Université Mohammed Seddik Ben yahia- Jijel
Faculté des Sciences Exactes et Informatique
Département Informatique

2016

Simulation & Prototypage

Support de Cours

Simulation & Prototypage

Spécialités

Système d'Information et Aide à la Prise de Décision (SIAD)

Auteur

Dr. KERKOUCHE EL-HILLALI

Ministère de l'Enseignement Supérieure et Recherche Scientifique
Université de Jijel

Faculté des Sciences Exactes et Informatique
Département Informatique

Support de Cours

Simulation & Prototypage

Spécialité

Système d'Information & Aide à la Prise de Décision (SIAD)

Présenté par

Dr. KERKOUCHE El-Hillali

Simulation & Prototypage

Le cours de Simulation & Prototypage est préparé pour servir comme support pédagogique d'étudiants inscrits en première année Master de spécialités système d'information et aide à la prise de décision (SIAD) filière informatique.

Table des Matieres

Table des Matières

Chapitre 01 : Modélisation & Simulation : une Introduction

Chapitre 02 : Machines d'Etats Finis

Chapitre 01 :

Modélisation & Simulation : une Introduction

1.1 Introduction

Durant la dernière décennie, les systèmes ont connu un développement sans précédent qui s'est accompagné d'un accroissement important de leur complexité. Pour maîtriser la complexité de tels systèmes, il est nécessaire de disposer d'outils performants permettant de comprendre leurs comportements dynamiques, prédire leurs performances, d'évaluer et de comparer les différentes configurations ou stratégies opératoires et de les optimiser.

Etudier un système consiste, dans la plupart des cas, à faire un ensemble de suppositions sur son fonctionnement. Ces suppositions, prennent généralement la forme de relations mathématiques ou logiques, constituent ainsi un *modèle* qui est utilisé comme support pour conduire des expériences afin d'évaluer et de déduire les performances du système qu'il le représente. Si les relations qui composent le modèle sont assez simples il peut être possible d'utiliser des méthodes mathématiques (telle que l'algèbre, la théorie des probabilités) pour obtenir des réponses exactes aux questions qui nous intéressent. Une telle solution s'appelle une solution analytique. Cependant, les systèmes qu'on trouve dans la réalité sont le plus souvent trop complexes pour pourvoir se prêter a une évaluation analytique, et leurs modèles doivent être étudiés au moyen de la *simulation*.

La simulation est reconnue comme une technique puissante pour la conception et l'analyse des systèmes. En générale, elle est perçue comme un outil d'aide à la décision dont le but principal est d'étudier les performances d'un système complexe avant sa construction, ou d'évaluer les performances d'un système existant pour l'améliorer s'il y a une grand différence au niveau de performances. Dans une simulation, on utilise l'ordinateur pour évaluer numériquement un modèle dans le but d'estimer les caractéristiques souhaitées du modèle.

La simulation peut être utilisée dans plusieurs domaines d'application, tels que :

- *Systèmes informatiques et télécommunications*
    - Etude des comportements des systèmes d'exploitation
    - Evaluation de protocoles de gestion des transactions de bases de données,
    - Etude de la file d'attente mémoire d'un serveur,
    - Etude des comportements des utilisateurs,
    - Conception et dimensionnement de hubs.
- Systèmes de flux de production ou de fabrication
    - Equilibrage de lignes d'assemblage,
    - Conception de systèmes de transfert entre des postes,
    - Dimensionnement des stocks d'un atelier,
    - Comparaison de pilotage,

- Evaluation de la charge prévisionnelle,
- Etude de la synchronisation entre les réceptions des pièces et l'assemblage.

– Systèmes de services
  - Etude de transactions bancaires,
  - Gestion de restaurants,
  - Comparaisons de politiques de maintenance des avions.

– Systèmes de transport et Flux logistiques
  - Conception et dimensionnement d'entrepôts,
  - Dimensionnement d'une flotte de camions,
  - Etude de procédures de contrôle des flux de véhicules en circulation.

– Les systèmes naturels (biologiques, écologiques,...)

1.2 Notion de Système

Le terme *système* couvre un large champ d'application, car il peut être utilisé dans de nombreux domaines. Il existe de nombreuses définitions qui lui ont été attribuées dans la littérature. En voici une liste non exhaustive :

- Ensemble de composants reliés entre eux.
- Combinaison de parties qui se coordonnent, pour concourir à un résultat de manière à former un ensemble (Larousse).
- Ensemble d'éléments interagissant entre eux selon certains principes ou règles.
- Ensemble d'éléments matériels ou immatériels en interaction, organisés pour l'accomplissement d'une certaine mission.

Le plus important ce n'est pas d'énoncer une définition précise du terme mais d'essayer de faire ressortir les mots clés qui semblent bien caractériser la notion de système. De là, on peut dire qu'un système est caractérisé par :

- C'est un tout composé de parties ordonnées.
- Chaque partie a ses lois et une certaine indépendance.
- Les parties ont des liens ou relations entre elles pour atteindre un but.
- Cet ensemble change au cours du temps.
- Il est influencé par le milieu dans lequel il existe et qui réagit sur lui.
- L'ensemble est le plus souvent soumis à des contraintes.

Ces caractéristiques font ressortir qu'un système est déterminé par :

- ✓ La connaissance de sa frontière, c'est-à-dire le critère d'appartenance au système déterminant si une entité appartient au système ou fait au contraire partie de son environnement.
- ✓ La connaissance de ses interactions avec son environnement.
- ✓ La connaissance de ses parties ou composants (la nature de ses éléments constitutifs).
- ✓ La connaissance des lois propres de chaque composant.
- ✓ La connaissance des lois d'interaction qui déterminent son but.

Chaque système est influencé par son environnement. Un système peut être ouvert, fermé, ou isolé selon son degré d'interaction avec son environnement. L'identification des frontières du système avec son environnement (son milieu) permet de délimiter la portée du système c'est-à-dire déterminer ce qui doit être à l'intérieur du système et ce qui doit rester à l'extérieur. Les relations entre un système et son environnement sont des *relations de cause à effet*. Certains facteurs (paramètres) externes peuvent influencer le comportement du système :

- Si ces facteurs contrôlent entièrement la dynamique du système, il n'y a pas d'intérêt à conduire des expérimentations avec ce système, et il faudra le redéfinir.
- Si par contre ces facteurs contrôle partiellement la dynamique du système, On peut alors soit :

1. Elargir le système de façon à les inclure.
2. Les ignorer (si on juge que leurs effets sont négligeables).
3. Les considérer comme des entrées du système.

Donc, décrire un système consiste à déterminer ses éléments, ses relations, leurs propriétés et les valeurs que peuvent prendre ainsi que son activité et l'organisation qui en découle.

1.2.1 Définition d'un Système

"Un système est une entité complexe traitée (en égard à certaines finalités) comme une totalité organisée, formée d'éléments et de relations entre ceux-ci, les uns et les autres étant définis en fonction de la place qu'ils occupent dans cette totalité et cela de telle sorte que son identité soit maintenue face à certaines évolutions." [AFCET 77].

La complexité d'un système provient de la multiplicité des éléments qui le composent, de leurs objectifs qui peuvent être en conflit, de la diversité de leurs relations (ou interactions), et de l'incertitude concernant l'évolution globale du système. L'analyse du système doit disposer de méthodes et d'outils capables de représenter ces systèmes. Ces représentations ne doivent pas être trop simples afin de ne pas perdre de l'information sur le fonctionnement dynamique, et ne doivent pas être trop complexes et inexplicables pour simplifier l'analyse.

Exemple

Une usine de production représente un bon exemple de système. Les parties de ce système pourraient être (voir la Figure 1.1) :

- ☞ Administration
- ☞ Employés
- ☞ Service Production (Ateliers, Machines, …).
- ☞ Service approvisionnement.
- ☞ Service ventes.
- ☞ Service de gestion de stocks.
- ☞ …

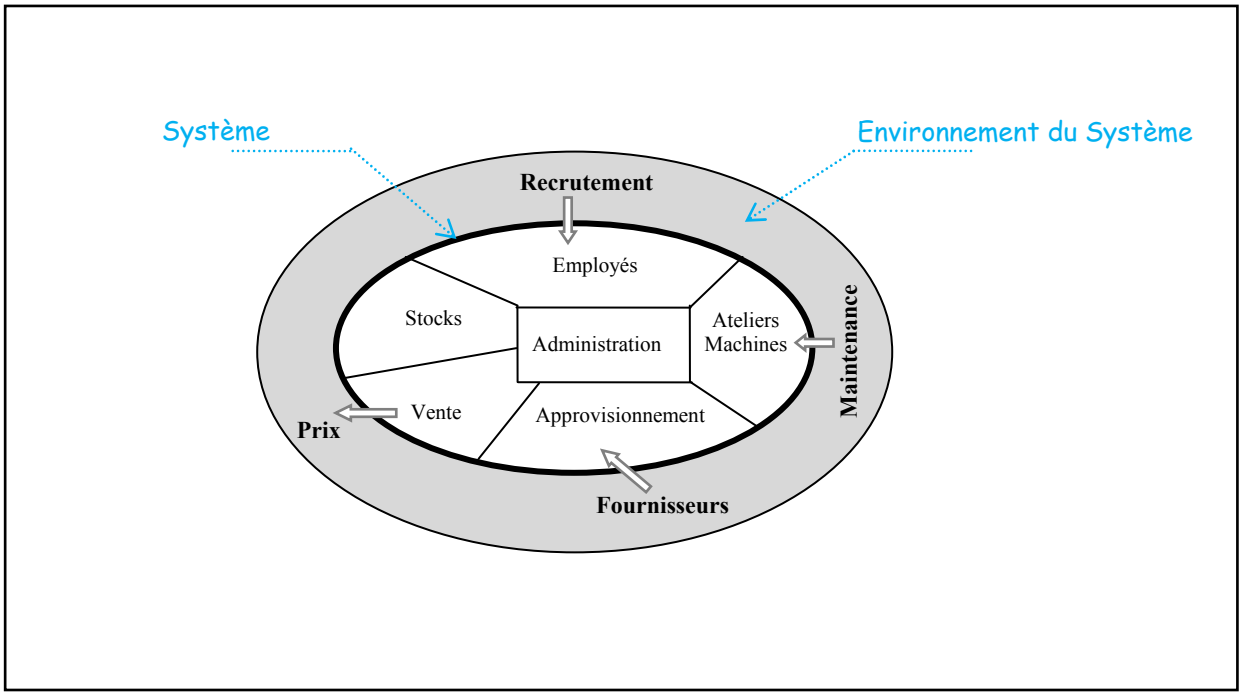


Figure 1.1 Le Système et Son Environnement.

Chaque partie du système a ses propres lois de fonctionnement et elle partiellement indépendante des autres parties. Toutes les parties du système sont en interaction pour atteindre l'objectif attendu. La portée du système est délimitée par l'identification de ses frontières et les différentes interactions possibles.

1.3 Modélisation et Modèle

La modélisation est une démarche consistant à élaborer une description simplifiée (à l'aide d'un langage mathématique ou logique) d'un phénomène, d'un processus ou d'un système appelée généralement un modèle, en vue d'en étudier ou d'en prévoir son fonctionnement.

1.3.1 Définition (Modèle)

"Un modèle est un schéma qui, pour un champ de questions, est pris comme représentation abstraite d'une classe de phénomènes dégagés de leur contexte par un observateur pour servir de support à l'investigation, et/ou à la communication." [AFCET 77] .

Un modèle est donc une représentation d'un système (réel ou imaginaire, existant ou en cours de conception) dont le but est d'*expliquer* et de *prédire* certains aspects du comportement de ce système. Cette représentation est plus ou moins *fidèle* pour répondre au compromis entre adéquation modèle-système et facilité d'analyse du modèle:

- le modèle devra être assez complet afin de pouvoir répondre aux diverses questions qu'on peut se poser sur le système qu'il représente.
- le modèle ne doit pas être trop complexe pour pouvoir être facilement manipulé.

Il est clair que lorsque le modèle est détaillé, les résultats seront précis mais il y aura quand même une difficulté à l'analysé. En revanche, si le modèle est moins détaillé, les résultats ne seront pas très exacts alors que l'analyse sera plus facile.

Le modèle ne tient compte que des aspects essentiels du fonctionnement du système, et négligeant les facteurs les moins influents. Par conséquent, on peut représenter un système par plusieurs modèles différents qui ne sont pas forcément comparables.

Les principaux avantages de manipuler un modèle plutôt que le système qu'il modélise sont:

- ✓ Un modèle évite la construction d'un système qui n'existe pas.
- ✓ Un modèle évite de faire des expérimentations directes sur un système existant (problèmes de sécurité, ou économiques, ou impossible : système solaire).

1.3.2 Processus de Modélisation

Le processus de construction d'un modèle peut être schématisé par la Figure 1.2 .

On distingue classiquement quatre phases distinctes: La *modélisation* (représentation du comportement d'un système), la *programmation*, l'*expérimentation* et *l'interprétation des résultats* (accompagnée d'actions). Il est à noter que ce processus est généralement itératif [Bank 98].

Le *modèle conceptuel* est une approximation mathématique ou logique du système réel conditionné pour une étude particulière. Il est obtenu dans la phase d'analyse et de modélisation. Le *modèle programmé ou de simulation* est la mise en œuvre du modèle conceptuel sur un calculateur. Il est obtenu dans la phase de programmation et de mise en œuvre. Les renseignements sur le système réel sont obtenus à la suite d'*expérimentations* sur le modèle programmé ou de simulation dans la phase d'expérimentation. L'e*xpérimentation* consiste à construire des théories, ou hypothèses, qui prennent en compte le comportement observé.

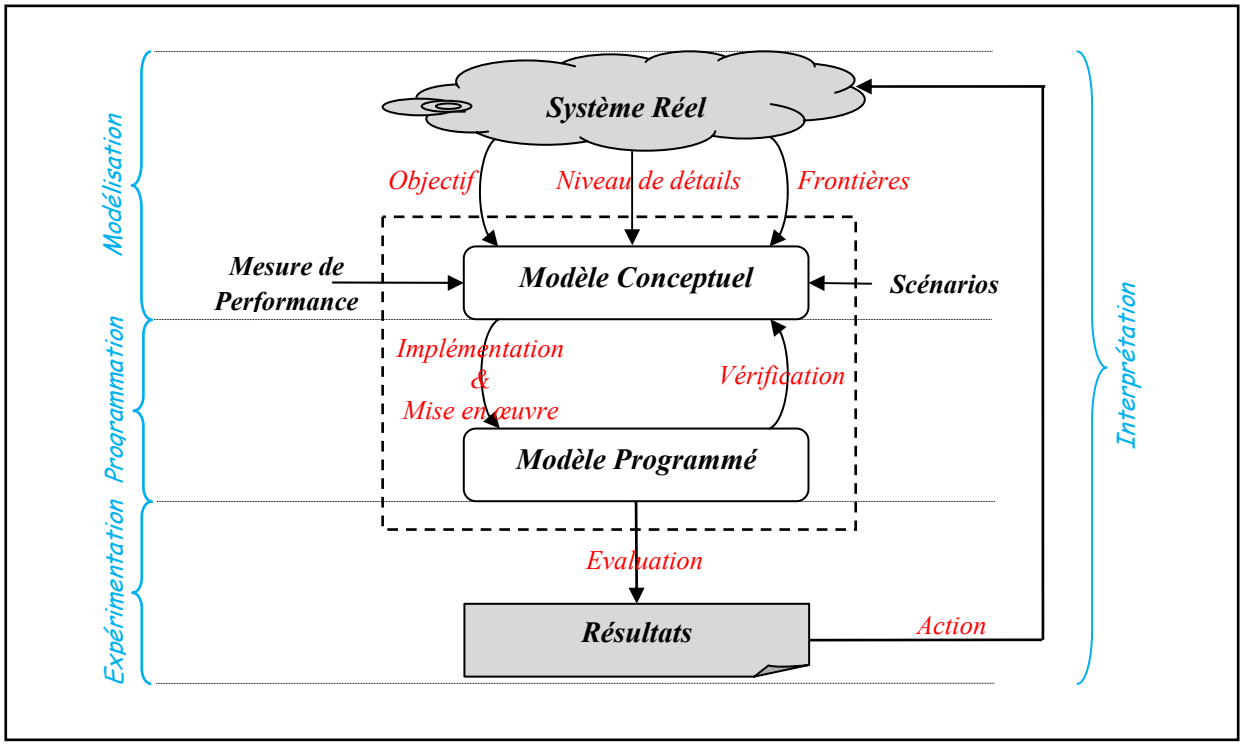


Figure 1.2 Processus de Modélisation.

Le passage du système au modèle conceptuel est une phase essentielle à la simulation. Différents points doivent être abordés :

- Définir l'objectif de la modélisation: Pourquoi modélise-t-on ? Qu'étudie-t-on ? Que veut-on améliorer, ou faire ?
- Définir le niveau de détails à incorporer dans le modèle (Abstraction: choisir les éléments du système pertinents pour l'étude).
- Définir les éléments du système
- Définir les interactions entre ces éléments (hiérarchie).
- Définir les frontières du modèle et donc des variables internes, d'entrées et de sorties nécessaires.
- Définir la dynamique du système (comportement du système au cours du temps).

1.3.3 Classification des Modèles (et de Systèmes)

En simulation, il existe deux grandes classes de modèles qui distinguent les modèles *Physiques* des modèles *symboliques* [Law 91].

