est supposé non-vide. Dans ce cas, il est possible de trouver une séquence ς de transitions qui est tirable à partir de $\mathbf{m}$ et qui augmente strictement le nombre de jetons dans toutes les places appartenant à P_a, sans le faire décroître dans les autres. Plus formellement, ς et P_a sont tels que : $\mathbf{m} \xrightarrow{\varsigma} \mathbf{m}'$; $\forall p \in P_a \colon \mathbf{m}(p) < \mathbf{m}'(p)$ et $\forall p \notin P_a \colon \mathbf{m}(p) = \mathbf{m}'(p)$. De plus, comme les réseaux de Petri sont strictement monotones et $\mathbf{m} \prec \mathbf{m}'$, ς est à nouveau tirable à partir de $\mathbf{m}'$, et cela permet donc d'augmenter de façon arbitraire le nombre de jetons dans les places de P_a, ce qui justifie la construction de $\mathbf{m}_\omega$.

On voit donc que la fonction d'accélération ne construit que des ω-marquages $\mathbf{m}$ tels que $\downarrow^{\preccurlyeq}(\mathbf{m}) \subseteq \mathsf{Cover}(\mathcal{N})$. Par ailleurs, il est aisé de voir que, si un nœud n n'a pas de successeur et qu'une transition est tirable à partir de $\Lambda(n)$, alors il possède un ancêtre n' tel que $\Lambda(n) = \Lambda(n')$, qui a été développé. On est ainsi certain de ne

pas « oublier » de marquages accessibles en arrêtant le développement d'un nœud. Ces constatations permettent de se convaincre que l'algorithme de Karp et Miller maintient l'invariant suivant :

$$\downarrow^{\preccurlyeq}(\Lambda\,(N) \cup \mathsf{Post}^*\,(\Lambda\,(en\_attente))) = \mathsf{Cover}\,(\mathcal{N}) \quad .$$

Ainsi, quand la frontière $en\_attente$ est vide, on obtient $\downarrow^{\preccurlyeq}(\Lambda\,(N)) = \mathsf{Cover}\,(\mathcal{N})$, ce qui prouve la correction de l'algorithme.

La terminaison de l'algorithme peut, quant à elle, être prouvée par le raisonnement suivant. Si l'algorithme ne termine pas, il construit un arbre de taille infinie, ce qui implique qu'il contient une branche infinie (car l'arbre est à branchement fini). Comme $\preccurlyeq$ est un beau pré-ordre, on peut extraire des étiquettes des nœuds de cette branche soit une séquence infinie de marquages égaux, soit une séquence infinie strictement croissante de marquages. Or :

– Une séquence infinie de marquages égaux n'est pas possible car l'algorithme arrête une branche dès qu'il rencontre deux marquages égaux sur cette branche.

– Une séquence infinie strictement croissante signifierait qu'un nombre non-borné d'accélérations a eu lieu sur la branche. Or chaque accélération ajoute au moins un ω dans les ω-marquages de la branche. Comme il n'y a qu'un nombre fini de places, cela n'est pas possible.

4. Calcul efficace d'un ensemble de couverture

Si l'algorithme de Karp et Miller est une solution très élégante pour montrer qu'on peut toujours calculer un ensemble de couverture d'un réseau de Petri, elle est malheureusement inapplicable dans bien des cas pratiques. En effet, les arbres générés par cet algorithme sont en général trop grands pour permettre la terminaison de l'algorithme dans des délais raisonnables, et cela, même pour des réseaux de petite taille.

Le problème du calcul *efficace* de l'ensemble de couverture est non trivial. C'est ainsi qu'il a été montré récemment (Geeraerts, 2007; Geeraerts *et al.*, 2007) que deux optimisations de l'algorithme de Karp et Miller (Finkel, 1993; Luttge, 1995) sont erronées, dans le sens où les algorithmes qui y sont présentés pourraient, soit ne pas calculer un ensemble de couverture, soit ne pas terminer. Dans cette section, nous présentons brièvement un nouvel algorithme (Geeraerts, 2007; Geeraerts *et al.*, 2007) pour calculer l'ensemble de couverture d'un réseau de Petri. Cet algorithme n'est pas basé sur la solution de Karp et Miller, mais est plus efficace en pratique. Nous renvoyons le lecteur intéressé aux références pour les détails des preuves, qui sont omises ici.

L'algorithme que nous présentons dans cette section a la particularité de manipuler des *ensembles de paires d'ω-marquages*. Nous devons donc redéfinir les opérateurs de successeur et d'accélération sur les paires d'ω-marquages. C'est l'objet des définitions suivantes :

– Étant donné une paire $(\mathbf{m}_1, \mathbf{m}_2)$ d'ω-marquages, nous définissons la fonction $\overline{\mathsf{Post}}((\mathbf{m}_1, \mathbf{m}_2)) = \{(\mathbf{m}_1, \mathbf{m}'), (\mathbf{m}_2, \mathbf{m}') \mid \mathbf{m}' \in \mathsf{Post}(\mathbf{m}_2)\}$. Cette fonction est étendue aux *ensembles* R de paires de la façon suivante : $\overline{\mathsf{Post}}(R) = \bigcup_{(\mathbf{m}_1, \mathbf{m}_2) \in R} \overline{\mathsf{Post}}((\mathbf{m}_1, \mathbf{m}_2))$,

– Étant donné une paire $(\mathbf{m}_1, \mathbf{m}_2)$ d'ω-marquages telle que $\mathbf{m}_1 \prec \mathbf{m}_2$, nous définissons $\overline{\mathsf{Accel}}((\mathbf{m}_1, \mathbf{m}_2)) = \{(\mathbf{m}_2, \mathsf{Acceleration}(\{\mathbf{m}_1\}, \mathbf{m}_2))\}$. $\overline{\mathsf{Accel}}((\mathbf{m}_1, \mathbf{m}_2))$ n'est pas définie si $\mathbf{m}_1 \not\prec \mathbf{m}_2$. Cette fonction est également étendue aux *ensembles* R de paires de la façon suivante : $\overline{\mathsf{Accel}}(R) = \bigcup_{(\mathbf{m}_1, \mathbf{m}_2) \in R}^{\mathbf{m}_1 \prec \mathbf{m}_2} \overline{\mathsf{Accel}}(\mathbf{m}_1, \mathbf{m}_2)$,

– Finalement, étant donné un ensemble R de paires de marquages, nous définissons : $\mathsf{Flatten}(R) = \{\mathbf{m} \mid \exists \mathbf{m}' \colon (\mathbf{m}', \mathbf{m}) \in R\}$.

La fonction d'accélération $\overline{\mathsf{Accel}}$ est *plus faible* que la fonction d'accélération de Karp et Miller. En effet, $\overline{\mathsf{Accel}}((\mathbf{m}_1, \mathbf{m}_2))$ est définie (*cfr. supra*) comme étant la fonction $\mathsf{Acceleration}$ appliquée à $\mathbf{m}_2$ (le marquage à accélérer), et prenant $\{\mathbf{m}_1\}$ comme ensemble de marquages permettant l'accélération. Donc, pour accélérer un marquage ($\mathbf{m}_2$), $\overline{\mathsf{Accel}}$ ne prend en compte qu'un seul autre marquage ($\mathbf{m}_1$), alors que dans l'algorithme de Karp et Miller, $\mathsf{Acceleration}$ est appelée avec tous les ancêtres de $\mathbf{m}_2$.

Néanmoins, en étudiant attentivement l'algorithme de Karp et Miller, il est possible d'établir des propriétés qui permettent d'affirmer que la fonction $\overline{\mathsf{Accel}}$, quand elle est correctement utilisée, permet d'obtenir les mêmes résultats que l'accélération de Karp et Miller. Comme mentionné dans la section précédente, lorsque l'accélération de Karp et Miller construit un marquage $\mathbf{m}_\omega$ en rajoutant des ω dans un marquage $\mathbf{m}$, il est toujours possible de construire une séquence de transitions ς qui est croissante sur les places où l'accélération a ajouté un ω, et constante sur les places ne contenant pas de ω. Donc, si nous supposons que $\varsigma = t_1 \cdots t_k$ et $\mathbf{m} \xrightarrow{t_1} \cdots \xrightarrow{t_k} \mathbf{m}'$, on a que $\overline{\mathsf{Accel}}((\mathbf{m}, \mathbf{m}')) = (\mathbf{m}', \mathbf{m}_\omega)$.

Nous pouvons maintenant expliquer comment calculer, de manière efficace, un ensemble de couverture d'un réseau de Petri $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$. Pour ce faire, nous définissons d'abord la séquence $\mathsf{CovSeq}(\mathcal{N}) = (V_i)_{i \geq 0}$ d'ensembles de paires d'ω-marquages comme suit :

$$V_0 = \{(\mathbf{m}_0, \mathbf{m}_0)\} \text{ et } \forall i \geq 1 \colon V_i = V_{i-1} \cup \overline{\mathsf{Post}}(V_{i-1}) \cup \overline{\mathsf{Accel}}(V_{i-1}) \quad .$$

En raison du lien que nous venons d'établir entre $\overline{\mathsf{Accel}}$ et la fonction d'accélération de Karp et Miller, il est facile de voir que, pour tout nœud n de l'arbre de Karp et Miller de $\mathcal{N}$, il existe $k \geq 0$ tel que $\Lambda(n) \in \mathsf{Flatten}(V_k)$. Comme, par ailleurs, tant $\overline{\mathsf{Post}}$ que $\overline{\mathsf{Accel}}$ ne construisent que des ω-marquages inclus dans un ensemble de couverture, on obtient le lemme suivant :

Lemme 5 ((Geeraerts *et al.*, 2007)) *Considérons un réseau de Petri $\mathcal{N}$ tel que $\mathsf{CovSeq}(\mathcal{N}) = (V_i)_{i \geq 0}$. Alors, il existe $k \geq 0$ tel que pour tout $0 \leq \ell < k$:*

$\downarrow^{\preccurlyeq}(\mathsf{Flatten}\,(V_\ell)) \subset \downarrow^{\preccurlyeq}(\mathsf{Flatten}\,(V_{\ell+1}))$ *et pour tout* $\ell \geq k$ *:* $\downarrow^{\preccurlyeq}(\mathsf{Flatten}\,(V_\ell)) = \mathsf{Cover}\,(\mathcal{N})$.

Le Lemme 5 stipule que la séquence $(\downarrow^{\preccurlyeq}(\mathsf{Flatten}\,(V_\ell)))_{\ell\geq 0}$ est strictement croissante (pour l'inclusion ensembliste) jusqu'à un indice k où $\downarrow^{\preccurlyeq}(\mathsf{Flatten}\,(V_k))$ est l'ensemble recouvrant du réseau de Petri. À partir de ce point, la séquence $(\downarrow^{\preccurlyeq}(\mathsf{Flatten}\,(V_\ell)))_{\ell\geq 0}$ se stabilise. L'algorithme consiste donc à calculer la séquence $\mathsf{CovSeq}(\mathcal{N})$ jusqu'à rencontrer un ensemble V_ℓ tel que $\downarrow^{\preccurlyeq}(\mathsf{Flatten}\,(V_\ell)) \subseteq \downarrow^{\preccurlyeq}(\mathsf{Flatten}\,(V_{\ell-1}))$.

Le calcul de cette séquence $\mathsf{CovSeq}(\mathcal{N})$ peut être effectué de façon efficace en introduisant un ordre $\sqsubseteq$ sur les paires de marquages. Nous allons utiliser cet ordre afin de conserver, dans les ensembles manipulés, uniquement des paires *maximales* par rapport à $\sqsubseteq$. Afin de définir cet ordre, nous devons d'abord définir l'opérateur $\ominus$, qui est un opérateur de différence sur les marquages.

