Chapitre 8

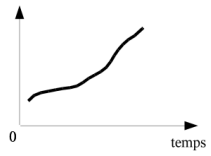
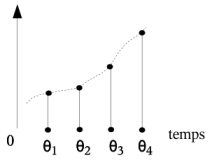
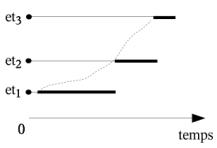
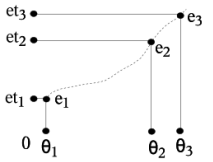
Modèles de la dynamique des systèmes

8.1 Modélisation de la dynamique d'un système

Modélisation de la dynamique d'un système

La modélisation statique s'intéresse à ce qu'il y a dans le système, à sa structure, etc. La modélisation de la dynamique traite de l'évolution du système dans le temps.. Il s'agit de modéliser l'état du système et sa transformation. Les variables qui représentent l'état peuvent prendre des valeurs continues (dans l'ensemble R) ou discrètes (il y a un nombre fini ou infini dénombrable de valeurs possibles). Il en va de même pour le temps. On obtient alors quatre types de modélisation :

Types de modélisation de systèmes

↓ var.T →	continu	discret
continues	systèmes continus 	systèmes échantillonnés 
discrètes	systèmes discrets 	*systèmes à évènements discrets* 

Parmi ces modélisations nous nous intéresserons aux systèmes à évènements discrets, pour lesquels nous étudierons deux formalismes de modélisation : les automates à états finis et les réseaux de Petri.

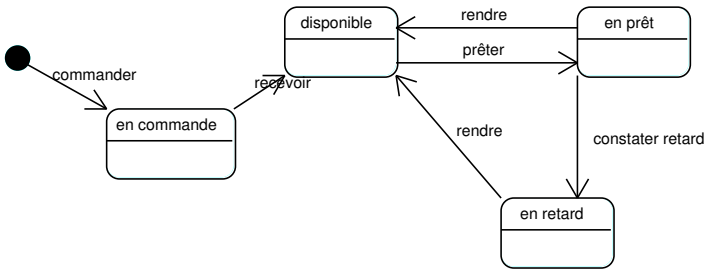
8.2 Automates à états

Le formalisme des automates à état représente un système à évènements discrets à l'aide

1. d'états
2. d'évènements
3. de transitions

Une transition $s \xrightarrow{e} t$ signifie : quand on est dans l'état s et que l'évènement e de produit on doit passer à l'état t.

Exemple 8.1. Cycle de vie d'un livre



Sémantique/Fonctionnement

Étant donné une séquence d'évènements $(e_1, \ldots, e_k)$
L'automate passe successivement par les états $(s_0, s_1, \ldots, s_k)$ si et seulement si
- s_0 est l'état initial
- il possède les transitions $s_0 \xrightarrow{e_1} s_1, \ldots, s_{k-1} \xrightarrow{e_k} s_k$.

Utilisation

1. trouver l'état atteint par une séquence d'évènements
2. vérifier si une séquence d'évènements atteint un état souhaité
3. définir toutes les séquences qui atteignent un état souhaité
4. etc.

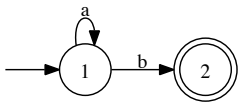
Définition formelle

Un automate est composé de
- un ensemble S d'états
- un état initial $s_0 \in S$
- un ensemble F d'états accepteurs (états finals)
- un alphabet d'entrée Σ (le vocabulaire d'évènements)
- une fonction de transitions $\delta : S \times \Sigma \rightarrow S$

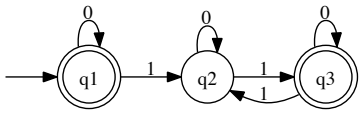
Acceptation d'une chaine de symboles

La chaine de symboles $x_1 \ldots x_n \in \Sigma^*$ est acceptée par l'automate si et seulement si il existe une séquence d'états $(s_0, s_1, \ldots, s_n)$ telle que
- s_0 est l'état initial
- $\delta(s_{i-1}, x_i) = s_i$ pour $i = 1, \ldots, n$
- $s_n \in F$

Exemple 8.2. L'automate suivant, représenté graphiquement (le double cercle indique un état accepteur et les flèches forment la fonction de transition) accepte toutes les chaines qui commencent par zéro, un ou plusieurs a et se terminent par un b.