1.3.3.1 Les Modèles Physiques

Sont ceux dans lesquels le système réel est représenté par une réplique ou une maquette, à une échelle différente et éventuellement à l'aide de matériaux différents. Ils sont utilisés à des fins

d'entraînement : simulateurs de vol, de conduite, maquettes de véhicules pour des essais aérodynamiques en soufflerie, ...

1.3.3.2 Les Modèles Symboliques ou Analytiques

Sont des modèles abstraits définis par des relations mathématiques, logiques ou symboliques qui sont manipulées et changées pour voir comment le modèle du système réel réagit. Il est en général exécuté sur un calculateur. C'est ce type de modèle qui sera utilisé dans la suite de ce cours. Dans les modèles symboliques, un système est décrit par un ensemble de variables appelés *Variables d'État*, ces derniers permettent de décrire l'évolution du système à chaque instant.

Selon la prise en compte du temps dans l'évolution des modèles, on distingue :

- ***Modèles Statiques***

  Les modèles statiques représentent les systèmes qui ne changent pas avec le temps, c'est-à-dire les variables d'état ne sont pas définies en fonction du temps. Par exemple : modèle comptable permettant de calculer un profit en fin d'exercice à l'aide d'un tableur, les systèmes décrits par la programmation linéaire (affectation, plus court chemin,…).

- ***Modèles Dynamiques***

  Dans lesquels le temps est un facteur essentiel de l'évolution de l'état d'un système, c'est-à-dire ses variables d'état changent avec le temps. Par exemple : réacteur chimique qui régi par des équations différentielles, système de manutention dans une usine, les mouvements des planètes du système solaire, etc. L'évolution de l'état d'un système a toujours un caractère continu, mais selon l'objectif de la modélisation et la logique de changement d'état, le modèle développé est de nature différente :

  - ***Modèle Continu***

    Lorsque le changement d'état du système est caractérisé par des paramètres dont la valeur évolue de façon continue dans le temps (voir la Figure 1.3). L'évolution de ces variables peut être décrite par un système d'équations algébriques ou différentielles. Par exemple, la pression atmosphérique, la position d'une planète dans son orbite, etc.

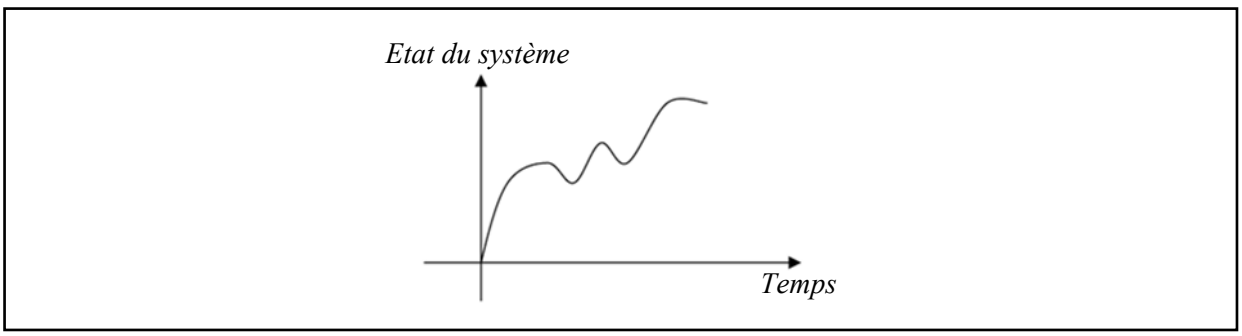


Figure 1.3 Evolution Continu.

- ***Modèle Discret***

  Dans lequel l'état ne change qu'à certaines dates. Autrement dit, les variables d'état sont discrètes (discontinues). Dans ce type de modèles, les systèmes évoluent de manière discrète dans le temps, c'est-à-dire qu'il existe une suite strictement croissante de nombres réels positifs représentant les instants : $t_{(i)}$, $i=0,1,2,...$, telle que entre l'instant $t_{(i)}$ et l'instant $t_{(i+1)}$ l'état d'un tel système n'évolue pas. Par exemple, la suite croissante des dates de début et de fin d'un ensemble d'opérations réalisées pour usiner une pièce dans un système de production, une file d'attente devant un guichet, les variables d'états comme la longueur de cette file d'attente changent de façon discrète, etc. A l'intérieur des modèles Discret, on trouve :

  1. ***Les modèles discrétisés :*** permettent l'observation de l'état à des instants réguliers. Ces modèles correspondent à une représentation discrète de systèmes continus.
  2. ***Les modèles à événements discrets :*** dans ces modèles, l'état du système n'est modifié que lors de l'occurrence de certains événements (voir la Figure 1.4).

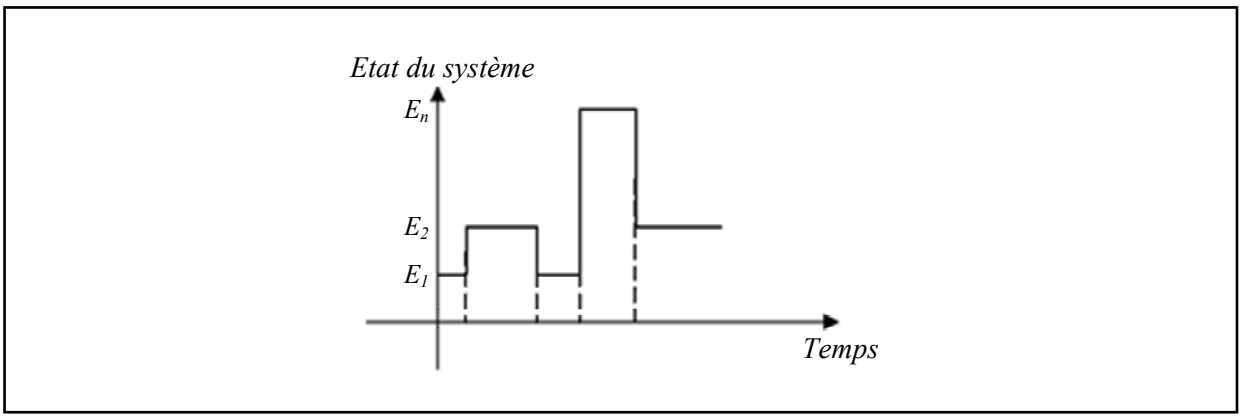


Figure 1.4 Evolution Discret.

- ***Modèle Combiné***

  Des modèles qui contiennent à la fois des composantes discrètes et d'autres continues.

Une autre distinction concerne la prise en compte de variations aléatoires dans les modèles dynamiques. En réalité, tous les systèmes dynamiques sont déterministes. C'est-à-dire, on peut prévoir leur évolution avec certitude. Cependant dans la pratique, on ne dispose pas de toutes les informations nécessaires ou ils sont trop complexes. On fait appelle alors à la notion d'aléatoire. Il s'agit de remplacer les informations manquantes ou trop complexes à exploiter par une caractérisation probabiliste dans leurs modèles.

Les modèles dynamiques peuvent être soit déterministes ou bien stochastiques (non déterministes, aléatoire ou probabilistes) :

- ***Modèles Déterministes***

  Sont des modèles où l'état futur est connu à l'avance ou prévisible, c'est-à-dire qu'il est facile de déterminer avec certitude l'état prochain à partir de l'état actuel (un seul état futur correspondant). Autrement dit, aucune variable de ce modèle n'est aléatoire.

- ***Modèles Stochastique***

  Dans ces modèles, l'état prochain n'est pas connu avec certitude, il est donc imprévisible. Le mécanisme de changement d'état est régi par les lois de probabilité. Par conséquent, plusieurs états futurs sont candidats.

La Figure 1.5 synthétise cette classification :

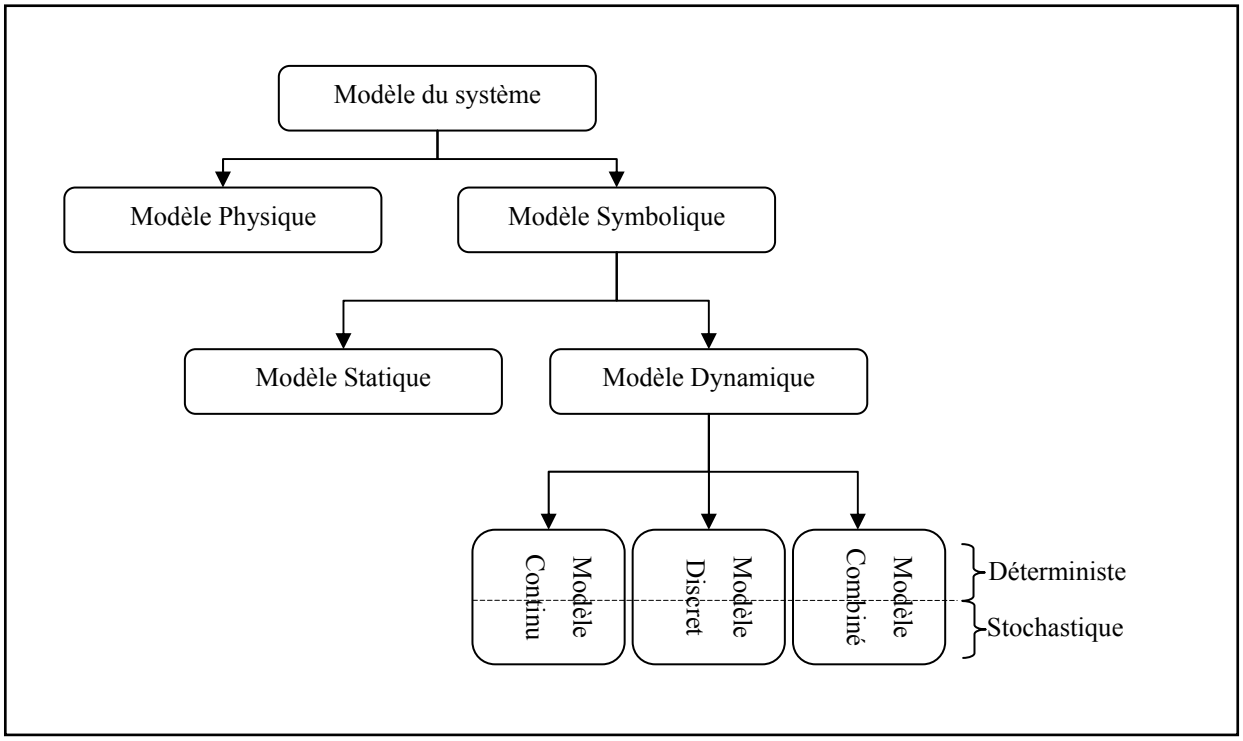


Figure 1.5 Classification des Modèles

Nous nous intéresserons par la suite à la modélisation et la simulation des systèmes dynamiques déterministes ou stochastiques.

1.3.4 Outils de Modélisation

Pour décrire un système d'une manière compréhensible, il est nécessaire d'utiliser un langage de modélisation. Ces langages sont basés sur des formalismes permettant d'utiliser des règles et des contraintes facilitant le passage du système réel au modèle. Il faut note que certains outils de modélisation ne se basent pas sur des formalismes pour l'écriture des modèles, mais autorise une représentation algorithmique du comportement des systèmes avec le degré de finesse souhaité. L'absence de formalisme rend la tâche de modélisation difficile et complexe.

Parmi les outils de modélisation utilisés, on distingue :

- ***Machines à états finis :*** un outil abstrait de conceptualisation et de modélisation formelle basé sur la théorie des automates. Il est utilisé dans divers domaines où l'évolution du système est séquentielle.
- ***Les StateChart* :** Élargissement de l'outil classique Machine à états avec les notions de Hiérarchie, Parallélisme et diffusion.
- **Réseaux de Petri :** un outil graphique de modélisation formelle exprimant la concurrence, la synchronisation et le parallélisme des activités du système.
- ***Processus Stochastique et Chaine de Markov* :** Outils mathématiques permettant de décrire des systèmes dynamiques évoluant dans un environnement aléatoire.
- ***Les files d'attente* :** la théorie des files permet de modéliser de nombreuses situations dans lesquelles des clients arrivent à intervalles aléatoires dans un système comportant plusieurs serveurs auxquels ils vont adresser une demande ou une requête.
- ***Réseaux de Petri Stochastique* :** Augmentation de l'outil de Réseaux de Petri par l'aspect temporal et aléatoire.
- **…**

1.3.5 Evaluation de Modèles

Comme indiqué précédemment, la modélisation vise, entre autres, à représenter afin d'analyser et d'évaluer les performances des systèmes. L'évaluation de modèles est l'étude des propriétés du modèle pour en déduire des propriétés associées au système qu'il le représente. Il existe deux types d'évaluation pour un système, l'évaluation *Qualitative* et l'évaluation *Quantitative* [Aissani 07].

1.3.5.1 Evaluation Qualitative

L'évaluation *qualitative* s'intéresse à la vérification d'une manière précise des propriétés structurelles et comportementales du système telles que:

- → Absence de blocage du modèle.
- → Certaines propriétés de vivacité d'atteignabilité pour des états donnés du système.
- → Bornitude des états du modèle dont l'absence révèle à une possibilité d'explosion du nombre de certaines entités dans le système réel.
- → Gestion de la concurrence.
- → Existence d'une solution.

1.3.5.2 Evaluation Quantitative

L'évaluation *quantitative* consiste à calculer les valeurs des critères de performances (c'est-à-dire des indices de performances) du système tels que :

- → Débit d'entrée et de sortie.

- → Le taux d'occupation, … .
- → Temps de séjour d'un client dans le système.
- → Nombre moyen de clients dans le système.
- → Le temps moyen d'attente.
- → Temps de réponse.

Pour calculer ces indices, on distingue deux grandes familles de méthodes d'analyse quantitative:

- ***Les méthodes analytiques:*** permettent le calcul des indices de performance de façon mathématique en résolvant des équations qui modélisent le fonctionnement du système. Leur avantage réside dans leur faible coût d'exploitation et qu'elles fournissent rapidement des résultats. Cependant, leur hypothèses sont très restrictives, elles nécessitent généralement de simplifier le fonctionnement du système pour leur mise en œuvre.
- ***La simulation:*** La simulation consiste à mesurer le comportement d'un modèle du système en simulant son évolution, par exemple par une génération aléatoire d'événements répartis dans le temps. La simulation possède un domaine d'application quasi-illimité, mais elle reste très coûteuse (en termes de temps humain et machine). Cette technique permet donc de simuler le fonctionnement de système existant ou non en étudiant une réalisation particulière de celui-ci, ainsi elle conduit à réaliser des expériences sur le modèle pour un objectif donné (études de fonctionnements transitoires, tests de différentes stratégies,…).

L'analyse qualitative utilise les langages formels tels que les Machines d'états finis, les Statecharts et les Réseaux de Petri. L'analyse quantitative fait appel aux chaines de Markov, aux Réseaux de files d'attente et aux Réseaux de Petri stochastique.

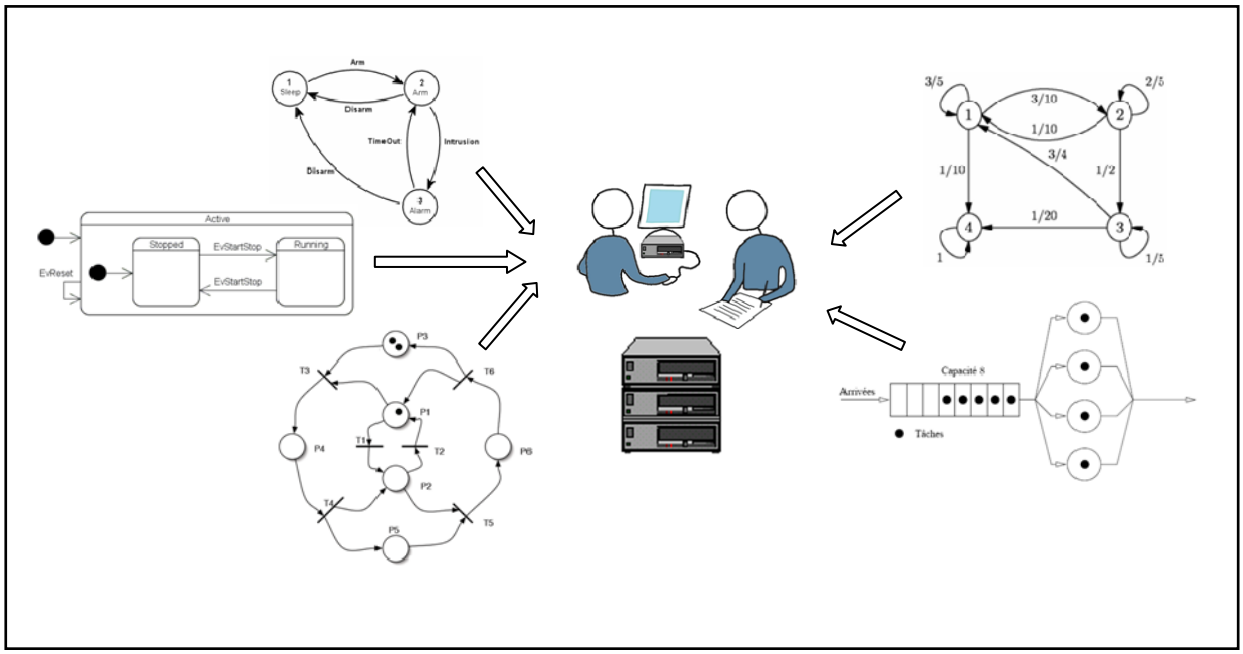


Figure 1.6 Analyse Qualitative et Analyse Quantitative.

