Vérification de propriétés invariantes par surapproximation

Florent Peres, Pierre-Olivier Ribet, François Vernadat, Bernard Berthomieu

Vérification de propriétés invariantes par surapproximation

FLORENT PERES, PIERRE-OLIVIER RIBET,
FRANÇOIS VERNADAT, BERNARD BERTHOMIEU
LAAS-CNRS
{fperes,ribet,francois,bernard}@laas.fr

8 mars 2005

Résumé. Ce papier expose une technique d'abstraction de réseaux de Petri permettant de réduire l'explosion combinatoire lors de la validation de propriétés. L'abstraction proposée repose sur une fonction de renommage qui va permettre de cacher/confondre certaines places du réseau. Nous montrerons les résultats que nous avons obtenus ainsi que les limites de cette abstraction, nous soumettrons celle-ci à des exemples connus pour finalement apporter une comparaison avec des techniques existantes.

1 Introduction

Le travail présenté dans ce papier s'intéresse à la réduction de l'explosion combinatoire qu'il est nécessaire de mettre en oeuvre pour pouvoir appliquer les techniques de vérification exhaustive. Différentes approches ont été suivies pour combattre cette explosion : symétries, ordres partiels,... Celles-ci offrent en général une procédure de décision effective : pour une classe de propriétés connue à l'avance, la vérification peut être menée de façon équivalente sur le graphe réduit ou sur le graphe exhaustif. L'approche suivie dans ce travail s'apparente aux techniques d'abstraction.

Nous pouvons distinguer deux approches dans les techniques d'abstraction pour les réseaux de Petri : une approche d'abstraction de l'espace d'états, où l'on retrouve les techniques de réduction de type ordre partiel, symétrie, etc..., ainsi qu'une approche d'abstraction du réseau de Petri lui-même. Notre technique se situe dans le second cas. Dans le même domaine d'application, on peut citer les travaux de Berthelot[Ber86] [Ber87] et d'André[And82] [And83] qui ont mis au point des techniques de réduction de réseaux. Parmi celles-ci, les transformations de Berthelot permettent de simplifier un réseau en ôtant certaines places (ou transitions) «inutiles» vis à vis de la propriété à vérifier. Ces transformations vont également permettre d'établir une relation de raffinement pour vérifier qu'une implémentation (ajout d'une partie dans un réseau de Petri) vérifie bien sa spécification (réseau de Petri sans cet ajout). Enfin, elles permettent de décomposer un réseau pour pouvoir le vérifier plus facilement, ces décompositions utilisant les travaux de C. André, qui présentent des relations d'équivalences permettant d'extraire un sous-réseau ayant des propriétés similaires à un sous-ensemble du réseau global. Ces techniques d'abstraction offrent des procédures de décision effectives.

Notre approche est différente en ce sens qu'elle n'offre uniquement qu'une procédure de décision semi-effective : lorsque la propriété est établie sur le graphe réduit (abstrait), on peut en déduire sa satisfaction sur le graphe exhaustif mais, dans la négative, on ne peut en général pas statuer. Ce papier présente un premier état des travaux que nous venons de commencer dans cette direction. Nous nous situons dans le cadre des réseaux de Petri Place/Transition, l'abstraction que nous proposons repose sur une fonction de renommage qui va permettre de cacher/confondre certaines places du réseau. On montre que l'espace d'états associé au réseau abstrait - obtenu par masquage/renommage de certaines de ses places - est un sur-ensemble de l'espace d'états concret et plus généralement que le réseau de Petri abstrait est simulé par le réseau concret initial. L'approche proposée permet donc de statuer uniquement sur des *"propriétés locales invariantes"*. Pour pouvoir expérimenter nous avons écrit un prototype que nous avons couplé avec le logiciel Tina[BRV03]. Nous présenterons les premières expérimentations et effectuerons une première comparaison avec les techniques de réduction *"ordre partiel"* que nous développons et utilisons habituellement.

Dans la section 2 nous présenterons par un exemple comment l'abstraction est obtenue et nous introduirons de manière formelle l'abstraction par renommage, ainsi que les propriétés qui sont préservées. Dans la section 3 nous donnerons deux exemples d'utilisation. Dans la section 4 nous nous intéressons aux propriétés d'accessibilité et au caractère borné, et nous verrons comment notre abstraction se comporte sur un réseau temporel. La section 5, enfin, conclura notre discussion.

2 Présentation

2.1 Les réseaux de Petri Place/Transition[HV01]

Définition 1. Un réseau de Petri marqué est un quintuplet $\mathcal{R} =< P, T, Pre, Post, \mathcal{M}_0 >$, où
- P est un ensemble fini de places,
- T est un ensemble fini de transitions,
- Pre : $P \times T \rightarrow \mathbb{N}$ est l'application places précédentes ou préconditions,
- Post : $P \times T \rightarrow \mathbb{N}$ est l'application places suivantes ou postconditions,
- $\mathcal{M}_0 : P \rightarrow \mathbb{N}$ est le marquage initial.

Définition 2. Un marquage est une application $\mathcal{M} : P \rightarrow \mathbb{N}$. $\mathcal{M}(p)$ est le nombre de jetons dans la place p.

Définition 3. Une transition t est tirable (ou franchissable ou sensibilisée) à partir d'un marquage $\mathcal{M}$ ssi $\forall p \in P : \mathcal{M}(p) \geqslant Pre(p,t)$. Cela sera noté $\mathcal{M} \xrightarrow{t}$.

Définition 4. Lorsqu'une transition t est tirable à partir d'un marquage $\mathcal{M}$, le tir de t donne un marquage $\mathcal{M}'$, tel que : $\forall p \in P : \mathcal{M}'(p) = \mathcal{M}(p) - Pre(p,t) + Post(p,t)$. On utilisera alors la notation $\mathcal{M} \xrightarrow{t} \mathcal{M}'$.

2.2 Présentation informelle de la surapproximation

L'approche suivie pour regrouper/cacher certains comportements du réseau consiste à faire abstraction de certaines contraintes le régissant. En l'occurence, nous allons introduire des places abstraites permettant de fusionner des places concrètes du réseau. A cet effet, on utilise une fonction de renommage de places qui nous permettra d'obtenir un réseau abstrait. Mettons en oeuvre notre abstraction sur un exemple :