Étant donné deux ω-marquages $\mathbf{m}_1$ et $\mathbf{m}_2$, nous définissons $\mathbf{m}_1 \ominus \mathbf{m}_2$ comme une fonction $P \mapsto \mathbb{Z} \cup \{-\omega, \omega\}$ telle que, pour toute place p :

$$(\mathbf{m}_1 \ominus \mathbf{m}_2)(p) = \begin{cases} \omega & \text{si } \mathbf{m}_1(p) = \omega \\ -\omega & \text{si } \mathbf{m}_2(p) = \omega \text{ et } \mathbf{m}_1(p) \neq \omega \\ \mathbf{m}_1(p) - \mathbf{m}_2(p) & \text{sinon} \end{cases}$$

Étant donné deux paires $(\mathbf{m}_1, \mathbf{m}_2)$ et $(\mathbf{m}'_1, \mathbf{m}'_2)$ d'ω-marquages, nous pouvons maintenant définir l'ordre $\sqsubseteq$ comme suit :

Définition 10 *Étant donné deux paires* $(\mathbf{m}_1, \mathbf{m}_2)$ *et* $(\mathbf{m}'_1, \mathbf{m}'_2)$ *d'*ω*-marquages,* $(\mathbf{m}_1, \mathbf{m}_2) \sqsubseteq (\mathbf{m}'_1, \mathbf{m}'_2)$ *si et seulement si :* (i) $\mathbf{m}_1 \preccurlyeq \mathbf{m}'_1$*, et* (ii) $\mathbf{m}_2 \preccurlyeq \mathbf{m}'_2$*, et* (iii) $\forall p \in P\colon (\mathbf{m}_2 \ominus \mathbf{m}_1)(p) \leq (\mathbf{m}'_2 \ominus \mathbf{m}'_1)(p)$.

L'ordre que nous venons d'introduire possède les propriétés suivantes :

Lemme 6 ((Geeraerts *et al.*, 2007)) *Pour toutes paires* $(\mathbf{m}_1, \mathbf{m}_2)$ *et* $(\mathbf{m}'_1, \mathbf{m}'_2)$ *d'*ω*-marquages telles que* $(\mathbf{m}_1, \mathbf{m}_2) \sqsubseteq (\mathbf{m}'_1, \mathbf{m}'_2)$ *:*

1) pour toute paire $(\overline{\mathbf{m}}_1, \overline{\mathbf{m}}_2) \in \overline{\mathsf{Post}}\,((\mathbf{m}_1, \mathbf{m}_2))$*, il existe une paire* $(\widehat{\mathbf{m}}_1, \widehat{\mathbf{m}}_2) \in \overline{\mathsf{Post}}\,((\mathbf{m}'_1, \mathbf{m}'_2))$ *telle que* $(\overline{\mathbf{m}}_1, \overline{\mathbf{m}}_2) \sqsubseteq (\widehat{\mathbf{m}}_1, \widehat{\mathbf{m}}_2)$*,*

2) pour toute paire $(\overline{\mathbf{m}}_1, \overline{\mathbf{m}}_2) \in \overline{\mathsf{Accel}}\,((\mathbf{m}_1, \mathbf{m}_2))$*, il existe une paire* $(\widehat{\mathbf{m}}_1, \widehat{\mathbf{m}}_2) \in \overline{\mathsf{Accel}}\,((\mathbf{m}'_1, \mathbf{m}'_2))$ *telle que* $(\overline{\mathbf{m}}_1, \overline{\mathbf{m}}_2) \sqsubseteq (\widehat{\mathbf{m}}_1, \widehat{\mathbf{m}}_2)$.

La première propriété découle directement de la propriété de monotonie des réseaux de Petri et de la définition 10 de $\sqsubseteq$ (plus particulièrement des points (i) et (ii))

La seconde propriété découle de la définition de $\sqsubseteq$, et plus particulièrement du point (iii). En effet, le point (iii) assure que pour chaque place la différence entre le nombre de jetons dans $\mathbf{m}'_2$ et dans $\mathbf{m}'_1$ est au moins aussi grande que la différence entre le nombre de jetons dans $\mathbf{m}_2$ et dans $\mathbf{m}_1$. Ainsi, lorsqu'il existe une différence

strictement positive entre $\mathbf{m}_2(p)$ et $\mathbf{m}_1(p)$ (ce qui implique qu'il y aura un ω dans la place p après l'accélération), on a la garantie que cette différence est également strictement positive entre $\mathbf{m}'_2(p)$ et $\mathbf{m}'_1(p)$.

Nous pouvons maintenant introduire l'Algorithme 3, qui calcule de façon efficace l'ensemble de couverture d'un réseau de Petri. L'algorithme fait usage de la fonction $\mathsf{Max}^{\sqsubseteq}$ qui, étant donné un ensemble de paires, renvoie les paires maximales pour l'ordre partiel $\sqsubseteq$. La correction de cet algorithme ainsi que sa terminaison découlent des lemmes 5 et 6.

Données : Un réseau de Petri $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$.
Résultat : Un sous-ensemble D fini de $(\mathbb{N} \cup \omega)^{|P|}$ tel que $\downarrow^{\preccurlyeq}(D) = \mathsf{Cover}\,(\mathcal{N})$.
$i := 0$; $V_0 := \{(\mathbf{m}_0, \mathbf{m}_0)\}$;
répéter
  $i := i + 1$;
  $V_i := \mathsf{Max}^{\sqsubseteq}\left(\overline{\mathsf{Post}}\,(V_{i-1}) \cup \overline{\mathsf{Accel}}\,(V_{i-1})\right)$;
jusqu'à $\downarrow^{\preccurlyeq}(\mathsf{Flatten}\,(V_i)) \subseteq \downarrow^{\preccurlyeq}(\mathsf{Flatten}\,(V_{i-1}))$;
retourner $\mathsf{Flatten}\,(V_i)$;

Algorithme 3 : L'algorithme pour calculer un ensemble de couverture de façon efficace.

Exemple 3 *Considérons le réseau de la Figure 1, avec le marquage initial* $\mathbf{m}_1 = \langle 0, 1, 0 \rangle$*, et exécutons l'Algorithme 3 sur cette instance. L'ensemble* V_0 *contient* $(\mathbf{m}_1, \mathbf{m}_1)$*. L'ensemble* V_1 *ne contient qu'une seule paire :* $(\langle 0, 1, 0 \rangle, \langle 2, 1, 0 \rangle)$*. Les ensembles* V_2*,* V_3 *et* V_4 *sont détaillés ci-dessous :*

V_2	V_3	V_4
$(\langle 2,1,0\rangle, \langle \omega,1,0\rangle)$	$(\langle \omega,1,0\rangle, \langle \omega,1,0\rangle)$	$(\langle \omega,1,0\rangle, \langle \omega,0,1\rangle)$
$(\langle 0,1,0\rangle, \langle 4,1,0\rangle)$	$(\langle \omega,1,0\rangle, \langle \omega,0,1\rangle)$	$(\langle \omega,0,1\rangle, \langle \omega,1,0\rangle)$
$(\langle 2,1,0\rangle, \langle 1,0,1\rangle)$	$(\langle 1,0,1\rangle, \langle 2,1,0\rangle)$	$(\langle \omega,1,0\rangle, \langle \omega,1,0\rangle)$
$(\langle 0,1,0\rangle, \langle 1,0,1\rangle)$	$(\langle 1,0,1\rangle, \langle 3,0,1\rangle)$	$(\langle \omega,0,1\rangle, \langle \omega,0,1\rangle)$

L'algorithme s'arrête au bout de quatre itérations car $\downarrow^{\preccurlyeq}(\mathsf{Flatten}\,(V_3)) = \downarrow^{\preccurlyeq}(\mathsf{Flatten}\,(V_4)) = \downarrow^{\preccurlyeq}(\{\langle \omega, 1, 0\rangle, \langle \omega, 0, 1\rangle\})$*, soit l'ensemble recouvrant du réseau. Il est facile de se convaincre sur cet exemple que l'ordre* $\sqsubseteq$ *permet effectivement de réduire la taille des ensemble manipulés.*

En pratique, on peut améliorer les performances de cet algorithme en adoptant les deux modifications suivantes : premièrement, cet algorithme peut être implémenté en utilisant une *frontière*, afin d'éviter de calculer les successeurs et les accélérations de *toutes les paires* à chaque étape. Ensuite, il est possible de modifier quelque peu l'ordre dans lequel les paires sont développées. En effet, l'Algorithme 3 développe, à chaque itération, les successeurs des paires se trouvant dans V_i, ce qui correspond à un parcours *en largeur d'abord*. On obtient de meilleurs performances avec un algorithme qui privilégie le développement de paires « contenant le plus d'ω » (c'est-à-dire

les paires dont une des deux coordonnées est un marquage dont le nombre de places marquées par ω est maximal). Cette modification dans l'ordre de développement doit néanmoins être réalisée avec soin afin de conserver la correction de l'algorithme. Selon les résultats expérimentaux présentés dans (Geeraerts *et al.*, 2007), ce nouvel algorithme permet de traiter des réseaux ayant deux fois plus de places que ceux que l'algorithme de Karp et Miller est capable d'analyser. Sur les réseaux que les deux algorithmes peuvent analyser, les gains en temps d'exécution peuvent atteindre un facteur 1000. Nous renvoyons le lecteur intéressé aux références citées (Geeraerts *et al.*, 2007) pour de plus amples détails.

5. Une approche générale en arrière

Dans les sections précédentes, nous avons présenté des algorithmes qui construisent un ensemble de couverture du réseau de Petri à analyser, ce qui permet de résoudre le problème de couverture (*cfr.* Proposition 1). Néanmoins, il est possible de résoudre le problème de couverture sans calculer cet ensemble. Dans cette section, ainsi que dans les sections 6 et 7, nous présentons de tels algorithmes. Le premier d'entre eux, appelé « approche en arrière », a été introduit par Abdulla *et al.* (Abdulla *et al.*, 1996). Nous en résumons les idées principales dans cette section, et renvoyons le lecteur intéressé aux références pour une présentation détaillée.

L'approche en arrière résout le problème de couverture pour une classe générale de modèles, appelée *systèmes bien structurés*, dont les réseaux de Petri forment un cas particulier. Dans cette section, afin de simplifier la présentation, nous présentons l'instanciation de cette approche au cas des réseaux de Petri.

Étant donné un réseau de Petri $\mathcal{N}$ l'approche en arrière consiste à calculer le plus petit point fixe suivant :

$$\mu X.\, U \cup \mathsf{Pre}\,(X) \quad = \quad \mathsf{Pre}^*\,(U) \qquad [1]$$

Ce point fixe est l'ensemble de tous les marquages à partir desquels U est accessible en un nombre quelconque de pas. Il est facile de voir qu'il nous permet de résoudre le problème de couverture dans la mesure où U est accessible à partir du marquage initial $\mathbf{m}_0$ de $\mathcal{N}$ si et seulement si $\mathbf{m}_0 \in \mathsf{Pre}^*\,(U)$.

