Réseau de Petri : modélisation et propriétés

Exercice 1 On considère le réseau de Petri donné à la figure 1.

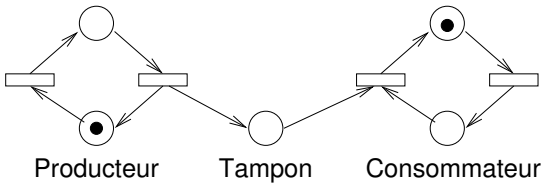

FIGURE 1 – Un exemple de réseau de Petri

1.1 Lire l'appendice A et lister les différents éléments qui constituent ce réseau de Petri et rappeler leur signification.

1.2 Faire évoluer le réseau de Petri en précisant à chaque étape les transitions sensibilisées.

1.3 On constate que la place nommée *Tampon* est non bornée (elle peut contenir un nombre quelconque de jetons). Proposer une évolution du réseau de Petri qui garantit que la place *Tampon* n'aura jamais plus de 3 jetons.

1.4 Construire le graphe de marquage pour ce réseau de Petri.

Exercice 2 : Modèle de processus
En lien avec le cours et pour illustration, nous nous servons d'un langage dédié (DSL) appelé SimplePDL. Ici, nous nous intéressons à des modèles de processus (du développement logiciel) composés d'activités et de dépendances entre ces activités. Nous souhaitons pouvoir vérifier la cohérence d'un tel modèle de processus, en particulier savoir si le processus décrit peut se terminer ou non.

Le modèle de procédé utilisé est inspiré de SPEM [1], standard de l'OMG. Nous donnons entre parenthèses le vocabulaire utilisé par cette norme. Dans un premier temps, nous nous intéressons à des processus simples composés seulement d'activités (WorkDefinition) et de dépendances (WorkSequence). La figure 2 donne un exemple de processus qui comprend quatre activités : Conception, RédactionDoc, Développement et RédactionTests. Les activités sont représentées par des ovales. Les arcs entre activités représentent les dépendances. Une étiquette permet de préciser la nature de la dépendance sous la forme « étatToAction » [2] qui précise l'état qui doit être atteint par l'activité source pour réaliser l'action sur l'activité cible. Par exemple, on ne peut commencer RédactionTests que si Conception est commencée. On ne peut commencer Développement que si Conception est terminée. On ne peut terminer RédactionTests que si Développement est terminé.

2.1 On considère les trois modèle de processus suivants où chaque "Ai" représente une activité quelconque :

1. `A1 --f2s--> A2`
2. `A1 --s2f--> A2`
3. `A1 --s2f--> A2 et A0 --f2f--> A2`

1. `http://www.omg.org/spec/SPEM/2.0/`
2. Par exemple, f2s est l'étiquette de finishToStart.

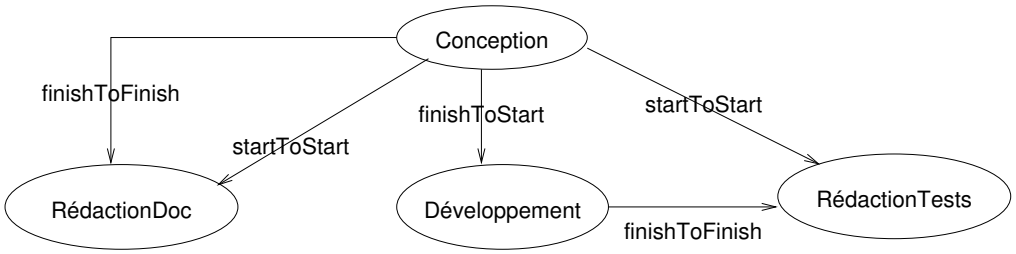

FIGURE 2 – Exemple de modèle SimplePDL

2.1.1

Proposer, pour chacun de ces modèles, plusieurs scénarios qui montrent que le processus peut se terminer (et que donc toutes ses activités se terminent). Il faut bien sûr que les contraintes de dépendance soient respectées.

2.1.2 Donner également des scénarios d'erreur qui montrent un non respect des contraintes de dépendance.

2.2 Proposer une traduction en réseau de Petri des trois modèles de processus précédents.

On essaiera de proposer une traduction systématique des éléments d'un processus (activité et dépendance) pour être capable de traduire tout modèle de processus en un réseau de Petri.

2.3 L'exécution d'un réseau de Petri correspond à une séquence d'états caractérisés par le marquage du réseau. Le passage d'un état au suivant immédiat correspond à une transition unique du réseau. Donner des contraintes sur le marquage d'un réseau de Petri pour que les propriétés suivantes qui portent sur le modèle de processus de la figure 2 soient satisfaites.

1. Le processus peut terminer.
2. Le processus ne peut pas terminer.

Remarque : La question suivante fait partie du mini-projet.