1.4 Simulation

La simulation est une méthode qui consiste à étudier le fonctionnement d'un système en imitant son fonctionnement au cours du temps en manipulant un modèle. Cette méthode est basée sur la génération d'un historique artificiel des changements d'état du système (une trajectoire d'état) et l'observation et l'analyse de cet historique pour faire des déductions sur ses caractéristiques de fonctionnement. La simulation est devenue incontournable et pratique permettant de modéliser aussi bien des systèmes à concevoir que des systèmes existants déjà pour analyser et valider des choix des solutions. Elle peut être utilisée pour:

- ✓ Décrire et analyser la dynamique d'un système.
- ✓ Répondre aux questions de type *"**What If** ? "* (***Qu'est-ce qui se passe si*** …) sur le système réel.
- ✓ Aider à la conception d'un système réel.
- ✓ …

1.4.1 Définition (Simulation)

" La simulation est l'étude du comportement dynamique d'un système, grâce à un modèle que l'on fait évoluer dans le temps en fonction de règles bien définies, à des fins de prédiction." [Pritsker 87].

À partir de cette définition, on peut dire que le terme de simulation pourrait être caractérisé par les mots clés suivants :

- Basée sur un élément fondamental qui est le *modèle*.
- Le modèle est *manipulé* sur ordinateur, cette manipulation fournissant des solutions.
- Les solutions trouvées sont celles du modèle et non du système modélisé.
- Son but est de choisir parmi les solutions celle qui semble être la meilleure.

La simulation est une expérimentation indirecte (voir la Figue 1.7) car elle est portée sur le modèle et non pas sur le système. Elle ne résout pas le problème posé en trouvant la bonne solution, mais elle aide seulement à prendre parmi plusieurs solutions la meilleure possible.

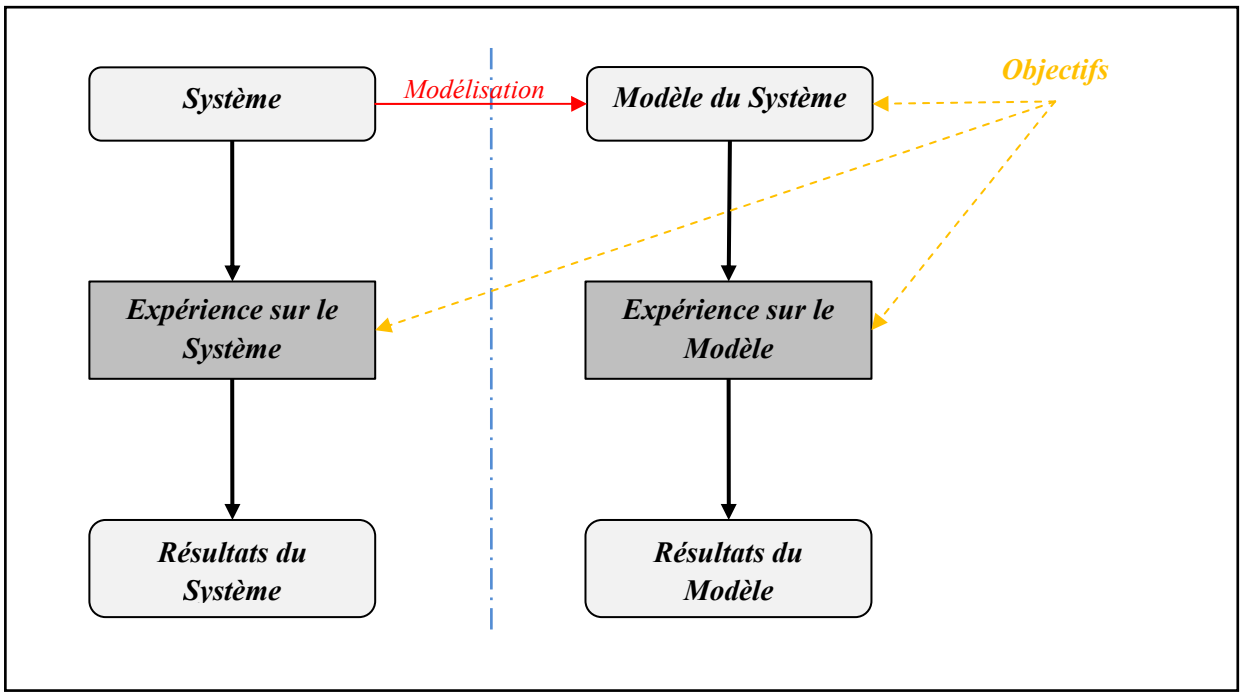


Figure 1.7 Expérimentation Directe et Indirecte.

1.4.2 Les Langages de Simulation

La programmation d'un modèle pour la simulation peut se faire dans un langage quelconque, on distingue :

- les ***langages universels impératifs*** (ou procéduraux) ayant un fort pouvoir d'expression. On trouve des modèles écrits en C, en lisp, en shell ou en PERL. La mise en œuvre peut être longue, par contre on dispose d'une grande flexibilité.
- les ***langages déclaratifs*** ou non procéduraux, on utilise alors un langage du même type pour décrire l'application. Le plus connu de ces langages est PROLOG. On ne décrit pas les algorithmes à exécuter pour produire les résultats mais les objets et les relations entre eux.
- Utilisation des ***Simulateurs*** dédiés à un type de systèmes et un type de problème. il suffit de paramétrer le modèle pour l'adapter au cas étudié. Cette alternative présente l'avantage de ne pas programmer (seules des données sont à entrer), par contre il n'est pas toujours simple de trouver le simulateur dédié adapté au système et au problème concernés.
- les ***langages spécifiques (les langages de simulation)*** qui proposent des primitives de modélisation particulièrement adaptées à un type de domaine. Ces primitives permettent à l'utilisateur de développer d'un modèle de simulation réalisé au travers d'un programme. Ce type de logiciel offre une grande flexibilité mais avec des coûts de développement parfois importants. par exemple, dans le domaine du temps réel on peut citer les langages LUSTRE, SIGNAL ou ESTEREL qui sont à la fois langages de spécification, prototypage et réalisation. Dans le domaine de modélisation des ressources et fonction de transport, SIMAN est un des principaux logiciels standards.

1.4.3 Quand Simuler

La simulation est souvent caractérisée dans la littérature comme la méthode de dernier ressort. Ceci veut dire que si on a la possibilité de résoudre le problème posé avec des méthodes analytiques, il serait préférable de les utiliser car elles conduisent à des solutions *optimales*. En effet, pour un problème donné et des objectifs fixés, on peut avoir le choix entre différentes approches pour le résoudre parmi lesquelles il y a la simulation comme le montre la Figure 1.8.

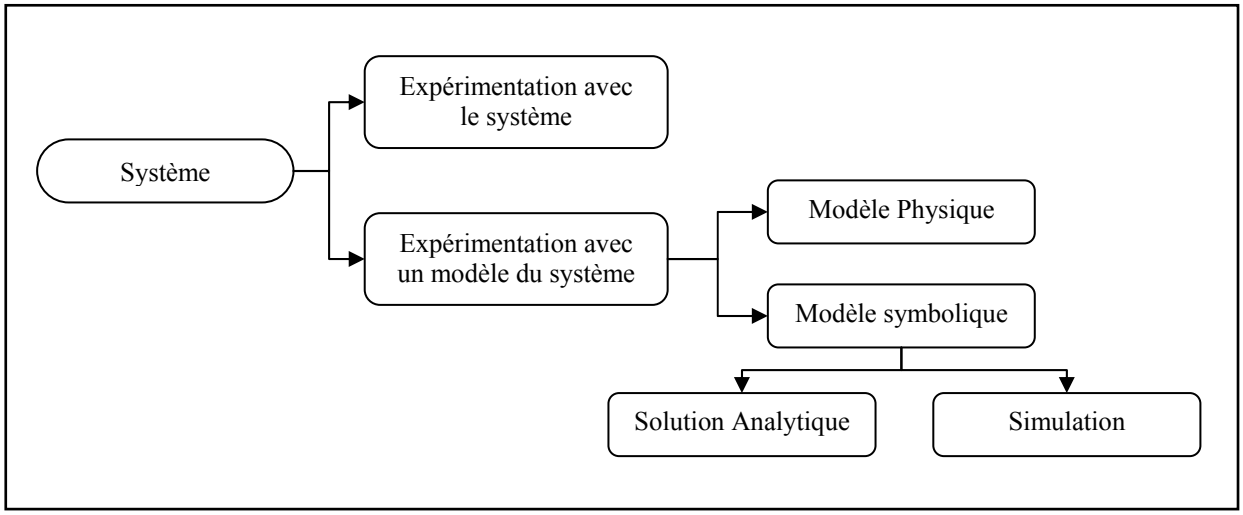


Figure 1.8 Approches pour Etudier un Système.

Le tableau ci-dessous présente la comparaison entre la simulation et les méthodes analytiques :

Simulation	***Méthodes Analytiques***
Résultats approximatif	Résultats exacte si possible
Applique aux systèmes complexes	Pas toujours
Gourmande en temps de calcul	Peut couteuse en temps de calcul

Les résultats avec les formules analytiques ne sont pas toujours existants, mais s'ils existent ça sera exact, contrairement à la simulation qui donne des résultats approximative.

1.4.4 Limites (Obstacles)

Malgré ses avantages, la simulation présente des inconvénients :

- ***Coûteuse en termes de temps*** : elle nécessite beaucoup de dépenses en temps pour la conception du modèles de simulation, la programmation, l'expérimentation et la validation. En outre, elle demande un certain niveau d'expertise; la qualité des résultats est liée à la qualité de la modélisation et de la programmation.

- ***Les résultats de simulation sont souvent complexes à interpréter*** : les modèles utilisés pour étudier les systèmes à grande échelle ont tendance à être très compliqués, et l'analyse des résultats fournis par la simulation des ces modèles peut être une tâche difficile.
- ***Non exhaustive*** : en effet, la simulation n'est pas une technique d'optimisation de performance d'un système au sens propre. Elle peut seulement évaluer les performances d'une solution conçue et imaginée par l'utilisateur. Cette technique est entièrement itérative qui ne propose pas de solution finale optimale, mais qui permet seulement à l'utilisateur d'envisager des choix possibles. En tout état de cause, c'est l'utilisateur qui devra décider de ce qui répond le mieux aux problèmes posés. la simulation ne peut résoudre des problèmes mais seulement fournir des indications à partir desquelles des solutions approximatives peuvent être déduites.

1.5 Résumé

Dans ce chapitre, nous avons introduit la terminologie de base que nous allons utiliser tout le long de ce cours. Nous avons montré que la simulation est un outil de résolution de problèmes, mais elle peut être inutile si elle n'est pas utilisée intelligemment. En effet, parce que la simulation parait facile à utiliser, l'utilisateur est tenté de ne pas consentir l'effort de comprendre ce qu'il fait et en conséquence elle résulte en des coûts très lourds.

Chapitre 02 :

Machines d'Etats Finis

2.1 Introduction

Les Machines d'états finis sont un outil de modélisation formelle basé sur la théorie des automates [Alur 94]. Le terme Machines d'états finis est surtout utilisé par ceux qui s'intéressent à la modélisation de systèmes. Le terme automate est plus général. Les automates constituent des techniques de base que l'informaticien se doit de connaître. Ils sont utilisés essentiellement dans la théorie des langages et dans la modélisation et vérification de systèmes complexes où il existe un nombre fini de possibilités (c'est-à-dire un nombre fini d'états).

La notion d'automates est utilisée dans différentes méthodes de conception (les diagrammes dynamiques d'UML "Unified Modeling Language", SDL "Specification and Description Language" et autres), avec des notations graphiques souvent différentes d'une méthode à l'autre.

La machine d'états finis est une machine séquentielle algorithmique ayant un état courant (elle est supposée se trouver dans un *état* précis à un moment précis) et d'autres états par lesquels elle est déjà passée ou par lesquels elle passera (éventuellement) dans le futur. Un *état* dans la machine est caractérisé par la ou les valeurs d'une ou plusieurs variables d'état qui correspond à une situation particulière du système. L'apparition d'un *évènement* (stimuli, sollicitation, …) est susceptible de faire passer la machine d'un état à un autre. Le changement d'état s'appelle *transition*. De manière simplifiée, un système se trouve dans un état initial et change d'état en fonction de l'arrivée des événements qui lui parviennent de son environnement.

2.2 Définition

Une Machine d'états finis ou Un automate est un quadruplet $A = (S, s_0, F, E, T)$,

Où :
- S = ensemble d'états.
- s_0 = l'état initial de l'automate ($s_0 \in S$).
- F = ensemble des états finaux de l'automate ($F \subseteq S$).
- E = ensemble fini des étiquettes des transitions.
- T = ensemble des transitions ($T \subseteq S \times E \times S$).

2.3 Représentation

Les machines d'états finis sont généralement représentées de manière graphique (c'est d'ailleurs l'un des atouts des automates). Cette représentation, comme illustre la Figure 2.1, permet de visualiser les transitions entre les états selon l'événement de simulation reçu par l'environnement :

→ Les *états* sont représentés par des cercles. Le *nom* rattaché à chaque état est à l'intérieur du cercle qui le représente.

→ Les *transitions* entre les états sont représentées par des arcs dirigés reliant les cercles.

→ Les *étiquettes* associées aux transitions peuvent être composées de trois types d'éléments selon la syntaxe suivante (ces éléments ne sont ni nécessaires ni exclusifs) :

Evénement [Garde]/ Liste d'actions

1. Un *événement* déclencheur, une fois reçu, permet le franchissement de la transition.
2. Une condition de *Garde*, une expression booléenne (exprimée généralement sur les variables de l'automate) qui doit être vraie lorsque l'évènement arrive pour que la transition soit déclenchée. C'est une condition nécessaire, mais pas suffisante.
3. Des *actions* effectuées par l'automate avant de changer d'état. Par exemple : affectations de variables, envois d'événements, ...

→ On attribue généralement à la machine un état de départ lui permettant de débuter son fonctionnement à partir d'un point fixe, et un ou plusieurs états finaux lui permettant de terminer son fonctionnement s'il existe. L'état initial est distingué par une petite flèche, et les états finaux sont représentés par un double cercle centrique.

Remarques :

- Si l'étiquette associée à une transition est vide, cela signifie que la transition se fait de manière aléatoire ou de manière implicite connue par celui qui a fait la modélisation.
- Si l'étiquette ne contient pas d'événement, cela signifie que le franchissement est aléatoire et est lié uniquement à la garde si elle existe. Pour des raisons de déterminisme, il faut éviter d'utiliser des transitions sans événements.

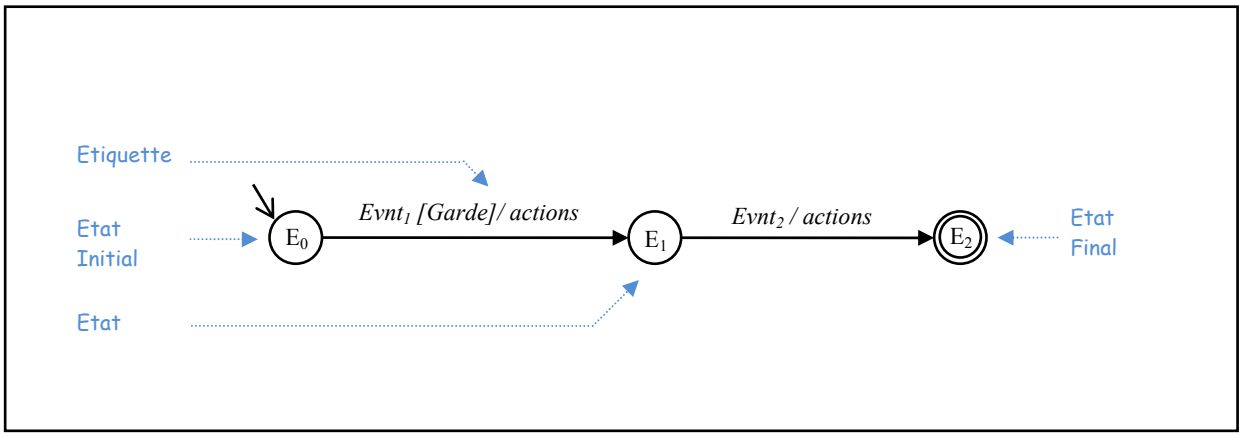


Figure 2.1 Représentation Graphique des Machines d'Etats Finis.

Un automate d'états finis peut aussi être représenté sous forme d'une table de transition, qui indique pour chaque état l'état à atteindre pour chaque nouvel évènement (voir la Figure 2.2) :

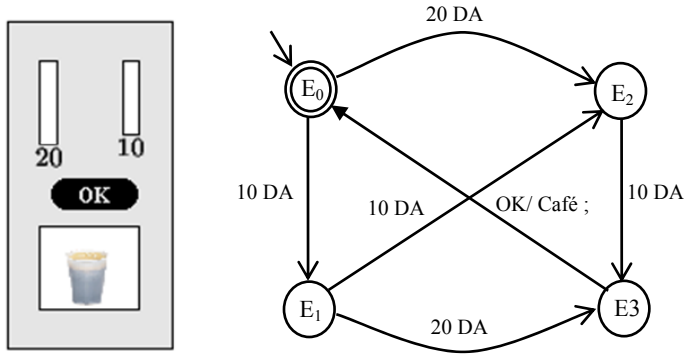
- Chaque colonne correspond à un évènement.
- Chaque ligne correspond à un état de l'automate (l'état initial est précédé d'une flèche "→"; l'état final d'une étoile "*").
- La valeur de la table à l'intersection de la ligne et de la colonne indique l'état à atteindre.

