Réseaux de Petri

Pour ce TME, on utilise TINA, un outil pour dessiner, simuler et analyser des réseaux de Petri. Il est installé sur les machines. Il suffit d'exécuter la commande `tina` depuis un terminal.

Prise en main de l'outil

Exercice 1 Prise en main avec `l6000.ndr`

Récupérer les fichiers `.ndr` qui sont dans le répertoire `/usr/local/tina-3.4.0/nets`.

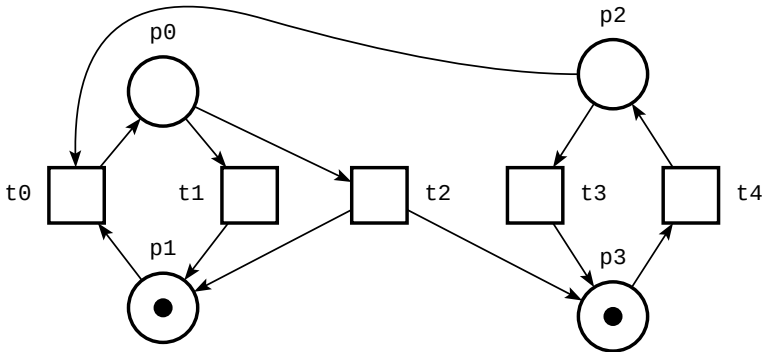
Lancer `tina` et ouvrir le fichier `l6000.ndr`.

1. Observer le graphe et essayer de prédire son fonctionnement. Note : un nombre près d'un arc indique son poids, un nombre dans une place indique le marquage de cette place (si le marquage est 1, un simple rond noir apparait dans la place, et si le marquage est 0, la place reste vide). 6K représente le nombre 6000.
2. Utiliser le simulateur pas à pas (menu `Tools\stepper simulator`) pour simuler le fonctionnement du graphe. Pensez-vous que le graphe soit borné ? vivant ? réversible ? Note : Les transactions franchissables sont en rouge et peuvent être franchies en cliquant dessus. Il est aussi possible d'appuyer sur le bouton `Rand` pour franchir aléatoirement en continu des transactions franchissables. La vitesse peut être réglée dans le menu `Mode` (rubrique `firing duration`. Pour revenir à la fenêtre du réseau de Petri, il faut quitter le stepper simulator (en utilisant le menu, l'icône de fermeture de fenêtre ou le raccourci `Crtl+q`).
3. Utiliser l'analyse d'accessibilité pour générer le graphe des marquages et vérifier si le réseau est borné, vivant, et/ou réversible. Note : il faut utiliser le menu (`Tools\reachability analysis`), qui ouvre une fenêtre de configuration. Choisir pour building `marking graph`, pour output `verbose` et cocher `liveliness analysis`. Lire les résultats pour retrouver les différentes conclusions.
4. Pour permettre une simulation plus rapide, remplacer tous les marquages de 6K et 2K par 6 et 2, puis les poids d'arc de 6K, 3K, 2K par 6,3,2. Note : Pour éditer les propriétés d'une place ou d'un arc, il faut cliquer dessus avec le clic droit dans l'éditeur pour faire apparaître un menu contextuel.
5. Générer et afficher le graphe des marquages accessibles en utilisant l'analyse d'accessibilité, en choisissant cette fois pour sortie `kts (.aut)`. L'analyse de vivacité n'est alors pas possible. Il est alors possible par un clic droit de sauvegarder le fichier automate généré et de l'ouvrir dans tina (`open in nd`). Le menu `Edit\draw` permet alors un affichage sous forme graphique (avec différents outils de visualisation pour répartir les nœuds à choisir dans une liste).

Syntaxe

Exercice 2 Graphe des marquages accessibles et vivacité

Cet exercice utilise Tina pour résoudre l'exercice 2 de la feuille de TD. On considère le réseau de Petri de cet exercice, que l'on rappelle ci-dessous :



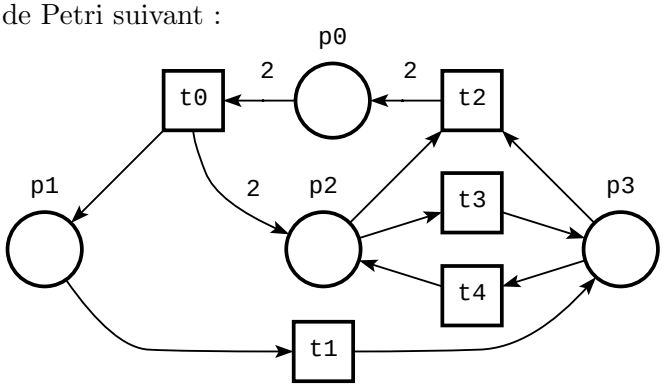
1. Tracer le réseau et renseigner le marquage initial. Sauvegarder le fichier `.ndr` produit.
2. Utiliser le simulateur pas à pas pour retrouver le marquage résultant du tir de la séquence des transitions ($t_4\ t_3\ t_4\ t_3\ t_4\ t_0\ t_2\ t_4\ t_0\ t_2\ t_4\ t_3\ t_4\ t_0\ t_1$). Sauvegarder aussi cet historique (fichier `.scn`).