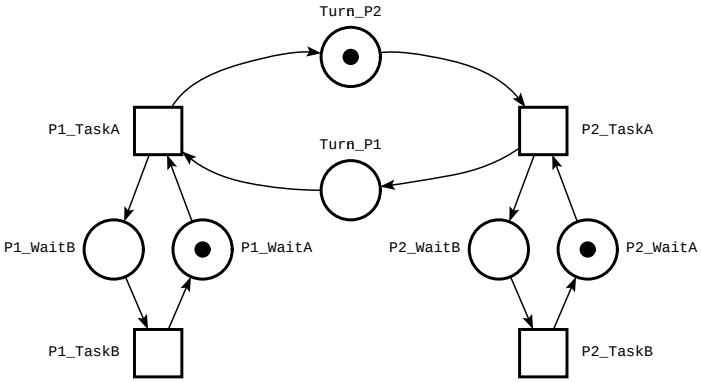

FIG. 1 – Scheduler de Milner

Ce réseau modélise le scheduler de Milner [Mil89] pour deux processus P1 et P2. Le scheduler va contrôler l'exécution des tâches A des différents processus : une tâche A d'un processus i ne peut être exécutée que lorsque les autres processus ont eux-mêmes exécuté cette tâche à tour de rôle. Une fois que les tâches A ont été exécutées par les processus, ceux-ci informent le scheduler, et ne sont plus soumis à celui-ci, jusqu'à ce qu'ils tentent de réexécuter la tâche A. Le scheduler est représenté par les places Turn_P_i. Elles permettent de dire à qui le scheduler va donner le droit d'exécuter A. Un processus i peut exécuter A s'il est en attente du droit d'exécution de A (P_i_WaitA) et si c'est à son tour d'exécuter A (Turn_P_i).
Essayons de vérifier la propriété : *"il n'est pas possible d'avoir plus d'un jeton sur l'ensemble des places Turn_P_i"* (i.e. : $\mathcal{M}(Turn\_P_1) + \mathcal{M}(Turn\_P_2) = 1$). Pour cela on va abstraire les places Turn_P_i en une seule place Turn. Il ne nous restera alors plus qu'à vérifier que pour toute exécution du système, il n'y a pas plus d'un jeton dans la place abstraite Turn. Voici le réseau que l'on obtient après fusion :

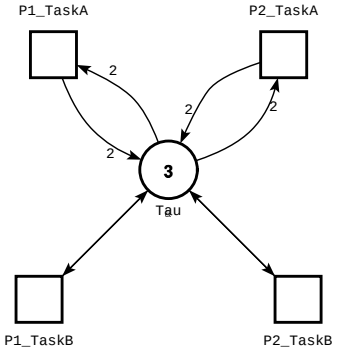

FIG. 3 – Scheduler de Milner - fusion de toutes les places.[1]

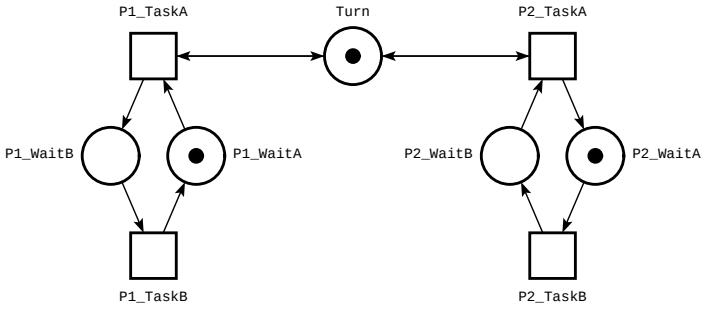

FIG. 2 – Scheduler de Milner - fusion des Turn_P$_i$.

Qu'avons-nous fait pour obtenir ce réseau ? Nous avons pris tous les jetons des places à renommer et nous les avons placés dans la place abstraite, le **marquage d'une place du réseau renommé est donc la somme des marquages des places renommées**. Il apparaît sur le graphe des marquages qu'il ne peut y avoir qu'un jeton dans la place Turn quelle que soit l'exécution considérée. Le nombre de jetons de Turn étant la somme des jetons des places concrètes, on peut en conclure qu'on aura toujours la propriété $\mathcal{M}(Turn\_P_1) + \mathcal{M}(Turn\_P_2) = 1$.
Compte tenu de la propriété qui nous intéresse, on n'a pas besoin d'observer l'alternance des P$_i$_Wait{A|B}. On pourrait donc être tenté de regrouper toutes les places en une seule τ. Le réseau associé est représenté ci-dessous. L'abstraction obtenue est bien sûr trop radicale pour permettre de statuer sur la propriété, puisque la place τ contient 3 jetons et donc l'abstraction nous permet que la somme des marquages des 4 places concrètes est égale à 3, ce qui est une conclusion trop faible pour pouvoir conclure que $\mathcal{M}(Turn\_P_1) + \mathcal{M}(Turn\_P_2) = 1$.
Un mécanisme d'abstraction plus sélectif va nous permettre de contourner cette difficulté : les places Turn_P$_i$ sont abstraites en une place unique Turn, les places P1_WaitA et P1_WaitB en une place P1_Wait, finalement les deux places P2_WaitA et P2_WaitB sont abstraites en P2_Wait. Voyons le réseau que nous obtenons :

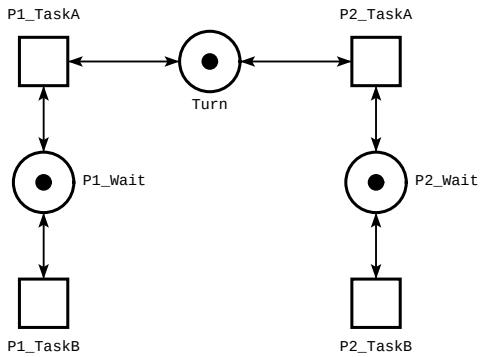

FIG. 4 – Scheduler de Milner - renommage final.

[1] Graphiquement, Pre(p,t) et Post(p,t) sont apposés sur les arcs reliant p à t (si ceux-ci sont supérieur à 1)

Il apparaît alors que la propriété est toujours vérifiée, et pour cause : le graphe des marquages n'a plus qu'un seul état, son état initial, et par conséquent il n'y a toujours qu'un seul jeton dans Turn.

2.3 Renommages

Le renommage s'utilise basiquement sur un ensemble de places. Néanmoins, pour décrire le comportement d'un réseau de Petri, on utilise le graphe des marquages. Les marquages étant des multiensembles de places, il nous faut donc étendre le renommage pour qu'il s'applique sur les multiensembles. De la même manière, un réseau de Petri est un graphe bipartite constitué de places et de transitions, il nous faut donc, pour généraliser le renommage de l'ensemble des places, étendre le renommage aux réseaux de Petri. Nous devrons utiliser le renommage de marquages dans la définition du renommage de réseau, pour la simple raison que Pre(.,t) et Post(.,t) sont des multiensembles.

Définition 5. Soit P et P_{ren} deux ensembles de places. On appelle **renommage de places** une application surjective[2] $\oslash : P \mapsto P_{ren}$.
Son inverse est l'application $\oslash^{-1} : P_{ren} \mapsto \mathcal{P}(P)$, telle que $\oslash^{-1}(p') = \{p \in P | \oslash (p) = p'\}$.

Définition 6. Soit P et P_{ren} deux ensembles de places et $\rho : P \mapsto P_{ren}$ un renommage de places. On appelle ***renommage de marquages***, une application surjective $\oslash_\rho$ paramétrée par ρ, telle que :
$\oslash_\rho : P^{\mathbb{N}} \mapsto P_{ren}^{\mathbb{N}}$
et
$\forall p' \in P_{ren} : \oslash_\rho(\mathcal{M})(p') = \sum_{p \in \rho^{-1}(p')} \mathcal{M}(p)$

Définition 7. Soit P et P_{ren} deux ensembles de places, $\rho : P \mapsto P_{ren}$ un renommage de places et $\pi : P^{\mathbb{N}} \mapsto P_{ren}^{\mathbb{N}}$ un renommage de marquages paramétré par ρ. On appelle ***renommage de réseau***, une application surjective $\oslash_\rho$ paramétrée par ρ, telle que pour tout réseau de Petri marqué R=<P,T,Pre,Post,$\mathcal{M}_0$>, on a : $\oslash_\rho(R) =< P_{ren}, T, Pre_\oslash, Post_\oslash, \pi(\mathcal{M}_0) >$,avec : $\forall t \in T : Pre_\oslash(.,t)$[3] $= \pi(Pre(.,t))$ et $Post_\oslash(.,t) = \pi(Pre(.,t))$

Nous utiliserons la notation ${}^\bullet t$ = Pre(.,t), et $t^\bullet$ = Post(.,t) par soucis de concision. Nous n'écrirons pas les applications définies ci-dessus avec leurs paramètres de renommage lorsqu'il n'y a pas ambiguïté.

