
L'outil TINA

Construction d'espaces d'états abstraits pour les réseaux de Petri et réseaux temporels

Bernard Berthomieu — Pierre-Olivier Ribet — François Vernadat

LAAS-CNRS
7, avenue du Colonel Roche
F-31077 Toulouse Cedex
{berthomieu,ribet,vernadat}@laas.fr

RÉSUMÉ. Outre les fonctionnalités classiques d'édition graphique, l'outil logiciel Tina propose un certain nombre de constructions de graphes de comportements pour les réseaux de Petri et les réseaux temporels. Différentes techniques sont utilisées pour extraire des vues du comportement du réseau préservant certaines classes de propriétés de l'ensemble des états. Pour les réseaux de Petri, ces abstractions permettent de limiter l'explosion combinatoire, leur construction utilise les techniques de pas couvrant et/ou les techniques à base d'ensembles persistants. Pour les réseaux temporels, dont le graphe d'états est en général infini, elles permettent d'obtenir des abstractions de comportement finies à base de classes d'états.

ABSTRACT. Beside the common graphic editing facilities, the software tool Tina proposes construction of a number of representations for the behavior of Petri nets or Time Petri nets. Various techniques are used to extract vues of the behavior of nets preserving certain classes of properties of their state spaces. For Petri nets, these abstractions help preventing combinatorial explosion, they rely on techniques using covering steps and/or persistent sets. For Time Petri nets, that have in general infinite state spaces, they allow to finitely represent their behavior, in terms of state classes.

MOTS-CLÉS : Réseaux de Petri, Réseaux temporels, Espaces d'états Abstraits, Pas couvrants, Ensembles persistants, Classes d'états, Mise en œuvre.

KEYWORDS: Petri nets, Time Petri nets, Abstract State Spaces, Covering Steps, Persistent Sets, State classes, Implementation.

1. Introduction

Tina (TIme Petri Net Analyser, <http://www.laas.fr/tina>) est un environnement logiciel permettant l'édition et l'analyse de réseaux de Petri [HAD 01] et réseaux de Petri temporels [MER 76].

Outre les fonctions classiques d'édition et d'analyse énumérative (graphe de marquages, arbre de couverture) ou structurelle (semi-flots), *Tina* propose la construction d'espaces d'états abstraits permettant la vérification de classes spécifiques de propriétés. Les classes de propriétés proposées incluent : les propriétés générales d'accessibilité (absence de blocage, vivacité), les propriétés spécifiques basées sur la structure linéaire de l'espace d'états concrets (celles exprimables en logique temporelle linéaire, ou capturées par les équivalences de test) ou sur sa structure arborescente (celles exprimables en logiques temporelles arborescentes, ou capturées par la bisimulation).

Les abstractions proposées opèrent sur des systèmes temporisés ou non. Dans le cas de systèmes temporisés, pouvoir considérer un espace d'états abstrait est un impératif car l'espace d'états concret est en général infini. Ces abstractions sont obtenues par la technique des graphes de classes et ses évolutions récentes. Dans le cas de systèmes atemporels, offrir un espace d'état abstrait permet de limiter les risques d'explosion combinatoire. Pour cela, *Tina* fait appel aux techniques de réduction à base d'"ordre partiel" que sont les ensembles persistants et les pas couvrants.

Tina n'est pas un "model-checker" au sens où il ne permet pas à lui seul - sauf bien sûr dans le cas des propriétés générales d'accessibilité - de décider de la satisfaction d'une certaine propriété. *Tina* intervient en amont du model-checker en lui fournissant un graphe d'états réduit sur lequel il pourra effectuer plus efficacement la vérification. La réduction opérée par *Tina* permet de préserver les propriétés "linéaires" ou "arborescentes" de l'espace d'états qui sera ultérieurement analysé. Afin d'obtenir une chaîne complète de "model-checking", *Tina* est couplé avec MEC [ARN 92] pour la vérification de formules du μ -calcul et Aldebaran [FER 91] pour la vérification de pré-ordres ou d'équivalences de comportement.

Cet article propose un panorama des possibilités de l'outil *Tina*, il obéit au plan suivant : La section 2 décrit brièvement les méthodes d'analyse classiques offertes par *Tina*. La section 3 s'intéresse aux systèmes atemporels et présente les différentes méthodes d'exploration "ordre partiel" proposées dans *Tina* en les associant aux différentes classes de propriétés, linéaires ou arborescentes, préservées. La section 4 s'intéresse à l'analyse des réseaux temporels. La technique des graphes de classe est rappelée. Les spécialisations de cette technique pour la vérification de propriétés linéaires ou arborescentes sont présentées. La section 5 décrit les interfaces de *Tina* et son architecture logicielle ; interface utilisateur et facilités d'édition, formats d'entrée et de sortie des composants, interopérabilité avec les outils de model-checking.

Pour plus de précisions sur les méthodes utilisées, le lecteur est invité à consulter les articles cités, notamment [RIB 02, VER 03] pour les aspects "ordre partiels", et [BER 91, BER 03] pour les aspects temporels.

2. Méthodes classiques

Un premier groupe d'outils de construction de comportements proposé par *Tina* est composé des constructions "classiques" que sont le graphe des marquages d'un réseau de Petri et son graphe de couverture [HAD 01].

La construction du graphe des marquages est interrompue si une place est détectée "non bornée". Le graphe de couverture permet de détecter toutes les places non bornées, sa construction est basée sur l'algorithme classique de Karp et Miller [KAR 69]. Comme cette dernière construction ne produit pas de graphe unique, plusieurs heuristiques sont proposées, visant à réduire soit le temps de calcul, soit le nombre de ω -marquages produits.

Enfin, parmi les méthodes classiques, *Tina* propose de calculer des ensembles générateurs de semi-flots sur les places et sur les transitions du réseau. Cet outil est le premier d'un groupe d'outils à venir qui concerneront les méthodes d'analyse dites "structurelles"(utilisant les invariants déduits des flots et semi-flots) [HAD 01].