Le calcul du point fixe [1] est basé sur une séquence $(R_i)_{i\geq 0}$ d'ensembles clos par le haut de marquages, définie comme suit. L'ensemble R_0 est U. Pour tout $i \geq 0$, l'ensemble R_{i+1} est $R_i \cup \mathsf{Pre}\,(R_i)$. Il est facile de voir que cette séquence est croissante pour l'inclusion ensembliste : pour tout $i \geq 0 : R_i \subseteq R_{i+1}$. Comme $\preccurlyeq$ est un beau pré-ordre, il est possible de montrer que cette séquence se stabilise nécessairement en un nombre fini de pas. En effet, le lemme suivant indique qu'il n'est pas possible de construire une séquence infinie d'ensembles clos par le haut qui est strictement croissante (pour $\subseteq$) :

Lemme 7 *Étant donné une séquence infinie $(U_i)_{i\geq 0}$ d'ensembles clos par le haut de marquages, il existe $i < j$ tel que $U_j \subseteq U_i$.*

Par ailleurs, nous avons la garantie que l'ensemble R_k obtenu au moment où la séquence converge est bien l'ensemble $\mathsf{Pre}^*(U)$:

Lemme 8 *Pour tout $i \geq 0$, $U \subseteq R_i \subseteq \mathsf{Pre}^*(U)$ et il existe $k \geq 0$ tel que $R_k = \mathsf{Pre}^*(U)$.*

Ceci nous permet d'énoncer l'Algorithme 4 qui résout le problème de couverture à l'aide de l'approche « en arrière ». Celui-ci calcule itérativement les ensembles de la séquence $(R_i)_{i\geq 0}$ jusqu'à la convergence, puis teste si $\mathbf{m}_0$ appartient à l'ensemble ainsi calculé pour conclure.

Données : Un réseau de Petri $\mathcal{N} = \langle P, T, \mathbf{m}_0\rangle$ et $U \subseteq \mathbb{N}^{|P|}$ clos par le haut.
Résultat : $vrai$ si $\mathsf{Cover}(\mathcal{N}) \cap U \neq \emptyset$, $faux$ sinon.
$i := 0, R_0 := U$;
répéter
  $i := i + 1$;
  $R_i := R_{i-1} \cup \mathsf{Pre}(R_{i-1})$;
jusqu'à $R_i \subseteq R_{i-1}$;
si $\mathbf{m}_0 \in R_i$ **alors retourner** $vrai$ **sinon retourner** $faux$

Algorithme 4 : L'algorithme en arrière pour résoudre le problème de couverture.

Exemple 4 *Considérons à nouveau le réseau de Petri $\mathcal{N}$ de la Figure 1, et l'ensemble clos par le haut $U = \uparrow^{\preccurlyeq}(\langle 0, 0, 2\rangle)$. Le tableau ci-dessous indique, pour chaque i, les éléments minimaux des ensembles R_i obtenus lors de l'exécution de l'Algorithme 4 sur cette instance :*

0	*1*	*2*	*3*	*4*
$\langle 0,0,2\rangle$	$\langle 0,0,2\rangle$	$\langle 0,0,2\rangle$	$\langle 0,0,2\rangle$	$\langle 0,0,2\rangle$
	$\langle 1,1,1\rangle$	$\langle 0,1,1\rangle$	$\langle 0,1,1\rangle$	$\langle 0,1,1\rangle$
		$\langle 2,2,0\rangle$	$\langle 0,2,0\rangle$	$\langle 0,2,0\rangle$

Comme on le voit, le marquage initial $\mathbf{m}_1 = \langle 0, 1, 0\rangle$ n'appartient pas à $\uparrow^{\preccurlyeq}(R_4)$, et U n'est donc pas accessible à partir de $\mathbf{m}_1$. Par contre, U est bien accessible à partir de $\mathbf{m}_2 = \langle 0, 1, 1\rangle$. En effet, $\mathbf{m}_2 \in R_4$.

6. *Expand, Enlarge and Check* : une approche générale en avant

Dans cette section, nous présentons les idées principales d'un second algorithme (Geeraerts, 2007; Geeraerts *et al.*, 2006), appelé *Expand, Enlarge and Check* (EEC) qui résout le problème de couverture sans calculer un ensemble de couverture. Comme l'algorithme de la section précédente, EEC est un algorithme générique qui peut être

appliqué à la classe des systèmes bien-structurés, donc aux réseaux de Petri. De nouveau, nous ne présentons pas EEC dans toute sa généralité mais seulement son instanciation aux réseaux de Petri.

EEC est un algorithme itératif où trois étapes se succèdent à chaque itération : la première étape construit une sous-approximation S_1 de l'ensemble des marquages accessibles. La seconde étape construit une sur-approximation S_2 des marquages accessibles. La troisième étape consiste à comparer les approximations S_1 et S_2 à U : si $S_1 \cap U \neq \emptyset$, on conclut positivement ; si $S_2 \cap U = \emptyset$, on conclut négativement. Si aucun de ces deux tests ne réussit, cela signifie que les approximations ne sont pas assez précises. Dans ce cas, les deux approximations sont raffinées et l'algorithme effectue une nouvelle itération.

Dans le cas d'un réseau de Petri $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$, les approximations de $\mathsf{Post}^*(\mathbf{m}_0)$ sont construites à partir de graphes *finis* paramétrés par des ensembles de marquages et d'ω-marquages. Pour cela, EEC utilise deux séquences infinies. La première est une séquence infinie d'ensembles finis de marquages $(C_i)_{i \geq 0}$ telle que $C_i = \{0, \ldots, i\}^{|P|} \cup \{\mathbf{m}_0\}$. Autrement dit, C_i est l'ensemble de tous les marquages dans lesquels chaque place contient au plus i jetons (plus $\mathbf{m}_0$).

La seconde séquence est une séquence infinie d'ensembles finis d'ω-marquages $(L_i)_{i \geq 0}$ telle que $L_i = \{0, \ldots, i, \omega\}^{|P|} \cup \{\mathbf{m}_0\}$. Donc, L_i contient tous les ω-marquages dans lesquels chaque place contient au plus i jetons, ou bien ω (plus $\mathbf{m}_0$).

À l'itération i, EEC construit le graphe $\mathsf{Sous}(\mathcal{N}, C_i)$ qui est le système de transition sous-jacent au réseau de Petri restreint aux marquages apparaissant dans C_i. Formellement, $\mathsf{Sous}(\mathcal{N}, C_i)$ est le graphe $\langle C_i, \mathbf{m}_0, \overset{sous}{\Longrightarrow} \rangle$ où la relation $\overset{sous}{\Longrightarrow} \subseteq C_i \times C_i$ est telle que $(\mathbf{m}_1, \mathbf{m}_2) \in \overset{sous}{\Longrightarrow}$ si et seulement si $\mathbf{m}_1 \rightarrow \mathbf{m}_2$. L'ensemble des éléments de C_i qui sont accessibles par $\overset{sous}{\Longrightarrow}$ à partir de $\mathbf{m}_0$ est dénoté par $\mathcal{R}(\mathsf{Sous}(\mathcal{N}, C_i))$.

EEC construit également à l'itération i le graphe $\mathsf{Sur}(\mathcal{N}, L_i) = \langle L_i, \mathbf{m}_0, \overset{sur}{\Longrightarrow} \rangle$ tel que $(\mathbf{m}_1, \mathbf{m}_2) \in \overset{sur}{\Longrightarrow}$ si et seulement si soit $\mathbf{m}_1 \rightarrow \mathbf{m}_2$; soit $\mathbf{m}_1 \rightarrow \mathbf{m}_2'$, $\mathbf{m}_2' \notin L_i$, $\mathbf{m}_2' \preccurlyeq \mathbf{m}_2$ et il n'existe pas $\mathbf{m}_2'' \in L_i$ tel que $\mathbf{m}_2' \prec \mathbf{m}_2'' \prec \mathbf{m}_2$. Donc, $\mathsf{Sur}(\mathcal{N}, L_i)$ approxime la relation de transition $\rightarrow$ du réseau de Petri lorsque l'ω-marquage d'arrivée ne se trouve pas dans L_i. Dans ce cas, l'ω-marquage d'arrivée est remplacé par le plus petit ω-marquage dans L_i qui le sur-approxime. Le fait que ce dernier ω-marquage est unique est assuré par le lemme suivant :

Lemme 9 *Pour tout $i, k \in \mathbb{N}$, pour tout $\mathbf{m} \in \mathbb{N}^k$, il existe un unique $\mathbf{m}' \in \{0, \ldots, i, \omega\}^k$ tel que (i) $\mathbf{m} \preccurlyeq \mathbf{m}'$ et (ii) pour tout $\mathbf{m}'' \in \{0, \ldots, i, \omega\}^k$: $\mathbf{m} \preccurlyeq \mathbf{m}''$ et $\mathbf{m}' \neq \mathbf{m}''$ impliquent que $\mathbf{m}'' \not\prec \mathbf{m}'$.*

L'ensemble des éléments de L_i accessibles par la relation $\overset{sur}{\Longrightarrow}$ à partir de $\mathbf{m}_0$ est dénoté par $\mathcal{R}(\mathsf{Sur}(\mathcal{N}, L_i))$. Les deux lemmes suivants donnent les propriétés de $\mathcal{R}(\mathsf{Sous}(\mathcal{N}, C_i))$ et $\mathcal{R}(\mathsf{Sur}(\mathcal{N}, L_i))$ qui permettent d'établir que ces ensembles constituent respectivement une sous et une sur-approximation des marquages accessibles :

Lemme 10 (Sous-approximation) *Pour tout réseau de Petri $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$, pour tout ensemble clos par le haut $U \subseteq \mathbb{N}^{|P|}$, et pour tout $i \geq 0$: $\mathcal{R}(\mathsf{Sous}(\mathcal{N}, C_i)) \cap U \neq \emptyset$ implique que $\mathsf{Post}^*(\mathbf{m}_0) \cap U \neq \emptyset$.*

Lemme 11 (Sur-approximation) *Pour tout réseaux de Petri $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$, pour tout ensemble clos par le haut $U \subseteq \mathbb{N}^{|P|}$, et pour tout $i \geq 0$: $\downarrow^{\preccurlyeq}(\mathcal{R}(\mathsf{Sur}(\mathcal{N}, L_i))) \cap U = \emptyset$ implique que $\mathsf{Post}^*(\mathbf{m}_0) \cap U = \emptyset$.*

Les approximations $\mathsf{Sous}(\mathcal{N}, C_i)$ et $\mathsf{Sur}(\mathcal{N}, L_i)$ peuvent être trop grossières pour conclure. Cependant, sous certaines conditions sur C_i et L_i ces approximations sont suffisamment précises pour conclure. Plus précisément, si C_i contient tous les marquages qui apparaissent dans une exécution aboutissant à U, alors cette exécution apparaîtra également dans $\mathsf{Sous}(\mathcal{N}, C_i)$, et $\mathcal{R}(\mathsf{Sous}(\mathcal{N}, C_i))$ aura une intersection non-vide avec U :

Lemme 12 *Pour tout réseau de Petri $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$, pour tout ensemble clos par le haut $U \subseteq \mathbb{N}^{|P|}$: si $\mathsf{Post}^*(\mathbf{m}_0) \cap U \neq \emptyset$, alors $\exists i \geq 0 \colon \mathcal{R}(\mathsf{Sous}(\mathcal{N}, C_i)) \cap U \neq \emptyset$.*

De façon symétrique, si l'ensemble des accessibles du réseau de Petri n'a pas d'intersection avec U, nous avons la garantie qu'il existe un ensemble L_i tel que $\downarrow^{\preccurlyeq}(\mathcal{R}(\mathsf{Sur}(\mathcal{N}, L_i))) \cap U = \emptyset$. En effet, on peut montrer que tout ensemble L_i qui contient un *ensemble de couverture* du réseau analysé possède cette propriété :

Lemme 13 *Pour tout réseau de Petri $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$, pour tout ensemble clos par le haut $U \subseteq \mathbb{N}^{|P|}$: si $\mathsf{Post}^*(\mathbf{m}_0) \cap U = \emptyset$, alors $\exists i \geq 0 \colon \downarrow^{\preccurlyeq}(\mathcal{R}(\mathsf{Sur}(\mathcal{N}, L_i))) \cap U = \emptyset$.*

L'Algorithme 5 décrit EEC. Comme indiqué au début de cette section, cet algorithme consiste à énumérer les sous et sur-approximations, et à tester si l'une d'elles est suffisamment précise. Les lemmes 12 et 13 garantissent que, dans tous les cas, une approximation suffisamment précise sera calculée au bout d'un temps fini. Ceci garantit la terminaison de l'algorithme. Sa correction provient des lemmes 10 et 11.

