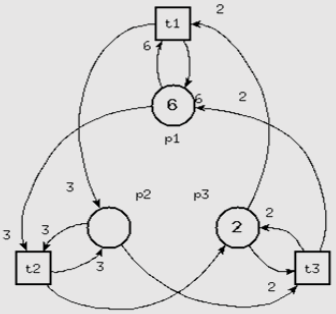
logique et REPRÉSENTATION **des connaissances**

Compte-rendu du septième TME : découverte des réseaux de Petri et de leur version temporisée.

16 décembre 2019

L////// S///////, L////// T///////

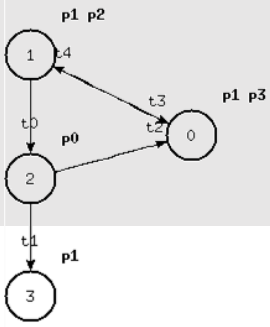
Prise en main de l'outil



Le réseau pris pour exemple semble tour à tour consommer et régénérer les tokens de trois places, en partant d'un stock de 6000. Il est borné, car chacune de ses places atteint respectivement un maximum de 6000, 2000 et 3000 tokens sans jamais le dépasser. Il est aussi vivant (toutes les transitions sont perpétuellement accessibles, sans état puits). Le retour à un état donné étant en particulier toujours possible, le réseau est aussi réversible.

L'affichage du logiciel TINA confirme ces trois intuitions. On affichera facilement, aussi, le graphe des marquages accessibles grâce à un menu dédié.

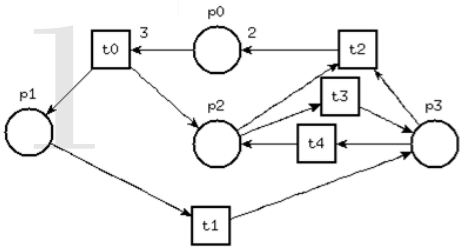
Syntaxe des réseaux de Petri



La suite $t_4 \rightarrow t_3 \rightarrow t_4 \rightarrow t_3 \rightarrow t_4 \rightarrow t_0 \rightarrow t_2 \rightarrow t_4 \rightarrow t_0 \rightarrow t_2 \rightarrow t_4 \rightarrow t_3 \rightarrow t_4 \rightarrow t_0 \rightarrow t_1$ transforme le marquage initial {0,1,0,1} du réseau de l'exercice 2 en {0,1,0,0}.

L'analyseur d'accessibilité donne le graphe des marquages accessibles ci-contre. {0,1,0,0} est un marquage puits, ce qui fait qu'il existe un blocage, et le réseau n'est donc pas vivant. Aussi est-il borné, puisque ses places ne dépassent jamais 1.
TINA permet aussi de savoir s'il est quasi-vivant : il suffit de vérifier que les transitions apparaissent toutes bien sur le graphe des marquages accessibles. C'est d'ailleurs le cas ici : ce graphe est bien quasi-vivant.

Propriétés des transitions, marquages et réseaux

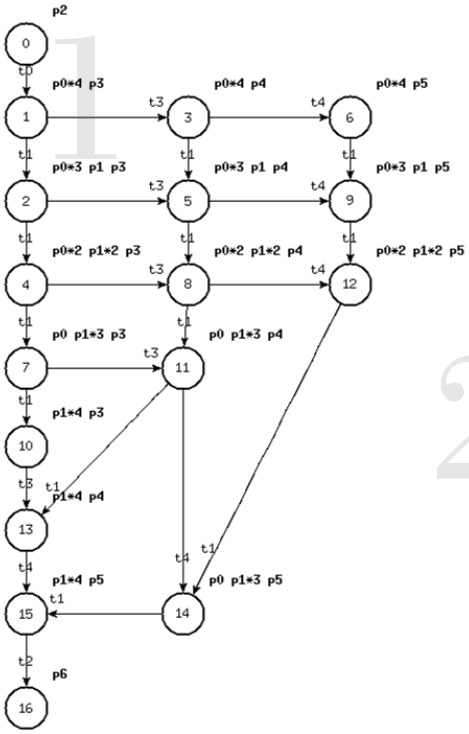


On part du graphe de Petri ci-contre :

