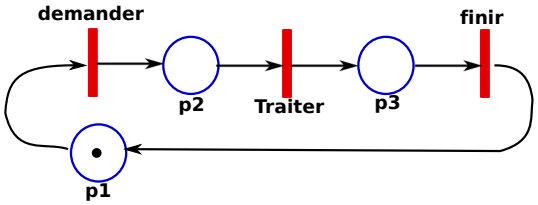
Cahier d'exercices 1 - Modélisation avec les réseaux de Petri *Place/Transition*
Tous les exos doivent être faits (finissez à la maison ceux qui ne sont pas traités en TD)
Nombre de séances : ... Compte-rendu demandé : ...selon consignes...



Rappels

Les réseaux de Petri *Place/Transition* permettent de décrire
- le comportement de processus (sous-systèmes) et les manières d'accéder à des ressources (données) partagées par des processus ;
- le comportement global d'un système ;
- l'effet de l'accès et de la manipulation des ressources sur le système global.

Pour traiter les exercices suivants, vous allez vous baser sur les exemples déjà vus en cours pour certains (producteur/consommateur, philosophes, lecteur/rédacteur).

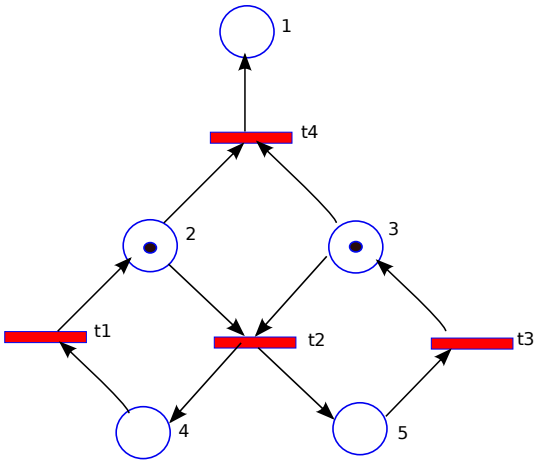
Méthode de modélisation

Selon le cahier de charges (ici les exercices) à traiter,
- **identifiez les ressources** et par conséquent les modes d'accès ;
  chaque ressource sera systématiquent modélisée par **au moins une place** ;
- **identifiez les processus**, et par conséquent leurs comportements (suites d'actions) ; précisez pour chaque processus, quelles ressources il utilise ;
  de façon générale une action est modélisée par **une transition** ; cependant pour certaines actions il faut une autorisation (représentée par **une transition**) quand les conditions pour effectuer l'action sont remplies. Ces conditions sont représentées par **des places**. Ainsi chaque action (du comportement) d'un processus est précédée d'**une transition** qui l'autorise et éventuellement suivie par une transition.
- réutilisez les techniques de base (pour l'accès à une ressource : demander, utiliser, libérer), **synchronisation, exclusion mutuelle**.
- Vous pouvez alors synthétiser avec un tableau comme le suivant :

	ressources$_1$	**ressources**$_2$	**ressources**$_3$	$\cdots$
Processus$_1$	actions$_i$,...	actions$_j$,...	actions$_r$...	actions$_u$
Processus$_2$	actions$_j$,...	actions$_i$	actions$_k$	actions$_i$
$\cdots$	actions,...	actions,...	actions,...	actions,...

Exercice - rappels du fonctionnement d'un réseau de Petri

Ouvrez votre support de cours et exploitez le pendant tout le TD. Soit le réseau de Petri suivant :



Q#1 Listez l'ensemble des places de ce réseau.

Q#2 Listez l'ensemble des transitions de ce réseau.

Q#3 Quelles sont les transitions *tirables* (ou *franchissables*) dans ce réseau ? Expliquez pourquoi.

Q#4 Ecrivez une bande dessinée constituée du réseau, dans les étapes successives de son fonctionnement (en tirant une transition à chaque étape).