Exemple 1. On est passé du réseau de la figure 1 au réseau de la figure 2 par le renommage de réseau paramétré par le renommage de places $\oslash_1$:

$$\oslash_1(p) = \begin{cases} \text{Turn, si p } \in \text{ \{ Turn\_P1, Turn\_P2 \}} \\ \text{p sinon.} \end{cases}$$

Le renommage de réseau paramétré par le renommage de places $\oslash_2$ transforme le réseau de la figure 1 en le réseau de la figure 4 :

$$\oslash_2(p) = \begin{cases} \text{Turn si p } \in \{Turn\_p1, Turn\_P2\} \\ \text{NonObs1 si p } \in \{P1\_WaitA, P1\_WaitB\} \\ \text{NonObs2 sinon.} \end{cases}$$

Théorème 1. $\mathcal{M} \in \mathcal{A}(\mathcal{R}, \mathcal{M}_0) \Rightarrow \oslash(\mathcal{M}) \in \mathcal{A}(\oslash(\mathcal{R}), \oslash(\mathcal{M}_0))$

De manière moins mathématique : Si un marquage $\mathcal{M}$ est accessible pour un RdP $\mathcal{R}$, alors le marquage obtenu après application du renommage sur $\mathcal{M}$ est accessible pour le RdP obtenu par application du renommage sur $\mathcal{R}$.

Démonstration. Soit $\mathcal{M} \in \mathcal{A}(R, \mathcal{M}_0)$, $\exists \sigma \in T^*$ tel que $\mathcal{M}_0 \xrightarrow{\sigma} \mathcal{M}$; montrons par récurrence sur la longueur de la séquence σ (notée $|\sigma|$), que $\oslash(\mathcal{M}_0) \xrightarrow{\sigma} \oslash(\mathcal{M})$.

Initialisation

$|\sigma| = 0$, évident car $\oslash(\mathcal{M}_0) \in \mathcal{A}(\oslash(R), \oslash(\mathcal{M}_0))$.

Hypothèse de récurrence :

$\mathcal{M} \in \mathcal{A}(\mathcal{R}, \mathcal{M}_0) \Rightarrow \oslash(\mathcal{M}) \in \mathcal{A}(\oslash(\mathcal{R}), \oslash(\mathcal{M}_0))$ avec $\mathcal{M}_0 \xrightarrow{\sigma} \mathcal{M} \wedge \oslash(\mathcal{M}_0) \xrightarrow{\sigma} \oslash(\mathcal{M}) \wedge |\sigma| = n$

[2] i.e. Dom($\oslash$)=P et Ran($\oslash$)=P_{ren}, avec Ran(f) est l'ensemble des images par f.
[3] $Pre(.,t) : P \rightarrow \mathbb{N}$, i.e. c'est le marquage minimum pour pouvoir tirer t.

Etape $n+1$

Soit $t \in T$ tel que $\mathcal{M}_0 \xrightarrow{\sigma} \mathcal{M} \xrightarrow{t} \mathcal{M}'$, on a donc : $|\sigma.t| = n+1$. Il faut d'abord vérifier que l'on peut tirer t à partir de $\oslash(\mathcal{M})$ — c'est à dire que, $\oslash({}^\bullet t)(p) \leqslant \oslash(\mathcal{M})(p)$, avec $p \in P_{ren}$ — puis vérifier que l'on peut atteindre le marquage $\oslash(\mathcal{M}')$ par le tir de la transition t.

a) Peut-on tirer la transition t à partir de $\oslash(\mathcal{M})$?

$\forall q \in P_{ren} : \oslash(\mathcal{M})(q) = \sum_{p \in \oslash^{-1}(q)} \mathcal{M}(p)$ et $\oslash({}^\bullet t)(q) = \sum_{p \in \oslash^{-1}(q)} Pre(p,t)$. Comme $\forall p \in P : Pre(p,t) \leqslant \mathcal{M}(p)$, alors, $\exists q \in P_{ren} : \sum_{p \in \oslash^{-1}(q)} Pre(p,t) \leqslant \sum_{p \in \oslash^{-1}(q)} \mathcal{M}(p)$, et par conséquent, $\oslash({}^\bullet t)(q) \leqslant \oslash(\mathcal{M})(q)$.

b) Peut-on, à partir de $\oslash(\mathcal{M})$ **et en tirant t, obtenir le marquage** $\oslash(\mathcal{M}')$ **?**

il nous faut prouver que $\oslash(\mathcal{M}') = \oslash(\mathcal{M}) - \oslash({}^\bullet t) + \oslash(t^\bullet)$.
$\forall p \in P : \exists q \in P_{ren} : \oslash(p) = q \wedge \oslash(\mathcal{M}) - \oslash({}^\bullet t)(q) + \oslash(t^\bullet)(q) = \sum_{p \in \oslash^{-1}(q)} \mathcal{M}(p) - \sum_{p \in \oslash^{-1}(q)} Pre(p,t) + \sum_{p \in \oslash^{-1}(q)} Post(p,t) = \sum_{p \in \oslash^{-1}(q)} \mathcal{M}'(p) = \oslash(\mathcal{M}')(q)$.

□

Nous connaissons donc maintenant un peu plus de chose sur notre renommage, mais pour le considérer comme une surapproximation (et donc une abstraction) il nous faut déterminer si le réseau obtenu par renommage simule le réseau concret. Cette simulation est une relation entre deux systèmes de transition étiquetés (STE). Le graphe des marquages est un tel STE. Définissons donc ce qu'est une relation de simulation entre deux systèmes de transitions étiquetés :

Définition 8. Soit deux STE, $\mathcal{STE} = \langle \Sigma, S, \rightarrow \subset (\Sigma \times S \times \Sigma) \rangle$ et $\mathcal{STE}' = \langle \Sigma', S', \rightarrow \subset (\Sigma' \times S' \times \Sigma') \rangle$, une relation binaire R, (R$\subset S \times S'$) est une simulation entre $\mathcal{STE}$ et $\mathcal{STE}'$ ssi elle vérifie :
$\forall (p,p') \in R : \forall t \in \Sigma : \forall q \in S$:Si $p \xrightarrow{t} q$ alors $\exists q' \in S' : p' \xrightarrow{t} q'$ et $(q,q') \in R$.