Le marquage initial de ce graphe peut être transformé pour obtenir
→ *un réseau non-borné : marquage {0,0,1,1}*
Il y a alors création de valeur sur toutes les places, c'est ce que veut dire l'absence de borne.
→ *un réseau borné avec blocage : marquage {1,0,0,0}*
Un tel réseau est bloqué dès le départ (il aurait fallu deux tokens sur la place 0 pour avancer).
→ *un réseau borné, sans blocage, réversible, mais ni vivant, ni quasi-vivant : marquage {0,0,0,1}*
Le token reste bloqué dans la partie droite du graphe, qui n'est donc même pas quasi-vivant. De plus, il n'y a pas de création de valeur, donc le réseau est borné, et le marquage initial est de nouveau atteint tous les deux mouvements (réversibilité).
→ *un réseau de mêmes propriétés qui ne soit pas réversible : marquage {0,1,0,0}*
Idem ; mais il est cette fois impossible de revenir au marquage initial (token vite bloqué dans la partie droite du graphe).

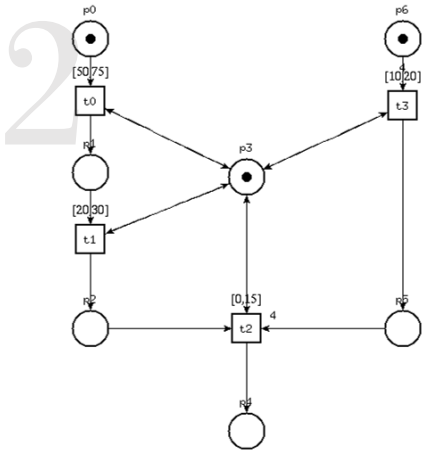
La modification du marquage peut être complétée par des opérations sur le poids des arcs autour de la transition initiale. On obtient ainsi
→ *un réseau borné, quasi-vivant, avec blocage : $p_0 \rightarrow t_0$ prend le poids 3, $t_0 \rightarrow p_2$ le poids 1, puis marquage {3,0,0,0}.*
Un token finit par être consommé après un tour de graphe, pas de création de valeur, et les deux qui restent sont insuffisants pour franchir la transition t_0 : donc, blocage en p_0.
→ *un réseau borné, quasi-vivant mais non vivant, sans blocage et non réversible : idem, marquage {3,1,1,1}*
En reprenant le précédent, il suffit d'ajouter des tokens sur les places restées vides. Au bout d'un temps, deux tokens sont bloqués en p_0 et un troisième finit par boucler éternellement sur la partie droite du graphe (non réversible, sans blocage, mais non-vivant).
→ *un réseau borné, quasi-vivant mais non vivant, et sans blocage, mais réversible : impossible.*
De fait, on attend qu'il soit possible de retourner à l'état initial (réversibilité), et qu'il existe une transition franchissable après toute séquence (sans blocage). Il faut donc qu'il soit possible de boucler pour revenir à l'état initial, et ainsi le graphe devrait être vivant, ce qui n'est pas permis.
→ *un réseau borné, vivant et réversible : graphe de départ avec $t_0 \rightarrow p_2$ de poids 1, marquage {2,0,0,1}.*
Pas d'ajout de valeur. Toutes les transitions sont accessibles, avec une boucle qui repasse par le marquage de départ.