3. Techniques des ordres partiels

Les techniques de réduction dites à "ordre partiel" réduisent l'explosion combinatoire en éliminant l'une de ses causes : la représentation du parallélisme par l'entrelacement des actions. Lorsque deux (ou n) composants offrent une action en parallèle, la sémantique d'entrelacement représente ce comportement par un losange (un hypercube dans le cas général) où les différentes séquences, constituées par les mêmes transitions mais apparaissant dans un ordre différent, convergent vers le même état. Comme tous ces chemins convergent vers le même état, l'idée directrice commune aux approches ordre partiel consiste à n'explorer - lorsque c'est suffisant - qu'un chemin particulier parmi tous les chemins équivalents possibles.

Cette stratégie de base a été initialement mise en oeuvre par A. Valmari avec la théorie des ensembles Stubborn [VAL 90] dans le cadre des réseaux de Petri, et généralisée ensuite par Godefroid et Wolper [GOD 91] pour donner lieu à la notion d'ensembles "persistants".

L'approche des "Pas couvrants" [VER 96, VER 03] prolonge l'approche "ordre partiel" classique de réduction des entrelacements de type "ensemble persistant". Comme les séquences possédant la même trace conduisent aux mêmes états, l'ordre d'occurrence de chaque événement dans la séquence est arbitraire, ainsi que les états intermédiaires la constituant. Ceci conduit à considérer, sous certaines conditions, certains événements "indépendants" comme un "pas de transitions" et à considérer la réalisation de ce pas de transitions comme un événement atomique.

La Figure 1 montre le bénéfice obtenu dans le cas de la dérivation de n événements parallèles. L'exploration exhaustive donne lieu à un hypercube d'ordre n : on obtient un nombre exponentiel d'arcs et d'états. En explorant un seul chemin par trace, le facteur exponentiel disparaît : nombre linéaire d'arcs et d'états. L'approche par

“pas” fournit un résultat optimal puisque le nombre d’états (d’arcs) est indépendant du nombre d’évènements.

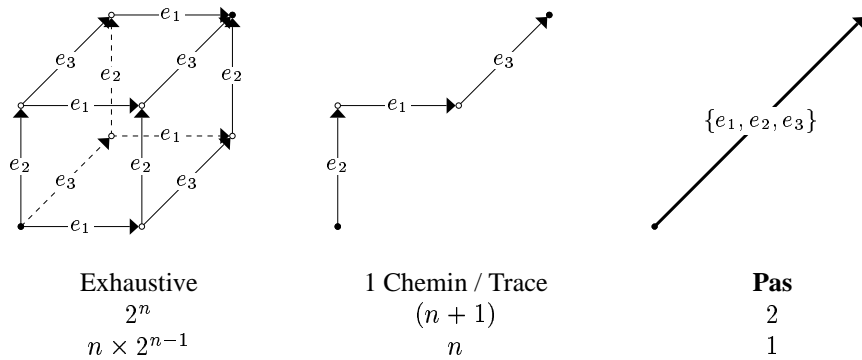


Figure 1. Dérivation de 3 événements indépendants

L’intérêt d’une technique de réduction, au-delà du facteur de compression qu’elle offre effectivement, réside dans la capacité d’analyse qu’elle procure. La stratégie dépend donc de la classe de propriétés que l’on cherche à vérifier. Les techniques “ordre partiel” sont des techniques assez générales qui peuvent être déclinées suivant le type de propriétés que l’on cherche à vérifier [GOD 96]. Nous illustrons maintenant les techniques mises en oeuvre dans Tina en les classant suivant les différentes classes de propriétés qu’elles permettent de vérifier.

Nous prendrons comme exemple conducteur le scheduler de Milner [MIL 85]. L’approximation structurale de la relation d’indépendance utilisée dans Tina, est : t_1 et t_2 sont indépendantes ssi $Pre(t_1) \cap Pre(t_2) = \emptyset$.

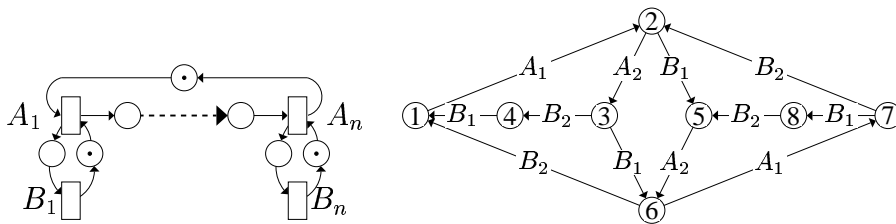


Figure 2. Scheduler de Milner et graphe complet $ST\mathcal{E}$ pour $n = 2$

Le système complet (scheduler + n sites) est représenté Figure 2, partie gauche. n sites exécutent cycliquement l’action A_i puis l’action B_i . Un scheduler contraint l’exécution du système complet de telle sorte que les n sites alternativement exécutent les actions A_1, A_2, \dots, A_n . Le système de transitions étiquetées correspondant à un scheduler à 2 sites est représenté Figure 2, partie droite.

3.1. Préservation des états de blocage

Un état de blocage est un état puits, i.e. un état à partir duquel plus aucun événement ne peut avoir lieu. Les méthodes suivantes permettent donc de construire un graphe $ST\mathcal{E}_r$ réduit, mais qui comporte exactement les mêmes états de blocage que le graphe complet $ST\mathcal{E}$.

3.1.1. Ensembles persistants [GOD 96]

L'approche des ensembles persistants consiste à ne considérer, dans chaque état, qu'un sous-ensemble des transitions sensibilisées. La façon de choisir ce sous-ensemble influence la taille et bien sûr les propriétés préservées.