Théorème 2. *Un renommage de marquages* $\oslash$ *est une relation de simulation, et* $\oslash(R) = \langle P_{ren}, T, \oslash(Pre), \oslash(Post), \oslash(\mathcal{M}_0) \rangle$ *est simulé par R=*$\langle P, T, Pre, Post, \mathcal{M}_0 \rangle$.

Démonstration. $R_\oslash$ simule R ssi il existe une simulation ρ telle que $(\oslash(\mathcal{M}_0), \mathcal{M}_0) \in \rho$. Or par définition de $\oslash$, on a $(\oslash(\mathcal{M}_0), \mathcal{M}_0) \in \oslash$.
Il reste donc à voir si $\oslash$ est une relation de simulation. D'après le théorème 1, on a : si $s \xrightarrow{t} s'$, alors $\oslash(s) \xrightarrow{t} \oslash(s')$. Par définition de $\oslash$ $(s, \oslash(s)) \in \oslash$ et $(s', \oslash(s')) \in \oslash$, par conséquent, on a bien : $s \xrightarrow{t} s' \wedge (s, \oslash(s)) \in \oslash \Rightarrow \oslash(s) \xrightarrow{t} \oslash(s') \wedge (s', \oslash(s')) \in \oslash$, et donc $\oslash$ est une simulation. □

Maintenant que l'on a prouvé que l'ensemble des marquages avant renommage est un sous-ensemble de l'ensemble des marquages après renommage, on peut parler de l'application de renommage comme d'une *abstraction par renommage* Nous renvoyons le lecteur aux documents [Mic96],[Loi94] pour les propriétés des simulations.

Exemple 2. Pour mieux comprendre ce qu'il se passe lorsque l'on génère le graphe des marquages d'une abstraction de renommage, voici les deux graphes des marquages concret et abstrait, ainsi que la relation d'abstraction par renommage (en pointillé) des marquages. Sur la droite on a le graphe des marquages du réseau concret et sur la gauche le graphe des marquages du réseau abstrait obtenu par le renommage de places $\oslash_1$ (transformation de la figure 1 en la figure 2). La relation figurant en pointillé entre les marquages n'est autre qu'une relation de simulation comme l'atteste le théorème 2. Sur la figure, les états qui s'abstraient en un seul sont groupés côte à côte. Ainsi, on peut voir que, globalement, les transitions qui partent (resp. arrivent) vers (resp. de) ces états groupés (qui vont ne former plus qu'un seul états après abstraction), sont aussi présentes dans l'abstraction, c'est ce que nous garantie la relation de simulation.

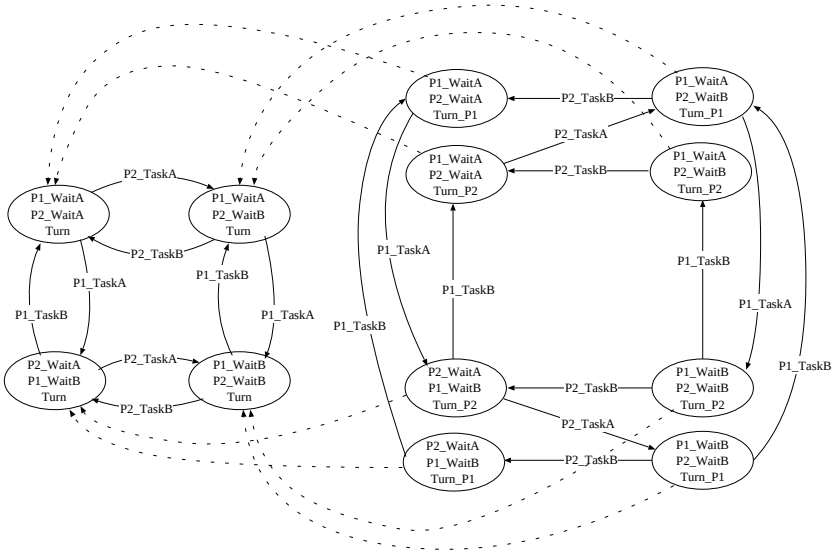

FIG. 5 – Graphes des marquages abstraits et concrets.

Corollaire 1. $\mathcal{L}(R, \mathcal{M}_0) \subset \mathcal{L}(\oslash(R), \oslash(\mathcal{M}_0))$

Définition 9. L'ensemble des concrétisations d'un marquage $\mathcal{M}' \in P_{ren}^{\mathbb{N}}$, paramétrées par l'application $\oslash^{-1}$: $P_{ren} \rightarrow P$, est défini par :
$Concr(\mathcal{M}') = \{\mathcal{M} | \forall p' \in P_{ren} : \sum_{p \in \oslash^{-1}(p')} \mathcal{M}(p) = \mathcal{M}'(p')\}$
C'est l'inverse de l'application de renommage de marquages.

Exemple 3. Soit $P = \{p1, p2, p3\}$ et $P_{ren} = \{p1, \tau\}$, $\oslash(p2) = \tau et \oslash(p3) = \tau$. Les concrétisations possibles du marquage $\{p1, 2 * \tau\}$ [4] sont :
Concr($\{p1, 2 * \tau\}$) = $\{\{p1, 2 * p2\}, \{p1, p2, p3\}, \{p1, 2 * p3\}\}$.

Corollaire 2. *En étendant Concr sur les ensembles de marquages, on a :*

$$\mathcal{A}(\mathcal{R}, \mathcal{M}_0) \subset Concr(\mathcal{A}(\oslash(\mathcal{R}), \oslash(\mathcal{M}_0)))$$

Définition 10. Deux marquages $\mathcal{M}_1$ et $\mathcal{M}_2$ sont **équivalent au renommage** $\oslash$ **près** (noté $\mathcal{M}_1 \equiv_{\oslash} \mathcal{M}_2$) ssi $\oslash(\mathcal{M}_1) = \oslash(\mathcal{M}_2)$.

La propriété de surapproximation permet d'identifier la classe de propriétés potentiellement préservée par notre mécanisme d'abstraction et le type de décision que l'on pourra mettre en oeuvre : on ne pourra statuer que sur des propriétés d'état, caractérisées par les marquages. De façon informelle, l'ensemble d'états abstraits construits représente un sur-ensemble - modulo la relation d'équivalence $\equiv_{\oslash}$ - de l'espace d'états concrets. Deux aspects sont donc à considérer : le fait que l'on travaille avec une surapproximation et le fait que l'on travaille avec des états abstraits agrégés par la relation d'équivalence induite par le renommage des places.
Impact de la sur-approximation : Si une propriété d'état est satisfaite sur le sur-ensemble alors elle sera satisfaite sur l'espace d'états accessibles et l'on pourra conclure. Dans la négative on ne pourra pas statuer car il faudrait savoir si les états violant la propriété sont effectivement accessibles ou non. De même si une propriété d'état n'est satisfaite pour aucun élément du sur-ensemble alors a fortiori elle n'est satisfaite pour aucun état accessible. De nouveau le fait qu'elle soit satisfaite sur des éléments du sur-ensemble nous empêchera de pouvoir conclure car rien ne garantit que ces éléments soient effectivement accessibles.
Impact de l'abstraction des états : Une propriété d'état φ sera satisfaite sur un état *abstrait* m si elle est vraie sur tous les états concrets qui lui sont associés : $m_a \vDash \varphi$ ssi $\forall m_c \in Concr(m_a) : m_c \vDash \varphi$. On voit l'incidence de la fonction de renommage qui sera utilisée : plus le renommage de places sera abstrait, plus la concrétisation d'un état sera grande et moins les "chances " de vérifier une propriété sur un état abstrait seront importantes.
Considérons maintenant ce que nous avons jusqu'à présent appelé "propriété d'état" : il s'agit uniquement de propriétés invariantes de nature "propositionnelle " en considérant les places du réseau comme des variables atomiques.