Exemple 8.3. Cet automate accepte les chaines qui contiennent des 0 et un nombre pair de 1.



8.3 Automates et langages réguliers

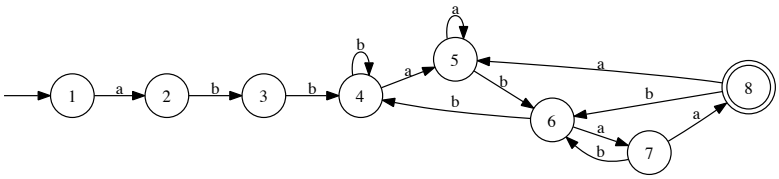
L'ensemble des chaines de Σ^* acceptée par un automate $\mathcal{A}$ d'alphabet Σ forme un langage. On le note $\mathcal{L}(\mathcal{A})$

On peut montrer que

– le langage d'un automate est toujours régulier

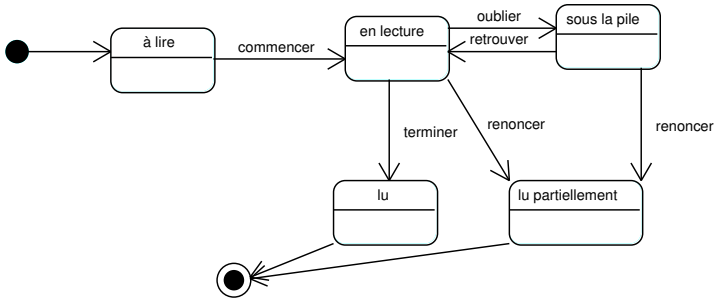
– tout langage régulier correspond à un automate

Exemple 8.4. Automate du langage $abb(a|b)^*abaa$



8.3.1 Retour à la modélisation dynamique

On considère le cycle de vie d'un livre du point de vue du lecteur



Si l'on veut modéliser avec un seul automate les deux aspects de la vie d'un livre on arrive à un très grand nombre d'états (le produit cartésien des états des automates partiels).

Dans l'exemple :

{en commande, disponible, en prêt, en retard}

×

{à lire, en lecture, sous la pile, lu, lu partiellement}

L'automate devient complexe (beaucoup de transitions)

Si l'on modélise le système avec plusieurs automates, il faut aussi représenter les interconnexions entre automates.Par exemple. La transition de à lire à en lecture de l'automate 2 ne peut avoir lieu que si l'automate 1 est dans l'état prêt. Il faut donc pouvoir représenter certaines formes de synchronisation entre automates.

8.4 Réseaux de Petri

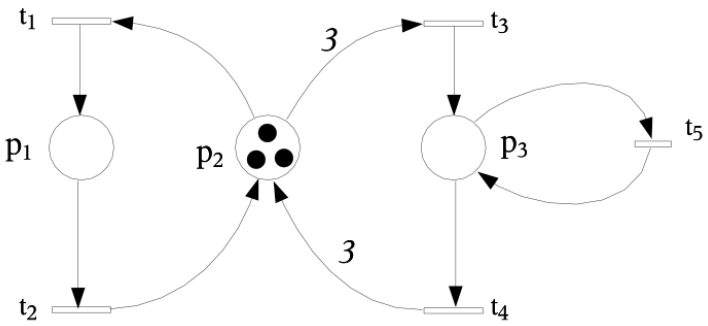
Pour représenter l'évolution d'un système on s'intéresse aux (pré)conditions de déclenchement des évènements et aux (post)conditions produites par les évènements. On peut également considérer que les évènements consomment et produisent des ressources.