Un ensemble d'actions T est *persistant* dans un état p ssi toute action sur une trace, partant de p et composée d'actions hors de T , est indépendante de toute action dans T . Les "ensembles persistants" peuvent être dérivés directement de la description formelle du système (réseaux Place/Transition, systèmes Variable/Transition, produit d'automates synchronisés). Le graphe réduit se calcule directement à partir de la description, en ajoutant la règle suivante à n'importe quel algorithme d'énumération classique : *pour chaque état atteint p , calculer un "ensemble persistant" associé à p et n'exécuter que les actions dans cet ensemble.*

La Figure 3 présente un exemple de graphe utilisant les ensembles persistants pour le scheduler de Milner à 2 sites : dans l'état initial 0 une seule transition A_1 est sensibilisée par conséquent l'ensemble persistant est $\{A_1\}$. Dans l'état 1, deux transitions B_1 et A_2 sont sensibilisées et indépendantes donc on peut choisir l'ensemble persistant $\{B_1\}$ ou $\{A_2\}$; dans l'exemple le second est choisi. Le Tableau 1 donne les résultats obtenus avec des ensembles persistants sur différents exemples. Dans la totalité des cas, le facteur exponentiel a disparu.

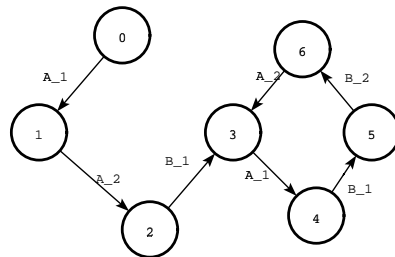


Figure 3. Scheduler de Milner - graphe Persistant pour $n = 2$

3.1.2. Graphe de pas couvrants

Les graphes de pas couvrants (*GPC*) ont été définis dans [VER 96]. Les états d'un *GPC* sont des états du graphe de marquages complet. Les transitions de ce graphe sont des "pas", i.e. des ensembles de transitions indépendantes. La présence d'un pas

de transition, entre deux états s et s' , dans le GPC représente de façon compacte toutes les séquences de transitions de ce pas qui existent dans le graphe complet entre les états s et s' . Réciproquement, les GPC offrent une propriété de couverture : toute séquence de transitions dans le graphe de marquages peut être prolongée de façon à être couverte par une séquence de pas dans le GPC . Notons que tout STE peut être considéré comme son propre graphe de pas couvrants : il suffit de prendre $\lambda = \emptyset$.

La Figure 4 représente un graphe de pas couvrants du scheduler de Milner pour $n = 2$. On peut illustrer la propriété de couverture sur cet exemple. Considérons la séquence de franchissement du graphe des marquages $A_1.A_2.B_2.B_1$ qui constitue un cycle en l'état 1 (cf Figure 2). Cette séquence n'existe pas dans le graphe de pas. Par contre, complétée par A_1 , elle figure dans la séquence de pas $\{A_1\}.\{A_2, B_1\}.\{A_1, B_2\}$. Ces deux séquences mènent de l'état 1 à 2 et possèdent la même trace de Mazurkiewicz (i.e $[A_1.A_2.B_2.B_1.A_1]_{(T, \lambda^c)} = [\{A_1\}.\{A_2, B_1\}.\{A_1, B_2\}]_{(T, \lambda^c)}$).

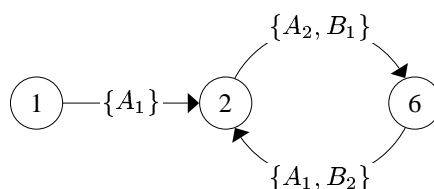


Figure 4. Scheduler de Milner - graphe de pas couvrants pour $n = 2$

Les graphes de pas couvrants préservent les états de blocage, mais également les traces maximales (modulo l'équivalence de traces de Mazurkiewicz [MAZ 86]). Ils permettent aussi de décider de la vivacité au sens réseau de Petri du terme. Ils gardent donc plus d'états que nécessaire pour la préservation des blocages.

Le Tableau 1 donne les résultats obtenus avec des GPC sur les exemples déjà traités avec les ensembles persistants. Dans la totalité des cas, le facteur exponentiel a disparu. Il est intéressant de noter que suivant le cas il vaudra mieux utiliser les ensembles persistants (ex : Philosophe) ou les GPC (ex : Scheduler). La section suivante présente les PSG , une technique mixte combinant "ensembles persistants" et "pas de transitions". Cette technique permet en pratique d'obtenir une réduction aussi bonne, voire meilleure, que celle obtenue en utilisant les "ensembles persistants" et les "graphes de pas".

3.1.3. Graphe de pas persistants

Les graphes de pas persistants sont une spécialisation des graphes de pas couvrants dédiés à l'unique préservation des états de blocages. La technique des PSG [RIB 02] consiste à utiliser de façon combinée les ensembles persistants et les pas de transitions. Dans chaque état un ensemble persistant est choisi, et les pas de transitions ne sont construits que sur ce sous-ensemble.

[RIB 02] montre que la méthode des graphes de pas persistants est une généralisation des ensembles persistants et des graphes de pas. Comme la technique des ensembles persistants, la technique des graphes de pas persistants doit utiliser une stratégie pour choisir, en chaque état exploré, les ensembles de transitions persistants. [RIB 02] montre qu'il ne peut pas exister une stratégie fixe d'exploration pour les graphes de pas persistants qui soit meilleure que chaque stratégie de choix d'ensembles persistants. En d'autres termes, une exploration à base d'ensembles persistants sans utiliser de pas peut être meilleure que toute stratégie utilisant des pas.