Définition 11. Une proposition atomique peut être extraite d'un marquage en construisant le nom des places indexé par leur nombre de jeton dans ce marquage : à partir de $\mathcal{M}(p) = k$, on construit la proposition atomique p_k.

[4] $2 * \tau$ est un raccourci pour dire que l'élément τ est présent deux fois dans le multiensemble.

Définition 12. Si un marquage est tel que : $\mathcal{M}(p) = k$, alors on dit que p_k est valide pour $\mathcal{M}$ (ou bien que la place p possède k jetons est valide pour $\mathcal{M}$). On le notera : $\mathcal{M} \vDash p_k$.

Définition 13. Soit $AP_p = \bigcup_{p \in P} P_i : i{>}0$ et $CP(AP_p)$ le calcul propositionnel construit sur AP_p.

Le fait de travailler avec des réseaux Place/Transition, non forcément bornés, nous conduit à considérer un ensemble potentiellement infini de variables propositionnelles.

Définition 14. Pour un ensemble d'états E, $\varphi \in CP(AP_p)$ on note $E \vDash \varphi$ ssi $\forall e \in E : e \vDash \varphi$.

Proposition 1. *L'ensemble des propriétés préservées par l'abstraction est caractérisé par :*
$f \in CP(AP_p) : \mathcal{A}(\oslash(\mathcal{R}), \oslash(\mathcal{M}_0)) \vDash f \Rightarrow \mathcal{A}(\mathcal{R}, \mathcal{M}_0) \vDash f$

3 Exemples

Nous allons voir des exemples dans un premier temps de l'utilisation de l'abstraction avec un renommage en une seule place : τ (i.e. on cache les places que l'on ne veut pas voir dans τ).

3.1 Les philosophes

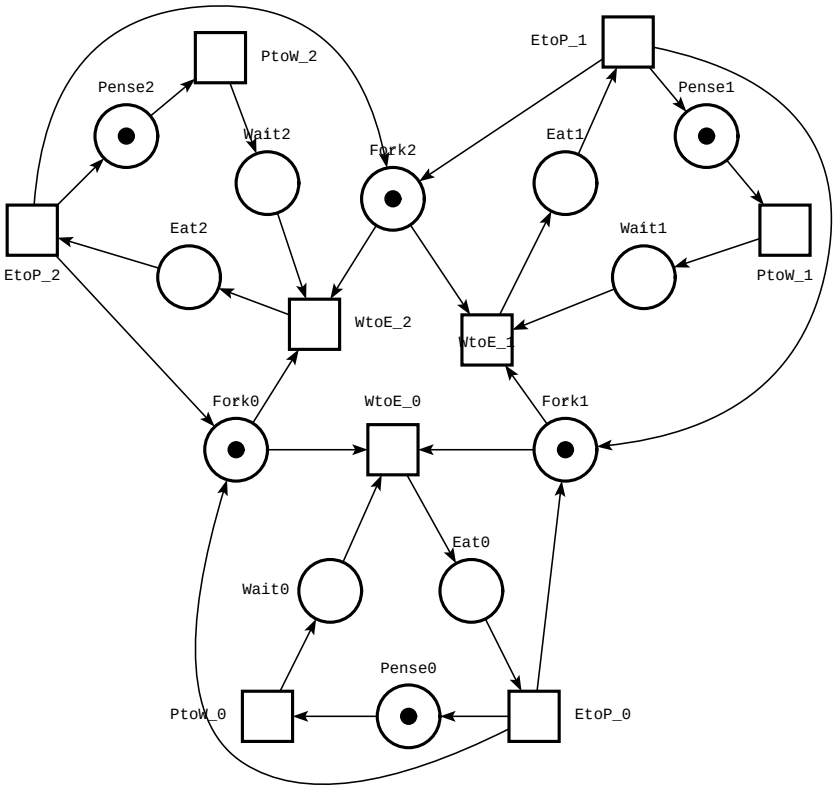

FIG. 6 – Les philosohes. Réseau concret.

Les philosophes peuvent penser, manger, ou attendre pour manger. Il y a une fourchette entre deux philosophes. Un philosophe ne peut manger qu'avec les deux fourchettes qui sont en face de lui. On va s'intéresser ici à la propriété : *Il ne peut y avoir deux philosophes voisins qui mangent en même temps* . Le réseau de la Figure 6 représente le réseau concret. En utilisant le renommage suivant $\oslash(p) = \begin{cases} \tau \text{ si } p \in \{Pense_i, Wait_i\} \\ p \text{ sinon.} \end{cases}$, nous obtenons le réseau de la Figure 7 :

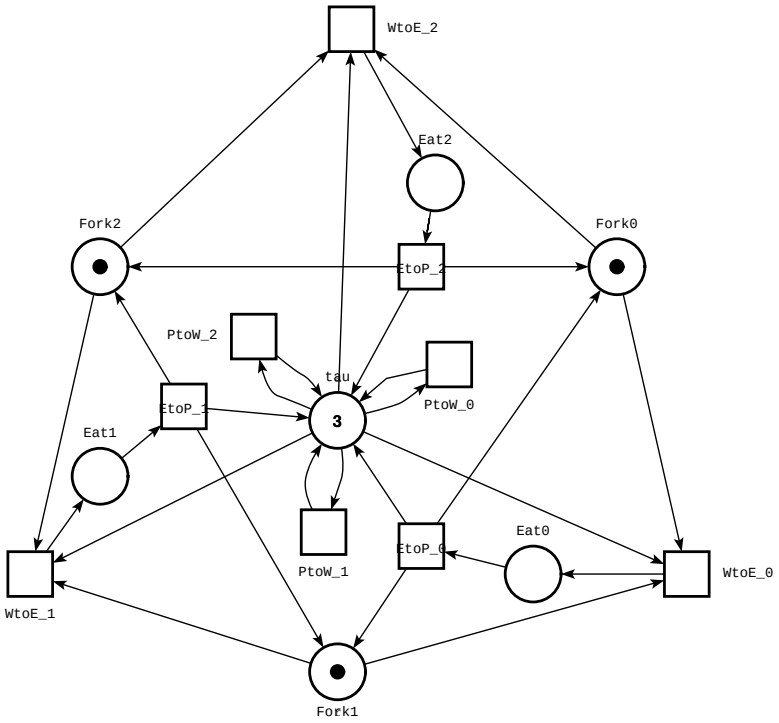

FIG. 7 – Les philosophes. Abstraction de Penser et Attendre.