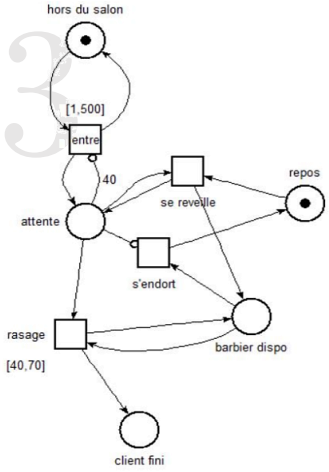
Restitution des modèles de TD

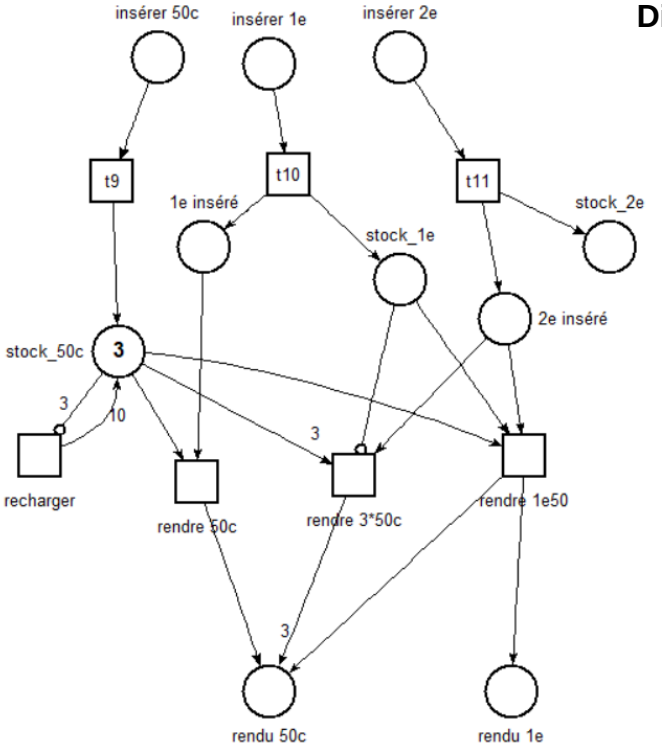


Ci-dessous, un modèle pour la création d'une voiture de Noël par un elfe. Selon l'outil dédie, il existe en tout dix-sept marquages accessibles, mais les états de départ (un elfe disponible) et de fin (une voiture créée) sont communs à tous les scenarii. Il y a donc quinze scenarii distincts possibles. C'est aussi ce que l'on obtient en beaucoup plus longtemps en appliquant un comptage.

A ce modèle, on a ajouté l'extension des réseaux de Petri, la temporisation, qui permet d'exprimer le temps passé par un elfe sur chaque poste de création. Le même traitement est aussi appliqué au problème du barbier :







Distributeur de café

La machine à café ci-contre offre des boissons à cinquante centimes (un rêve éveillé). L'intérêt de l'exercice est dans la gestion du rendu de monnaie. La préférence dans le rendu va aux pièces d'un euro, puis à celles de cinquante centimes.

Selon l'action de l'utilisateur (insertion d'un token dans l'une des trois places du haut), s'il n'a pas fait l'appoint, la machine décide de ce qu'elle fait de sa réserve de pièces : à supposer que ceux des pièces de 50 se rechargent automatiquement, et si l'utilisateur a inséré deux euros, il suffit de vérifier qu'on a suffisamment de pièces d'un euro pour rendre 1€50. On utilise alors un arc inhibiteur (un arc qui sert à inactiver une transition si un token est présent à telle place) : sur insertion d'une pièce de deux euros, on évite de rendre trois pièces de cinquante centimes si l'on a de quoi faire autrement.

Un autre arc inhibiteur a été utilisé pour le rechargement des stocks de 50, qui n'a pas lieu tant que le stock total n'est pas descendu sous la barre des trois pièces. On évite ainsi d'avoir à attendre l'épuisement total des ressources, et on pare à toute éventualité en en gardant toujours une quantité suffisante sous la main.

conclusion

Le dernier TME concerne les réseaux de Petri et les automates temporisés. Le logiciel TINA, développé par le LAAS (Laboratoire d'Analyse et d'Architecture des Systèmes), s'ajoute à notre panoplie d'outils. Ce sera notre allié spécifique pour le traitement automatique de tels objets. Après une phase de familiarisation, et en plus des traditionnels exercices de modélisation repris du TD, nous avons eu l'occasion d'exercer notre imagination – et de représenter le comportement de la machine à café du patio de la vie étudiante (avant l'augmentation du prix du café, datée de 2017).

Sur un graphe personnalisé (choix des poids des transitions et des marquages de départ), il est possible de simuler des séquences de transitions, de vérifier quelques propriétés, comme la vivacité des transitions, ou encore de générer l'arbre des marquages accessibles. TINA, on l'a vu dans le quatrième exercice, permet en plus de prendre en compte des contraintes temporelles – la durée de chaque étape – dans l'exécution, donc d'étudier précisément l'enchaînement d'un ensemble de tâches.

Dans cette expérience, on déplore seulement l'impossibilité matérielle d'exporter quoi que ce soit de ce que nous avons fait dans un fichier séparé (bug logiciel ?). L'absence de captures d'écran, pourtant précieuses pour compenser ce manque, a été expressément, professoralement requise ; on s'y est tenu au possible.