

Réseaux de Petri temporels, SE-LTL et boîte à outils TINA

Résumé

Ce document présente les différents éléments que nous proposons de mettre en oeuvre pour la vérification (par *model-checking*) des modèles TCL. Parmi ceux-ci, seront utilisés les réseaux de Petri (temporels) pour donner une sémantique par traduction à la description des modèles TCL, la logique temporelle SE-LTL pour exprimer (sur les réseaux de Petri) les propriétés à vérifier sur les modèles TCL, et l'environnement de description et de vérification TINA qui supporte ces différents éléments. Nous ne décrivons pas dans le détail chacune de ces parties et nous renvoyons le lecteur intéressé à [3] pour une présentation de SE-LTL, [2] pour une présentation des réseaux temporels et [1] pour une description de TINA.

1 Réseaux de Petri

Les explications données ici sont récupérées de http://en.wikipedia.org/wiki/Petri_net.

Définition Un réseau de Petri est un tuple (S, T, F, M_0, W) où :

- S définit une ou plusieurs places.
- T définit une ou plusieurs transitions.
- F définit un ou plusieurs arcs (flèches).
- Un arc ne peut pas être connecté entre 2 places ou 2 transitions ; plus formellement : $F \subseteq (S \times T) \cup (T \times S)$.
- $M_0 : S \rightarrow \mathbb{N}$ appelé place initiale, où, pour chaque place $s \in S$, il y a $n \in \mathbb{N}$ jetons.
- $W : F \rightarrow \mathbb{N}^+$ appelé ensemble d'arcs primaires, assignés à chaque arc $f \in F$ un entier positif $n \in \mathbb{N}^+$ qui indique combien de jetons sont consommés depuis une place vers une transition, ou sinon, combien de jetons sont produits par une transition et arrivent pour chaque place.

De nombreuses définitions formelles existent. Cette définition concerne un réseau place-transition (ou P-T).

Représentation Un réseau de Petri se représente par un graphe orienté composé d'arcs reliant des places et des transitions. Deux places ne peuvent pas être reliées entre elles, ni deux transitions.

Les places peuvent contenir des jetons. La distribution des jetons dans les places est appelée le marquage du réseau de Petri.

Les entrées d'une transition sont les places desquelles part une flèche pointant vers cette transition, et les sorties d'une transition sont les places pointées par une flèche ayant pour origine cette transition.

La figure 1 proposent quelques exemples de réseaux de Petri.

Dynamique d'exécution Un réseau de Petri évolue lorsqu'on exécute¹ une transition : des jetons sont pris dans les places en entrée de cette transition et envoyés dans les places en sortie

¹On emploie aussi les verbes franchir ou tirer.

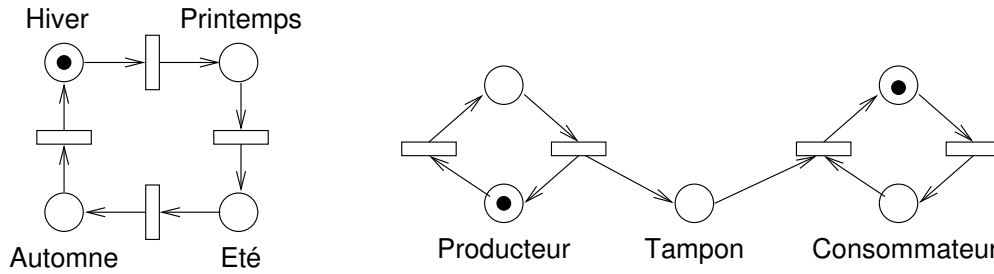


FIG. 1 – Exemples de réseaux de Petri

de cette transition. Le nombre de jetons pris dans les places d'entrée et mis dans les places de sortie est celui indiqué par l'arc correspondant. Une transition ne peut être exécutée que si elle est exécutable², c'est-à-dire qu'il y a dans chaque place d'entrée un nombre de jetons au moins égal au nombre de jetons indiqué sur l'arc.

L'exécution d'une transition est une opération indivisible qui est déterminée par la présence du jeton sur la place d'entrée.

L'exécution d'un réseau de Petri n'est pas déterministe, car il peut y avoir plusieurs possibilités d'évolution à un instant donné.

Si chaque transition dans un réseau de Petri a exactement une entrée et une sortie alors ce réseau est un automate fini.