Notons que, même si la présence d'un ensemble de couverture dans un des ensembles L_i est une *condition suffisante* pour assurer la terminaison dans le cas des instances négatives, elle ne constitue pas une *condition nécessaire*. Ainsi, EEC pourrait (correctement) conclure négativement même si L_i ne contient pas d'ensemble de couverture du réseau.

Exemple 5 *Considérons à nouveau le réseau $\mathcal{N}$ de la Figure 1, équipé du marquage initial $\mathbf{m}_1 = \langle 0, 1, 0 \rangle$, et appliquons* EEC *sur cet exemple. Pour $i = 1$, la sous-approximation $\mathsf{Sous}(\mathcal{N}, C_1)$ ne contient que le seul marquage initial, étant donné que seule t_1 est activée dans ce marquage, et qu'elle crée deux jetons dans p_1. La sur-approximation $\mathsf{Sur}(\mathcal{N}, L_1)$ qu'on obtient lors de la phase* Enlarge *est présentée à la*

Données : Un réseau de Petri $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$ et $U \subseteq \mathbb{N}^{|P|}$ clos par le haut.
Résultat : $vrai$ si Cover $(\mathcal{N}) \cap U \neq \emptyset$, $faux$ sinon.
pour chaque $i = 1, 2, \ldots$ **faire**
  « Expand »
    Calculer $\mathcal{R}(\mathsf{Sous}(\mathcal{N}, C_i))$;
  « Enlarge »
    Calculer $\mathcal{R}(\mathsf{Sur}(\mathcal{N}, L_i))$;
  « Check »
    si $\mathcal{R}(\mathsf{Sous}(\mathcal{N}, C_i)) \cap U \neq \emptyset$ **alors**
      retourner $vrai$;
    sinon si $\downarrow^{\preccurlyeq}(\mathcal{R}(\mathsf{Sur}(\mathcal{N}, L_i))) \cap U = \emptyset$ **alors**
      retourner $faux$;

Algorithme 5 : L'algorithme en avant *Expand, Enlarge and Check* pour résoudre le problème de couverture.

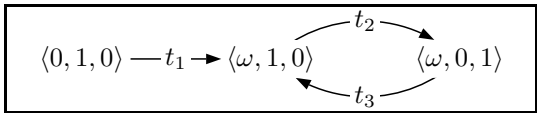


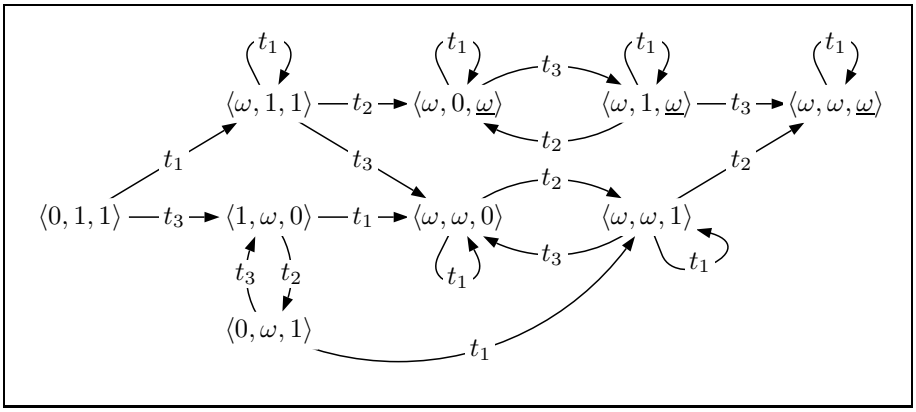
Figure 4. *La sur-approximation* $\mathsf{Sur}(\mathcal{N}, L_1)$ *obtenue en appliquant* EEC *sur le réseau* $\mathcal{N}$ *de la Figure 1, avec le marquage initial* $\mathbf{m}_1 = \langle 0, 1, 0 \rangle$.

Figure 4. Comme on le voit, celle-ci permet de prouver que $U = \uparrow^{\preccurlyeq}(\langle 0, 0, 2 \rangle)$ *n'est pas accessible.*

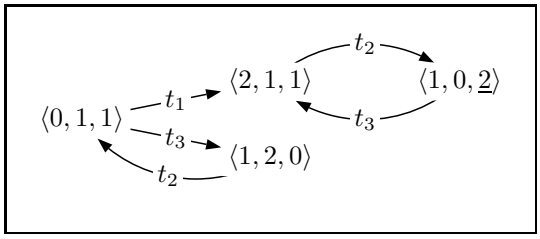
Considérons maintenant un autre marquage initial, à savoir $\mathbf{m}_2 = \langle 0, 1, 1 \rangle$. *À nouveau, la sous-approximation* $\mathsf{Sous}(\mathcal{N}, C_1)$ *ne contient que le marquage initial. Par contre, la sur-approximation* $\mathsf{Sur}(\mathcal{N}, L_1)$*, présentée à la Figure 5(a) permet d'atteindre* $U = \uparrow^{\preccurlyeq}(\langle 0, 0, 2 \rangle)$. *Il faut donc raffiner ces approximations. À l'étape* $i = 2$*, on obtient la sous-approximation* $\mathsf{Sous}(\mathcal{N}, C_2)$ *telle que présentée à la Fig. 5(b). Celle-ci permet de prouver que* U *est bien accessible.*

7. Une combinaison des approches en avant et en arrière

Dans les deux sections précédentes, nous avons présenté deux approches totalement différentes pour résoudre le problème de couverture. La première, dite « en arrière » consiste à calculer un point fixe basé sur l'opérateur Pre. La seconde, dite « en avant », repose sur le calcul, à l'aide de points fixes basés sur l'opérateur Post (ou d'une sur-approximation de Post), d'approximations des accessibles. Nous avons également présenté, dans les sections 3 et 4 deux algorithmes qui calculent un en-



(a) La sur-approximation $\mathsf{Sur}(\mathcal{N}, L_1)$



(b) La sous-approximation $\mathsf{Sous}(\mathcal{N}, C_2)$

Figure 5. *Deux étapes de l'application d'*EEC *sur le réseau $\mathcal{N}$ de la Figure 1, avec le marquage initial $\mathbf{m}_2 = \langle 0, 1, 1\rangle$.*

semble de couverture du réseau, et qui sont basés sur l'opérateur Post. Dans ce sens, ils constituent également des méthodes « en avant ».

Comme nous l'avons déjà dit, les méthodes « en avant » se comportent en général mieux que les méthodes « en arrière » (Henzinger *et al.*, 2003). Néanmoins, quand on veut seulement résoudre un problème de couverture le calcul d'un *ensemble de couverture* est une opération trop coûteuse qui ne se justifie pas en pratique. Dans cette section, nous allons donc chercher à améliorer les idées introduites dans EEC. Son principal inconvénient est que la séquence des approximations construites est fixée *a priori*. On pourrait donc s'attendre à obtenir de meilleurs résultats si les approximations étaient construites en fonction de la sémantique du réseau.

Sur la base de cette idée, nous introduisons maintenant l'Algorithme 6, une nouvelle approche inspirée de (Ganty, 2007; Cousot *et al.*, 2007). Elle consiste essentiellement à construire « en avant » des approximations successives des accessibles du réseau. Ces approximations sont raffinées d'une façon qui dépend du réseau analysé, et ce à l'aide d'un point fixe « en arrière ». Cette nouvelle méthode peut donc être vue comme une combinaison des approches « en avant » et « en arrière ».

Étant donné un réseau de Petri $\mathcal{N}$ et un ensemble clos par le haut U, cet algorithme construit en parallèle deux séquences d'approximations $(\mathcal{F}_i)_{i\geq 0}$ et $(\mathcal{B}_i)_{i\geq 0}$. La première séquence est constituée de sur-approximations de $\mathsf{Post}^*(\mathbf{m}_0)$: $\forall i \geq 0 \colon \mathsf{Post}^*(\mathbf{m}_0) \subseteq \downarrow^{\preccurlyeq}(\mathcal{F}_i)$. La seconde séquence $(\mathcal{B}_i)_{i\geq 0}$ est constituée de sous-approximations de plus en plus précises de $\mathsf{Pre}^*(U)$: $\forall i \geq 0 : \mathcal{B}_{i+1} \supseteq \mathcal{B}_i$ et $\mathcal{B}_i \subseteq \mathsf{Pre}^*(U)$. On voit donc que cet algorithme est bien une combinaison des approches « en avant » et « en arrière ».

Les constructions de ces deux séquences sont intimement liées. Nous expliquons maintenant plus en détails ces constructions, ainsi que la façon dont ces séquences sont utilisées pour décider le problème de couverture :

1) Les sur-approximations $\mathcal{F}_i$ des accessibles, calculées par la première séquence, sont construites de manière similaire à ce qui est fait dans EEC. Considérons l'itération i, et supposons que nous disposions, pour chaque place p, d'une borne B_p^i (la manière dont ces bornes sont calculées dépend de la séquence $(\mathcal{B}_i)_{i\geq 0}$, et sera expliquée dans la suite). Remarquons que les bornes peuvent ici différer d'une place à l'autre, alors que, dans le cas d'EEC, on fixait à chaque itération une borne unique valable pour toutes les places. Une fois ces bornes fixées, on obtient $\mathcal{F}_i$, en calculant l'ensemble des marquages accessibles du réseau $\mathcal{N}$, et en remplaçant, dans chaque ω-marquage $\mathbf{m}$ calculé, chaque coordonnée $\mathbf{m}(p) > B_p^i$ par ω. On obtient donc un ensemble *fini* d'ω-marquages, dont la clôture par le bas est bien une sur-approximation de $\mathsf{Post}^*(\mathbf{m}_0)$. Ce calcul correspond au point fixe de la ligne 3 de l'algorithme.

À chaque étape i, la sur-approximation $\mathcal{F}_i$ peut servir à détecter tant des instances *négatives* que des instances *positives* du problème de couverture.

En ce qui concerne les instances *négatives*, il est facile de voir que, pour tout ensemble de marquages E tel que $U \subseteq E \subseteq \mathsf{Pre}^*(U) : \downarrow^{\preccurlyeq}(\mathcal{F}_i) \cap E = \emptyset$, implique que $\mathsf{Post}^*(\mathbf{m}_0) \cap U = \emptyset$. C'est vrai en particulier si on prend $E = \mathcal{B}_i$, puisque chaque itéré de la séquence $(\mathcal{B}_i)_{i\geq 0}$ est une sous-approximation de $\mathsf{Pre}^*(U)$ qui contient U (ligne 5).

Par ailleurs, $\mathcal{F}_i$ nous permet de détecter des instances *positives* grâce à la propriété suivante : s'il existe $\mathbf{m} \in \mathcal{F}_i$ tel que $\mathbf{m}(p) \neq \omega$ pour toute place p, alors nous sommes certains qu'*aucune approximation* n'a été effectuée le long de l'exécution qui va de $\mathbf{m}_0$ à $\mathbf{m}$. Donc, si nous détectons un tel marquage $\mathbf{m}$ qui est aussi dans une sous-approximation $\mathcal{B}_i$ de $\mathsf{Pre}^*(U)$, nous pouvons conclure positivement. Dans l'algorithme, c'est la fonction Concrete qui permet de détecter les ω-marquages qui sont des marquages ordinaires (ligne 4).

2) La séquence $(\mathcal{B}_i)_{i\geq 0}$ est une sous-séquence de la séquence des itérés du point fixe $\mu X. U \cup \mathsf{Pre}(X)$, que l'on avait déjà exploité dans l'algorithme « en arrière ». On sait donc déjà que cette séquence converge, en un temps fini, vers $\mathsf{Pre}^*(U)$.