Q#5 Donnez le marquage initial de ce réseau (en utilisant la fonction $(\mu(p1), \mu(p2), \cdots)$).

Q#6 Réduisez votre bande dessinée en utilisant les marquages successifs comme abstraction du réseau.

Exercice 1 : Producteur / Consommateur (vu en cours)

Un système producteur/consommateur met en jeu un *processus producteur*, un *processus consommateur* et une *ressource partagée* sous forme d'un tampon ; le processus producteur y dépose des produits qui sont ensuite retirés par le processus consommateur avec la contrainte de **synchronisation** exigeant au consommateur d'attendre au moins une production avant de pouvoir consommer.

Q#7 Construisez le réseau de Petri correspondant à un système producteur/consommateur avec un tampon de **taille illimitée**.

Q#8 Construisez le réseau de Petri correspondant à un système producteur/consommateur avec un tampon de **taille limitée**.

Exercice 2 : Accès à une ressource en exclusion mutuelle

Rappel du principe de l'exclusion mutuelle entre processus : à un moment donné un seul processus accède/utilise une ressource qu'il a demandée et obtenue. Après usage de la ressource, le processus la libère.

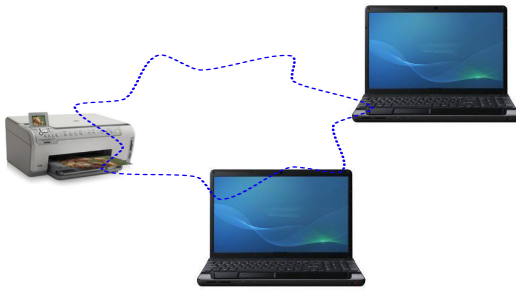


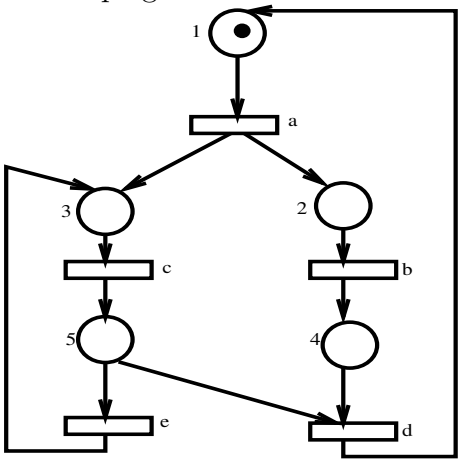
FIGURE 1 – Accès à une imprimante partagée

Q#9 Des usagers sur un réseau partagent entre autres, une imprimante ; ils envoient des messages pour imprimer des documents. Construisez le réseau de Petri modélisant l'accès à l'imprimante par deux processus p_1 et p_2 (représentant les usagers). On veut que les impressions de chaque processus se déroulent en exclusion mutuelle. Il n'y a aucune sorte de priorité entre les processus.

Q#10 On veut gérer l'accès à une ressource par deux processus. Ces processus peuvent être soit au repos (par rapport à la ressource), soit en demande de la ressource, soit en utilisation de la ressource.
Construisez le réseau de Petri modélisant cette gestion.

Exercice 3 : graphe de marquage, expression régulière

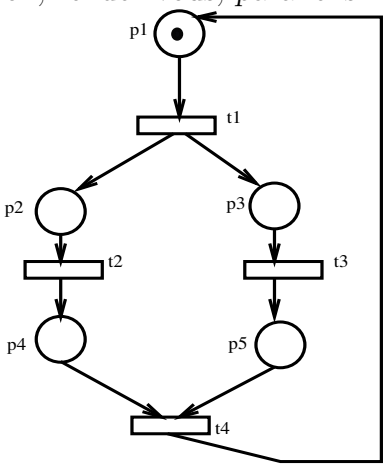
Q#11 Construisez le graphe de marquage associé au réseau suivant.



Q#12 Donnez une expression régulière décrivant le langage associé au réseau (ou comportement du système modélisé).