Sur ce réseau on voit qu'un philosophe ne va manger que si la garde de WtoE_i est satisfaite, c'est à dire qu'il faut que l'on ait tau,Fork_i, et Fork_i+1, ce qui correspond bien au fonctionnement que l'on attend, et en effet si l'on construit le graphe des marquages, on voit que l'on n'a pas de marquages violant la propriété à vérifier.
Voyons maintenant le gain de l'espace d'état.

philosophes	états réseau complet	états réseau abstrait
3	20	4
4	56	7
5	152	11
6	416	18
7	1136	29
8	3104	47
9	8480	76
20	??	15127

L'espace d'état abstrait explose toujours mais de manière moins importante.

3.2 Comparaisons et compositions de techniques

Nous avons comparé trois techniques d'ordres partiels avec notre abstraction : deux d'entre elles ne préservent que les blocages, la dernière préserve LTL. Les deux premières techniques (Stubborn sets et Persistent steps) ne préservent que les blocages. Ces techniques sont pour autant utilisables pour vérifier des propriétés locales : il suffit simplement d'ajouter une transition qui ne sera tirable qu'en cas de violation de la propriété à vérifier et dont le tir conduira à un état de blocage. Pour notre exemple, on va ajouter une transition ayant pour précondition Eat1 et Eat2 et sans postcondition. Par cette astuce, la violation de la propriété introduit un état de blocage.
Nous avons obtenu, pour les philosophes, les résultats du tableau ci-dessous :

philosophes	gdm[5]	LTL_Csg [6]	st [7]	psg [8]	gdm_a [9]
3	20	20	11	8	4
4	56	49	23	16	7
5	152	117	39	27	11
6	416	281	67	49	18
7	1136	667	111	86	29
8	3104	1156	185	152	47
9	8480	3941	308	266	76
20	??	??	98955	98748	15127

De plus, vu que l'on a modifié le réseau pour les deux techniques qui ne préservaient que les blocages, l'on a un ensemble de techniques, qui ont au moins pour base commune de préserver l'invariance des propriétés locales. Il est donc possible de combiner les techniques d'ordre partiels avec la technique d'abstraction par renommage, dans l'espoir d'en voir les bénéfices cumulés. Malheureusement, sur les exemples que nous avons présentés ici, les techniques d'ordre partiel ne permettent pas de gagner encore un peu sur la taille de l'espace d'états.

3.3 Base de données

Cet exemple a été introduit dans [Jen87] pour illustrer la méthode des marquages équivalents dans le contexte des réseaux de Petri colorés. Il a été repris par la suite dans [Val89] pour mettre en oeuvre les ensembles persistants, puis dans [VAM96] pour la méthode des pas couvrants, et [RVB02] pour le graphe des pas persistants. On modélise ici la demande de mise à jour d'un manager de base de données parmi un ensemble de managers. Le manager est l'entité qui gère l'accès en écriture à une base de donnée. Pour pouvoir mettre à jour, le manager i va d'abord demander leur accord aux autres managers j (i_send). Ceux-ci vont alors recevoir le message (i_j_rec) et envoyer leur accord (i_j_resp), le manager i peut alors faire l'update, i.e. passer en section critique (i_goSC). Quand il a fini l'update il revient dans son état initial idle_i en faisant i_ra. La figure 8 montre le réseau de Petri pour deux managers.

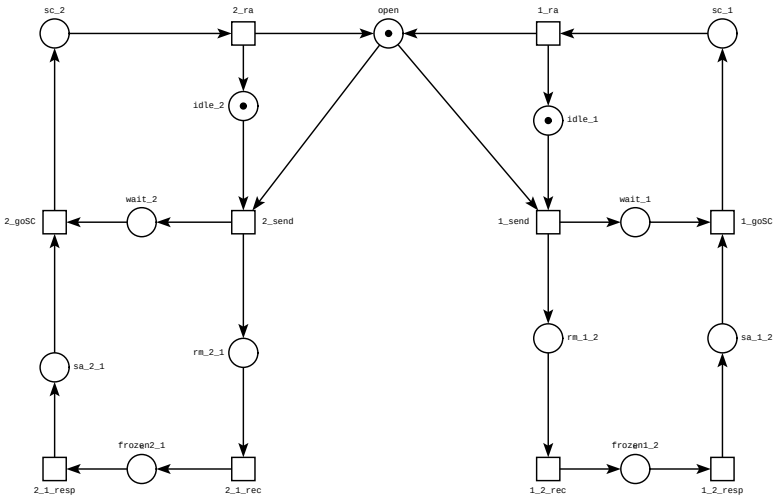

FIG. 8 –

On souhaite vérifier que l'on a bien qu'*un seul manager à la fois en section critique.* Pour cela on doit observer i_sc et open (sans le open l'abstraction devient non bornée). La figure 9 montre le réseau obtenu après abstraction du réseau de la figure 8.

[9] Graphe des marquage du réseau concret
[9] Graphe des pas couvrants préservant LTL
[9] Stubborn sets
[9] Persistent steps graph
[9] Graphe des marquages abstrait

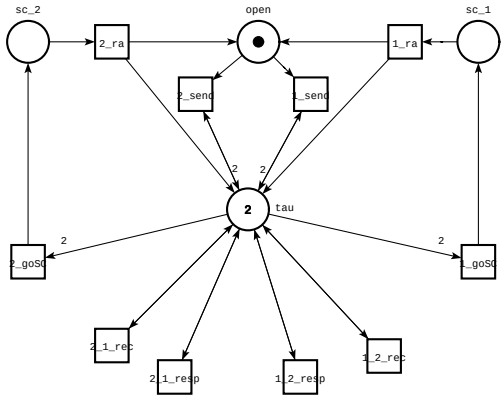

FIG. 9 –

On obtient des résultats positifs : le nombre d'états de l'abstraction n'augmente pas de façon exponentielle mais linéaire (nombre d'états = 2∗nombre de managers + 3). On est passé à un accroissement linéaire de l'espace d'états grâce au fait que les i_j_resp ne modifient plus le marquage, alors que ceux sont clairement des transitions coûteuses, dans le sens où il y a $n(n-1)$ de ces transitions, qui modifient chacune leur jeton.

nombre managers	réseau complet	réseau abstrait
3	31	9
4	113	11
5	411	13
6	1465	15
7	5111	17
8	17505	19
9	59059	21

4 Accessibilité, borne et temps

4.1 Caractère Borné

Nous allons maintenant aborder le problème de la décision du caractère borné lorsque l'on passe par une abstraction. Commençons par un exemple qui pose problème : un réseau borné peut très bien avoir une abstraction non-bornée.

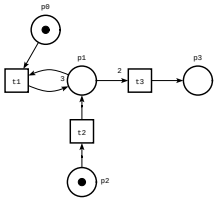

FIG. 10 –

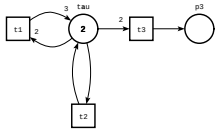

FIG. 11 –

Le réseau de la figure 10 est le réseau concret. Il est borné et il a un état de blocage. En faisant l'abstraction de ce réseau, avec $P_{obs} = \{p_3\}$ (fig. 11), on a aussi un état de blocage, mais il est non borné.
La question se pose alors pour un réseau non borné : le réseau abstrait est-il aussi systématiquement non borné ?