	<i>Modèle</i>	<i>Exhaustif</i>	<i>Persistant</i>	<i>GPC</i>	<i>PSG</i>
1	<i>Scheduler</i> 300	$2^n * n \approx 6.10^{92}$	1 394	301	301
2	<i>Philosopher</i> 8	103 681	233	31 231	227
3	<i>Data base</i> 10	196 831	191	31	31
4	<i>Token Ring</i> 10	35 840	99	52	51
5	<i>Manuf. system</i>	2 034	455	979	360
6	<i>Async. Buffer</i> 7	972	37	12	12

Tableau 1. *Table de comparaison*

Le Tableau 1 montre que dans la pratique le *PSG* peut donner des résultats aussi bons que le meilleur résultat obtenu par chacune des deux techniques de base. Le Tableau 2 montre sur le modèle de la piscine¹ [BÉR 99] qu'ils peuvent même être bien meilleurs (en nombre d'états et en temps de calcul).

<i>K</i>	<i>Persistant</i>		<i>GPC</i>		<i>PSG</i>	
10	857	0:00:00	367	00:00	87	00:00
235	4 602 707	9:47:00	219 742	01:32	2 112	00:01
240	X	—	229 217	01:42	2 157	00:01
500	X	—	997 517	31:08	4 497	00:02
600	X	—	X	—	5 397	00:02
200 000	X	—	X	—	1 799 997	24:11

Tableau 2. *Évaluation sur l'exemple de la piscine*

3.2. Préservation de la structure linéaire

Nous présentons ci-dessous les abstractions réalisées préservant la structure linéaire de l'espace d'états. Dans le premier cas, l'abstraction réalisée est un point d'entrée des outils permettant la comparaison de modèles tels *Aldébaran* [FER 91]. Dans le second cas, la technique visée est le model-checking de formules logiques pour des outils tels que *MEC* [ARN 92].

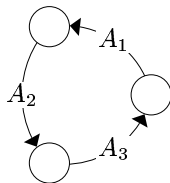
1. Le modèle de la piscine est paramétré par un entier K représentant à la fois le nombre de cabines disponibles, et le nombre de paniers à vêtements disponibles.

3.2.1. Sémantiques de refus

[VER 97] présente une spécialisation des *GPC* permettant la préservation des “sémantiques de refus” [VGL 90]. L’abstraction est opérée en fonction des transitions que l’on désire observer. De façon intuitive, la préservation de la structure linéaire de l’espace d’états est obtenue au prix de deux conditions supplémentaires par rapport aux *GPC* généraux :

- Un pas ne comporte aucune transition observable,
- Un pas ne peut contenir une transition en conflit avec une transition observable.

Nous considérons de nouveau le scheduler de Milner avec une observation de nature globale : on observe les synchronisations entre le scheduler et les “différents” sites $T_{Obs} = \{A_i : i \in Sites\}$.



Ci-contre est représenté le graphe minimal équivalent pour $n = 3$. Dans le cas général, celui-ci est constitué par n états et n arcs tandis que la taille du *GPC* préservant l’équivalence de refus évolue de façon quadratique $n^2 + n$ états et $2 \times n^2$ arcs.

3.2.2. Logique temporelle linéaire LTL_{-X}

À partir d’une formule LTL_{-X} - et plus précisément de ses variables atomiques - on peut définir un sous-ensemble de transitions “significatives” par rapport à cette formule [PEL 98] ; ce sont les transitions dont l’exécution peut faire changer l’évaluation de la formule. [RIB 03] montre que la conservation des traces observables (finies et infinies) entre deux structures de Kripke étiquetées est une condition suffisante pour assurer qu’elles sont “stuttering” équivalentes. Cette propriété garantissant la préservation de l’évaluation des formules de LTL_{-X} [PEL 98] permet d’envisager le model-checking de formules LTL_{-X} sur les *LGPC*.

3.3. Préservation de la structure arborescente

[VER 96] propose une spécialisation des *GPC* permettant la préservation de la bisimulation faible. L’abstraction opère de nouveau en fonction des transitions que l’on désire observer. De façon intuitive, la préservation de la structure arborescente de l’espace d’états est obtenue au prix de trois conditions supplémentaires par rapport aux *GPC* généraux :

- Les pas de transitions ne sont constitués qu’à partir de transitions “libres de tout conflit” (transition indépendante de toutes les autres),
- Un pas de transition peut comporter au plus une transition observable,
- Pour ne pas risquer de perdre la structure arborescente, un sous-pas constitué uniquement par les transitions inobservables est aussi considéré.

4. Graphes de classes des réseaux temporels

4.1. Réseaux temporels, états, classes d'états

Réseaux temporels : Les réseaux temporels (ou Time Petri nets) sont des réseaux de Petri dans lesquels un intervalle réel non négatif à bornes rationnelles $I_s(t)$ est associés à chaque transition t [MER 76]. La fonction I_s est appelée fonction *Intervalle statique*.

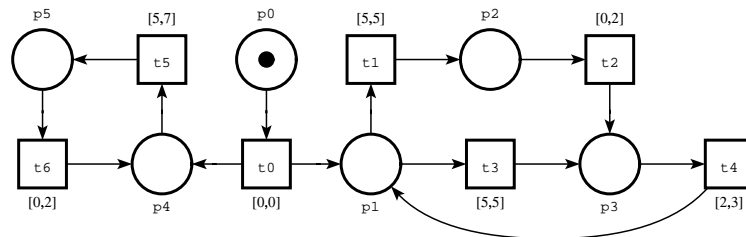


Figure 5. Un réseau temporel

États, échéanciers : Un état d'un réseau temporel est un marquage m associé à une fonction I , qui associe un intervalle réel non négatif à chaque transition sensibilisée par m . Initialement, $s_0 = (m_0, I_0)$, avec $I_0(t) = I_s(t)$ pour chaque transition sensibilisée par le marquage initial m_0 . $Min(I(t))$ et $Max(I(t))$ désignent les bornes inférieures et supérieures de l'intervalle $I(t)$, respectivement (si $I(t)$ est non supérieurement borné, on posera $Max(I(t)) = \infty$).