2.4 *Ressources.*

Pour réaliser une activité, des ressources peuvent être nécessaires. Une ressource peut correspondre à un acteur humain, un outil ou tout autre élément jouant un rôle dans le déroulement de l'activité. Ici, nous nous intéressons simplement aux types de ressources nécessaires et au nombre d'occurrences d'un type de ressource. Par exemple, il peut y avoir deux développeurs, deux concepteurs, trois machines, un bloc-note, etc. Un type de ressource sera seulement caractérisé par son nom et la quantité d'occurrences de celle-ci.

Pour pouvoir être réalisée, une activité peut nécessiter plusieurs ressources (éventuellement aucune). Par exemple, l'activité *RedactionTest* nécessite un développeur (une occurrence de la ressource de type Développeur) et deux machines (deux occurrences de la ressource Machine). Les occurrences de ressources nécessaires à la réalisation d'une activité sont prises au moment de son démarrage et rendues à la fin de son exécution. Bien entendu, une même occurrence de ressource ne peut pas être utilisée simultanément par plusieurs activités. Les types de ressource et leur nombre d'occurrences sont définis en même temps que le procédé lui-même.

2.4.1 Modéliser la notion de ressources.

2.4.2 Les propriétés proposées à la question 2.3 doivent elles être adaptées ?

2.4.3 Donner des contraintes sur le marquage d'un réseau de Petri qui expriment que le processus peut se terminer sans utiliser une des occurrences de la ressource Développeur. Ceci signifie que l'on pourrait retirer un développeur au projet sans dommage.

2.5 On considère maintenant que chaque activité doit se dérouler dans un intervalle de temps donné précisant un temps minimal et un temps maximal. Proposer une évolution du réseau de Petri pour prendre en compte cet aspect. On utilisera les aspects temporels des réseaux de Petri.

L'exercice suivant est optionnel

Exercice 3 Le problème classique des lecteurs/rédacteurs consiste à modéliser deux groupes d'acteurs agissant sur des ressources et disposant de capacités différentes :

- *Les lecteurs* lisent (accèdent à) une ressource. Plusieurs lecteurs peuvent lire une même ressource sans qu'aucun problème de partage de la ressource n'apparaisse.
- *Les rédacteurs* écrivent des informations dans une ressource. Un seul rédacteur peut avoir accès à une ressource à la fois. Si un rédacteur a accès à une ressource alors personne (y compris les lecteurs) n'a accès à cette ressource.

3.1 Proposer un réseau de Petri permettant de modéliser le problème des lecteurs/rédacteurs.

3.2 Construire le graphe de marquage pour ce réseau de Petri.

3.3 Vérifier sur le graphe de marquage qu'un rédacteur ne peut jamais avoir accès à la ressource en même temps qu'un lecteur ou un autre rédacteur.

A Rappel : Réseaux de Petri

Le rappel donné ici est extrait de `http://fr.wikipedia.org/wiki/R%C3%A9seau_de_Petri`

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 : $F \subseteq (S \times T) \cup (T \times S)$.
- $M_0 : S \to \mathbb{N}$ appelé marquage initial (ou parfois place initiale), où, pour chaque place $s \in S$, il y a $n \in \mathbb{N}$ jetons.
- $W : F \to \mathbb{N}^+$ 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 produis 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. Le marquage d'un réseau de Petri caractéristique l'état de ce réseau à une étape de son exécution.

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 3 propose quelques exemples de réseaux de Petri.

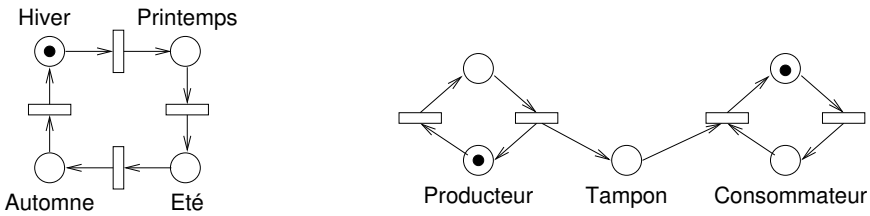

FIGURE 3 – Exemples de réseaux de Petri

Dynamique d'exécution Un réseau de Petri évolue lorsqu'on exécute [3] une transition : des jetons sont pris dans les places en entrée de cette transition et envoyés dans les places en sortie 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 exécutable [4], 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.

3. On emploie aussi les verbes franchir ou tirer.
4. On dit aussi franchissable, sensibilisée, validée ou tirable.

Une exécution possible d'un réseau de Petri est une séquence de marquage correspond aux états du réseau au fur et à mesure de son exécution. Le passage d'un état à l'état suivant se fait par l'exécution d'une seule transition.

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 On définit deux extensions sur ces réseaux de Petri. Tout d'abord nous définissons une notion de temps sur les transitions sous la forme d'un intervalle $[temps_{min}, temps_{max}]$. Le temps commence à s'écouler à partir du moment où la transition est exécutable. Elle doit alors s'exécuter dans l'intervalle de temps indiqué. On parle alors de réseau de Petri temporel.

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 les transitions qui ont cette place pour entrée.

La figure 4 présente un exemple de réseau de Petri temporel avec read-arc.

FIGURE 4 – Le même réseau de Petri avec un read-arc et sans