Etats \ *Transitions*	$evnt_1$	$evnt_2$	…	$evnt_n$
$Etat_1$				
$Etat_2$				
⋮				
$Etat_m$				

Figure 2.2 Table de Transition.

2.4 Exemple de Modélisation de Distributeur de Café

La machine de la Figure 2.3 comporte deux chargeurs de pièces pour les pièces de 10 et 20 Dinars, et un bouton *OK* pour obtenir un café. Au début, le bouton *OK* est bloqué, les deux chargeurs de pièces sont ouverts. Le café coute 30 Dinars. Si on a mis suffisamment de pièces (la machine accepte une seule pièce de 10 DA et une autre de 20 DA), les chargeurs de pièces se ferment, le bouton *OK* est libéré, on ne peut plus mettre de pièces, et on peut enfoncer le bouton *OK*. Alors, un gobelet tombe dans l'espace vide au bas de la machine, et le café se verse dans le gobelet.



Etats \ *Transitions*	10 DA	20 DA	OK
→*E_0	E_1	E_2	-
E_1	E_2	E_3	-
E_2	E_3	-	-
E_3	-	-	E_0

Figure 2.3 : Modélisation de Distributeur de Café.

L'automate correspondant comporte 4 états. Dans les états E_0, E_1, E_2, les chargeurs de pièces sont ouverts, le bouton *OK* est bloqué. Dans l'état E_3, les chargeurs de pièces sont fermés, le bouton OK est débloqué. Les états E_0, E_1, E_2, E_3 enregistrent le crédit courant en multiples de 10 Dinars. On passe d'un état à un autre soit en insérant une pièce dans une des deux chargeurs de pièces, soit en appuyant sur le bouton *OK*. Au début, le distributeur est dans l'état E_0. Ceci est résumé par le schéma et la table de transition de la Figure 2.3.

2.5 Propriétés des Machine d'Etats Finis

2.5.1 Automate fini & Automate infini

Lorsque l'ensemble des états, S, est fini on parle *d'automate d'états finis* et lorsqu'il est infini on parle d'*automate d'états infinis* (ou d'*automate à états* tout simplement) [Arnold 92].

Par exemple, un automate qui modélise les états (valeurs) d'une variable réelle est infini (puisque le nombre de valeurs est infini). Par contre, l'automate de la Figure 2.4 qui modélise l'état d'une lampe est fini, si on ne s'intéresse qu'aux états *Lampe Allumée* et *Lampe Eteinte*.

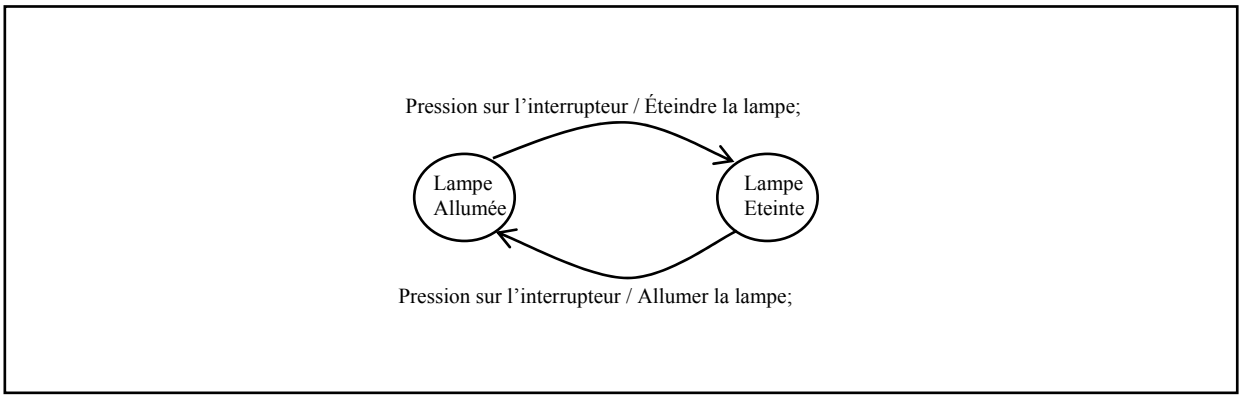


Figure 2.4: Modélisation de l'Etat d'une Lampe, un Automate Fini.

Dans ce qui suit, nous nous intéressons aux automates d'états finis seulement.

2.5.2 Automate Déterministe

Un automate est dit *déterministe* si pour tout couple (état, étiquette), le choix de la transition est unique. Il est *indéterministe* s'il existe au moins un état qui a deux transitions étiquetées de la même manière mais qui mènent vers deux états différents [Arnold 92] .

L'automate de la Figure 2.4 est déterministe. Par contre, celui de la Figure 2.5 n'est pas déterministe, car à la réception d'un message, on peut soit traiter le message soit l'ignorer. L'indéterminisme peut être volontairement choisi pour modéliser des aspects indéterministes comme la perte de messages dans un réseau ou la défaillance de machines dans un système de production.

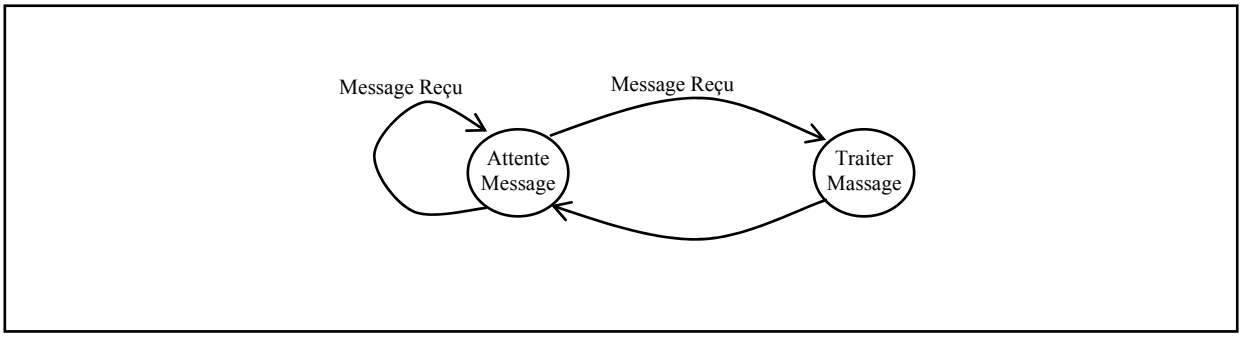


Figure 2.5: Modélisation d'Envoie de Message, un Automate Indéterministe.

2.6 Structure de Kripke

Etant donné un ensemble π de propositions atomiques et un ensemble E d'étiquettes, une structure de Kripke K est un système de transitions étiquetées où les états sont décorés par des ensembles des propositions de π, $K = (S, s_0, E, T, L)$,

Où :
- S = ensemble d'états.
- s_0 = l'état initial de l'automate ($s_0 \in S$).
- F = ensemble des états finaux de l'automate ($F \subseteq S$).
- E = ensemble fini des étiquettes des transitions.
- T = ensemble des transitions ($T \subseteq S \times E \times S$).
- L= une application qui associe à chaque état l'ensemble des propriétés de π vérifiées dans cet état.

C'est généralement pour pouvoir prouver des propriétés sur un système que l'on rajoute le cinquième élément l'automate, celui qui associe à tout état de S, l'ensemble des propriétés élémentaires vérifiées dans cet état. Cet élément n'est pas nécessaire pour utiliser les automates. Si toutes (ou certaines) propriétés liées aux états sont connues à l'avance, il est préférable de les noter sur l'automate, cela simplifiera la preuve de certaines propriétés sur l'automate.

2.7 Sémantique & Exécution

2.7.1 Chemin d'Exécution ou Trace

Un *chemin* d'un automate A est une suite σ, finie ou infinie, de transitions (s_i, e_i, s_i') de A qui s'enchaînent. On note souvent une telle chaîne sous la forme: $S_1 \xrightarrow{e1} S_2 \xrightarrow{e2} S_3 \xrightarrow{e3} S_4 \xrightarrow{e4} \dots$. La longueur d'un chemin, $|\sigma|$, désigne le nombre de transitions qui le composent. *Chemin* et *trace* sont des termes interchangeables [Hopcroft 01].

Par exemple, $E_0 \xrightarrow{10\,DA} E_1 \xrightarrow{10\,DA} E_2 \xrightarrow{10\,DA} E_3 \xrightarrow{OK} E_0$ est un chemin de l'automate de la Figure 2.3.

2.7.2 Exécution

Une *exécution partielle* d'un automate est un chemin partant de l'état initial de cet automate. Une *exécution complète* est une exécution que l'on ne peut plus prolonger (car le système a atteint un état de blocage ou bien l'exécution est infinie) [Wolper 91].

2.7.3 Arbre d'Exécution

Un *arbre* d'exécution est l'ensemble des exécutions possibles décrivant la dynamique d'un système. L'arbre d'exécution peut être infini [Wolper 91].

2.7.4 Atteignabilité

Un état est dit *atteignable* (ou *accessible*) s'il apparaît au moins une fois dans l'arbre d'exécution de l'automate [Wolper 91].

La notion d'atteignabilité est importante, car elle est souvent utilisée pour démonter des propriétés du système. En effet, le passage par un état (donc cet état est atteignable) peut correspondre à une propriété particulière du système ou bien ce passage est nécessaire pour que la propriété soit vérifiée. Par exemple, si dans un automate correspondant à la commande d'un moteur d'un véhicule, il y a un état *moteur_Eteint*, démonter la propriété « Le moteur peut être éteint» revient à démontrer que l'état *moteur_Eteint* est atteignable .

2.8 Raisonnement sur les Machine d'Etats Finis

Les propriétés générales attendues concernant le comportement d'un système sont souvent regroupées en deux grandes classes :

- ☑ **Propriétés de sûreté** (*safety*): le fonctionnement d'un système ne doit pas conduire à des situations catastrophiques ou dangereuses. Le non-blocage est un exemple de propriétés de sûreté.
- ☑ **Propriétés de vivacité** (*liveness*) : le fonctionnement de l'automate doit éventuellement conduire à bonnes situations. Par exemple, la terminaison d'un programme est souvent considérée comme propriété de vivacité.

les propriétés les plus élémentaires (qui sont liées aux automates) sont souvent vérifiées pour démontrer des propriétés plus globales. Il s'agit notamment des propriétés suivantes [Arnold 92]:

- ***Accessibilité*** : un état est accessible s'il existe un chemin l'atteignant depuis l'état initial.
- ***Réinitialisabilité*** : un automate est réinitialisable s'il existe un chemin depuis chaque état vers l'état initial.
- Manque de ***blocage*** : une situation de blocage (*deadlock*) indique qu'aucune transition n'est plus possible à un instant donné.

- Manque de ***famine*** : la famine est une situation où l'automate boucle sur une partie, ce qui empêche tout accès à d'autres parties de l'automate.
- ***Equité*** *(fairness)* : l'équité indique que si deux chemins non-déterministes sont possibles à partir d'un état, ce n'est pas toujours le même chemin qui est emprunté.

2.9 Exercices

Exercice N° 1

Spécifier à l'aide d'une machines d'états finis le comportement d'un système de communication téléphonique : *Décrocher*, *Composer le numéro*, *Répondre à un appel*, *Parler*, *Raccrocher*, ... ?

Exercice N° 2

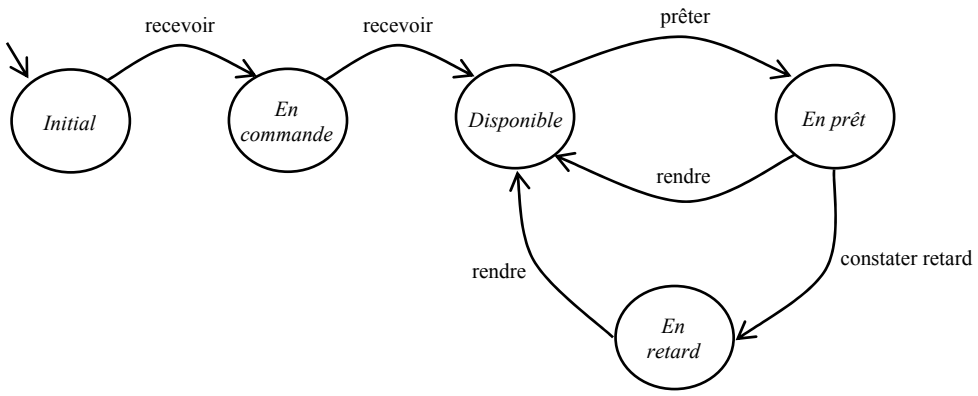
Soit une machine d'états finis dont la relation de transition est donnée par la table suivante :

	a	b
$\rightarrow E_0$	E_1	E_4
E_1	E_4	E_2
$*E_2$	E_3	E_4
E_3	E_4	E_2
E_4	E_4	E_4

– Représenter cette machine sous forme de diagramme ?

Exercice N° 3

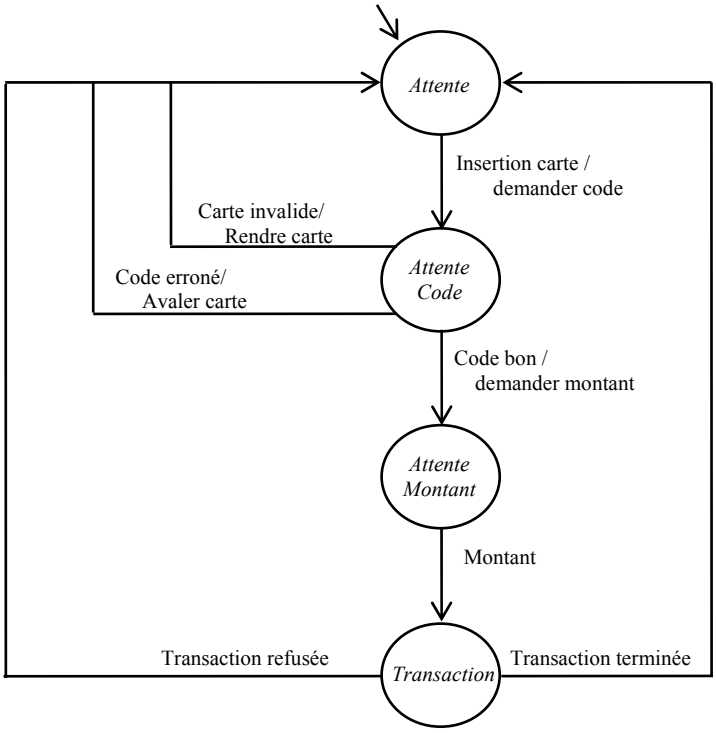
La machine d'états finis ci-dessous représente le cycle de vie d'un livre :



1. Donner la table de transition de cette machine ?
2. Etant donné la séquence d'évènements (commander, recevoir, prêter, rendre, prêter), trouver l'état atteint par cette séquence d'évènements ?
3. Dessiner l'arbre d'exécution de cette machine ?

Exercice N° 4

Soit la machine d'états finis suivante qui représente le fonctionnement d'un Guichet Automatique de Banque :



1. Donner la table de transition de cette machine ?
2. Dessiner l'arbre d'exécution de cette machine ?

Chapitre 03 :

StateCharts

3.1 Introduction

Les StateCharts sont un formalisme défini au début des années 80 par David Harel [Harel 87] pour la spécification de systèmes complexes. Ils représentent un des premiers formalismes graphiques pour modéliser le comportement des systèmes larges (en termes d'états et de transitions) constitués d'un grand nombre d'entités ou de composants en interaction. Ils décrivent les séquences possibles d'états et d'actions qu'une entité peut traiter au cours de son cycle de vie en réaction à des évènements. La vision globale du système est moins apparente sur ces diagrammes, car la liaison entre les parties du système est assurée implicitement par le mécanisme d'envoie d'évènements.

Un StateChart est un automate d'états finis dont les états peuvent contenir un ou plusieurs automates représentant ainsi un sous-système ou une sous-fonction du système global. Lorsque le système entre dans un tel état, le sous-système ou la sous-fonction du système est démarré. Le mécanisme de structuration des états dans les StateCharts limite l'explosion combinatoire des états et des transitions. L'originalité des StateCharts provient non seulement de la description graphique des changements d'états du système, mais aussi de la représentation graphique de la hiérarchie (plusieurs niveaux d'abstraction), du parallélisme et de la concurrence. Il est à noter que nous présentons dans ce cours qu'un sous ensemble de concepts et notations des StateCharts.

3.2 Définition

Les StateCharts sont une représentation graphique et formelle permettant de factoriser les informations contenues dans un automate d'états finis. Plus précisément, il s'agit d'une extension des automates d'états finis avec trois notions pour décrire des comportements complexes :

- ***La Hiérarchie :*** Une des limitations du modèle de automates d'états est que la complexité des diagrammes augmente de façon dramatique avec le nombre d'états. Les StateCharts permettent d'imbriquer des états les uns dans les autres (*super-états* ou *états composites* qui englobent chacun plusieurs états), ce qui rend le diagramme plus compréhensible et lisible et permet plusieurs niveaux d'abstraction.
- ***L'Orthogonalité ou Parallélisme:*** Le modèle des automates d'états ne dispose d'aucune construction pour représenter la concurrence.
- ***La Diffusion :*** Le mécanisme de communication des événements est la Diffusion (*broadcast*). Lorsqu'un événement survient, l'ensemble des automates actifs d'un StateCharts perçoivent cet événement. Ce mécanisme permet à plusieurs états orthogonaux de communiquer par des événements. Les automates d'un StateCharts peuvent réagir à des événements provenant soit de l'environnement soit d'autres automates du modèle.