Les réseaux temporels évoluent de la façon suivante : supposons que t soit sensibilisée dans l'état $s = (m, I)$, et le soit devenue pour la dernière fois à une date τ . Alors t ne peut être tirée depuis s avant la date $\tau + Min(I(t))$ et doit l'être au plus tard à la date $\tau + Max(I(t))$, sauf si le tir d'une autre transition désensibilise la transition t avant que celle-ci ne soit tirée. Le tir des transitions est de durée nulle.

Cette règle définit sur l'ensemble des états une relation d'accessibilité temporisée notée $\xrightarrow{t@\theta}$. On a $s = (m, I) \xrightarrow{t@\theta} s'$ si la transition t peut être tirée depuis l'état s à la date relative θ ($\theta \in I(t)$). L'ensemble des états d'un réseau temporel est l'ensemble des états accessibles depuis l'état initial s_0 , muni de la relation d'accessibilité temporisée $\xrightarrow{t@\theta}$. On notera $s \xrightarrow{t} s'$ la relation $(\exists \theta)(s \xrightarrow{t@\theta} s')$.

Un échéancier est une suite de transitions datées $(t_i @ \theta_i)_{1 \leq i \leq n}$. Son *support* est la séquence de transitions $t_1 \dots t_n$. Le *domaine de tir* d'un état (m, I) est l'ensemble de vecteurs $\{\phi | (\forall k)(\phi_k \in I(k))\}$, avec leurs composantes indexées par les transitions sensibilisées.

Classes d'états : Les transitions pouvant être franchies à toute date dans leurs intervalles temporels, les états d'un réseau temporel admettent en général une infinité de successeurs. Toute représentation finie de l'espace d'états passe par des groupements d'états. Ces groupements sont appelés *classes d'états*.

Dans leur définition la plus générale, les espaces de classes d'états sont des recouvrements de l'ensemble des états munis d'une relation de transition \xrightarrow{t} satisfaisant la propriété (EE) ci-dessous (c and c' désignent des ensembles d'états). Nous supposons de plus que tous les états de chaque classe ont même marquage.

$$(EE) (\forall t \in T)(\forall c, c')(c \xrightarrow{t} c' \Leftrightarrow (\exists s \in c)(\exists s' \in c')(s \xrightarrow{t} s'))$$

Dans ce cadre, plusieurs espaces de classes différents peuvent être définis, selon les propriétés de l'espace d'états qu'ils préservent. Notons que les transitions entre classes d'états ne portent plus d'information temporelle, les classes d'états permettent d'abstraire le temps du comportement d'un réseau temporel. *Tina* propose deux groupes de constructions, préservant respectivement les propriétés exprimables en logiques temporelles à temps linéaire, et celles à temps arborescent.

Systèmes caractéristiques : Pour donner une définition synthétique des espaces de classes discutés, il est commode d'introduire les systèmes caractéristiques : pour chaque séquence de tir σ , les dates (relatives) $\underline{\theta}$ auxquelles peuvent être franchies les transitions de la séquence, et l'état atteint (décrit par son domaine de tir, variables $\underline{\phi}$), sont liés par un système d'inéquations de la forme suivante, appelé *système caractéristique de la séquence* σ :

- (1) $P\underline{\theta} \leq \underline{p}$
- (2) $\underline{0} \leq \underline{\phi}, \underline{e} \leq \underline{\phi} + M\underline{\theta} \leq \underline{l}$, avec $\underline{e}_k = \text{Min}(I_s(k)), \underline{l}_k = \text{Max}(I_s(k))$

Le sous système (1) décrit les vecteurs de dates relatives possibles $\underline{\theta}$ pour les transitions de σ , et, pour chaque $\underline{\theta}$, (2) décrit le domaine de tir de l'état atteint.

Ces systèmes, aisément calculés [BER 03], forment un arbre KG de racine K_ϵ . Le système K_σ caractérise l'ensemble des états (généralement infini) accessibles depuis l'état initial par des échéanciers de support σ . L'arbre KG est à branchement fini, mais il a autant de nœuds qu'il existe de séquences de tirs tirables, une infinité donc, en général.

4.2. Préservation des propriétés LTL, classes d'états linéaires

4.2.1. Classes d'états linéaires, construction LSCG [BER 83, BER 91]

La première construction fournie par *Tina*, concernant les réseaux temporels, est celle, classique, introduite dans [BER 83]. Elle peut être expliquée comme suit.

Pour toute séquence de tir σ , soit C_σ l'ensemble des états accessibles par des échéanciers de support σ (comme caractérisé par le système K_σ). Considérons la relation d'équivalence \cong satisfaite par deux de ces ensembles d'états si ceux-ci ont même marquage et même domaine de tir, le marquage d'un ensemble étant défini comme celui de ses états, et son domaine de tir comme la réunion des domaines de tirs des états le constituant.

Le *graphe des classes d'états linéaires* ($LSCG$) est l'ensemble des ensembles C_σ , pour toute séquence σ tirable, considérés modulo \cong , et muni de la relation de transition : $c \xrightarrow{t} c'$ ssi $(\exists s \in c)(\exists s' \in c')(s \xrightarrow{t} s')$.

Il coïncide avec le graphe obtenu depuis l'arbre des systèmes caractéristiques en identifiant les nœuds équivalents par \cong (c.a.d. dont les systèmes d'inéquations ont même ensemble de solutions après élimination des variables θ). Une construction directe est proposée dans [BER 83]. Le $LSCG$ du réseau représenté Figure 5 admet ainsi 83 classes et 160 transitions.

Le graphe $LSCG$ préserve les propriétés du graphe d'états exprimables en logiques temporelles à temps linéaire (comme LTL), d'où son nom. Il peut être montré, en effet, que si deux systèmes caractéristiques K_σ et $K_{\sigma'}$ sont équivalents par \cong , σ et σ' conduisant au même marquage, alors les sous arbres de KG qu'ils définissent sont isomorphes. Ceci implique la préservation par le $LSCG$ des traces et traces maximales du graphe des états, et donc des propriétés LTL .