Exercice 4 : graphe de marquage

Q#13 Construisez le graphe de marquage associé au réseau suivant et discutez le cas échéant les aspects suivants : *synchronisation, rendez-vous, parallélisme, exclusion mutuelle.*



Q#14 Donnez l'expression régulière correspondant au comportement du réseau.

Exercice 5 : le problème des philosophes

Quatre philosophes ϕ_1, ϕ_2, ϕ_3, ϕ_4 partagent leur temps entre repos (pensée) et repas. Avant de manger ils doivent **se munir successivement du couvert situé à leur droite puis du couvert situé à leur gauche**. On a vu qu'on obtient un interblocage très rapidement avec cette stratégie (qui est une pouvant surgir dans le cas général).

Q#15 Modéliser à l'aide de réseau de Petri le problème des philosophes en évitant soigneusement la situation d'*interblocage*.

Exercice 6 : modélisation d'une application de *e-commerce*

On veut analyser les interactions entre un client (un processus, ou une application) qui interagit avec un serveur d'un site marchand.

Cas élémentaire.

Un **client** adresse une demande d'achat d'un produit au serveur (marchand). Le serveur acquite la réception de la demande. A la réception de cet acquitement, le client effectue le paiement de son achat tout en attendant la livraison de son achat, pour clore cette transaction avec le site marchand. Le paiement de l'achat est attendu par le serveur.

A la suite du paiement du client auprès du serveur, le client reçoit la livraison de son achat (venant d'un gestionnaire de stock, par exemple).

Un **serveur**, en attente de clients, lorsqu'il reçoit une demande d'achat, envoie une requête au gestionnaire de stock qui fournira le produit, puis acquite la bonne réception de la demande au client. Le serveur se met ensuite en attente du paiement par le client. Suite au paiement, le serveur autorise le gestionnaire de stock à livrer l'achat.

Le **gestionnaire de stock**, en attente des requêtes venant du serveur, lorsqu'il reçoit une requête, prépare la livraison d'un produit/achat, puis attend l'autorisation du serveur pour le livrer.

Nous modélisons uniquement les aspects comportementaux ; nous n'allons donc pas nous préoccuper des aspects quantitatifs (les produits, leur nature, leur quantité, leur prix, etc).

Q#16 Modéliser cette interaction à l'aide de réseaux de Petri, en procédant par étapes ; modélisation du processus client, modélisation du serveur, et enfin la modélisation du gestionnaire de stock.

Q#17 Modéliser la composition des (trois) processus. Est-il possible de faire des compositions deux à deux ?

Q#18 A partir du réseau global obtenu, analyser la structure de composition des processus (par exemple, est-il possible de distinguer les places ou transitions sur lesquelles repose la composition) ?

Q#19 Réflechisser à la dérivation de programmes/applications à partir de cette modélisation (comment elle vous aide à structurer votre application ?).

Généralisation.

Q#20 Proposer des améliorations du cas élémentaire, avec des hypothèses réalistes et raisonnables. Essayer la modélisation avec des réseaux de Petri. Quelles difficultés particulières se posent ?

Q#21 Imaginer une phase d'inscription préalable du client. Il y a des clients déjà inscrits sur le site marchand et des clients non encore inscrits.

Outils de modélisation

Nous allons effectuer les expérimentations de modélisation et d'analyse avec les outils suivants essentiellement : romeo et pipe

Utilisation sur les machines (virtuelles) de l'IUT :
romeo : (lancer dans un terminal)

pipe : /usr/lib/jvm/java-8-openjdk-amd64/bin/java -jar /opt/app/PIPE.jar

Autres outils en ligne : APO
`https://apo.adrian-jagusch.de/` et `https://github.com/stromhalm/apo`

Retour sur la page du cours :
`http://pagesperso.ls2n.fr/%7Eattiogbe-c/mespages/programmation-applis-reparties.html`