Théorème 3. *Si un réseau est non-borné, alors son abstraction est non-bornée.*

Démonstration. Un réseau marqué est non borné si $\exists p \in P : \forall k \in \mathbb{N} : \exists \mathcal{M} \in \mathcal{A}(\mathcal{R}, \mathcal{M}_0) : \mathcal{M}(p) > k$.
Montrer que le réseau est non borné est équivalent à exhiber une séquence répétitive croissante [KM69], il vient qu'un réseau est non borné ssi :
$\exists \mathcal{M} \in \mathcal{A}(\mathcal{R}, \mathcal{M}_0) : \exists \sigma \in T^* : \exists p \in P : \mathcal{M} \xrightarrow{\sigma} \mathcal{M}' \wedge \mathcal{M}' \geqslant \mathcal{M} \wedge \mathcal{M}'(p) > \mathcal{M}(p)$.
D'après le théorème 1, $\mathcal{M} \xrightarrow{\sigma} \mathcal{M}' \Rightarrow \oslash(\mathcal{M}) \xrightarrow{\sigma} \oslash(\mathcal{M}')$, il reste à prouver que le marquage augmente aussi strictement pour l'abstraction :
$\mathcal{M}' \geqslant \mathcal{M} \wedge \exists p \in P : \mathcal{M}'(p) > \mathcal{M}(p)$

$$\Leftrightarrow \sum_{p\in P} \mathcal{M}'(p) \geqslant \sum_{p\in P} \mathcal{M}(p) \wedge \exists p \in P : \mathcal{M}'(p) > \mathcal{M}(p)$$
$$\Leftrightarrow \sum_{p\in P} \mathcal{M}'(p) > \sum_{p\in P} \mathcal{M}(p)$$
$$\Leftrightarrow \sum_{q\in P_{ren}} \left(\sum_{p\in \oslash^{-1}(q)} \mathcal{M}'(p) \right) > \sum_{q\in P_{ren}} \left(\sum_{p\in \oslash^{-1}(q)} \mathcal{M}(p) \right)$$
$$\Leftrightarrow \sum_{q\in P_{ren}} \oslash(\mathcal{M}')(q) > \sum_{q\in P_{ren}} \oslash(\mathcal{M})(q)$$

□

Remarque 1. La contraposée est plus intéressante : ***Si une abstraction est bornée, alors ses réseaux concrets sont aussi bornés.***
Il est aussi à noter l'utilisation du signe d'équivalence dans la preuve du théorème. Cela ne veut pas dire que la réciproque est vrai, car si on a $\mathcal{M} \geqslant \mathcal{M}' \bigwedge \exists p \in P : \mathcal{M}(p) > \mathcal{M}'(p)$ pour l'abstraction (c'est à dire que l'abstraction est non bornée), la concrétisation vérifiera aussi cela (les signes $\Leftrightarrow$nous assurent cela)... mais cette concrétisation ne sera pas forcément accessible.

4.2 Non préservation des états de blocages

Le réseau de la figure 12 admet un état de blocage - pour le marquage {p1}. Or si l'on décide de tout cacher, on se retrouve avec le réseau abstrait de la figure 13, qui lui n'a pas d'état de blocage.

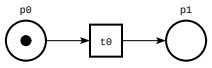

FIG. 12 – Réseau concret : blocage.

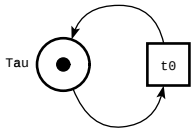

FIG. 13 – Réseau abstrait : pas de blocage

Voici maintenant la situation inverse : le réseau de la figure 14 ne possède pas d'état de blocage car p0 ne peut jamais avoir 2 jetons et donc on ne pourra jamais tirer fin. Par contre, si l'on renomme les deux places en τ, on obtient une place qui a le nombre requis de jetons pour pouvoir tirer fin, comme on peut le voir sur le réseau abstrait de la figure 15.

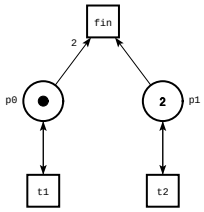

FIG. 14 – Réseau concret : pas de blocage.

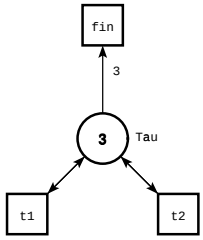

FIG. 15 – Réseau abstrait : blocage.

De ces situations problématiques, où l'on remarque que l'on ne peut rien décider à partir de l'abstraction, nous avons pu obtenir une propriété :

Théorème 4. *Toute concrétisation d'un état de blocage dans un réseau abstrait est un état de blocage dans le réseau concret.*

Démonstration. Soit $Bad(t, \mathcal{M}) = \{p \in P | Pre(p, t) > \mathcal{M}(p)\}$. Un blocage s'écrit alors : $\forall t \in T : \exists q \in P_{ren} : q \in Bad(t, \mathcal{M})$. Prouvons par l'absurde que la concrétisation est un état de blocage : il faut donc démontrer que l'on ne peut avoir : $\exists t \in T : \forall p \in P : p \in Bad(t, \mathcal{M})$ dans le réseau concret. On a alors : $\exists t \in T : \forall p \in P : \sum_{p\in \oslash^{-1}(q)} \mathcal{M}(p) \geqslant \sum_{p\in \oslash^{-1}(q)} Pre(p, t)$, ce qui est la définition de l'abstraction $\oslash$, d'où : $\exists t \in T : \forall q \in P_{ren} : \oslash(\mathcal{M})(q) \geqslant \oslash(^\bullet t)(q)$, et donc : $\exists t \in T : \forall q \in P_{ren} : q \notin Bad(t, \mathcal{M})$, ce qui contredit notre hypothèse de départ qui était qu'on avait un blocage dans le réseau abstrait. □

Ce théorème exprime le fait que toutes les concrétisations d'un état de blocage sont aussi des états de blocages dans le réseau concret, mais il ne dit pas si ces états de blocages sont réels ou non, c'est à dire si ils sont accessibles ou non (dans le réseau concret).

4.3 Accessibilité.

Il n'est pas direct de décider de l'accessibilité d'un état à partir de l'abstraction. Il y a néanmoins selon les cas des perspectives pour se sortir de cette situation.
D'une part considérons le cas où l'abstraction ne détecte pas l'accessibilité de l'état demandée. Ce cas est très gênant car en n'ayant que la donnée fournie par l'abstraction, on ne peut absolument rien dire (il nous est nécessaire de parcourir tout l'espace d'état concret pour s'assurer de la réelle accessibilité sans posséder aucune information pouvant nous guider dans le processus).
D'autre part, considérons le cas où l'état est présent dans l'abstraction (état but). Il nous faut alors décider si cet état existe vraiment dans le concret. De quelle informations dispose-t-on ? Le graphe des marquages de l'abstraction va nous fournir une information grossière du chemin à prendre pour arriver potentiellement à l'état but. On peut ainsi imaginer une exploration à la volée de l'espace d'état concret guidée par les chemins menant à l'état but de l'abstraction. De cette exploration à la volée on obtiendra (potentiellement) plus rapidement un résultat qu'avec une exploration exhaustive si l'état but est accessible dans le réseau concret. S'il n'est pas réellement accessible, il est obligatoire d'explorer tout l'espace d'états concrets pour s'en rendre compte.