Cette séquence est également utilisée pour détecter des instances *positives* et *négatives* du problème de couverture, comme nous l'avons déjà expliqué au point précédent (voir lignes 4 et 5). De plus, à chaque étape i, les bornes B_p^i qui permettent d'obtenir $\mathcal{F}_i$, sont calculées à partir de $\mathcal{B}_i$, de la façon suivante. On considère tous les marquages

$\mathbf{m}$ qui apparaissent dans les éléments minimaux $\mathsf{Min}^{\preccurlyeq}(\mathcal{B}_i)$ et, pour chaque place p, la borne B_p^i est définie comme le maximum des $\mathbf{m}(p)$. On est ainsi assuré de pouvoir représenter de façon *précise* toutes les exécutions qui accèdent à un élément minimal de $\mathcal{B}_i$ en ne visitant que des marquages au-dehors de $\mathcal{B}_i$.

Afin de garantir que les bornes B_p^i sont raffinées à chaque étape i, $\mathcal{B}_i$ est calculé en fonction de $\mathcal{B}_{i-1}$ en appliquant l'opérateur Pre, soit jusqu'à ce qu'on obtienne un nouvel ensemble clos par le haut qui modifie les valeurs des bornes, soit jusqu'à stabilité (cela signifie qu'on a calculé $\mathsf{Pre}^*(U)$). L'opérateur Pre peut être appliqué plusieurs fois à chaque étape i : ceci explique que la séquence $(\mathcal{B}_i)_{i\geq 0}$ est une *sous-séquence* de celle calculée par l'algorithme en arrière (lignes 6 à 11).

Avant de prouver la correction de l'Algorithme 6, nous définissons les fonctions qu'il utilise :

– $\mathsf{Bound}\colon 2^{\mathbb{N}^{|P|}} \mapsto \mathbb{N}^{|P|}$ est utilisée dans l'algorithme pour définir une borne supérieure sur le nombre de jetons pour chaque place du réseau. Les bornes sont définies en fonction des éléments minimaux d'un ensemble clos par le haut U de la façon suivante : pour toute place $p \in P$, $\mathsf{Bound}(U)(p) = \max\Big(\{\mathbf{m}(p) \mid \mathbf{m} \in \mathsf{Min}^{\preccurlyeq}(U)\}\Big)$,

– $\mathsf{Widen}[f]\colon (\mathbb{N}\cup\{\omega\})^{|P|} \mapsto (\mathbb{N}\cup\{\omega\})^{|P|}$ est paramétrée par une fonction $f\colon P \mapsto \mathbb{N}$ et définie comme suit. Pour tout ω-marquage $\mathbf{m}$, $\mathsf{Widen}[f](\mathbf{m})$ est un nouvel ω-marquage tel que :

$$\forall p \in P\colon \mathsf{Widen}[f](\mathbf{m})(p) = \begin{cases} \mathbf{m}(p) & \text{si } \mathbf{m}(p) \leq f(p) \\ \omega & \text{sinon} \end{cases}$$

La fonction Widen est étendue de façon naturelle aux ensembles d'ω-marquages : $\mathsf{Widen}[f](S) = \{\mathsf{Widen}[f](\mathbf{m}) \mid \mathbf{m} \in S\}$,

– $\widehat{\mathsf{Post}}[f]\colon 2^{(\mathbb{N}\cup\{\omega\})^{|P|}} \mapsto 2^{(\mathbb{N}\cup\{\omega\})^{|P|}}$ est paramétrée par une fonction $f\colon P \mapsto \mathbb{N}$. Elle peut être vue comme un Post approximé, et est définie comme suit : pour tout ensemble S d'ω-marquages, $\widehat{\mathsf{Post}}[f](S) = \mathsf{Widen}[f](\mathsf{Post}(S))$.

– $\mathsf{Concrete}\colon (\mathbb{N}\cup\{\omega\})^{|P|} \mapsto \{vrai, faux\}$. Cette fonction permet de savoir si un ω-marquage $\mathbf{m}$ contient des ω (ce qui permet de détecter des instances positives). Pour tout marquage $\mathbf{m}$: $\mathsf{Concrete}(\mathbf{m}) = vrai$ ssi $\forall p \in P\colon \mathbf{m}(p) \neq \omega$.

Comme cet algorithme n'a pas encore été présenté dans la littérature, nous donnons les preuves de correction et de terminaison en détail.

Commençons par montrer que les points fixes basés sur $\widehat{\mathsf{Post}}$ sont calculables :

Lemme 14 *Dans l'Algorithme 6, à l'itération i, $\mathcal{F}_i$ existe et est calculable.*

Preuve. Remarquons que $\widehat{\mathsf{Post}}[\mathsf{Bound}(\mathcal{B}_i)]$, définie comme la composition des deux fonctions monotones Post et Widen, est une fonction monotone. On peut dès lors appliquer le théorème de Knaster–Tarski (Tarski, 1955), qui assure l'existence de $\mathcal{F}_i$.

Données : Un réseau de Petri $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$ et $U \subseteq \mathbb{N}^{|P|}$ clos par le haut.
Résultat : $vrai$ si $\mathsf{Cover}\,(\mathcal{N}) \cap U \neq \emptyset$, $faux$ sinon.

$\mathcal{B}_0 := U$;
pour chaque $i = 0, 1, \ldots$ **faire**
  $\mathcal{F}_i := \mu X.\, \{\mathbf{m}_0\} \cup \widehat{\mathsf{Post}}[\mathsf{Bound}\,(\mathcal{B}_i)]\,(X)$;
  si $\exists \mathbf{m} \in \mathcal{F}_i$ *t.q.* $\mathsf{Concrete}\,(\mathbf{m}) \wedge \mathbf{m} \in \mathcal{B}_i$ **alors retourner** $vrai$;
  si $\downarrow^{\preccurlyeq}(\mathcal{F}_i) \cap \mathcal{B}_i = \emptyset$ **alors retourner** $faux$;
  $j := 0, \mathcal{R}_0 := \mathcal{B}_i$;
  répéter
    $j := j + 1$;
    $\mathcal{R}_j := \mathcal{R}_{j-1} \cup \mathsf{Pre}\,(\mathcal{R}_{j-1})$;
  jusqu'à $\mathsf{Bound}\,(\mathcal{R}_j) \neq \mathsf{Bound}\,(\mathcal{R}_{j-1})$ *ou* $\mathcal{R}_j \subseteq \mathcal{R}_{j-1}$;
  $\mathcal{B}_{i+1} := \mathcal{R}_j$;

Algorithme 6 : L'algorithme qui combine les approches en avant et en arrière pour résoudre le problème de couverture.

Remarquons ensuite que les fonctions Post et $\mathsf{Widen}[\mathsf{Bound}\,(\mathcal{B}_i)]$ sont calculables. Par ailleurs, le co-domaine de la fonction $\widehat{\mathsf{Post}}[\mathsf{Bound}\,(\mathcal{B}_i)]$ est fini. Dès lors, le point fixe $\mathcal{F}_i$ défini par la séquence $(X_j)_{j\geq 0}$ telle que $X_0 = \emptyset$ et pour tout $j \geq 1 : X_{j+1} = \{\mathbf{m}_0\} \cup \widehat{\mathsf{Post}}[\mathsf{Bound}\,(\mathcal{B}_i)]\,(X_j)$ est calculable. □

Montrons ensuite que les séquences d'approximations respectent bien les propriétés que nous avons introduites ci-dessus. Nous prouvons d'abord que les points fixes $\mathcal{F}_i$ sont bien des sur-approximations des accessibles.

Lemme 15 *Dans l'Algorithme 6, à l'itération* i, $\mathsf{Post}^*\,(\mathbf{m}_0) \subseteq \downarrow^{\preccurlyeq}(\mathcal{F}_i)$.

Preuve. Observons d'abord que $\mathsf{Post}^*\,(\mathbf{m}_0) = \mu X.\, \{\mathbf{m}_0\} \cup \mathsf{Post}\,(X)$ et que, à l'étape i, $\mathcal{F}_i = \mu X.\, \{\mathbf{m}_0\} \cup \widehat{\mathsf{Post}}[\mathsf{Bound}\,(\mathcal{B}_i)]\,(X)$. Nous allons donc prouver le lemme en établissant que $\mu X.\, \{\mathbf{m}_0\} \cup \mathsf{Post}\,(X) \subseteq \downarrow^{\preccurlyeq}\Big(\mu X.\, \{\mathbf{m}_0\} \cup \widehat{\mathsf{Post}}[\mathsf{Bound}\,(\mathcal{B}_i)]\,(X)\Big)$.

Pour ce faire, considérons les séquences $(X_j)_{j\geq 0}$ et $(Y_j)_{j\geq 0}$ qui convergent respectivement vers $\mathsf{Post}^*\,(\mathbf{m}_0)$ et $\mathcal{F}_i$ et qui sont définies comme suit : $X_0 = \emptyset$, pour tout $j \geq 1 : X_{j+1} = \{\mathbf{m}_0\} \cup \mathsf{Post}\,(X_j)$; $Y_0 = \emptyset$ et pour tout $j \geq 1 : Y_{j+1} = \{\mathbf{m}_0\} \cup \widehat{\mathsf{Post}}[\mathsf{Bound}\,(\mathcal{B}_i)]\,(Y_j)$.

Prouvons maintenant par induction sur j, que $X_j \subseteq \downarrow^{\preccurlyeq}(Y_j)$, pour tout $j \geq 0$, ce qui établit que $\mu X.\, \{\mathbf{m}_0\} \cup \mathsf{Post}\,(X) \subseteq \downarrow^{\preccurlyeq}\Big(\mu X.\, \{\mathbf{m}_0\} \cup \widehat{\mathsf{Post}}[\mathsf{Bound}\,(\mathcal{B}_i)]\,(X)\Big)$.

Case de base. Trivial par définition de X_0 et Y_0.

Cas inductif.

$$\downarrow^{\preccurlyeq}(Y_{j+1}) = \downarrow^{\preccurlyeq}\Big(\{\mathbf{m}_0\} \cup \widehat{\mathsf{Post}}[\mathsf{Bound}\,(\mathcal{B}_i)]\,(Y_j)\Big) \quad [2]$$

$$= \downarrow^{\preccurlyeq}(\{\mathbf{m}_0\}) \cup \downarrow^{\preccurlyeq}\Big(\widehat{\mathsf{Post}}[\mathsf{Bound}\,(\mathcal{B}_i)]\,(Y_j)\Big) \quad [3]$$

$$= \downarrow^{\preccurlyeq}(\{\mathbf{m}_0\}) \cup \downarrow^{\preccurlyeq}\Big(\widehat{\mathsf{Post}}[\mathsf{Bound}\,(\mathcal{B}_i)]\,\big(\downarrow^{\preccurlyeq}(Y_j)\big)\Big) \quad [4]$$

$$\supseteq \downarrow^{\preccurlyeq}(\{\mathbf{m}_0\}) \cup \downarrow^{\preccurlyeq}\Big(\widehat{\mathsf{Post}}[\mathsf{Bound}\,(\mathcal{B}_i)]\,(X_j)\Big) \quad [5]$$

$$\supseteq \{\mathbf{m}_0\} \cup \mathsf{Post}\,(X_j) \quad [6]$$

$$= X_{j+1} \quad [7]$$

[2] est la définition d'un itéré. Ensuite, on passe de [3] à [4] par monotonie des réseaux de Petri ; de [4] à [5] par hyp. d'induction et monotonie de $\widehat{\mathsf{Post}}[\mathsf{Bound}\,(\mathcal{B}_i)]$; de [5] à [6] car $\mathsf{Post}\,(X) \subseteq \downarrow^{\preccurlyeq}\Big(\widehat{\mathsf{Post}}[\mathsf{Bound}\,(\mathcal{B}_i)]\,(X)\Big)$; et de [6] à [7] par déf. de X_{j+1}. □

Montrons maintenant qu'on peut utiliser les marquages $\mathbf{m} \in \mathcal{F}_i$ tels que $\mathsf{Concrete}\,(\mathbf{m}) = vrai$ pour détecter des instances *positives*. Pour ce faire, nous établissons que ces marquages font partie des accessibles du réseau considéré :

Lemme 16 *Dans l'Algorithme 6, à l'itération i, pour tout marquage $\mathbf{m} \in \mathcal{F}_i$: si $\mathsf{Concrete}\,(\mathbf{m}) = vrai$, alors $\mathbf{m} \in \mathsf{Post}^*\,(\mathbf{m}_0)$.*

Preuve. Considérons la séquence $(Y_j)_{j\geq 0}$ définie comme suit : $Y_0 = \emptyset$ et, pour tout $j \geq 1$: $Y_{j+1} = \{\mathbf{m}_0\} \cup \widehat{\mathsf{Post}}[\mathsf{Bound}\,(\mathcal{B}_i)]\,(Y_j)$. Cette séquence converge vers $\mu X.\,\{\mathbf{m}_0\} \cup \widehat{\mathsf{Post}}[\mathsf{Bound}\,(\mathcal{B}_i)]\,(X) = \mathcal{F}_i$. Montrons maintenant, par induction sur j, que pour tout $j \geq 0$: $\mathbf{m} \in Y_j$ et $\mathsf{Concrete}\,(\mathbf{m}) = vrai$ impliquent que $\mathbf{m} \in \mathsf{Post}^*\,(\mathbf{m}_0)$.