4.2.2. Classes d'états linéaires avec multi-sensibilisation [BER 01]

Dans la construction du $LSCG$, chaque transition sensibilisée est associée à au plus une variable temporelle. Une variante de la construction $LSCG$ est présentée dans [BER 01] dans laquelle les transitions plusieurs fois sensibilisées sont associées à autant de variables temporelles.

Cette interprétation trouve plusieurs usages pratiques, notamment lorsque la présence de jetons dans les places du réseau est interprétée comme l'arrivée d'événements. Le traitement de [BER 01] consistant à ordonner les instances de sensibilisation des transitions selon leur âge, gère une forme de symétrie de l'espace d'états. Cette construction, le $LSCG$ avec interprétation de la multi-sensibilisation, est aussi proposée par Tina.

4.2.3. Classes d'états linéaires fortes, construction $SSCG$ [BER 03]

Une troisième construction préservant aussi les propriétés LTL , est proposée. Comme précédemment, soit C_σ l'ensemble des états accessibles par des échanciers de support σ . Les *classes d'états linéaires fortes* ($SSCG$) sont exactement ces ensembles C_σ , pour toute σ tirable, c'est à dire considérés sans autre relation d'équivalence que l'égalité d'ensembles d'états.

Une construction du $SSCG$ est proposée dans [BER 03]. Les classes linéaires fortes sont représentables par un marquage associé à un système d'inéquations "en horloges". Le système associé à la classe C_σ est le sous-système (1) de K_σ , auquel les équations $\underline{\gamma} = M\underline{\theta}$ sont ajoutées (les $\underline{\gamma}$ sont les variables d'horloge, une par transition sensibilisée), les variables $\underline{\theta}$ sont ensuite éliminées.

Il est montré dans [BER 03], qu'une relation d'équivalence \equiv est calculable pour ces systèmes, telle que $C_\sigma \equiv C_{\sigma'}$ si et seulement si ces classes dénotent des ensembles d'états égaux. En résumé, si toutes les transitions ont un intervalle statique borné, alors

l'équivalence \equiv coïncide avec l'égalité des ensemble de solutions des systèmes d'horloges. Si ce n'est pas le cas, alors une opération supplémentaire, dite de "relaxation" doit être appliquée aux systèmes d'horloges avant leur comparaison.

Comme le *LSCG*, le *SSCG* préserve les propriétés *LTL*, mais il est moins compact et son calcul est plus complexe. La construction *SSCG* ne présente donc que peu d'intérêt par elle-même. En fait, elle n'est fournie que parce qu'elle constitue le point de départ de la construction préservant les propriétés de branchement, décrite dans ce qui suit. Le *SSCG* du réseau Figure 5 admet 107 classes et 205 transitions.

4.3. *Préservation des propriétés CTL*, classes d'états atomiques [BER 03]*

Les propriétés de branchement sont celles exprimables dans les logiques temporelles à temps arborescent comme *CTL**, ou dans les logiques modales comme *HML* ou le μ -calcul. En l'absence de transitions silencieuses, on sait que la préservation de ces propriétés est une conséquence de la bisimulation [BRO 88]. Tout graphe de classes bisimilaire au graphe des états d'un réseau temporel préserve donc ses propriétés de branchement.

La construction d'un tel graphe est offerte par *Tina*, sous le nom de *graphe des classes d'états atomiques*, ou *ASCG*. Classiquement, ce graphe est construit par une technique similaire à la technique du "raffinement de partition" de [PAI 87][TRI 96]. Une classe est atomique, ou stable, si elle l'est par rapport à toute autre classe. Une classe est stable par rapport à une autre si aucun de ses états n'a de successeur dans cette dernière, ou tous ses états en ont un. Le graphe des classes atomiques est obtenu par raffinement du graphe des classes linéaires fortes, ses classes sont partitionnées jusqu'à ce que toute classe soit stable.

Toute classe instable est partitionnée en exhibant une contrainte linéaire non redondante dans son système d'horloges, et telle que cette contrainte est nécessaire pour qu'un état de la classe ait un suivant dans la classe cible considérée.

Cette construction est en général coûteuse, mais elle permet de vérifier le plus grand ensemble de propriétés, quelques résultats d'expérimentations figurent dans [BER 03]. Les classes atomiques ont même représentation que les classes linéaires fortes, c'est à dire un marquage associé à un système d'horloges. Le *ASCG* du réseau Figure 5 admet 101 classes et 431 transitions.

4.4. *Préservation des propriétés temporisées*

Enfin, il est une classe de propriétés d'un grand intérêt pratique, mais pour laquelle aucune construction dédiée n'est proposée : celle des propriétés "quantitatives", comme exprimées dans les logiques temporelles "temporisées".

Bien qu'aucune construction ne soit proposée qui permettrait de vérifier toutes les propriétés de $TCTL$, par exemple, certaines de ces propriétés peuvent être vérifiées à l'aide de la technique standard des observateurs.

L'idée est de coder une propriété "temporisée" p du réseau à vérifier en une propriété p' de LTL ou CTL^* du réseau augmenté par un réseau auxiliaire "observateur", de façon à ce que p soit vraie sur le réseau de départ ssi p' l'est sur le réseau étendu. p' étant une propriété de LTL ou CTL^* , il suffit ensuite d'invoquer la construction de *Tina* qui préserve ce groupe de propriétés. La technique est applicable pour une large famille de propriétés, notamment celles se réduisant à des propriétés d'accessibilité.

5. Interface utilisateur

La boîte à outil *Tina* est conçue de façon modulaire. Les modules incluent :

- Un éditeur graphique (de réseaux, d'automates), incluant des fonctions de dessin de réseaux ou d'automates ;
- Un outil de constructions de comportements ;
- Un outil d'analyse structurelle (préliminaire).