Statechart = Automate d'Etats Finis + Hiérarchie + Parallélisme + Diffusion

3.3 Représentation

Un StateChart rassemble et organise les états et les transitions d'une entité ou d'un composant du système pendant sa durée de vie. Bien entendu, le modèle dynamique du système comprend plusieurs StateCharts. Tous les automates d'états finis des StateCharts d'un système s'exécutent concurremment et peuvent donc se communiquer et changer d'état de façon indépendante. Toutes ces notions sont représentées graphiquement ce qui offre une vue synthétique du système. La Figure 3.1 illustre les principaux éléments qui composent un StateChart.

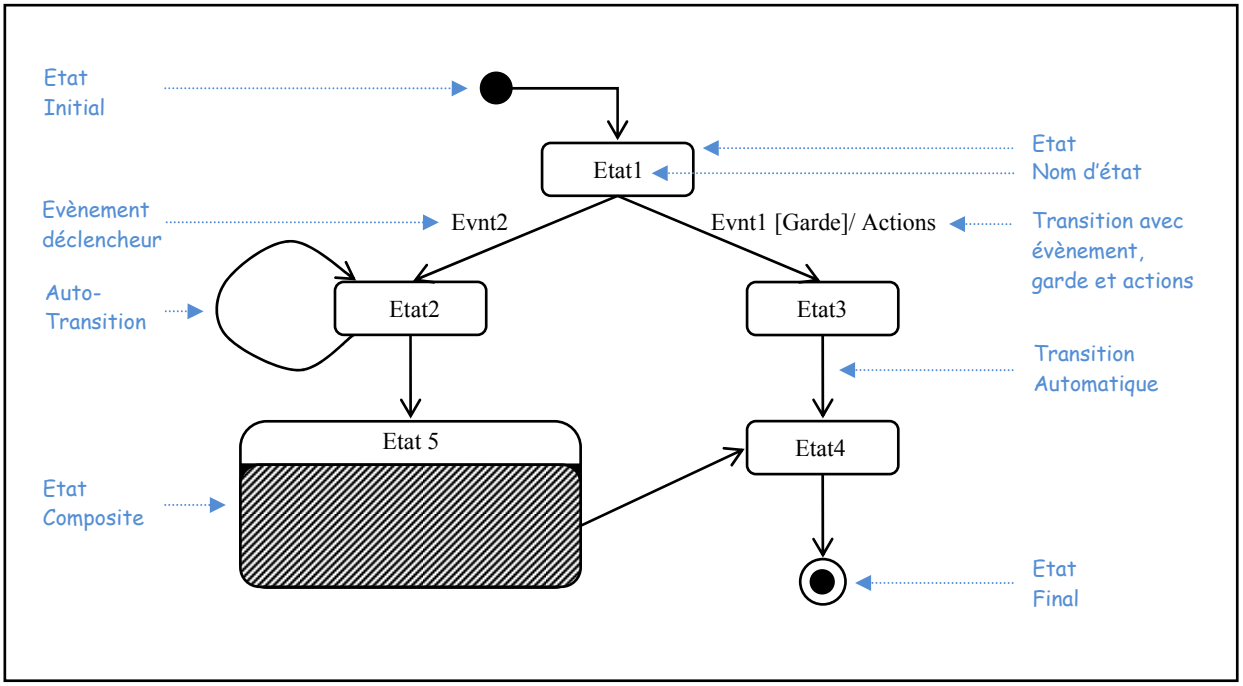


Figure 3.1 : Représentation Graphique des Statechart.

3.3.1 Etat

Un état est une situation stable au cours de la vie d'un système (ou d'un sous-système) qui satisfait certaines conditions. Un système reste dans un état pendant une durée déterminée pendant laquelle il exécute une *activité* ou attend un *évènement*. Deux états sont particuliers : l'*état initial* qui indique l'état de départ de l'automate et l'*état final* qui marque la fin de son exécution. Un état se compose essentiellement de 5 parties dont certaines sont optionnelles (Figure 3.2) :

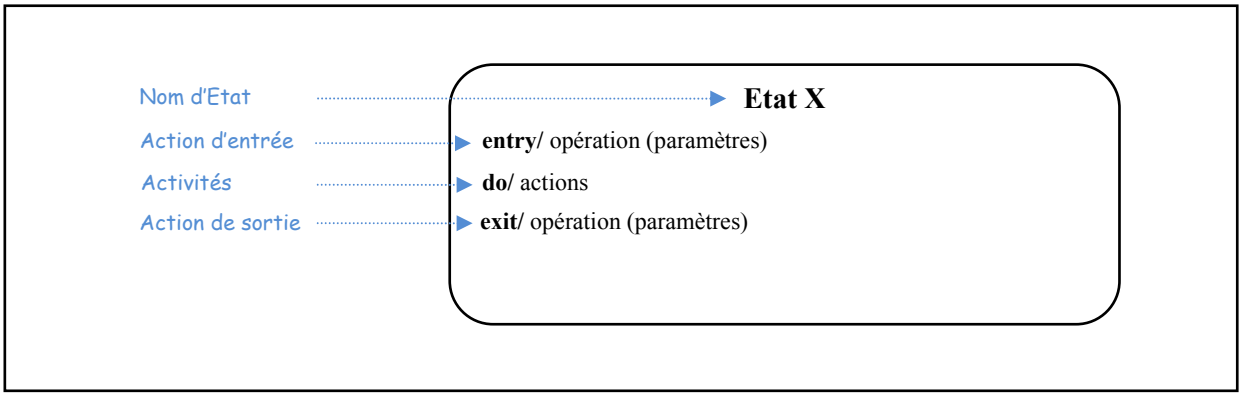


Figure 3.2 : Représentation Graphique d'un Etat.

- Un *nom* qui permet d'identifier clairement l'état.
- Une *action d'entrée*: action exécutée quand le système entre dans l'état. Elle est spécifiée via le mot clé "*entry*".
- Une *action de sortie* : action exécutée quand le système sort de l'état. Elle est spécifiée via le mot clé "*exit*".
- Des *activités* : ce sont des activités effectuées une fois l'action d'entrée est exécutée. Quand les activités sont terminées, le système attend un évènement pour changer d'état. Il est à noter que les activités ont une certaine durée, et elles peuvent être par conséquent interrompues. le mot clé "*do*" est utilisé pour spécifier les activités.
- Des *sous-états* : Un *sous-état* est un état emboîté dans un autre état (dit *état composite*). un *état composite* peut être hiérarchisé en plusieurs sous-états. Un *état composite* a une sous-structure qui permet de masquer à un certain niveau le fonctionnement interne de l'état lorsque celui-ci est complexe (voir dans la section 3.3.3 les détails sur la manière de spécifier des états composites).

3.3.2 Transition

Une transition définit la réponse d'un système à l'arrivée d'un *événement*. Elle indique qu'un système qui se trouve dans un état (un état source) peut transiter vers un autre état (un état cible) et exécuter certaines *Actions*, si un événement déclencheur se produit. Elle indique aussi sous quelle condition de garde le système passe de l'état source à l'état cible.

Une transition entre deux états est représentée par un trait droit fléché de l'état source vers l'état cible et étiqueté par un triplet (*Evénement [Garde]/ Liste d'actions*). Une transition comporte cinq parties qui ne sont pas toutes obligatoires (Figure 3.3):

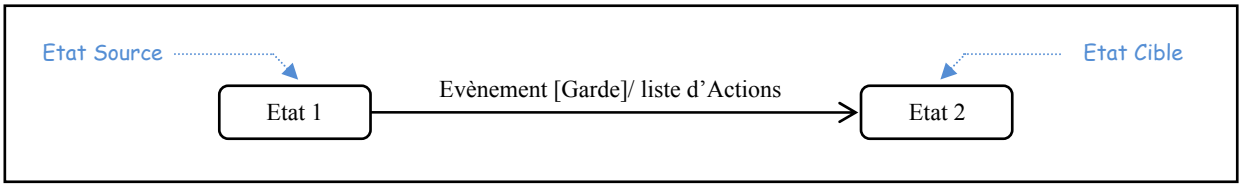


Figure 3.3: Représentation Graphique d'une Transition Entre deux Etats.

- Un *état source* de franchissement de la transition (obligatoire);
- Un *événement déclencheur* (optionnel) : la réception de cet événement déclenche la transition si la condition de garde le permet. L'événement qui détermine le franchissement de la transition est indiqué sous forme de texte. Il est possible d'avoir une transition sans événement qui est franchie de manière implicite (appelée *transition automatique*) quand l'état source a terminé ses *activités*. Le même *événement* peut être le déclencheur de plusieurs transitions quittant un même état. Chaque transition avec le même événement doit avoir une *condition de garde* différente. Pendant l'exécution, une seule transition peut se déclencher. Si deux transitions sont activées en même temps par un même événement, une seule se déclenche et le choix n'est pas prévisible (c'est-à-dire indéterministe).
- Une *garde* (optionnelle) : expression booléenne évaluée après la réception de l'*événement*. Si elle est vraie, la transition est effectuée, sinon le système reste dans le même état et l'événement est perdu. L'expression booléenne est placée entre crochets.
- Une *action* (optionnelle) : calcul exécuté par le système avant de passe à l'état cible. Il s'agit généralement d'une opération qui peut être: envois d'événements, modification de variables de l'automate, … . Il est à noter que les actions sont considérées comme instantanées (c'est-`a-dire dont le temps d'exécution est négligeable) et atomique (c'est-à-dire non interruptible)
- Un *état cible* de la transition (obligatoire).

3.3.3 Etats Composites

Un *état composite* est un état qui contient d'autres états à l'intérieur (*sous-états*). Il existe deux mécanismes de composition des *sous-états* afin de former des *états composites : Orthogonale* ou *Séquentielle.* Un *état composite* est décomposé en régions contenant chacune un ou plusieurs *sous-états*. Quand un *état composite* comporte plus d'une région, il est qualifié *d'état orthogonal*. Un *état composite* ne comportant qu'une région est qualifié d'*état séquentiel*. Un *sous-état* est un état, il peut par conséquent être à son tour un *état composite*. L'utilisation d'*états composites* permet :

- ✓ De structurer les automates pour les rendre plus lisibles.
- ✓ De factoriser des transitions similaires qui partent de plusieurs états.
- ✓ De développer une spécification par raffinements.

3.3.3.1 Etat Composite Séquentiel (Hiérarchique ou XOR)

Un état composite *séquentiel* est constitué de *sous-états* qui se suivent (voir la Figure 3.4). A partir d'un état source externe, une transition peut cibler l'*état composite* ou un *sous-état*. Si la cible est un *sous-état*, l'automate imbriqué commence au *sous-état* spécifié après exécution de l'action d'entrée associée à l'*état composite* suivi par l'exécution de l'action d'entrée du *sous-état*. Si la cible est l'*état composite*, la transition est équivalente à une transition ayant pour cible l'état initial de l'état composite. L'automate imbriqué commence à l'état initial après exécution de l'action d'entrée associée à l'état composite.

Une transition qui sort d'un état composite peut avoir pour source l'*état composite* ou un *sous-état*. Dans les deux cas, l'action de sortie de l'*état composite* est exécutée avant de quitter l'état. Si la source est un *sous-état*, l'action de sortie de ce *sous-état* est exécutée avant celle de l'état composite. Si la source est l'*état composite*, si la transition ne porte pas d'évènement déclencheur explicite, elle sera franchissable quand l'état final de l'*état composite* est atteint. Sinon (la transition avec un évènement déclencheur), il est équivalente à des transitions qui s'appliquent à tout sous-état de l'état composite source, cette transition interrompt l'activité de l'automate imbriqué (tous les sous-états deviennent inactifs). Cette relation est transitive: la transition est franchissable depuis tout état imbriqué, quelle que soit sa profondeur (c'est-à-dire, les différents niveaux d'imbrication) en traversant les frontières des états composites. Un automate imbriqué peut avoir au maximum un état initial et un état final.

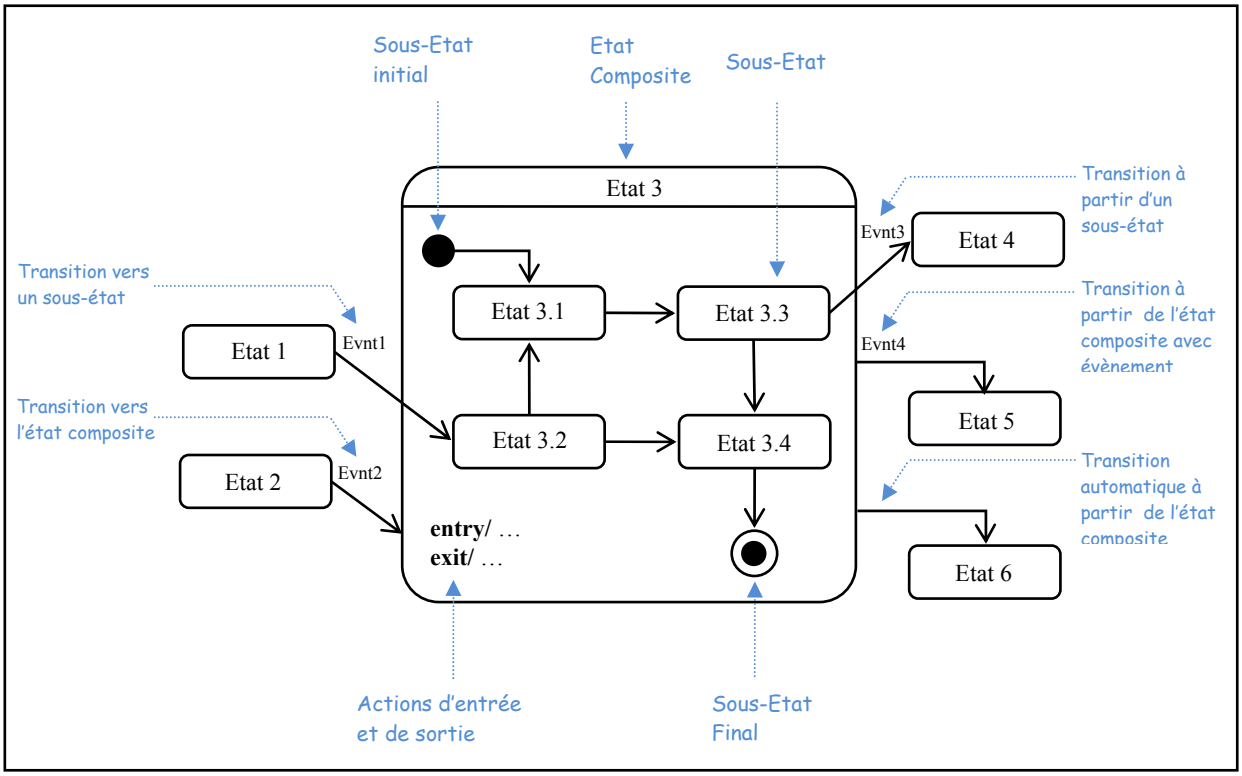


Figure 3.4 : Représentation Graphique d'un Etat Composite Séquentiel.

La Figure 3.5 présente un modèle de la transmission automatique de vitesse en voiture. Les vitesses sont passées dans l'ordre, et de chacune la transmission peut revenir au point mort. Le modèle de la transmission de vitesses est composé de trois états *Marche_avant*, *Marche_arrière* et *Point_mort*. L'état *Marche_avant* est un composite séquentiel incluant cinq états représentant les cinq rapports de vitesse. La notion d'état initial et final devient relative au niveau d'imbrication. La transition *passerPM* de l'état composite *Marche_avant* vers l'état *Point_mort* exprime que quelle que soit la vitesse enclenchée, la transmission passe au point mort. La transition *passerP* de l'état *Point_mort* vers l'état composite *Marche_avant* envoie dans l'état initial de l'état composite *Marche_avant* (c'est-à-dire, dans l'état Première).

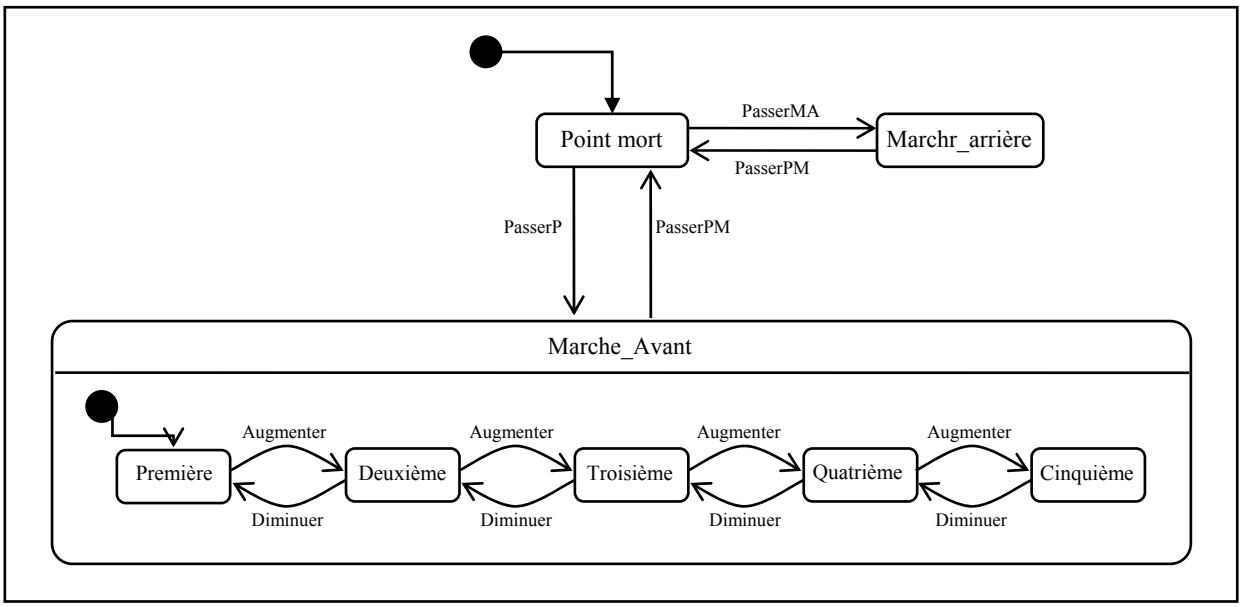


Figure 3.5 : Statechart de la Boite de Vitesse.