Cas de base. Trivial comme $Y_0 = \emptyset$.

Cas Inductif. Par hypothèse d'induction on sait que si $\mathbf{m} \in Y_j$ et $\mathsf{Concrete}\,(\mathbf{m})$ alors $\mathbf{m} \in \mathsf{Post}^*\,(\mathbf{m}_0)$. Considérons que $\mathbf{m} \in Y_{j+1}$ et $\mathsf{Concrete}\,(\mathbf{m}) = vrai$. Cela signifie que $\mathbf{m} \in \{\mathbf{m}_0\} \cup \widehat{\mathsf{Post}}[\mathsf{Bound}\,(\mathcal{B}_i)]\,(Y_j)$ par définition de $(Y_j)_{j\geq 0}$. Donc, soit $\mathbf{m} \in \{\mathbf{m}_0\}$ et $\mathbf{m} \in \mathsf{Post}^*\,(\mathbf{m}_0)$; soit $\mathbf{m} \in \widehat{\mathsf{Post}}[\mathsf{Bound}\,(\mathcal{B}_i)]\,(Y_j) = \mathsf{Widen}[\mathsf{Bound}\,(\mathcal{B}_i)](\mathsf{Post}\,(Y_j))$ par définition de $\widehat{\mathsf{Post}}$. Comme $\mathsf{Concrete}\,(\mathbf{m}) = vrai$, on trouve qu'il existe $\mathbf{m}' \in Y_j$ tel que $\mathsf{Concrete}\,(\mathbf{m}') = vrai$ et tel que $\mathbf{m}' \rightarrow \mathbf{m}$. De là, $\mathbf{m} \in \mathsf{Post}^*\,(\mathbf{m}_0)$ par hypothèse d'induction. □

Nous pouvons maintenant prouver l'adéquation de l'algorithme : s'il retourne $vrai$, alors l'ensemble clos par le haut U est bien accessible.

Proposition 2 *Si l'Algorithme 6 retourne $vrai$, alors $\mathsf{Post}^*\,(\mathbf{m}_0) \cap U \neq \emptyset$.*

Preuve. Supposons que l'algorithme termine à l'itération i.

L'algorithme retourne $vrai$

$\Leftrightarrow \exists \mathbf{m} \in \mathcal{F}_i : \mathsf{Concrete}(\mathbf{m}) \wedge \mathbf{m} \in \mathcal{B}_i$ ligne 4

$\Rightarrow \exists \mathbf{m} \in \mathsf{Post}^*(\mathbf{m}_0) : \mathbf{m} \in \mathcal{B}_i$ Lemme 16

$\Rightarrow \exists \mathbf{m} \in \mathsf{Post}^*(\mathbf{m}_0) : \mathbf{m} \in \mathsf{Pre}^*(U)$ $\mathcal{B}_i \subseteq \mathsf{Pre}^*(U)$, Lemme 8

$\Leftrightarrow \mathsf{Post}^*(\mathbf{m}_0) \cap \mathsf{Pre}^*(U) \neq \emptyset$

$\Leftrightarrow \mathsf{Post}^*(\mathbf{m}_0) \cap U \neq \emptyset$ sémant. $\mathsf{Post}^*(\mathbf{m}_0)$ et $\mathsf{Pre}^*(U)$

□

Pour établir la correction de l'algorithme nous devons encore montrer sa complétude :

Proposition 3 *Si l'Algorithme 6 retourne $faux$, alors* $\mathsf{Post}^*(\mathbf{m}_0) \cap U = \emptyset$.

Preuve. Supposons que l'algorithme termine à l'itération i. La ligne 5 montre que l'algorithme retourne $faux$ ssi $\downarrow^{\preccurlyeq}(\mathcal{F}_i) \cap \mathcal{B}_i = \emptyset$. Par le Lemme 15 on conclut que $\mathsf{Post}^*(\mathbf{m}_0) \cap \mathcal{B}_i = \emptyset$; et de là que $\mathsf{Post}^*(\mathbf{m}_0) \cap U = \emptyset$ par le Lemme 8. □

Pour finir, nous prouvons la terminaison de l'algorithme. Pour cela, nous avons besoin du lemme technique suivant.

Lemme 17 *Soit un ensemble U de marquages qui est clos par le haut et soit un ω-marquage $\mathbf{m}$ tel que $\downarrow^{\preccurlyeq}(\mathbf{m}) \cap U = \emptyset$. Nous avons que $\downarrow^{\preccurlyeq}(\mathsf{Widen}[\mathsf{Bound}(U)](\mathbf{m})) \cap U = \emptyset$.*

Preuve. La preuve exploite le fait que, pour tout ensemble clos par le haut U, $\uparrow^{\preccurlyeq}\Big(\mathsf{Min}^{\preccurlyeq}(U)\Big) = U$.

$\downarrow^{\preccurlyeq}(\mathbf{m}) \cap U = \emptyset$ hypothèse

$\Leftrightarrow \forall \mathbf{m}_U \in \mathsf{Min}^{\preccurlyeq}(U) : \mathbf{m}_U \not\preccurlyeq \mathbf{m}$ $\uparrow^{\preccurlyeq}(U) = U$

$\Leftrightarrow \forall \mathbf{m}_U \in \mathsf{Min}^{\preccurlyeq}(U)\ \exists p\colon \mathbf{m}(p) < \mathbf{m}_U(p)$ def. de $\preccurlyeq$

$\Rightarrow \forall \mathbf{m}_U \in \mathsf{Min}^{\preccurlyeq}(U)\ \exists p\colon \mathsf{Widen}[\mathsf{Bound}(U)](\mathbf{m})(p) < \mathbf{m}_U(p)$ def. Bound

$\Rightarrow \forall \mathbf{m}_U \in \mathsf{Min}^{\preccurlyeq}(U) : \mathbf{m}_U \not\preccurlyeq \mathsf{Widen}[\mathsf{Bound}(U)](\mathbf{m})$ def. de $\preccurlyeq$

$\Rightarrow \downarrow^{\preccurlyeq}(\mathsf{Widen}[\mathsf{Bound}(U)](\mathbf{m})) \cap U = \emptyset$ $\uparrow^{\preccurlyeq}(U) = U$

□

Proposition 4 *Pour tout réseau de Petri $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$ et tout ensemble clos par le haut $U \subseteq \mathbb{N}^{|P|}$, l'Algorithme 6 a une exécution finie.*

Preuve. Supposons que la proposition est fausse et qu'il existe un réseau de Petri $\mathcal{N}$ et un ensemble clos par le haut U pour lesquels l'Algorithme 6 ne termine pas. Notons que chaque itération de la boucle principale prend un temps fini. En effet, chaque étape de la boucle est exécutée en un temps fini. Le Lemme 14 assure que le point fixe de la ligne 3 est calculable en un temps fini. La boucle de la ligne 10 est similaire à la boucle principale de l'Algorithme 4. Pour cette raison, les arguments de terminaison sont les mêmes. En particulier, les ensembles $\mathcal{R}_j$ sont toujours par construction des ensembles clos par le haut (d'après les lignes 1,6-11) qui croissent à chaque itération de la boucle (ligne 9). Par le Lemme 7, on est assuré qu'après un nombre d'itérations fini on a bien que $\mathcal{B}_j \subseteq \mathcal{B}_{j-1}$.

Donc, si l'algorithme ne termine pas, c'est nécessairement parce qu'il exécute un nombre infini d'itérations de la boucle principale. Le Lemme 8 montre qu'après un nombre fini k d'itérations on obtient $\mathcal{B}_k = \mathsf{Pre}^*(U)$. Nous montrons maintenant que l'algorithme doit nécessairement terminer.

On considère deux cas : $\mathsf{Post}^*(\mathbf{m}_0) \cap U = \emptyset$ ou non. Pour le premier cas, comme $\mathcal{B}_k$ est clos par le haut et est un point fixe pour la fonction $f(X) = U \cup \mathsf{Pre}(X)$ on trouve que $\downarrow^{\preccurlyeq}\left(\mathsf{Post}\left(\overline{\mathcal{B}_k}\right)\right) \subseteq \overline{\mathcal{B}_k}$ où $\overline{\mathcal{B}_k}$ est le complément de $\mathcal{B}_k$. De là, on déduit :

$$\forall \mathbf{m} \in (\mathbb{N} \cup \omega)^{|P|} \colon \downarrow^{\preccurlyeq}(\mathbf{m}) \subseteq \overline{\mathcal{B}_k} \Rightarrow \downarrow^{\preccurlyeq}(\mathsf{Post}(\mathbf{m})) \subseteq \overline{\mathcal{B}_k} \ . \qquad [8]$$

Pour tout ensemble X, on a que $X \cap \mathcal{B}_k = \emptyset$ si et seulement si $X \subseteq \overline{\mathcal{B}_k}$. En appliquant le Lemme 17 à [8], on trouve que pour tout ω-marquage $\mathbf{m}$

$$\downarrow^{\preccurlyeq}(\mathbf{m}) \subseteq \overline{\mathcal{B}_k} \Rightarrow \downarrow^{\preccurlyeq}(\mathsf{Widen}[\mathsf{Bound}(\mathcal{B}_k)](\mathsf{Post}(\mathbf{m}))) \subseteq \overline{\mathcal{B}_k}$$

est équivalent à

$$\downarrow^{\preccurlyeq}(\mathbf{m}) \subseteq \overline{\mathcal{B}_k} \Rightarrow \downarrow^{\preccurlyeq}\left(\widehat{\mathsf{Post}}[\mathsf{Bound}(\mathcal{B}_k)](\mathbf{m})\right) \subseteq \overline{\mathcal{B}_k} \ .$$

Par hypothèse, on sait que $\mathcal{B}_k = \mathsf{Pre}^*(U)$ et $\mathsf{Post}^*(\mathbf{m}_0) \cap U = \emptyset$, donc que $\mathbf{m}_0 \in \overline{\mathcal{B}_k}$. Donc, par le théorème de Knaster–Tarski (Tarski, 1955), on sait que l'algorithme calcule à la ligne 3 un point fixe tel que $\downarrow^{\preccurlyeq}(\mathcal{F}_k) \subseteq \overline{\mathcal{B}_k}$, et donc que $\downarrow^{\preccurlyeq}(\mathcal{F}_k) \cap \mathcal{B}_k = \emptyset$. Ceci implique que le test de la ligne 5 est vérifié et l'algorithme termine en retournant $faux$.