3*. Utiliser l'analyseur d'accessibilité et les sorties en automate pour générer le graphe des marquages accessibles et le sauvegarder après l'avoir éventuellement réarrangé sous forme lisible.

4. Utiliser l'analyseur d'accessibilité avec sortie textuelle pour vérifier si ce réseau de Petri est borné, vivant, ou sans blocage et donner une transition vivante ou un marquage puits.
5. Comment déterminer à partir de Tina si le graphe est ou non quasi-vivant ?

Exercice 3 Propriété des transitions, marquages et réseaux

On considère le réseau de Petri suivant :



1. Tracer ce réseau dans tina et le sauvegarder sous `r_base.ndr`.

2*. En ne modifiant que le marquage initial, construire
- un réseau non borné que l'on nomme `r_b0.ndr`.
- un réseau borné avec blocage `r_b1s0.ndr`.
- un réseau borné et sans blocage et réversible, mais ni vivant, ni quasi-vivant (`r_b1s1r1.ndr`)
- un réseau borné et sans blocage et non réversible, ni vivant ou quasi-vivant (`r_b1s1r0.ndr`)

Repérer à chaque fois quelles sont les transitions vivantes.

3*. En s'autorisant maintenant de plus à changer les poids des arcs entrants ou sortants de la transition `t0` en plus du marquage initial, construire (ou prouver qu'il est impossible de construire) :
- un réseau borné, quasi vivant mais avec blocage `r_b1s0r0q1v0.ndr`.
- un réseau borné, quasi vivant, sans blocage et non réversible ni vivant `r_b1s1r0q1v0.ndr`.
- un réseau borné, quasi vivant, sans blocage et réversible, mais pas vivant `r_b1s1r1q1v0.ndr`.
- un réseau borné, vivant et réversible `r_all1.ndr`.

Modélisation

Exercice 4 Variantes des modèles de TDs

1*. Reprendre le premier réseau de Petri pour modéliser le processus de montage d'une voiture en bois par les elfes. Utiliser le simulateur pas à pas pour observer les différents scénarii possibles. Combien y en a-t-il ? Peut-on le déduire du graphe des marquages accessibles ?

2. Reprendre le réseau qui modélise la production de plusieurs jouets en parallèle. Simuler le processus.
3. On se propose maintenant d'utiliser une extension des réseaux de Petri, les réseaux de Petri temporisés, pour spécifier des intervalles de temps aux transitions. L'intervalle $[n, m]$ apposé à une transition, signifie que le temps de franchissement associé à la transition est entre n et m. On précise donc qu'un elfe met 10 à 20 unités de temps à fabriquer une roue, 50 à 75 unités de temps à sculpter la carrosserie, 20 à 30 à la peindre, et 0 à 15 unités de temps à assembler les pièces. Rajouter des intervalles aux transitions pour traduire cela et simuler le réseau en mode temporisé (`stepper simulator` en mode `timed`).
3. Modéliser de même le salon de barbier (version avec N clients) en considérant que l'entrée d'un client peut survenir entre 1 et 500 unités de temps et que le rasage d'un client prend en pratique 40 à 70 unités de temps.

Exercice 5 Distributeur de café

On désire modéliser le fonctionnement d'un distributeur de café, un café valant 50 centimes.

Représenter avec un réseau de Petri `renduMonnaie.ndr` le sous-système de *rendu de monnaie* (on ne modélisera pas la confection et livraison du café par exemple).

La machine rend de préférence des pièces de 1 euro, puis ensuite des pièces de 50 centimes. Par exemple sur une pièce de 2 euros, la machine rend de préférence (si le stock de pièces le permet) une pièce de 1 euro et une pièce de 50 centimes.

On dispose de trois transitions "extérieures" : `insert50c`, `insert1e` et `insert2e` franchies quand une pièce est entrée dans le distributeur alors qu'il est en attente de pièce. Le modèle pourra utiliser diverses transitions internes, et devra franchir des transitions `rendu1e` ou `rendu50c` (ces transitions pouvant être présentes en plusieurs exemplaires) quand des pièces sont rendues.

On autorise les arcs inhibiteurs.

On considère que la machine peut automatiquement recharger son stock de pièce de 50 centimes quand il est vide à l'aide d'une transition `recharge` qui remet 10 pièces dans le stock. On ne gère pas dans ce sous-système l'insertion de pièces de 10 ou 20 centimes, bien que cela puisse être envisagé comme extension.