# The tool TWINA construction d'espaces d'états abstrait pour l'intersection de Time Petri nets

Éric Lubat

# The tool TWINA — construction d'espaces d'états abstrait pour l'intersection de Time Petri nets

Éric Lubat

LAAS-CNRS, Université de Toulouse, CNRS, INSA, Toulouse, France
elubat@laas.fr

## 1 Introduction

L'outil TINA [2] a été développé au sein de l'équipe Vertics du LAAS-CNRS, en grande partie par Bernard Berthomieu. TINA permet l'édition et l'analyse des réseaux de Petri et de leurs extensions ; en particulier sur les aspects temporels, avec l'étude des réseaux de Petri temporisés (ou *Time Petri nets*, TPN, en anglais). L'aspect le plus distinctif de l'outil TINA est de proposer diverses constructions d'espaces d'états abstraits, basées sur la notion de *graphes de classes* (State Class Graph ou SCG), qui préservent des classes spécifiques de propriétés sur les réseaux : atteignabilité, absence de deadlocks, propriétés de logiques temporelles, ... Il s'agit d'une contribution intéressante. En effet, à cause de l'utilisation d'un modèle de temps dense, l'espace d'état concret d'un TPN est généralement de taille infinie.

Contrairement à d'autres modèles, tel que les automates temporisés [1], il n'est pas toujours possible de construire (syntaxiquement) le produit de deux TPN dans le but de calculer l'intersection de leurs comportements. On dit que les TPN ne sont pas *composables*. Dans ce travail, nous présentons un nouvel outil, TWINA [3], qui permet de calculer une abstraction par graphes de classes pour « l'intersection » de deux TPN ; c'est-à-dire une abstraction permettant d'analyser de manière exacte les séquences de tirs, ou *traces*, qui sont communes à deux réseaux.

Dans notre contexte, on ne retient que les étiquettes (les *labels*) des transitions. Ainsi, une trace est commune à deux TPN lorsqu'il est possible de tirer une séquence de transitions ayant les bons labels, dans le bon ordre et au même dates, dans les deux réseaux.

En tant que tel, TWINA permet de raisonner sur l'intersection du langage de plusieurs TPN. Plus généralement, TWINA permet l'analyse du « produit synchrone » de TPN, un modèle appelé *Product Time Petri Net* dans [5]. Intuitivement, il s'agit d'une extension des TPN dans laquelle on impose que toutes les transitions de certains ensembles donnés soient tirées en même temps. Cette opération est utile dans la pratique, par exemple pour des applications de vérification des modèles—afin de pouvoir vérifier des propriétés temporisées par l'utilisation d'observateurs, mais sans risquer d'interférences avec le système à étudier—ou bien encore, dans le domaine du diagnostic, afin de définir une construction équivalente à celle de la *twin plant* [4] dans le cas des TPN.

## 2 L'outil TWINA

L'outil TWINA est un outil en ligne de commande développé intégralement en Go. Son utilisation ne repose pas sur l'installation de librairies ou de logiciels annexes particulier. Des versions binaires du logiciel sont disponibles pour de nombreuses architectures. Les binaires ainsi que la documentation et des cas d'études sont disponibles sur le site Web de l'outil [3].

TWINA utilise les mêmes formats d'entrée et de sortie que ceux de la boîte à outil TINA. En particulier, il est possible d'utiliser des arcs spéciaux, tel que les arcs inhibiteurs (avec ou sans

capacités) ou les *read arcs*. Comme pour TINA, notre méthode d'exploration est exhaustive ; on se restreint donc à des réseaux de Petri ayant un marquage borné.

Par défaut, TWINA prend un réseau en entrée (au format textuel `.net`) et affiche des statistiques sur la taille du graphe de classes généré. Dans ce cas, la sortie est la même que celle obtenue avec l'option `-W` de l'outil *sift*, qui est disponible dans la boîte à outil TINA. (Les performances de TWINA sont en général meilleures que celle de TINA.)