Extensions De nombreuses extensions des réseaux de Petri ont été proposées. Parmi celles-ci, nous décrivons les réseaux temporels supportés par les outils de TINA³. Les réseaux temporels, introduits dans [4], sont obtenus depuis les réseaux de Petri en associant deux dates min et max à chaque transition. Supposons que t soit devenue sensibilisée pour la dernière fois à la date θ , alors t ne peut être tirée avant la date $\theta + min$ et doit l'être au plus tard à la date $\theta + max$, sauf si le tir d'une autre transition a désensibilisé t avant que celle-ci ne soit tirée. Le tir des transitions est de durée nulle. Les réseaux temporels expriment nativement des spécifications "en délais". En explicitant débuts et fins d'actions, ils peuvent aussi exprimer des spécifications "en durées". Leur domaine d'application est donc large.

On ajoute aussi un nouveau type d'arc appelé *read-arc*. Il s'agit d'un arc qui relie nécessairement une place d'entrée à une transition. Il consiste à vérifier que la place a bien au moins le nombre de jetons indiqué sur cet arc. Si c'est le cas, la transition est exécutable. Lors de l'exécution de la transition, les jetons ne sont pas enlevés de la place d'entrée du *read-arc*. Cette notion de *read-arc* n'a de sens que dans le cas de réseau de Petri temporel. Sinon, elle pourrait être simulée par un arc remettant les jetons consommés par la transition dans la place concernée. Dans le cas d'un réseau de Petri temporel, le temps serait remis à 0 pour toutes les transitions qui ont cette place pour entrée.

La figure 2 présente un exemple de réseau de Petri avec *read-arc*.

2 La logique temporelle SE-LTL

La logique *LTL* étend le calcul propositionnel en permettant l'expression de propriétés spécifiques sur les séquences d'exécution d'un système⁴. *SE-LTL* est une variante de *LTL* récemment introduite [3], qui permet de traiter de façon homogène des propositions d'états et

²On dit aussi franchissable, sensibilisée, validée ou tirable.

³Mais pas nécessairement à utiliser dans le cadre de la vérification des modèles TCL. Cela dépendra du mapping que vous allez définir.

⁴Pour plus d'informations, voir http://en.wikipedia.org/wiki/Linear_temporal_logic

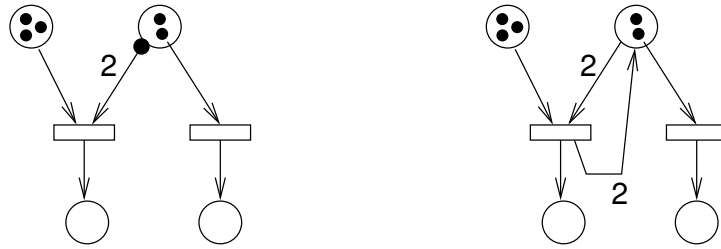


FIG. 2 – Le même réseau de Petri avec un read-arc et sans

des propositions de transitions. Les modèles pour la logique $SE-LTL$ sont des structures de Kripke étiquetées (ou SKE), aussi appelées systèmes de transitions de Kripke (ou KTS). En voici quelques formules :

- (Pour tout chemin)
- P P vraie au départ du chemin (pour l'état initial),
- $\square P$ P vraie tout le long du chemin,
- $\diamond P$ P vraie une fois au moins le long du chemin,
- $P U Q$ Q sera vraie dans le futur et P est vraie jusqu'à cet instant,
- $\square \diamond P$ P vraie infiniment souvent.

Nous pouvons considérer un chemin comme la séquence d'états d'une exécution possible du système.

3 La boîte à outils TINA (*Time Petri Net Analyzer*)

Il s'agit d'un environnement logiciel permettant l'édition et l'analyse de réseaux de Petri et réseaux temporels [1]. Les différents outils constituant l'environnement peuvent être utilisés de façon combinée ou indépendante. Parmi les outils utilisés dans le cadre de cette étude on citera :

`nd` (*NetDraw*) est un outil d'édition de réseaux temporels et d'automates, sous forme graphique ou textuelle. La syntaxe textuelle est décrite dans le listing 1. Il intègre également un simulateur "pas à pas" (graphique ou textuel) pour les réseaux temporels et permet d'invoquer les outils ci-dessous sans sortir de l'éditeur.

`tina` cet outil construit des représentations de l'espace d'états d'un réseau de Petri, temporel ou non. Aux constructions classiques (graphe de marquages, arbre de couverture), `tina` ajoute la construction d'espaces d'états abstraits, basés sur les techniques d'ordre partiel. Pour les réseaux temporels, `tina` propose toutes les constructions de graphes de classes discutées dans [2].