Définition 8.1. Un réseau de Petri (RdP) est composé

- d'un ensemble de place
- d'un ensemble de transitions
- d'un ensemble d'arcs qui associent les places (d'entrée) aux transitions et les transitions aux places (de sortie)
- de poids (entiers) associés aux arcs

L'**état** d'un réseau est défini par son **marquage**. Un marquage associe à chaque place un nombre entier positif ou nul de jetons.

Exemple 8.5. Le réseau ci-dessous possède les places p_1, p_2, p_3, les transitions $t_1, \ldots, t_5$ et les arcs indiqués par les flèches. Par défaut le poids d'un arc vaut 1. Dans ce réseau tous arcs ont un poids égal à 1 sauf les arcs (p_2, t_3) et (t_4, p_2) qui ont un poids de 3. Le marquage de ce réseau est $(p_1 \mapsto 0, p_2 \mapsto 3, p_3 \mapsto 0)$, que l'on peut aussi écrire sous forme d'un vecteur : $(0, 3, 0)$.



Dynamique d'un réseau de Petri

Etant donné un marquage, une transition est **sensibilisée** (ou déclenchable ou franchissable) si dans chacune de ses place d'entrée il y a au moins le nombre de jetons indiqué par le poids de l'arc correspondant.

Le **déclenchement** (ou franchissement) d'une transition sensibilisée consomme des jetons de ses places d'entré et ajoute des jetons dans ses places de sortie. Le nombre de jetons consommés et produits correspond aux poids des arcs. À la suite d'un déclenchement on obtient un nouveau marquage, qui n'a pas forcément le même nombre total de jetons que le précédent.

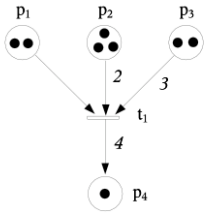


FIG. 3: t_1 non sensibilisée

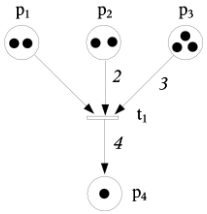


FIG. 4: t_1 sensibilisée

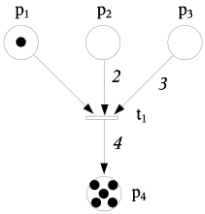
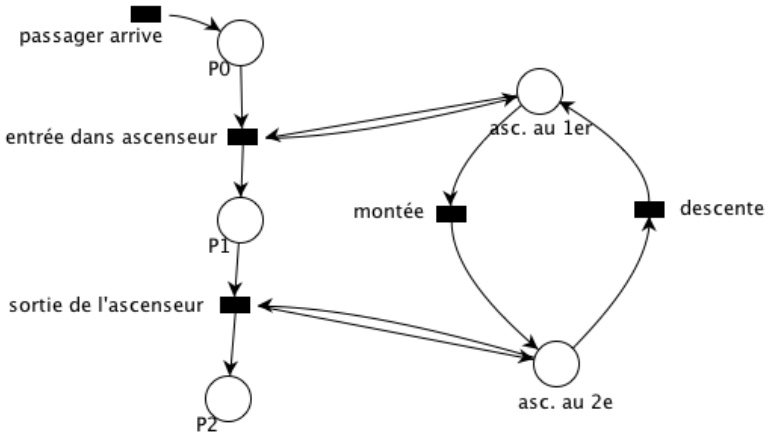


FIG. 5: t_1 franchie

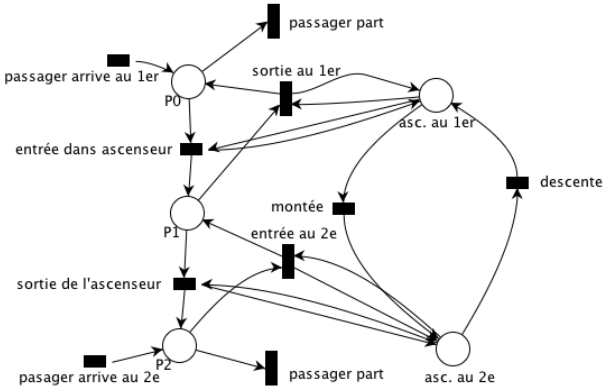
Étant donné que plusieurs transitions peuvent être sensibilisées dans un même marquage, l'évolution du réseau dépend du choix de la transition à déclencher. Ainsi, à partir d'un état le réseau peut évoluer selon différents scénarios, en fonction des choix faits à chaque étape. L'idée et l'intérêt du réseau de Petri est précisément de représenter potentiellement toutes les évolutions possibles d'un système puis de calculer des propriétés qui restent valables quelle que soit l'évolution.