Le format de sortie principal, obtenu par l'option `-aut`, est sous forme de graphes d'état au format AUT. Il s'agit d'un format textuel simple pour décrire les « labeled transition systems » (LTS) utilisé par plusieurs outils de vérification. Ce format peut être affiché sous forme graphique grâce à notre outil *nd* (netDraw), voir Fig. 1. Ce même format peut être exploité, à l'aide des outils *selt* et *muse* fournis par TINA, dans le but de vérifier des formules de logiques temporelles (LTL et CTL) sur l'ensemble des traces d'un système ou d'une composition de systèmes.

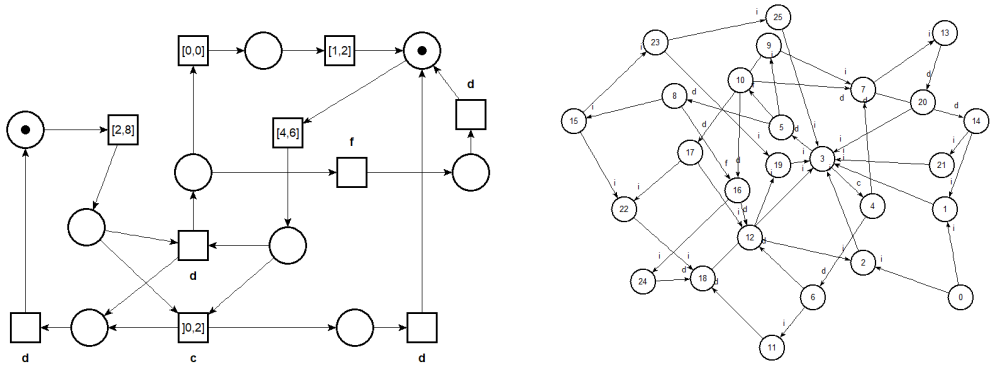


FIGURE 1 – Exemple de TPN et son SCG au format AUT

L'option principale de TWINA, `-I`, permet de calculer le graphe de classes du produit synchrone de deux TPN (dans lequel on doit tirer, en même temps, les transitions ayant des labels identiques).

Les autres options disponibles permettent de raisonner sur la *diagnosticabilité* d'un TPN. Par exemple, il est possible de distinguer un label particulier, disons `f`, à l'aide de l'option `--fault=f`. Intuitivement, ce label distingue les fautes du système des autres actions observables. L'option `-twin` de TWINA permet de calculer l'intersection du langage d'un réseau, $N$, avec une copie de lui même, $N_{|f}$, dans lequel on a omis les transitions « de fautes. » Cette option, qui donne son nom à l'outil, permet d'analyser un équivalent de la construction du *twin-plant* [4] pour les TPN. Il s'agit d'une construction utile pour décider si un système est diagnosticable (pour la faute `f`). TWINA fourni d'ailleurs une option, `-diag` qui permet de répondre à cette question.

## 2.1 Conclusion

Nous avons présenté quelques-unes des fonctionnalités de TWINA. Ce logiciel s'ajoute à la boîte à outil TINA et permet d'analyser le comportement du produit synchrone de plusieurs TPN, même lorsque ceux-ci ne sont pas composables. (Ce qui est le cas lorsque des transitions observables possèdent des contraintes de temps non-triviales). Pour la suite, nous comptons améliorer les capacités de cet outil afin de répondre à des questions de diagnostic plus complexes, notamment la détection de motifs [6].

# Références

[1] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2) :183–235, 1994.

[2] 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), 2004.

[3] Silvano Dal Zilio. TWINA : A realtime model-checker for analyzing Twin-TPN. https://projects.laas.fr/twina/, 2019.

[4] S. Jiang, Z. Huang, V. Chandra, and R. Kumar. A polynomial algorithm for testing diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 46(8), 2001.

[5] Éric Lubat, Silvano Dal Zilio, Didier Le Botlan, Yannick Pencolé, and Audine Subias. A State Class Construction for Computing the Intersection of Time Petri Nets Languages. In *17th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, volume 11750 of *LNCS*. Springer, August 2019.

[6] Yannick Pencolé and Audine Subias. Diagnostic de motifs de comportements dans les systèmes temporels. In *Modélisation des Systèmes Réactifs (MSR 2017)*, 2017.