3.3.3.2 Etat Historique

Dans certaines situations, on a besoin que l'automate imbriqué se souvienne du dernier *sous-état* où il était avant de le quitter pour la dernière fois. On peut modéliser ce besoin à l'aide d'*état historique*. Un *état historique* est un pseudo-état qui mémorise le dernier sous-état actif d'un état composite. Graphiquement, il est représenté par un cercle contenant la lettre *H*. Une transition ayant pour cible l'état historique est équivalente à une transition qui a pour cible le dernier état visité de l'état englobant. Dans le cas où un état composite contient d'autres sous-états qui eux-mêmes sont composites, il est possible de définir un *état historique profond* représenté graphiquement par un cercle contenant la lettre H*. Cet état permet d'atteindre le dernier état visité quel que soit son niveau d'imbrication.

La Figure 3.6 montre un exemple de Statechart correspondant au fonctionnement simplifié d'un ascenseur. L'ascenseur a trois états: *Démarrage*, *EnMouvement* et *Stoppé*. L'état *EnMouvement* est un super-état composite de trois sous-états *Pallier*, *Montée* et *Descente*. Si la cabine de l'ascenseur l'est

stoppée (arrêt d'urgence, coupure d'électricité, …), lorsqu'elle est remise en route, elle repart du même endroit mémorisé dans l'état historique.

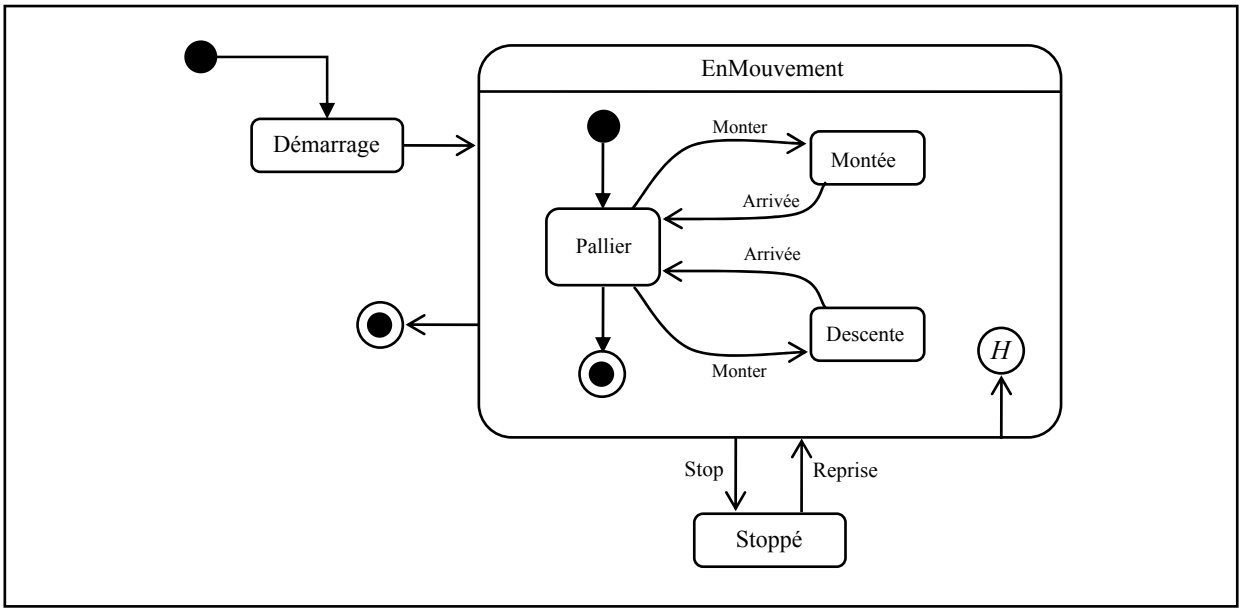


Figure 3.6 : Statechart Modélisant le Fonctionnement Simplifié d'un Ascenseur.

3.3.3.3 Etat Composite Orthogonal (Concurrent ou AND)

Un *état composite orthogonal* permet d'obtenir plusieurs automates imbriqués actifs simultanément. Il permet donc de réaliser des tâches en *parallèle*. Un *état orthogonal* est un état composite comportant plus d'une région. Pour séparer les régions d'un *état orthogonal,* une ligne horizontale en pointillée est utilisée allant du bord gauche au bord droit. Chaque région représente un flot d'exécution séquentiel et possède un état initial et un état final (voir la Figure 3.7).

Lorsqu'un état composite orthogonal est atteint, les exécutions de tous ces régions sont déclenchées à partir de leurs états initiaux, et se poursuivent en parallèle. Toutes les régions doivent atteindre leur état final pour sortir de l'état composite. Il est à note qu'un état composite orthogonal n'a pas d'état initial car chacune de ses régions concurrents a un état initial, ni d'état final car chacune de ses régions concurrents a un état final.

À tout instant plusieurs automates imbriqués peuvent être actifs. Dans ce cas, plusieurs états peuvent être actifs en même temps, l'état global étant alors caractérisé par l'ensemble des états actifs.

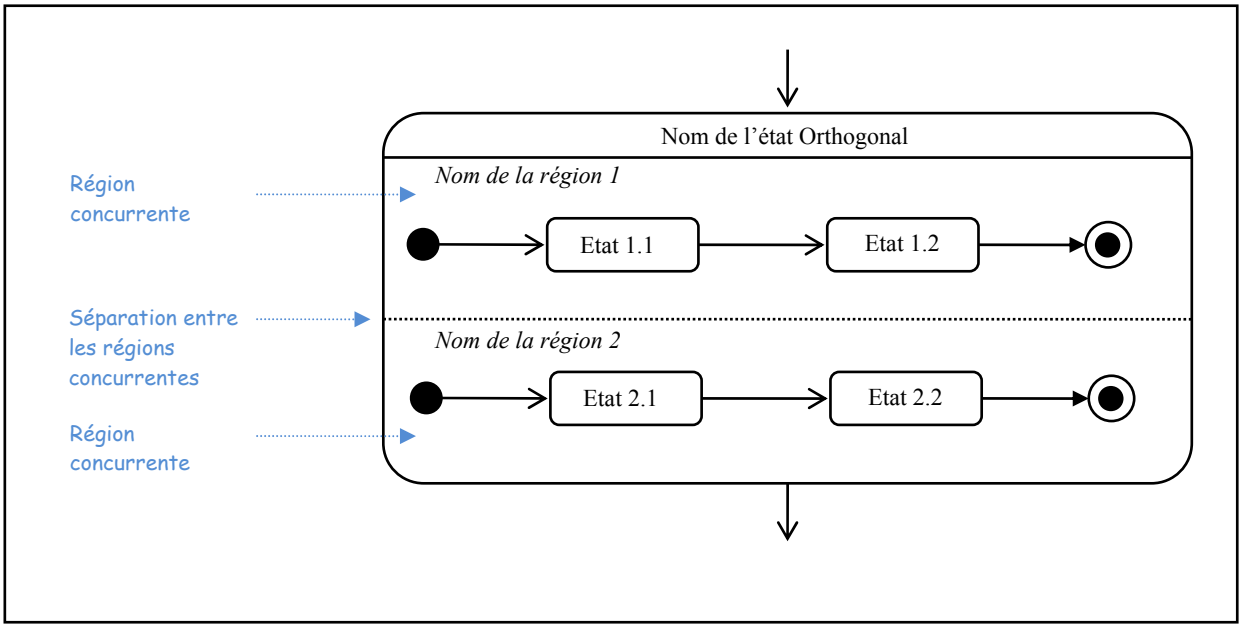


Figure 3.7: Représentation Graphique d'un Etat Composite Orthogonal.

Dans la Figure 3.8, le modèle présente l'utilisation d'un état composite orthogonal pour décrire le fonctionnement d'un distributeur automatique de boissons. Après avoir sélectionné la boisson et validé le montant par rapport au crédit, deux séquences d'actions sont déclenchées en parallèle : la préparation de la boisson et le rendu de la monnaie.

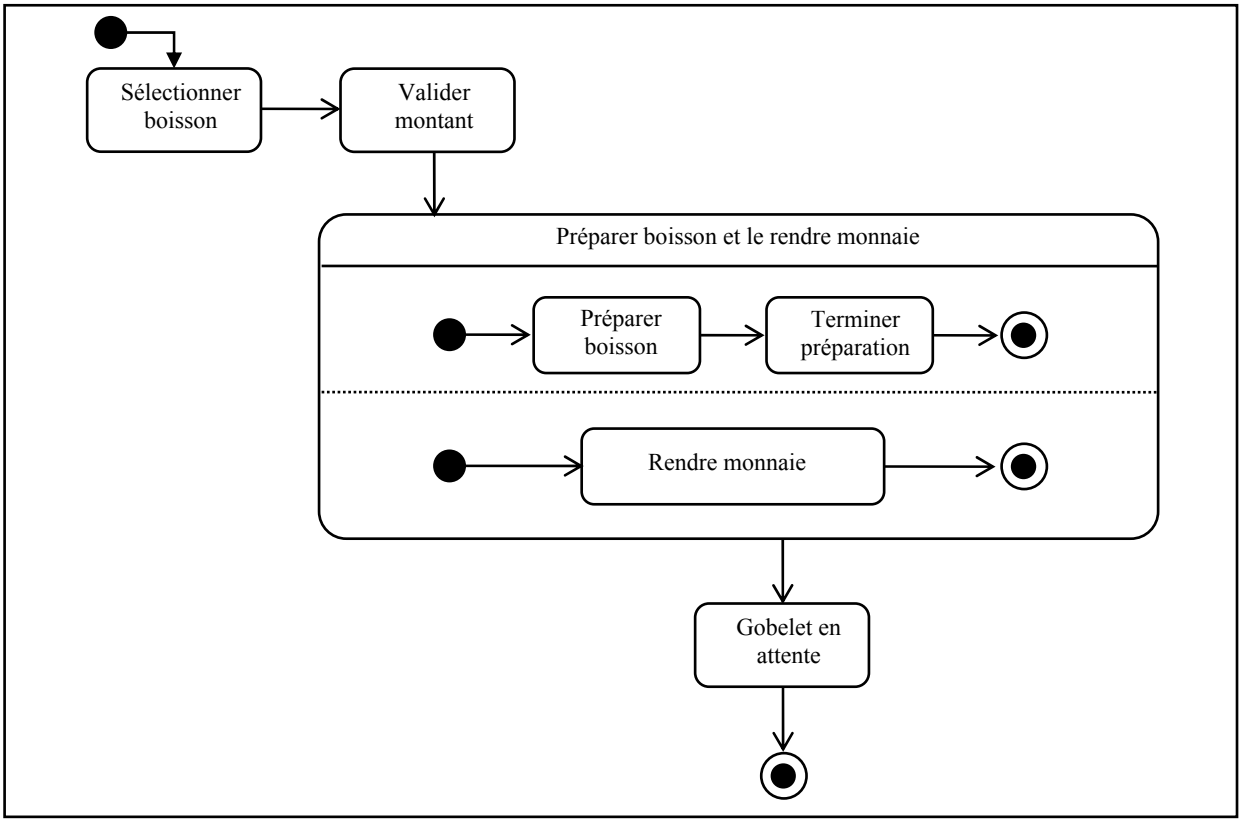


Figure 3.8: Statechart d'un Distributeur Automatique de Boissons.

3.4 Sémantique et Exécution

Les StateCharts ont donnée lieu à diverses interprétations au niveau de la sémantique opérationnelle. Nous ne donnerons ici qu'un bref aperçu de la sémantique opérationnelle des StateCharts. Le modèle d'exécution des StateCharts est synchrone. L'état du système avant un pas est défini par la configuration courante C ainsi que par l'ensemble E des événements générés au pas précédent (cet ensemble étant initialement vide).

L'exécution d'un pas se déroule de la manière suivante :

1) Ajouter à E d'éventuels événements de provenance externe (environnement du système).
2) Calculer l'ensemble T des toutes les transitions déclenchables.
3) Extraire de T un ensemble maximal non conflictuel T'. Les transitions les plus extérieures ont la priorité, en cas de conflit à priorité égale, un tirage au sort est effectué.
4) Exécuter les actions de T'. En particulier, remplacer E par l'ensemble des évènements générés par les éléments de T.
5) Adapter la configuration courante C.

Remarques

- L'exécution d'un pas présenté ci-dessus ne couvre qu'une partie du formalisme de Harel. Il existe encore d'autres types d'états et de transitions tels que les états historiques qui tiennent compte de l'histoire, les branchements conditionnels, etc.
- De ce point de vue, un StateChart n'est qu'une abréviation d'un automate d'états finis. La composition hiérarchique des états permet de diminuer le nombre de transitions et la composition parallèle permet de diminuer le nombre d'états.
- Les StateChart est un formalisme assez lisible, simple d'emploi et d'une grande expressivité. On peut par contre lui reprocher de se prêter moins bien que certains de ses concurrents à la vérification formel de propriétés. Pour remédier à ce défaut, deux voies sont possibles : la première est de redéfinir la sémantique opérationnelle de manière à la rendre plus facilement manipulable. La seconde est de définir une traduction des statecharts vers un formalisme possédant de bons outils de vérification formelle, tels que les réseaux de Petri, les systèmes de transitions.

3.5 Exercices

Exercice N° 1

Donner la machine d'états finis équivalente au Statecharts de la Figure 3.5 représentant la transmission de vitesse en voiture ?

Exercice N° 2

Reprendre l'exemple de la Figure 3.6, donner la machine d'états finis équivalente ?

Exercice N° 3

Dessiner un Statecharts résumant les états possibles d'un "contrat" tel que décrit dans l'énoncé suivant. Un ensemble de personnes décident d'établir un contrat. Pour ce faire elles rédigent un projet par itération successive. Le contrat est ensuite informellement accepté par les parties, et devient ce que l'on appelle un préaccord. A ce stade il peut toujours être l'objet de modification et revenir à l'état de projet. Une fois le préaccord définitivement établi, le contrat est signé par les parties. Dès ce moment les partenaires sont liés. Une fois signé le contrat peut être rendu exécutoire par une décision d'une des parties. Un contrat en exécution peut faire l'objet de discussions qui sont réglées par un arbitre désigné à cet effet. Le contrat une fois exécuté prend fin.

Exercice N° 4

Dessiner un Statecharts résumant le fonctionnement d'une montre digitale. La montre affiche l'heure, si on appuie 2 fois sur le *bouton 1*, la montre passe en mode "*modification*". Chaque pression sur le *bouton 2*, incrémente l'heure d'une unité. Si on appuie encore une fois sur le *bouton 1*, on peut régler les minutes de la même façon que les heures. Si on appuie une quatrième fois sur le *bouton 1*, la montre affiche à nouveau l'heure courante.

Chapitre 04 :

Réseaux de Petri

4.1 Introduction

Les Réseaux de Petri (*RdP*) ont été introduit par Carl Adam Petri en 1962 à l'université Darmstadt dans sa thèse intitulée : "*Communication avec des automates*" comme un outil de modélisation graphique et mathématique permettant la modélisation et l'analyse des systèmes dynamiques à événements discrets. En tant qu'outil graphique, les réseaux de Petri permettent la visualisation du comportement dynamique du système. En tant qu'outil mathématique, ils permettent l'analyse des propriétés du système. Ce formalisme bénéficie d'une riche panoplie de techniques d'analyse et d'outils de simulation. Les résultats de cette analyse sont utilisés pour évaluer le système et en permettre la modification ou l'amélioration [David 92].

4.2 Définition

4.2.1 Définition Informelle

Comme l'illustre le réseau de Petri de la Figure 4.1, un réseau de Petri est un graphe biparti orienté (ayant deux types de nœuds): des *places* représentées par des cercles et des *transitions* représentées par des rectangles. Les arcs du graphe ne peuvent relier que des places vers des transitions, ou des transitions vers des places. Chaque arc possède un entier positif qui représente son *poids* [Brams 83]. Par convention, lorsque le poids n'est pas précisé sur un arc, alors ce poids vaut 1.

Un réseau de Petri décrit un système dynamique à événements discrets. Les places permettent la description des états possibles du système et les transitions permettent la description des événements ou les actions qui causent le changement de l'état. Un réseau de Petri est un graphe muni d'une sémantique opérationnelle, c'est-à-dire qu'un comportement est associé au graphe, ce qui permet de décrire la dynamique du système représenté. Pour cela un troisième élément est ajouté aux places et aux transitions: les *jetons*.