Exemple 8.6. On modélise un ascenseur qui se déplace entre le 1^{er} étage et le 2^{e} et des passagers qui arrivent au 1^{er} et se rendent au 2^{e}. On ne représente pas l'appel de l'ascenseur ni le choix de l'étage de destination (de toute manière il n'y en a qu'un).



On constate : l'ascenseur peut monter et descendre, qu'il soit vide ou non. Par contre, un passager ne peut entrer que si l'ascenseur est au 1^{er} et il ne peut sortir que si l'ascenseur est au 2^{e}. L'ascenseur peut monter avant que tous les passagers ne soient entrés. Dans ce réseau les marquages des places $P0, P1, P2$ représentent des nombres de personnes (en attente, dans l'ascenseur, sorties) alors que le marquage de de asc. au 1er et asc. au 2e représentent la présence (1) ou l'absence (0) de l'ascenseur.

Exemple 8.7. On étend le réseau pour représenter également des passagers qui arrivent au 2^{e} et veulent se rendre au 1^{er}.

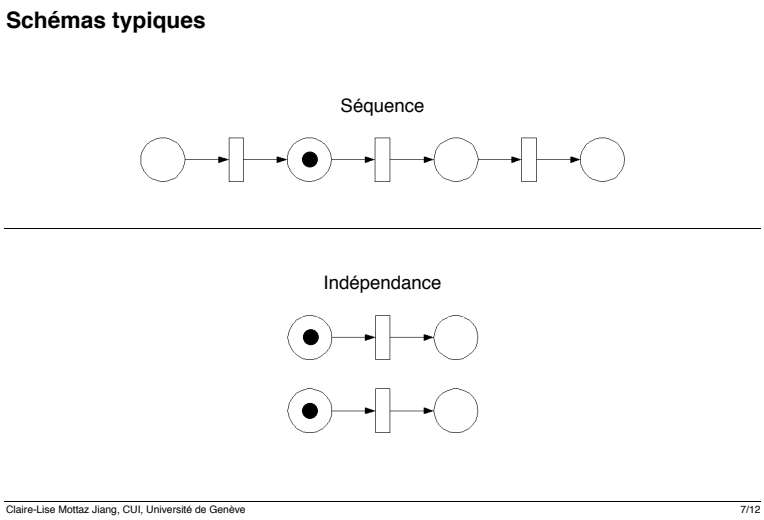


On constate sur ce réseau qu'un même passager peut monter et descendre autant de fois qu'il veut avant de quitter le système.

(Dessins réalisés avec PIPE 2 : http ://pipe2.sourceforge.net/)

8.5 Schémas pour la construction « top down » des réseaux

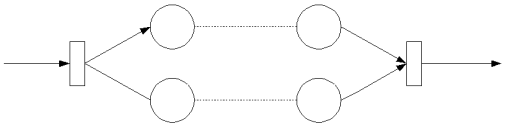
Lorsqu'on modélise des systèmes dynamiques on rencontre des situations typiques qui induisent des schémas de réseau de Petri. La *séquence* est une situation dans laquelle une série de transitions doivent obligatoirement se dérouler l'une après l'autre. Par exemple : *écrire un message* doit avoir lieu avant *envoyer le message*. Par contre d'autres transitions peuvent être indépendantes, on peut déclencher l'une sans tenir compte de l'autre. Par exemple *écrire un message* et *ouvrir la porte*. Ces deux situations correspondent aux schémas ci-dessous.