4.4 Réseaux de Petri temporels

Pour la prise en compte du temps, nous utilisons les réseaux de Petri temporels [MF76]. Ces réseaux étendent les réseaux classiques en associant à chaque transition un intervalle de tir pendant lequel une transition peut être tirée.

Définition 15. Un réseau de Petri temporel est un tuple <P,T,Pre,Post,$\mathcal{M}_0$,I_s> où <P,T,Pre,Post,$\mathcal{M}_0$> est un réseau de Petri marqué, et $I_s : T \rightarrow \mathbf{I}^+$ est une fonction appelée *Intervalle Statique*, avec $\mathbf{I}^+$ l'ensemble des intervalles réels non vides à bornes rationnelles non négatives.

Définition 16. Un état d'un Réseau de Petri temporel est un couple c=(m,I), où m est un marquage et I est une fonction appelée la fonction intervalle. $I : T \rightarrow \mathbf{I}^+$ associe un intervalle de temps avec chaque transition sensibilisée par m.

Appliquer notre méthode telle quelle sur un réseau de Petri temporel n'est pas possible. En effet, si l'on considère le réseau de la fig 16, avec l'abstraction par le renommage : $\begin{cases} \oslash(p0) = \tau \\ \oslash(p1) = \tau \end{cases}$, on obtient le réseau de la figure 17. Dans le réseau concret seule la transition t1 est tirable, or dans le réseau abstrait seule la transition t0 l'est. On a donc perdu la surapproximation : l'ensemble d'états abstraits n'est plus un surensemble de l'ensemble d'états concrets.

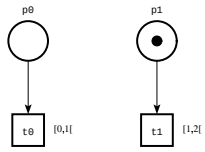

FIG. 16 – réseau temporel concret

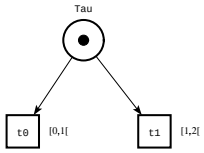

FIG. 17 – réseau temporel abstrait

Ce contre-exemple semble suggérer qu'il faille au minimum relâcher les contraintes de date de tir au plus tard des transitions possédant des préconditions qui auront été abstraites. Des études sont en cours pour envisager des solution et expérimenter leurs intérêts.

5 Conclusion

Ce papier a présenté les premiers résultats d'une étude qui vient de débuter sur l'utilisation de surapproximation pour réduire l'explosion combinatoire dans le cadre de modèles décrits par des réseaux de Petri Place/Transition. Cette surapproximation est obtenue en considérant une fonction de renommage qui va permettre de cacher/confondre certaines places du réseau. De façon informelle, l'ensemble d'états de la surapproximation représente un surensemble - modulo une relation d'équivalence - de l'espace d'états effectivement accessible. Plus précisément, le graphe des états abstraits est simulé par le graphe des états concrets. L'approche proposée permet uniquement de vérifier des propriétés propositionnelles invariantes.

Le fait de travailler avec une surapproximation n'offre qu'un mécanisme de semi-décision : si une propriété établie sur le sur-ensemble est satisfaite, elle le sera à fortiori également sur chacun de ces sous-ensembles, dans la négative on ne peut statuer. L'approche n'est que semi-automatique dans la mesure où elle se base sur une fonction de renommage dont le choix est crucial. Ce choix repose sur l'expertise du modélisateur : un renommage conduisant à une abstraction trop radicale peut conduire à une surapproximation trop grossière pour permettre de statuer sur la propriété voire à un espace d'états abstraits infini.
Pour pouvoir expérimenter, un premier prototype a été écrit et couplé avec l'environnement Tina. Le prototype est extrêmement simple puisqu'il ne s'agit que de mettre en place un renommage des places du réseau : la partie fusion des places (confondues par le renommage) est une fonctionnalité déjà offerte par Tina.
Quelques expérimentations ont été menées sur des benchmarks classiques du domaine. Des comparaisons ont été menées avec les approches de réduction basées sur les techniques "ordre partiel " que nous utilisons habituellement. Il faut noter que ces deux approches sont complémentaires et peuvent être combinées facilement. Nous exploiterons ces pistes ultérieurement. L'extension à d'autres propriétés (présence de blocages) nous semble difficile et l'application de cette approche par renommage à des réseaux temporels nécessite un aménagement que nous devons étudier et expérimenter.

Références

[And82] Charles André. Behaviour of a place - transition net on a subset of transitions. In *Selected Papers from the First and the Second European Workshop on Application and Theory of Petri Nets*, pages 131–135. Springer-Verlag, 1982.

[And83] Charles André. Structural transformations giving b-equivalent pt-nets. In *Selected Papers from the 3rd European Workshop on Applications and Theory of Petri Nets*, pages 14–28. Springer-Verlag, 1983.

[Ber86] G. Berthelot. Checking properties of nets using transformations. *Lecture Notes in Computer Science : Advances in Petri Nets 1985*, 222 :19–40, 1986. NewsletterInfo : 24.

[Ber87] G. Berthelot. Transformations and decompositions of nets. In *Advances in Petri nets 1986, part I on Petri nets : central models and their properties*, pages 359–376. Springer-Verlag, 1987.

[BRV03] B. Berthomieu, P-O. Ribet, and F. Vernadat. L'outil tina – construction d'espaces d'états abstraits pour les réseaux de petri et réseaux temporels. In *Modélisation des Systèmes Réactifs, MSR'2003*. Hermes, 2003.

[HV01] S. Haddad and F. Vernadat. Méthodes d'analyse des réseaux de petri. In *Les Réseaux de Petri. Modèles fondamentaux*, pages 69–117. Hermes Sciences, 2001.

[Jen87] Kurt Jensen. Coloured petri nets. In Brauer, W., Reisig, W., and Rozenberg, G., editors, *Lecture Notes in Computer Science : Petri Nets : Central Models and Their Properties, Advances in Petri Nets 1986, Part I, Proceedings of an Advanced Course, Bad Honnef, September 1986*, volume 254, pages 248–299. Springer-Verlag, 1987.

[KM69] R. Karp and R. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3 :147–195, 1969.

[Loi94] C. Loiseaux. *Vérification symbolique de programmes réactifs à l'aide d'abstractions.* PhD thesis, Université Joseph Fournier de Grenoble, Janvier 1994.

[MF76] P. M. Merlin and D. J. Farber. Recoverability of communication protocols : Implications of a theoretical study. *IEEE Trans. Comm.*, 24(9) :1036–1043, September 1976.

[Mic96] F. Michel. *Symétries d'architecture et de données. Validation de systémes répartis.* PhD thesis, LAAS-CNRS, 1996.

[Mil89] R. Milner. *Communication and concurrency.* Prentice-Hall, Inc., 1989.

[RVB02] P. Ribet, F. Vernadat, and B. Berthomieu. On combining the persistant sets method with persistent method with the covering steps graph method. In *LNCS 2529 : In Proceedings of Forte 2002*, 2002.

[Val89] Antti Valmari. Stubborn sets for reduced state space generation. In *Proceedings of the 10th International Conference on Application and Theory of Petri Nets, 1989, Bonn, Germany ; Supplement*, pages 1–22, 1989.

[VAM96] F. Vernadat, P. Azéma, and F. Michel. Covering step graph. In *LNCS 1091 : 17th Int. Conf. on Application and Theory of Petri Nets 96, Osaka, Japan*. Springer Verlag, 1996.