Une répartition des jetons dans les places à un instant donné est appelée *marquage* du réseau de Petri. Un marquage donne l'état du système. Le nombre de jetons contenus dans une place est un entier positif ou nul. Pour un marquage donné, une transition est franchissable si chacune de ses places d'entrée contient au moins un jeton. L'ensemble des transitions franchissables pour un marquage donné défini l'ensemble des changements d'états possibles du système depuis l'état correspondant à ce marquage. C'est un moyen de définir l'ensemble des événements auxquels ce système est réceptif dans cet état.

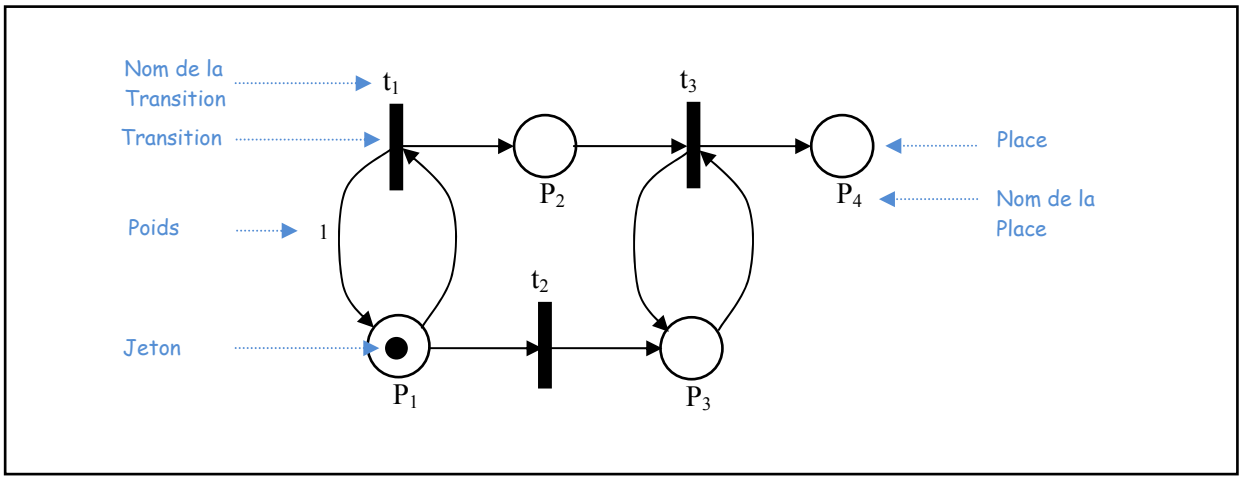


Figure4.1: Exemple de Réseau de Petri Marqué.

4.2.2 Définition Formelle

Un réseau de Petri est défini par le quintuplet $RdP = (P, T, Pré, Post, M_0)$,

Où :

- P = ensemble fini de places du réseau. Il contient des éléments (p_i) qui vont modéliser les ressources utilisées dans le système.
- T = ensemble fini de transitions du réseau. Il contient des éléments (t_i) qui vont modéliser les actions sur les ressources.
- $Pré$ = une application de $P\ x\ T \rightarrow \mathbb{N}$ appelée application d'incidence avant. $Pré\ (p, t)$ est une valeur (un poids ≥ 0) associée à l'arc allant de la place p à la transition t.
- $Post$ = une application de $T\ x\ P \rightarrow \mathbb{N}$ appelée application d'incidence arrière. $Post\ (t, p)$ est une valeur un poids ≥ 0) associée à l'arc allant de la transition t à la place p.
- M_0 = une application de $P \rightarrow \mathbb{N}$ appelée application du marquage initial. $M_0(p)$ est une valeur (≥ 0) représente le nombre initial de jetons dans la place p.

4.2.3 Représentation Matricielle

La représentation matricielle d'un réseau de Petri permet de définir les matrices correspondantes aux applications *Pré* et *Post* ainsi que le vecteur de marquage.

Soit un réseau de Petri $RdP = (P, T, Pré, Post, M_0)$, avec $P = \{p_1, p_2, \dots p_m\}$ et $T = \{t_1, t_2, \dots, t_n\}$. On appelle matrice d'incidence avant (des pré-conditions) $Pré$, la matrice $m\ x\ n$ (m représente le nombre de places et n le nombre de transitions) à coefficients dans $\mathbb{N}$ tel que $Pré\ (i, j) = Pré\ (p_i, t_j)$, qui indique le nombre de jetons que doit contenir la place p_i pour que la transition t_j devienne franchissable. De la même manière, on définit la matrice d'incidence arrière (des post-conditions) $Post$, la matrice $m\ x\ n$ tel que $Post(i, j) = Post(t_j, p_i)$ contient le nombre de jetons

déposées dans p_i lors du franchissement de la transition t_j. La matrice $C = Post - Pré$ est appelée matrice d'incidence du réseau. Le marquage d'un réseau de Petri est représenté par un vecteur de dimension m à coefficients dans $\mathbb{N}$. La représentation matricielle du réseau de Petri de la Figure 4.1 est comme illustrée dans la Figure 4.2.

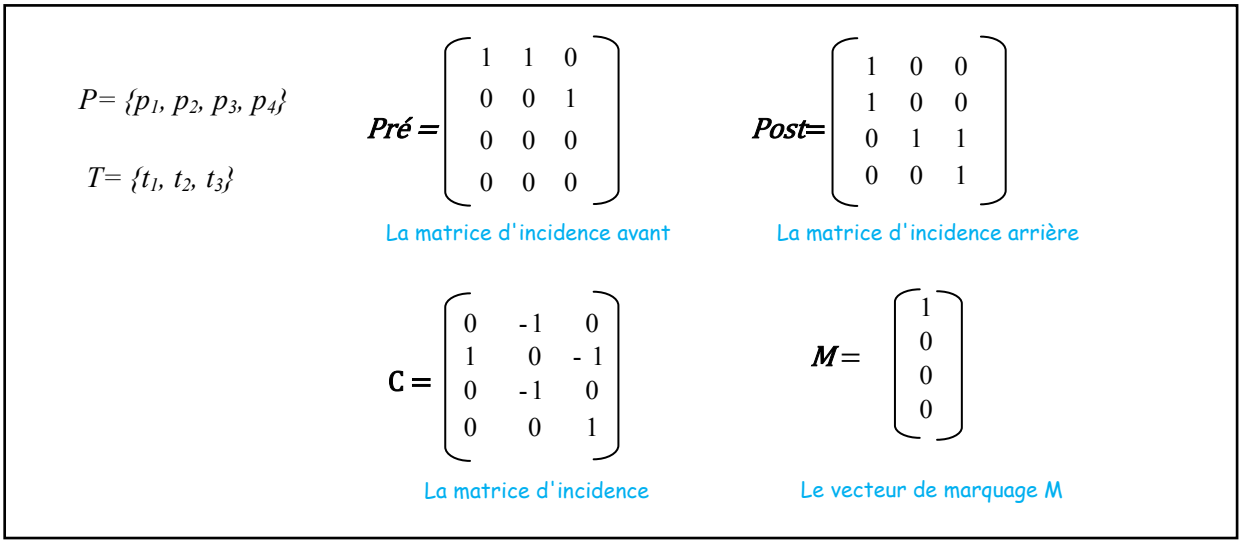


Figure 4.2: Matrice d'Incidence et Vecteur de Marquage du RdP de la Figure 4.1.

4.3 Sémantique & Exécution

4.3.1 Dynamique

La dynamique d'un réseau de Petri est déterminée par sa structure statique et par son état. L'état d'un réseau de Petri se modélise à l'aide d'un marquage que l'on fait évoluer en franchissant des transitions, ce qui correspond à exécuter les actions qui lui sont associées. Les règles suivantes permettent de représenter l'évolution des systèmes à modéliser [Brams 83]:

I. Une transition t est dite franchissable pour le marquage M si pour toute place p telle que $Pré\,(p,t) \neq 0$ est marquée d'au moins $Pré\,(p,t)$ jetons, C'est-à-dire, si $\forall p \in P\,, M(p) \geq Pré(p,t)$ ($M \geq Pré(.,t)$ pour la notation matriciel). On note ceci $M[t>$.

II. Une transition t est franchie en retirant $Pré\,(p,t)$ jetons de chaque place p telle que $Pré\,(p,t) \neq 0$, et en ajoutant $Post\,(t,p')$ jetons à chaque place p' telle que $Post(t,p') \neq 0$, c'est-à-dire, si t est franchissable pour M, alors le franchissement de t fait passer le marquage M au marquage M' de la façon suivante :

$$\forall p \in P\,, M'(p) = M\,(p) - Pré\,(p,t) + Post\,(t,p).$$

$M' = M + C(.,t)$ pour la notation matriciel.

On note ceci $M[t > M'$, ce qui veut dire que lorsque la transition t est franchie à partir d'un marquage M, il faut saisir $Pré\ (p,t)$ jetons à partir de chaque place d'entrée à la transition t et déposer $Post\ (t,p)$ jetons dans chaque place de sortie de la transition t ce qui permet de produire un nouveau marquage M'.

La notion de franchissement de transitions peut alors être étendue aux séquences de transitions. Soit $s = t_1\ t_2\ \ldots\ t_n$ une séquence de transitions. La séquence s est franchissable à partir de M et conduit au marquage M' ce qui sera note $M[t > M'$ si et seulement s'il existe des marquages $M_0 = M, M_1, \ldots\ldots, M_n = M'\ tels\ que\ \forall i : 0 \leq i \leq n-1 :\ M_i(t_{i+1}\rangle\ M_{i+1}$.

4.3.2 Graphe de Marquages

L'idée la plus naturelle pour étudier le comportement d'un réseau de Petri est de construire le graphe de tous ses marquages accessibles. Le graphe des marquages accessibles est un graphe dont chaque sommet correspond à un marquage accessible et dont chaque arc correspond au franchissement d'une transition permettant de passer d'un marquage à l'autre. On appelle $^*M$ l'ensemble des marquages accessible d'un réseau de Petri à partir d'un marquage initial. La Figure 4.3 présente le graphe de marquage du réseau de Petri à droite de la Figure. Deux situations peuvent alors se présenter :

1. ***Le graphe est fini.*** C'est la situation la plus favorable car dans ce cas toutes les propriétés peuvent être déduites simplement par inspection de ce graphe.
2. ***Le graphe est infini.*** Dans ce cas, on construit un autre graphe appelé "graphe de couverture" permettant de déduire certaines propriétés.

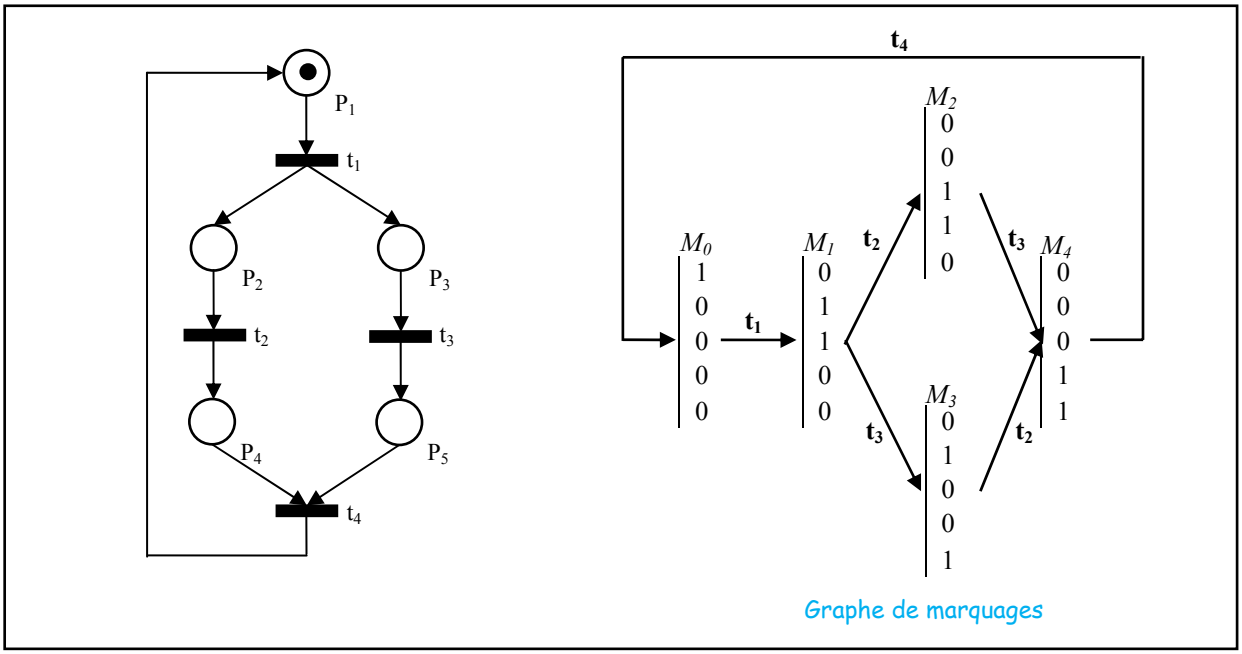


Figure 4.3: Exemple de Graphe de Marquages.

4.3.3 Graphe de Couverture

Un graphe de marquage ne peut plus être construit quand le nombre de marquages accessibles est infini. D'où le recourt au graphe dit de couverture. C'est un graphe à nombre de marquages fini. L'algorithme de construction du graphe de couverture est comme suit :

Pas 01: A partir du marquage initial M_0, on indique toutes les transitions validées et les marquages accessibles successeurs correspondants. Si un des marquages est strictement supérieur à M_0, on met la variable ω pour chacune des composantes supérieures aux composantes de M_0. La variable ω matérialise le fait que la place peut contenir autant de jetons que souhaité.

Pas 02 : Pour chaque nouveau marquage M_i, on fait soit le pas 2.1 soit le pas 2.2 suivants :

Pas 2.1: S'il existe sur le chemin de M_0 jusqu'à M_i (ce dernier exclut) un marquage

$$M_j = M_i \ (\forall p \in P, M_k(p) = M_j(p))$$

, alors M_i n'a pas de successeurs.

Pas 2.2: s'il n'existe pas de marquage $M_k = M_i$ sur le chemin de M_0 à M_i, alors on prolonge le graphe en ajoutant tous les successeurs de M_i. Pour chaque successeur M_k de M_i :

- Une composante ω de M_i reste une composante ω de M_k .
- S'il existe un marquage M_j sur le chemin de M_0 à M_k tel que M_k telque

$$M_k > M_j \ (\forall p \in P, M_k(p) > M_j(p))$$

, alors on met ω pour chacune des composantes supérieures aux composantes de M_i .

La Figure 4.4 présente un exemple de graphe de couverture.

Remarque

- Comme pour le graphe de marquages accessible, on peut déduire de l'observation du graphe de couverture un certain nombre de propriétés pour le réseau de Petri.

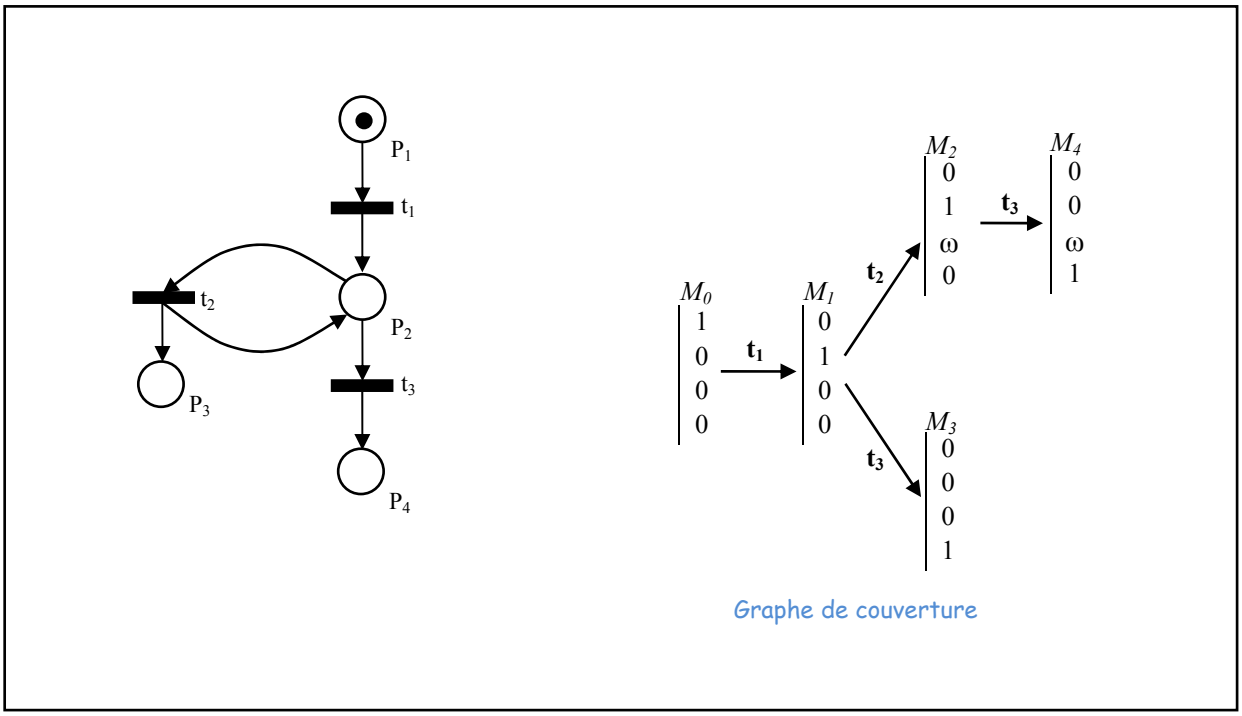


Figure 4.4: Exemple de Graphe de Couverture.