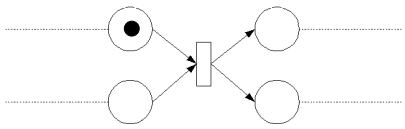
La décomposition parallèle signifie qu'un processus peut se décomposer en deux processus s'exécutant indépendamment l'un de l'autre. Pour que le processus soit considéré comme terminé il faut que les deux sous-processus soient terminés, quel qu'ait été l'ordre de leur exécution. Il se peut que deux processus séquentiels qui se déroulement indépendamment aient besoin de se synchroniser. Dans ce cas on crée un rendez-vous (voir figure) qui force chacun à attendre que l'autre ait atteint un certain point avant de continuer.

Schémas typiques

Parallélisme



Rendez-vous



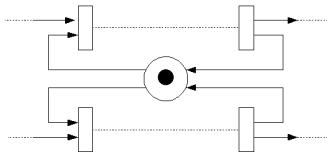
Claire-Lise Mottaz Jiang, CUI, Université de Genève 8/12

Deux processus indépendants peuvent avoir besoin d'une même ressource, qui ne peut être utilisée par les deux à la fois. On représente cette situation par une place correspondant à la disponibilité de la ressource. Si la ressource est disponible (marquage à 1) l'un des processus peut la prendre, ce qui bloque l'autre. Quand il a fini d'utiliser la ressource il remet la place correspondant à 1, libérant ainsi l'autre processus.

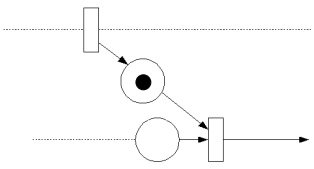
Le sémaphore est utilisé lorsqu'un processus ne peut avancer que si l'autre a déjà franchi une transition donnée.

Schémas typiques

Partage de ressource (exclusion mutuelle)

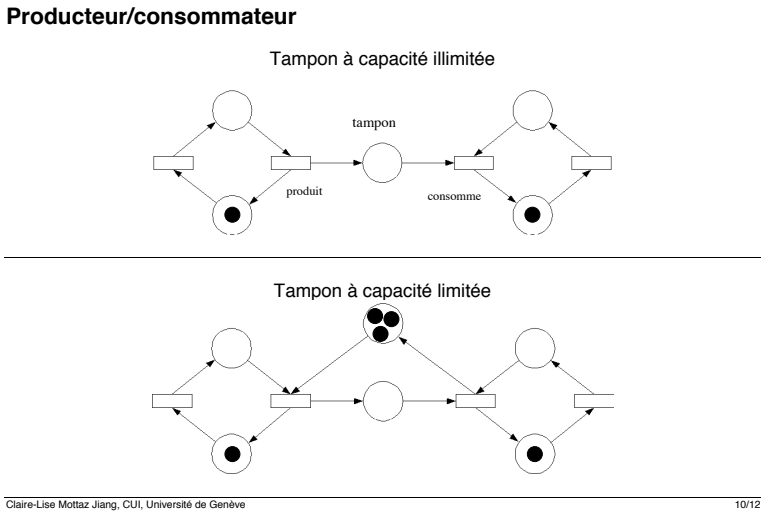


Sémaphore



Claire-Lise Mottaz Jiang, CUI, Université de Genève 9/12

Le schéma producteur/consommateur correspond à une situation où l'un des processus produit des ressources nécessaires à l'autre. On utilise une place « tampon » pour matérialiser le nombre de ressource produites mais pas encore consommées. On peut limiter la capacité du tampon pour éviter que le producteur ne « prenne trop d'avance » sur le consommateur. On ajoute une place « compteur » initialisée à la capacité désirée du tampon. Chaque ajout au tampon diminue le compteur d'une unité, chaque retrait l'augmente. Lorsque le compteur est à zéro le producteur se bloque jusqu'à ce que le consommateur ait consommé au moins une ressource du tampon.