Considérons maintenant le second cas : $\mathsf{Post}^*(\mathbf{m}_0) \cap U \neq \emptyset$. Remarquons d'abord que $\mathsf{Concrete}(\mathbf{m}_0) = vrai$. Ensuite,

$\mathsf{Post}^*(\mathbf{m}_0) \cap U \neq \emptyset$	hypothèse
$\Leftrightarrow \mathbf{m}_0 \in \mathsf{Pre}^*(U)$	sémantique de $\mathsf{Post}^*(\mathbf{m}_0)$
$\Leftrightarrow \mathbf{m}_0 \in \mathcal{B}_k$	$\mathcal{B}_k = \mathsf{Pre}^*(U)$
$\Leftrightarrow \mathbf{m}_0 \in \mathcal{B}_k \wedge \mathsf{Concrete}(\mathbf{m}_0)$	$\mathsf{Concrete}(\mathbf{m}_0) = vrai$
$\Rightarrow \exists \mathbf{m} \in \mathcal{F}_k : \mathsf{Concrete}(\mathbf{m}) \wedge \mathbf{m} \in \mathcal{B}_k$	$\mathbf{m}_0 \in \mathcal{F}_k$ voir ligne 3
$\Rightarrow$ L'algorithme retourne $vrai$ à l'itération k	test, ligne 4

□

En combinant les propositions 2, 3 et 4, nous obtenons le Théorème 2. Nous pouvons donc conclure :

Théorème 2 *Pour tout réseau de Petri $\mathcal{N} = \langle P, T, \mathbf{m}_0 \rangle$ et tout ensemble clos par le haut U, l'Algorithme 6 termine toujours et répond $vrai$ ssi $\mathsf{Post}^*(\mathbf{m}_0) \cap U \neq \emptyset$.*

Pour terminer cette section, notons que nous avons volontairement gardé la présentation de cet algorithme (et de ceux qui précèdent) la plus simple possible. En particulier, l'Algorithme 6 n'utilise pas l'information calculée lors des itérations précédentes pour réduire le calcul des points fixes comme dans (Ganty, 2007; Cousot *et al.*, 2007). De nombreuses autres heuristiques peuvent également être appliquées. Elles font l'objet de la prochaine section.

8. Heuristiques

Les algorithmes que nous avons présentés dans les sections précédentes permettent de résoudre le problème de couverture de façon efficace. Néanmoins, afin d'implémenter ces algorithmes de façon à ce qu'ils soient aussi efficaces que possible dans les cas pratique, il est souvent nécessaire de recourir à différentes heuristiques. Dans cette section, nous présentons un survol de la littérature présentant de telles heuristiques, et ceci, afin de bien montrer que ces techniques d'implémentation apportent un gain appréciable au niveau des performances, et ne sont donc pas à négliger. Nous renvoyons le lecteur intéressé aux articles originaux pour plus de détails.

8.1. *Structures de données symboliques*

Dans (Van Begin, 2003; Delzanno *et al.*, 2004), les auteurs ont présenté une structure de données, appelée *Covering Sharing Tree*[5] (CST), qui permet de représenter et

5. voir aussi (Delzanno *et al.*, 2006) pour une extension

de manipuler les ensembles clos par le haut et par le bas. Un CST représente, de façon *compacte*, les marquages minimaux d'un ensemble clos par le haut ou un ensemble de couverture d'un ensemble clos par le bas. Un exemple de CST est donné à la Figure 6.

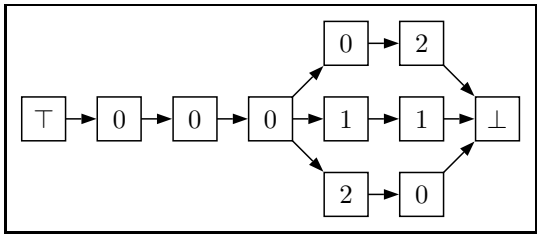


Figure 6. *Un exemple de* CST *qui représente l'ensemble clos par le haut U grâce à son générateur minimal* $\mathsf{Min}^{\preccurlyeq}(U) = \{\langle 0,0,0,0,2\rangle, \langle 0,0,0,1,1\rangle, \langle 0,0,0,2,0\rangle\}$

D'autres structures de données symboliques existent pour représenter des ensembles de marquages. Les *dépliages de réseaux de Petri* en sont un exemple. Ces derniers ont été initialement définis pour des réseaux de Petri bornés (Esparza *et al.*, 1996), mais dans (Abdulla *et al.*, 2000), l'utilisation des dépliages de réseaux de Petri est étendue aux réseaux de Petri non-bornés. Ils sont alors utilisés comme structure de données symbolique pour calculer $\mathsf{Pre}^*(U)$ (voire Section 5). Les algorithmes de construction des dépliages ont la particularité d'exploiter la concurrence du réseau de Petri analysé.

8.2. *Utilisation des invariants de place*

Dans (Delzanno *et al.*, 2004), les auteurs montrent comment les *invariants de place* (Silva *et al.*, 1998) du réseau de Petri considéré peuvent être utilisés pour réduire les ensembles R_i calculés par l'Algorithme 4. Un invariant de place est une contrainte linéaire qui caractérise une sur-approximation des marquages accessibles du réseau. La technique développée dans (Delzanno *et al.*, 2004) consiste à supprimer des ensembles R_i des marquages qui ne satisfont aucun invariant du réseau.

8.3. *Utilisation d'abstractions pour réduire la dimension des ensembles manipulés*

L'efficacité des solutions algorithmiques au problème de couverture décroît avec l'augmentation de la taille des réseaux de Petri considérés. Des méthodes proposent de définir, à partir d'un réseau $\mathcal{N}$, un réseau $\widehat{\mathcal{N}}$ *plus petit* tel que l'accessibilité d'un ensemble clos par le haut dans $\mathcal{N}$ peut être résolue en analysant $\widehat{\mathcal{N}}$.

Dans, (Ganty *et al.*, 2007), les auteurs définissent une telle méthode. Étant donné un réseau $\mathcal{N}$, appelé réseau *concret*, cette méthode consiste à « *fusionner* » des ensembles de places de $\mathcal{N}$, ce qui aboutit à la construction du réseau *abstrait* $\widehat{\mathcal{N}}$ (plus

petit) où chaque place de $\widehat{\mathcal{N}}$ correspond à un ensemble de places de $\mathcal{N}$. Par exemple, si $\mathcal{N}$ possède les places $\{p_1, p_2, p_3, p_4\}$ et que la partition $\{\{p_1, p_2\}, \{p_3, p_4\}\}$ décrit la fusion de places, alors $\widehat{\mathcal{N}}$ possède deux places et le marquage $\langle 1, 1, 1, 1\rangle$ de $\mathcal{N}$ est représenté par le marquage $\langle 2, 2\rangle$ de $\widehat{\mathcal{N}}$.

Dans ce cas, l'ensemble des accessibles de $\widehat{\mathcal{N}}$ représente une *sur-approximation* de l'ensemble des accessibles de $\mathcal{N}$. Si cette sur-approximation est suffisamment précise, elle peut être concluante sinon l'analyse doit être raffinée. On fixe alors une partition plus fine des places de $\mathcal{N}$ et on construit un nouveau $\widehat{\mathcal{N}}$. Le choix de la partition est guidé par la *sémantique* de $\mathcal{N}$.

D'autres travaux permettent de réduire la taille des réseaux en se basant sur leur syntaxe, et consiste essentiellement à remplacer des sous-réseaux par d'autres plus petits (voir par exemple (Berthelot *et al.*, 1975)).

Toutes les heuristiques présentées dans cette section ont montré leur intérêt en pratique et leur utilisation permet d'analyser des réseaux de Petri complexes de plus de 50 places.

9. Résultats expérimentaux

Nous avons implanté le nouvel algorithme (Algorithme 6) présenté dans la Section 7 ainsi que EEC (Section 6) et l'Algorithme 4 (en arrière) de la Section 5 dans un prototype qui utilise la structure de données des CST (Van Begin, 2003) pour manipuler des ensembles clos par le haut et par le bas. Aucune des autres heuristiques discutées à la Section 8 n'a été utilisée. Les résultats que nous avons obtenus sont rapportés au Tableau 1. La comparaison entre les temps d'exécution rapportés dans la colonne **Arrière** et les colonnes **Combi** et **EEC** est à relativiser dans le sens où les implantations de EEC et du nouvel algorithme dans notre prototype sont assez naïves. Par exemple, les techniques proposées dans (Geeraerts *et al.*, 2005) permettent de réduire les temps d'exécution de EEC (et de l'Algorithme 6) de façon dramatique mais n'ont pas été implantées dans notre prototype. Sur la plupart des exemples, l'Algorithme 6 qui combine les techniques *en avant* et *en arrière* se comporte mieux que l'algorithme EEC. De plus, les résultats expérimentaux donnés dans (Geeraerts *et al.*, 2005) montrent que EEC se comporte avantageusement par rapport au calcul de $\mathsf{Pre}^*(U)$ lorsqu'il est implenté de façon efficace. Les résultats donnés dans le Tableau 1 montrent donc l'intérêt de l'Algorithme 6.

Tous les exemples sont des instances négatives, c'est-à-dire que l'ensemble clos par le haut n'est pas accessible à partir du marquage initial, exceptés les exemples *kanban* et *pncsa*. Dans ce dernier exemple, l'Algorithme 6 effectue cinq raffinements alors que la propriété est rapidement trouvée par l'algorithme EEC lors de la première itération (c'est-à-dire en n'utilisant que des ω-marquages dont les coordonnées sont 0, 1 ou ω).

Examples	Places	Trans.	Combi.	EEC	Arrière
Basic ME	5	4	<0,01s	<0,01s	<0,01s
MultiME	12	11	<0,01s	<0,01s	<0,01s
CSM	15	13	0,06s	0,06s	0,06s
multipoll	18	21	9,4s	19,2s	1, 63s
fms	22	20	8,4s	>60min	4,5s
kanban	16	16	125,6s	>60min	28min30s
mesh2x2	32	32	57,6s	17min29s	0,6s
pncsa	31	36	0,7s	0,2s	6,8s

Tableau 1. *Tableau de résultats.* **Places** *: nombre de places,* **Trans.** *: nombre de transitions,* **Combi.** *: temps d'exécution de l'Algorithme 6,* **EEC** *: temps d'exécution de* EEC *(Algorithme 5),* **Arrière** *: temps d'exécution de l'Algorithme 4. Les temps d'exécutions ont été obtenus sur un* Intel Xeon *3Ghz.*

10. Travaux connexes

Nous terminons cet article en mentionnant d'autres résultats qui sont, de près ou de loin, liés au calcul de l'ensemble de couverture des réseaux de Petri, et que, par manque de place, nous n'avons pas pu présenter en détail dans les sections précédentes. Plus précisément, nous commençons par rappeler brièvement d'autres techniques pour résoudre le problème de couverture, puis nous abordons différents modèles qui généralisent les réseaux de Petri, et au sujet desquels le problème de couverture (ou un problème similaire) a également été étudié.