Ces modules peuvent être utilisés de façon combinée ou indépendante :

Utilisé seul, l'éditeur produit des fichiers exploitables ultérieurement par les outils de construction de comportement ou d'analyse. Mais les outils de constructions peuvent aussi être invoqués sans sortir de l'éditeur, et celui-ci permet d'éditer ou de dessiner les résultats.

Utilisés seuls, les outils de construction et d'analyse fonctionnent comme des filtres, ils sont donc facilement insérables dans des chaînes de développement existantes. Leur ligne de commande permet de sélectionner les paramètres de construction. Ils admettent en entrée des réseaux de Petri ou temporels décrits dans un format graphique (produit par l'éditeur), ou textuel (écrit à la main ou par programme), et produisent des résultats selon un choix de formats de sortie.

Le format textuel d'entrée est simple, intuitif, et compact. Plusieurs formats de sortie sont disponibles. L'outil *Tina* n'impose l'usage d'aucun vérificateur de modèle spécifique. A ce jour, *Tina* produit des résultats dans divers formats, incluant un format "en clair" pour des utilisations pédagogiques, le format d'entrée "automate" de l'outil *Aldébaran* pour des analyses d'équivalences, ainsi que le format d'entrée "système de transitions" de l'outil *MEC*, pour la vérifications de formules du μ -calcul. Ainsi, *Tina* peut être utilisé comme "front-end" par bon nombre d'outils de vérification, éventuellement au prix de l'écriture d'un filtre spécifique traduisant l'une des sorties possibles de *Tina* dans un format compris par le vérificateur utilisé.

Une capture d'écran d'une session *Tina* typique est reproduite en Figure 6, avec un réseau temporel en cours d'édition, un résultat textuel de construction de comportement, et une représentation graphique du comportement produit.

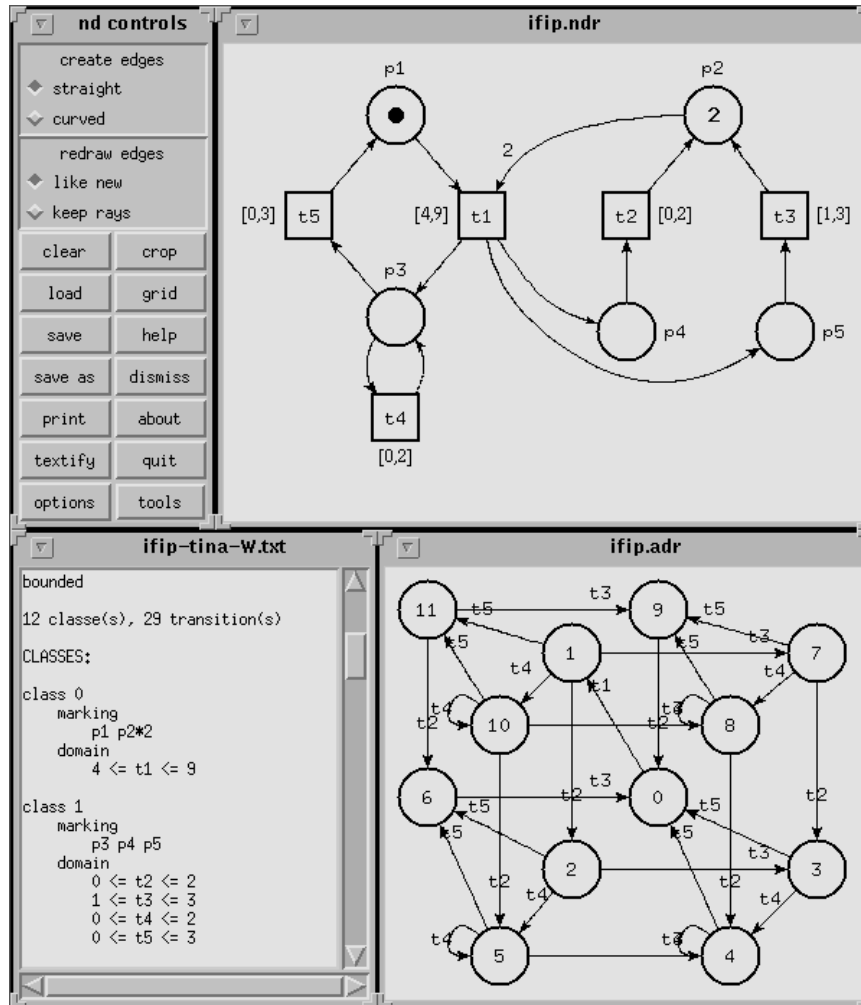


Figure 6. Capture d'écran d'une session Tina

6. Conclusion

Cet article introduit *Tina*, un environnement logiciel permettant l'édition et l'analyse de réseaux de Petri et temporels. Outre les fonctionnalités standard d'édition et d'analyse classique, l'originalité de *Tina* est de proposer la construction d'espaces d'états abstraits préservant la structure linéaire ou arborescente des espaces d'états concrets.

Différentes abstractions sont proposées pour permettre de vérifier différentes classes de propriétés allant des propriétés générales d'accessibilité, aux propriétés spécifiques - linéaires ou arborescentes - exprimées par des logiques temporelles ou des équiva-

lence de comportement. Deux formes d'abstractions fondées sur les "ordres partiels" et les "graphes de classes" permettent respectivement de prendre en compte des systèmes atemporels ou temporisés. Dans le cas de systèmes temporisés, pouvoir considérer un espace d'états abstrait est un impératif car l'espace d'états concret est en général infini. Dans le cas de systèmes atemporels, offrir un espace d'état abstrait permet de limiter les risques d'explosion combinatoire.

Le domaine d'utilisation de *Tina* est large. La variété de constructions qu'il propose en fait un outil utile pour l'enseignement de ces techniques. D'autre part, *Tina* est utilisé dans plusieurs projets à caractère industriel ; il figure notamment parmi les environnements de vérification retenus dans le cadre du projet RNTL COTRE (Composants Temps Réel, <http://www.laas.fr/COTRE>).