Atteignabilité

Pour un réseau R et un marquage M, le marquage M' est directement atteignable (un successeur de M) s'il existe une une transition t déclenchable qui produit le marquage M'. On notera

$$M \xrightarrow{t} M'$$

Pour un réseau R et un marquage M, le marquage M' est atteignable s'il existe une séquence correcte de déclenchements de transitions $s = t_1, \ldots t_n$ telle que

$$M \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \cdots \xrightarrow{t_n} M_n = M'$$

Graphe d'atteignabilité

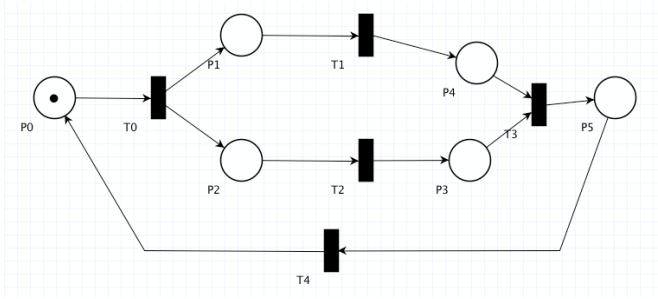
Le graphe d'atteignabilité de R pour le marquage M est composé de

sommets tous les marquages M_j atteignables depuis M

arcs s'il existe une transition t telle que $M_1 \xrightarrow{t} M_2$ on a un arc de M_1 à M_2 étiqueté par t.

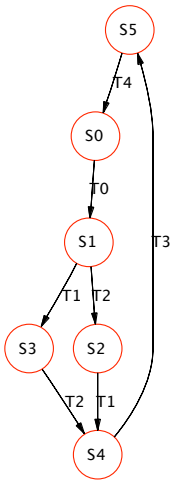
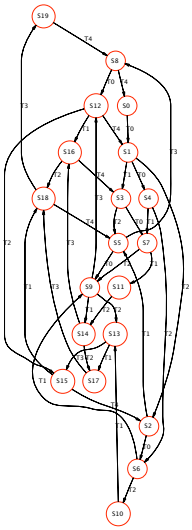
Exemple de graphe d'atteignabilité

Pour le réseau et le marquage



Les marquages atteignables sont :
S0 = (1, 0, 0, 0, 0, 0) (marquage initial)
S1 = (0, 1, 1, 0, 0, 0)
S2 = (0, 1, 0, 1, 0, 0)
S3 = (0, 0, 1, 0, 1, 0)
S4 = (0, 0, 0, 0, 0, 1)

Le tableau ci-dessous montre le graphe d'atteignabilité obtenu pour le marquage initial S0 puis pour le marquage $(2, 0, 0, 0, 0, 0)$. On constate que l'ajout d'un jeton complexifie énormément le graphes. Dans certains cas la taille du graphe augmente exponentiellement avec le nombre de jetons du marquage.

Marquage initial (1,0,0,0,0,0)	Marquage initial (2,0,0,0,0,0)
	

Vivacité des RdP

De nombreux systèmes dynamiques sont destinés à fonctionner en permanence, si possible sans blocages. Que l'on pense au système de signalisation routière d'une ville, au système d'information d'une organisation, au système d'exploitation d'un PC ou d'un serveur. Ce bon fonctionnement peut être représenté par des propriétés d'un modèle du système sous forme d'un réseau de Petri.

Quelques définitions

Une **transition** t est **quasi-vivante** pour un réseau R avec un marquage M si elle peut être déclencée au moins une fois. C'est-à-dire s'il existe une séquence de déclenchements à partir de M qui contient t.

Un réseau R avec un marquage M est **quasi-vivant** si toutes ses transitions sont quasi-vivantes.

Une **transition** t est **vivante** si, quel que soit le marquage M atteignable depuis le marquage initial M_0 du réseau, t est quasi vivante à partir de M.

Autrement dit, tout état peut évoluer de manière à franchir à nouveau t.

Un **réseau** est vivant si toutes ses transitions sont vivantes.

Un réseau avec **blocage** est un réseau qui peut être mis dans une configuration telle qu'aucune transition ne peut plus être effectuée.

♢

Il existe des algorithmes pour tester ces propriétés et donc pour garantir qu'un système conforme à un réseau donné ne se bloque jamais.