10.1. *Autres techniques pour résoudre le problème de couverture*

Commençons par mentionner l'existence de techniques *structurelles* qui se basent sur la structure du réseau pour décider le problème de couverture et qui ne sont en général pas complètes. Dans (Silva *et al.*, 1998) et plus récemment dans (Sankaranarayanan *et al.*, 2004), des invariants sont générés automatiquement. Ceux-ci sur-approximent l'ensemble de marquages accessibles d'un réseau de Petri et permettent de décider dans certains cas que des ensembles de marquages (clos par le haut) ne sont pas accessibles. Dans (Esparza *et al.*, 2000), des techniques de programmation linéaire en nombres entiers sont utilisées pour décider le problème de couverture. Comme dans le cas de l'utilisation d'invariants, ces techniques permettent de conclure dans certains cas qu'un ensemble de marquages n'est pas accessible.

10.2. *Généralisation des réseaux de Petri*

Une grande attention a été donnée à des classes de modèles plus expressives que les réseaux de Petri, comme les automates qui manipulent des compteurs (Minsky, 1967).

Ces automates sont aussi expressifs que les machines de Turing et la vérification de nombreuses propriétés sur ceux-ci, comme l'accessibilité d'un ensemble semi-linéaire de configurations (une généralisation du problème de couverture), est en général indécidable.

Dans (Boigelot, 1999), une structure de données basée sur les automates finis, appelée *Number Decision Diagrams* (NDD), est proposée pour représenter des ensembles semi-linéaires de configurations d'automates à compteurs. Dans (Boigelot, 1999; Bardin *et al.*, 2004), des semi-algorithmes manipulant des NDD sont définis afin de construire l'ensemble des configurations accessibles d'un automate à compteurs à partir d'un ensemble semi-linéaire de configurations initiales.

Des extensions de réseaux de Petri ont été étudiées dans le cadre de la résolution du problème de couverture. Le résultat principal de (Abdulla *et al.*, 1996) permet de conclure que le problème de couverture est décidable pour toutes les extensions monotones des réseaux de Petri. Dans (Van Begin, 2003; Delzanno *et al.*, 2002), des techniques sont développées pour obtenir un algorithme efficace en pratique pour résoudre le problème de couverture pour des extensions de réseaux de Petri permettant de décrire des programmes *multi-threads*. Dans (Ciardo, 1994), la technique des invariants de places de (Silva *et al.*, 1998) est généralisée aux *self-modifying nets*, une large extension des réseaux de Petri.

Les réseaux de Petri ont également été étudiés dans le cadre de la théorie des jeux. Dans (Raskin *et al.*, 2005), des résultats montrent que la plupart des problèmes deviennent indécidables sur les réseaux de Petri lorsqu'on rajoute un deuxième joueur. Les auteurs donnent aussi des résultats de décidabilité pour le problème de couverture en présence de deux joueurs pour une classe particulière de réseaux de Petri.

11. Bibliographie

Abdulla P. A., Cerans K., Jonsson B., Tsay Y.-K., « General decidability theorems for infinite-state systems », *LICS '96 : Proc. 11th Annual IEEE Symp. on Logic in Computer Science*, IEEE Computer Society, p. 313-321, 1996.

Abdulla P. A., Iyer S. P., Nylén A., « Unfoldings of Unbounded Petri Nets. », *CAV '00 : Proc. 12th Int. Conf. on Comp. Aided Verification*, vol. 1855 of *LNCS*, Springer, p. 495-507, 2000.

Bardin S., Finkel A., Leroux J., « FASTer Acceleration of Counter Automata in Practice », *TACAS'04 : Proc. 10th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, vol. 2988 of *LNCS*, Springer, p. 576-590, 2004.

Berthelot G., Roucairol G., Valk R., « Reductions of Nets and Parallel Programs », *Advanced Course : Net Theory and Applications*, vol. 84 of *LNCS*, Springer, p. 277-290, 1975.

Boigelot B., Symbolic Methods for Exploring Infinite State Spaces, PhD thesis, Université de Liège, 1999.

Ciardo G., « Petri nets with marking-dependent arc multiplicity : properties and analysis », *APN '94 : Proc. of 15th Int. Conf. on Application and Theory of Petri Nets*, vol. 815 of *LNCS*, Springer, p. 179-198, 1994.

Cousot P., Ganty P., Raskin J.-F., « Fixpoint-Guided Abstraction Refinements », *SAS'07 : Proc. 14th Int. Static Analysis Symp.*, vol. 4634 of *LNCS*, Springer, p. 333-348, 2007.

Delzanno G., Ganty P., Kalyon G., Meuter C., Raskin J.-F., Van Begin L., Symbolic Data Structure for sets of k-uples of integers, Technical report, Université Libre de Bruxelles, 2006.

Delzanno G., Raskin J.-F., Van Begin L., « Towards the Automated Verification of Multithreaded Java Programs », *TACAS '02 : Proc. 8th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, vol. 2280 of *LNCS*, Springer, p. 173-187, 2002.

Delzanno G., Raskin J.-F., Van Begin L., « Covering sharing trees : a compact data structure for parameterized verification. », *Software Tools for Technology Transfer*, vol. 5, n° 2-3, p. 268-297, 2004.

Dickson L. E., « Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors », *Amer. J. Math.*, vol. 35, p. 413-422, 1913.

Esparza J., Melzer S., « Verification of safety properties using integer programming : Beyond the state equation », *Formal Methods in System Design*, vol. 16, p. 159-189, 2000.

Esparza J., Römer S., Vogler W., « An improvement of McMillan's unfolding algorithm », *TACAS'96 : Proc. 2nd Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, vol. 1055 of *LNCS*, Springer, p. 87-106, 1996.

Finkel A., « A generalization of the procedure of Karp and Miller to well structured transition system », *ICALP '87 : Proc. of 14th Int. Colloquium on Automata, Languages and Programming*, vol. 267 of *LNCS*, Springer, p. 499-508, 1987.

Finkel A., « Reduction and Covering of Infinite Reachability Trees », *Information and Computation*, vol. 89, n° 2, p. 144-179, 1990.

Finkel A., « The Minimal Coverability Graph for Petri Nets. », *APN '91 : Proc. of 12th Int. Conf. on Appl. and Theory of Petri Nets*, vol. 674 of *LNCS*, Springer, p. 210-243, 1993.

Finkel A., Schnoebelen P., « Well-structured transition systems everywhere ! », *Theoretical Computer Science*, vol. 256, n° 1-2, p. 63-92, 2001.

Ganty P., The Fixpoint Checking Problem : An Abstraction Refinement Perspective, PhD thesis, Université Libre de Bruxelles, 2007.

Ganty P., Raskin J.-F., Van Begin L., « A Complete Abstract Interpretation Framework for Coverability Properties of WSTS », *VMCAI '06 : Proc. 7th Int. Conf. on Verification, Model Checking and Abstract Interpretation*, vol. 3855 of *LNCS*, Springer, p. 49-64, 2006.

Ganty P., Raskin J.-F., Van Begin L., « From Many Places to Few : Automatic Abstraction Refinement for Petri Nets », *ICATPN '07 : Proc. of 28th Int. Conf. on Appl. and Theory of Petri Nets and Other Models of Concur.*, vol. 4546 of *LNCS*, Springer, p. 124-143, 2007.

Geeraerts G., Coverability and Expressiveness Properties of Well-structured Transition Systems, PhD thesis, Université Libre de Bruxelles, 2007.

Geeraerts G., Raskin J.-F., Van Begin L., « Expand, Enlarge and Check... Made Efficient. », *in* K. Etessami, S. K. Rajamani (eds), *CAV*, vol. 3576 of *Lecture Notes in Computer Science*, Springer, p. 394-407, 2005.

Geeraerts G., Raskin J.-F., Van Begin L., « Expand, Enlarge and Check : New algorithms for the coverability problem of WSTS. », *Journal of Computer and System Sciences*, vol. 72, n° 1, p. 180-203, 2006.

Geeraerts G., Raskin J.-F., Van Begin L., « On the efficient Computation of the Minimal Coverability set of Petri nets », *ATVA '07 : Proc. of 5th Int. Symp. on Automated Technology for Verification and Analysis*, vol. 4762 of *LNCS*, Springer, p. 98-113, 2007.

German S. M., Sistla A. P., « Reasoning about Systems with Many Processes », *Journal of ACM*, vol. 39, n° 3, p. 675-735, 1992.

Hack M., « The equality problem for vector addition systems is undecidable, », *Theoretical Computer Science*, vol. 2, p. 77-95, 1976.

Henzinger T. A., Kupferman O., Qadeer S., « From *Pre*historic to *Post*modern Symbolic Model Checking », *Formal Methods in System Design*, vol. 23, n° 3, p. 303-327, 2003.

Karp R. M., Miller R. E., « Parallel Program Schemata », *Journal of Computer and System Sciences*, vol. 3, p. 147-195, 1969.

Luttge K., *« Zustandsgraphen von Petri-Netzen »*, Master's thesis, Humboldt-Universität zu Berlin, 1995.

Minsky M., *Finite and Infinite Machines*, Englewood Cliffs, N.J., Prentice-Hall, 1967.

Rackoff C., « The Covering and Boundedness Problems for Vector Addition Systems », *Theor. Comput. Sci.*, vol. 6, p. 223-231, 1978.

Raskin J.-F., Samuelides M., Van Begin L., « Games for Counting Abstraction », *Electronic Notes in Theoretical Computer Science*, vol. 128, n° 6, p. 69-85, 2005.

Reisig W., *Petri Nets. An introduction*, Springer, 1986.

Sankaranarayanan S., Sipma H. B., Manna Z., « Petri Net Analysis Using Invariant Generation », *Verification : Theory and Practice (Essays Dedicated to Zohar Manna on the Occasion of His 64th Birthday)*, n° 2772 in *LNCS*, Springer, p. 682-701, 2004.

Silva M., Teruel E., Colom J. M., « Linear Algebraic and Linear Programming Techniques for Analysis of Place/Transition Net Systems », *ICATPN '98 : Proc. of 19th Int. Conf. on Application and Theory of Petri Nets*, vol. 1491 of *LNCS*, Springer, p. 308-309, 1998.

Tarski A., « A lattice-theoretical fixpoint theorem and its applications. », *Pacific Journal of Mathematics*, vol. 5, n° 2, p. 285-309, 1955.

Valk R., Vidal-Naquet G., « Petri Nets and Regular Languages », *J. Comput. Syst. Sci.*, vol. 23, n° 3, p. 299-325, 1981.

Van Begin L., Efficient Verification of Counting Abstractions for Parametric Systems, PhD thesis, Université Libre de Bruxelles, Belgium, 2003.

Article reçu le 4 janvier 2008
Version révisée le 17 juin 2008
Rédacteur responsable : Gilles Geeraerts

Pierre Ganty *est docteur en Sciences, spécialité Informatique (Université Libre de Bruxelles, 2007). Il s'intéresse à la vérification de systèmes à espaces d'états infinis comme les réseaux de Petri ou les automates à pile. Ses travaux portent en particulier sur les techniques de vérification par raffinement d'abstractions.*

Gilles Geeraerts *est docteur en Sciences, spécialité Informatique (Université Libre de Bruxelles, 2007) et Professeur Assistant au Département d'Informatique de la Faculté des Sciences de l'ULB. Il s'intéresse aux méthodes de vérification adaptées aux systèmes à espaces d'états infinis comme les réseaux de Petri ou les automates temporisés.*

Jean-François Raskin *est docteur en Informatique (Université de Namur, 1999) et Professeur au Département d'Informatique de la Faculté des Sciences de l'ULB. Ses travaux embrassent les principaux domaines de la vérification assistée par ordinateur : logiques, automates, interprétation abstraite, théorie des jeux…*

Laurent Van Begin *est docteur en Sciences, spécialité informatique (Université Libre de Bruxelles, 2004). Il a travaillé comme chercheur au Département d'Informatique de la Faculté des Sciences de l'ULB, s'intéressant tout particulièrement à la vérification de systèmes infinis. Il exerce aujourd'hui dans l'industrie, dans le domaine de la sécurité banquaire.*