Dans le cadre des approches "ordre partiel", le travail en cours consiste à étudier les complémentarités entre les approches existantes. Nous considérons plus particulièrement la possibilité de combiner graphes de pas et "ensembles amples" dans le cadre du model-checking de formules de logique temporelle linéaire.

Dans le cadre temporisé, les travaux en cours visent à permettre la vérification de propriétés temporelles quantifiées telles que celles exprimables dans la logique *TCTL*. A terme, nous envisageons aussi de combiner les approches "ordre partiel" et "graphe de classes" afin de réduire l'explosion combinatoire dans le cadre de systèmes temporisés.

7. Bibliographie

- [ARN 92] ARNOLD. A. *Systèmes de transitions finis et sémantique des processus communicants*. Masson Paris, 1992.
- [BÉR 99] BÉRARD B., FRIBOURG. L., « Reachability Analysis of (Timed) Petri Nets Using Real Arithmetic », *10th International Conference on Concurrency Theory (CONCUR'99)*, Lecture Notes in Computer Science, août 1999, p. 178–193.
- [BER 83] BERTHOMIEU B., MENASCHE M., « An Enumerative Approach for Analyzing Time Petri Nets. », *IFIP Congress Series*, vol. 9, 1983, p. 41–46, Elsevier Science Publ. Comp. (North Holland).
- [BER 91] BERTHOMIEU B., DIAZ M., « Modeling and Verification of Time Dependent Systems Using Time Petri Nets. », *IEEE Transactions on Software Engineering*, vol. 17, n° 3, 1991, p. 259–273.
- [BER 01] BERTHOMIEU B., « La méthode des classes d'états pour l'analyse des réseaux temporels – Mise en œuvre, Extension à la multi-sensibilisation », *Proc. Modélisation des Systèmes Réactifs, Toulouse, France*, octobre 2001.
- [BER 03] BERTHOMIEU B., VERNADAT F., « State Class Constructions for Branching Analysis of Time Petri Nets », *Proc. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2003)*, Warsaw, Poland, 2003.
- [BRO 88] BROWNE M. C., CLARKE E. M., GRÜMBERG O., « Characterizing finite Kripke structures in propositional temporal logics », *Theoretical Computer Science*, vol. 59, 1988, p. 115–131.

- [FER 91] FERNANDEZ J.-C. , MOUNIER L. A toolset for deciding behavioral equivalences. *In proceedings of CONCUR'91 - LNCS*, 1991.
- [GOD 91] GODEFROID P., WOLPER P., « Using Partial Orders for the Efficient Verification of Deadlock Freedom and Safety Properties », *Proceedings of CAV'91*, Springer Verlag, LNCS 575, July 1991.
- [GOD 96] GODEFROID P. *Partial-Order Methods for the Verification of Concurrent Systems*, LNCS 1032, 1996.
- [HAD 01] HADDAD S., VERNADAT F. *Méthodes d'analyse des réseaux de Petri in Les Réseaux de Petri. Modèles fondamentaux*. Hermes Science, Traité IC2 Information-Commande-Communication, ISBN 2-7462-0250-6, 2001.
- [KAR 69] KARP R., MILLER R. Parallel program schemata. *Computer and System Sciences* 3, 1969.
- [MAZ 86] MAZURKIEWICZ A., « Trace Theory », *Petri Nets : Applications and Relationships to Other Model of Concurrency, Advances in Petri nets 1986, Part II ; Proceedings of an advanced Course*, Springer Verlag, LNCS 255, 1986, p. 279-324.
- [MER 76] MERLIN P. M., FARBER D. J., « Recoverability of Communication Protocols : Implications of a Theoretical Study. », *IEEE Trans. Comm.*, vol. 24, n° 9, 1976, p. 1036-1043.
- [MIL 85] MILNER R.M. *Communication and Concurrency*, Prentice Hall., 1985.
- [PAI 87] PAIGE P., TARJAN R. E., « Three Partition Refinement Algorithms », *SIAM Journal on Computing*, vol. 16, n° 6, 1987, p. 973-989.
- [PEL 98] PELED, « Ten years of partial order reduction », *Proceedings of Computer-Aided Verification (CAV)*, Springer Verlag, LNCS 1427, 1998.
- [RIB 02] RIBET P., VERNADAT F., BERTHOMIEU B., « On combining the Persistent Sets Method with the Covering Steps Graph Method », *FORTE 2002. Springer Verlag, LNCS 2529, ISBN 3-540-00141-7*, 2002, p. 344-359,
- [RIB 03] RIBET P., VERNADAT F., BERTHOMIEU B., « Graphe de pas couvrant préservant $LT L_X$ », *Rapport interne LAAS*, 2003.
- [TRI 96] TRIPAKIS S., YOVINE S., « Analysis of timed systems based on time-abstracting bisimulations », *8th Conference Computer-Aided Verification, CAV'96, Springer LNCS 1102*, jul 1996, p. 232-243.
- [VAL 90] VALMARI A., « A stubborn Attack on State Explosion », *Proceedings of CAV'90*, ACM, DIMACS volume 3, 1990, p. 25-42.
- [VER 96] VERNADAT F., AZÉMA P., MICHEL F., « Covering Step Graph », *Proceedings of ATPN'96*, Springer Verlag, LNCS 1091, 1996.
- [VER 97] VERNADAT F., MICHEL F., « Covering Step Graph Preserving Failure Semantics », *Proceedings of ATPN'97*, Springer Verlag, LNCS 1248, 1997.
- [VER 03] VERNADAT F., RIBET P., *Graphes de Pas couvrants : une approche ordre partiel in Vérification et mise en oeuvre des réseaux de Petri*, Hermes Science, Traité IC2 Information-Commande-Communication, ISBN 2-7462-0445-2, 2003.
- [VGL 90] VAN GLABBEEK. ROB. J. The linear time-branching time spectrum. In *CONCUR 1990*, 1990.