`selt` en plus des propriétés générales d'accessibilité vérifiées à la volée par `tina` (caractère borné, présence de blocage, pseudo-vivacité et vivacité), il est le plus souvent indispensable de pouvoir garantir des propriétés spécifiques relatives au système modélisé. L'outil `selt` est un vérificateur de modèle (*model-checker*) pour les formules d'une extension de la logique temporelle $SE-LTL$ (State/Event LTL) de [3]. En cas de non-satisfaction d'une formule, `selt` peut fournir une séquence contre-exemple en clair ou sous un format exploitable par le simulateur de TINA.

Les structures de Kripke étiquetées manipulées sont obtenues à partir d'un réseau de Petri. Afin de conserver l'information de multiplicité de marques exprimée par les marquages, `selt` travaille sur des structures de Kripke enrichies, basées sur des multi-ensembles de propriétés atomiques plutôt que des ensembles. Afin d'exploiter au mieux l'information contenue dans ces structures de Kripke enrichies, on substitue au calcul propositionnel (bi-valué par $\{true, false\}$),

Listing 1 – Syntaxe des fichiers d'entrée des outils de Tina

```

1 .net          ::= (<trdesc >|<pldesc >|<lbdesc >|<netdesc >)*
2 netdesc     ::= 'net' <net>
3 trdesc      ::= 'tr' <transition > {":" <label >} {<interval >}
4                                     {<tinput > -> <toutput >}
5 pldesc      ::= 'pl' <place > {":" <label >} {(<marking >)}
6                                     {<pinput > -> <poutput >}
7 lbdesc      ::= 'lb' [<place >|<transition >] <label >
8 interval    ::= ('['|']')INT', 'INT('['|']') | ('['|']')INT', 'w['
9 tinput      ::= <place >{<arc >}
10 toutput    ::= <place >{<normal_arc >}
11 pinput     ::= <transition >{<normal_arc >}
12 poutput    ::= <transition >{arc}
13 arc        ::= <normal_arc > | <test_arc > | <inhibitor_arc > |
14             <stopwatch_arc > | <stopwatch-inhibitor_arc >
15 normal_arc ::= '*'<weight >
16 test_arc   ::= '?'<weight >
17 inhibitor_arc ::= '?-'<weight >
18 stopwatch_arc ::= '!<weight >
19 stopwatch-inhibitor_arc ::= '!-'<weight >
20 weight, marking ::= INT{'K'|'M'}
21 net, place, transition, label ::= ANAME | '{'QNAME'}'
22 INT        ::= unsigned integer
23 ANAME      ::= alphanumeric name, see Notes below
24 QNAME      ::= arbitrary name, see Notes below

```

un calcul propositionnel multi-valué et on étend le langage d'interrogation de `selt` avec des opérateurs logico-arithmétiques. Le model-checker `selt` permet aussi la déclaration d'opérateurs dérivés ainsi que la redéclaration des opérateurs existants ; un petit nombre de commandes sont aussi fournies pour contrôler l'impression des résultats ou encore permettre l'utilisation de bibliothèques de formules.

Quelques formules de `selt` :

- $\square (p2 + p4 + p5 = 2)$ un invariant de marquage linéaire,
- $\square (p2 * p4 * p5 = 0)$ un invariant de marquage non linéaire,
- infix $q R p = \square (p \Rightarrow \diamond q)$ déclare l'opérateur "répond à", noté R .

Références

- [1] B. Berthomieu, P.-O. Ribet, and F. Vernadat. The tool TINA – construction of abstract state spaces for Petri nets and time Petri nets. *International Journal of Production Research*, 42(14) :2741–2756, 15 juillet 2004.
- [2] B. Berthomieu and F. Vernadat. *Réseaux de Petri temporels : méthodes d'analyse et vérification avec TINA*. Traité IC2 1, Ed. Nicolas Navet, Hermes, 2006.
- [3] S. Chaki, M E. Clarke, J. Ouaknine, N. Sharygina, and N. Sinha. State/event-based software model checking. In *4th International Conference on Integrated Formal Methods (IFM'04)*, Springer LNCS 2999, pages 128–147, avril 2004.
- [4] P. M. Merlin. *A Study of the Recoverability of Computing Systems*. Irvine : Univ. California, PhD Thesis, 1974.