4.3.4 Vecteur d'Occurrence et l'Equation de Changement d'Etat

Soit $s = t_1\ t_2\ \ldots\ t_n$ une séquence de transitions ($s \in T^*$). $\vec{s}$ est un vecteur appelé le vecteur d'*occurrences*, où chaque élément représente $\vec{s}(t)$ de ce vecteur représente le nombre d'occurrence de t dans s.

Par exemple, si on a un graphe contenant les transitions $t_1, t_2,\ t_3$. si on prend la séquence de transition : $s = t_1\ t_2\ t_1$ le vecteur d'occurrence $\vec{s} = [\vec{s}(t_1), \vec{s}(t_2), \vec{s}(t_1)] = [2, 1, 0]$.

A partir d'un marquage M , on peut tirer une séquence de transitions s et on trouve le marquage M'.

L'équation de changement d'état est :

$$M' = M\ C.\vec{s}$$

Remarque

- Les résultats de l'équation, même s'ils sont toujours calculables, n'ont de sens que si la séquence *s* est effectivement franchissable. Pour le réseau de Petri de la Figure 4.3, calculer le marquage après la séquence de franchissement : $s = t_1\ t_3\ t_4\ t_1\ t_3\ t_4\ t_1\ t_2$?

4.4 Propriétés des Réseaux de Petri

4.4.1 Les Propriétés Structurelles

Les propriétés structurelles dépendent uniquement de la topologie du réseau. Il s'agit de faire ressortir les propriétés statiques du système étudié. Ces différentes propriétés sont indépendantes du marquage [Murata 89].

- ***Les conflits*** dans un réseau de Petri (Figure 4.4(a) et (b)): Si une place ce trouve en amont de plusieurs transitions. On note le conflit de la place P_i : $K = (P_i, \{T_1, T_2, \ldots\})$; Où $T_1, T_2, \ldots$ sont les transitions en concurrence. On parle de *conflit structurel* car cela ne dépend pas du marquage. Dans certains cas, le franchissement de l'une des transitions peut empêcher le franchissement de l'autre (la Figure 4.4(a)). Le conflit devient *conflit effectif* quand il y a effectivement conflit, cela dépend du marquage (la Figure 4.4(b)).
- ***RdP à choix libre*** (Figure 4.4(c)) : Un réseau de Petri à choix libre est un réseau dans lequel pour tout conflit ($K = (P_i, \{T_1, T_2, \ldots, T_n\})$ aucune des transitions $T_1, T_2, \ldots, T_n$ ne possède aucune autre place d'entrée que P_i.
- ***RdP simple*** (Figure 4.4(d)) : Un réseau de Petri simple est un réseau de Petri dans lequel chaque transition ne peut être concernée que par un conflit au plus.
- ***RdP pur*** (Figure 4.4(e)): Un réseau de Petri pur est un réseau dans lequel il n'existe pas de transition ayant une place d'entrée qui soit à la fois place de sortie de cette transition.
- Un réseau de Petri est un ***graphe d'état*** (Figure 4.4(f)) si et seulement si toute transition a exactement une seule place d'entrée et une seule place de sortie.
- Un réseau de Petri est un ***graphe d'événement*** (Figure 4.4(g)) si et seulement si chaque place possède exactement une seule transition d'entrée et une seule transition de sortie.

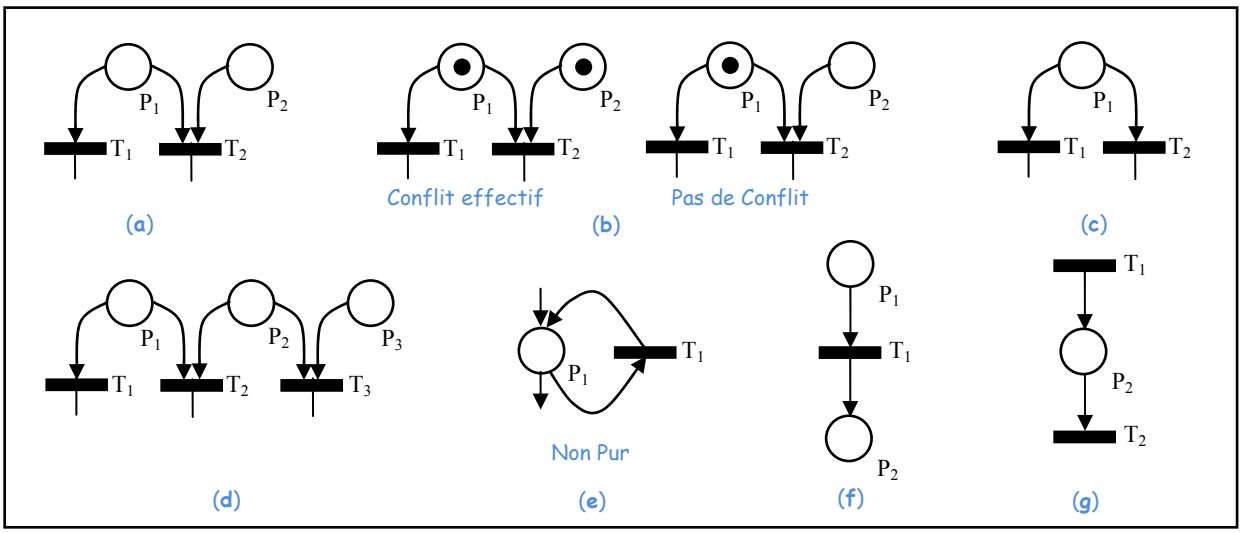


Figure 4.5: Détails des Propriétés Structurelles des RdPs.

4.4.2 Les Propriétés Comportementales

Ces propriétés dépendent à la fois du marquage initial M_0 et de la structure du réseau. Il s'agit ici de faire ressortir les propriétés dynamiques du système étudié [Murata 89].

- ***Existence d'un marquage:*** Pour toute séquence de transitions s, il existe un marquage M tel que celle-ci soit franchissable.
- ***Monotonie:*** L'augmentation de jetons dans les places d'un marquage préserve la possibilité de franchissement d'une séquence de transitions.
- ***Séquence répétitive:*** Une séquence de transitions est dite répétitive si pour tout marquage M tel que : $M[s >$ alors $M[s^* >$. La notion de séquence répétitive permet de définir une condition nécessaire et suffisante pour qu'un réseau marqué ait la possibilité d'être infiniment actif.
- ***Caractère borné:*** Cette propriété définit et caractérise la possibilité pour une place d'accumuler une quantité bornée ou pas de jetons au cours de l'évolution d'un réseau.
- ***Place k-bornée, non bornée:*** Pour un réseau réseau de Petri et un marquage M_0, une place p du réseau marqué (RdP, M_0) est k-bornée si pour tout marquage M accessible depuis $M_0, M(p) \leq k$. Dans le cas contraire la place p est dite non-bornée.
- ***Réseau borné :*** Un réseau marqué est borné si toutes ses places sont bornées. Les réseaux 1-bornés sont appelés des réseaux *saufs*.
- ***Activité d'un réseau :*** La notion d'activité d'un réseau recouvre deux classes de définitions. La première concerne l'activité individuelle des transitions, la seconde concerne l'activité globale d'un réseau (Indépendamment de transitions particulières).
- ***Pseudo-vivacité:*** Un réseau de Petri (RdP, M_0) est dit pseudo-vivant si pour tout marquage accessible depuis le marquage initial, il existe toujours une transition t qui puisse être franchie.
- ***Quasi-vivacité d'une transition:*** La quasi-vivacité d'une transition signifie que depuis le marquage initial, cette transition peut être franchie au moins une fois. Par conséquent, une transition qui n'est pas quasi-vivante est inutile.
- ***Quasi-vivacité d'un réseau:*** Un réseau est quasi-vivant si toutes ses transitions le sont.
- ***Monotonie et quasi-vivacité:*** La propriété de monotonie présentée plus haut, implique qu'une transition quasi-vivante pour (RdP, M) le reste pour (RdP, M').
- ***Vivacité :*** Les deux propriétés précédentes assurent une certaine correction du système mais elles ne permettent pas d'affirmer que, dans n'importe quel état atteint, le système dispose encore de toutes ses fonctionnalités. Autrement dit, si toute transition peut toujours être ultérieurement franchie à partir d'un état quelconque du système. Par exemple, dans un réseau quasi-vivant une transition pourra être franchie une seule fois.

- ***Vivacité d'une transition:*** La vivacité d'une transition exprime le fait que quelque soit l'évolution du réseau à partir du marquage initial, le franchissement à terme de cette transition est toujours possible.
- ***Vivacité d'un réseau:*** Un réseau est vivant si toutes ses transitions le sont.
- ***État d'accueil:*** Un réseau de Petri possède un état d'accueil M_a pour un marquage initial M_0 si pour tout marquage accessible M_i il existe une séquence s telle que $M_i\ [s > M_a$.
- ***RdP réversible:*** Un réseau de Petri est réversible pour un marquage initial M_0 si M_0 est un état d'accueil.
- ***Absence de blocage:*** Cette propriété est plus faible que celle de vivacité. Elle implique seulement que le réseau a toujours la possibilité d'évoluer.
- ***Marquage puit:*** Un marquage puit est un marquage à partir duquel aucune transition n'est tirable. Un réseau marqué est sans blocage si aucun de ses marquages accessibles n'est un marquage puit.

4.5 Modélisation des Systèmes

Les réseaux de Petri permettent de modéliser un certain nombre de comportements importants dans les systèmes complexes tels que le parallélisme, la synchronisation, le partage de ressources, la mémorisation et la lecture d'information, la limitation d'une capacité de stockage, etc [Vidal 92].

4.5.1 Parallélisme

Le parallélisme représente la possibilité que plusieurs processus évoluent simultanément au sein du même système. On peut provoquer le départ simultané de l'évolution de deux processus à l'aide d'une transition ayant plusieurs places de sortie.

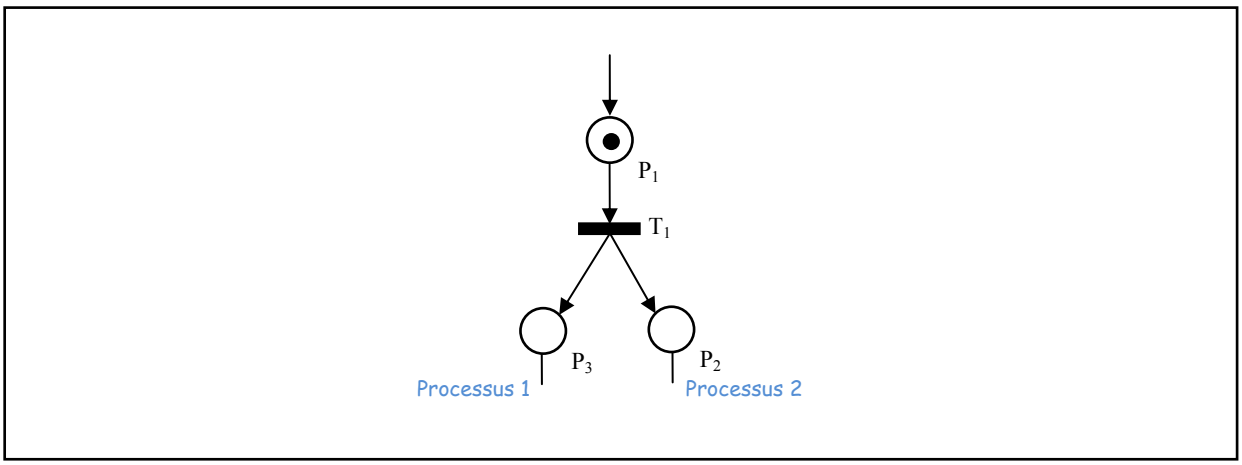


Figure 4.6: Structure du Parallélisme.

Le franchissement de la transition T_1 met un jeton dans la place P_2 (ce qui marque le déclenchement du processus 1) et un jeton dans la place P_3 (ce qui marque le déclenchement du processus 2).

4.5.2 Synchronisation

La synchronisation est modélisée sous deux formes :

- **Synchronisation mutuelle (Par rendez vous) :** La synchronisation mutuelle ou par rendez-vous permet de synchroniser les opérations de deux processus. La Figure 4.7 montre un exemple de deux processus. Le franchissement de la transition T_7 ne peut se faire que si la place P_{12} du processus1 et la place P_6 du processus 2 contiennent chacune au moins un jeton. Si ce n'est pas le cas, par exemple la place P_{12} ne contient pas de jetons, le processus 2 est bloqué sur la place P_6; il attend que l'évolution du processus 1 soit telle qu'au moins un jeton apparaisse dans la place P_{12}.

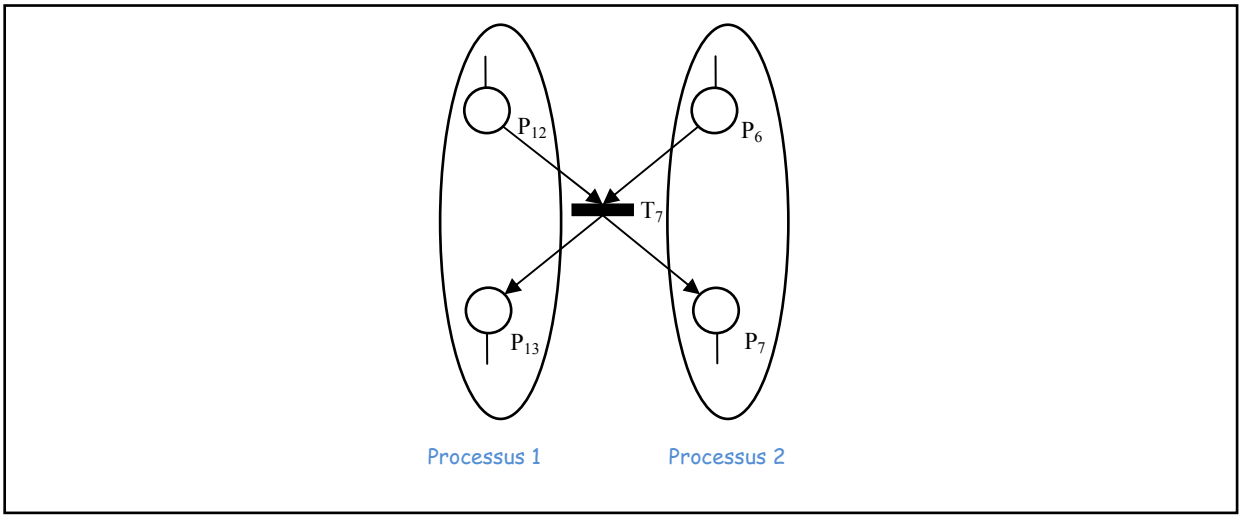


Figure 4.7: Synchronisation Mutuelle

- **Synchronisation par signal (sémaphore):** Dans la synchronisation par signal, les opérations du processus 2 ne peuvent se poursuivre que si le processus 1 a atteint un certain niveau dans la suite de ses opérations. Par contre, L'avancement des opérations du processus 1 ne dépend pas de l'avancement des opérations du processus 2.

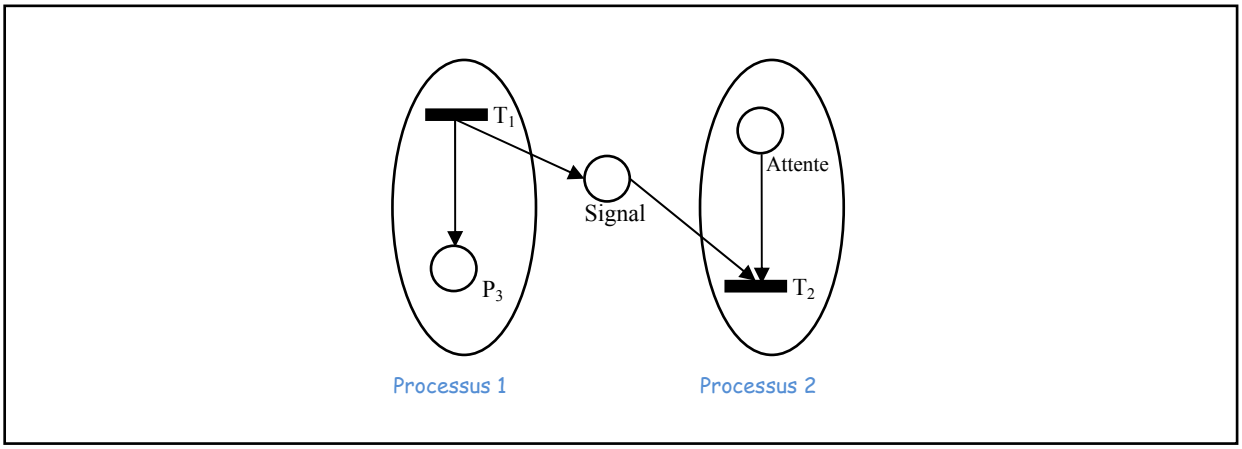


Figure 4.8: Synchronisation par Sémaphore

Si la place *Signal* est marquée et la place *Attente* ne l'est pas, cela signifie que le processus 1 a envoyé le signal mais le processus 2 ne l'a pas encore reçu. Si, par contre, la place *Signal* n'est pas marquée et que la place *Attente* est marquée, cela signifie que le processus 2 est en attente du signal.

4.5.3 Partage de ressources

Cette structure va modéliser le fait qu'au sein du même système plusieurs processus partagent une même ressource en utilisant le principe de l'exclusion mutuelle. Dans la Figure 4.9, Le jeton dans la place P_0 présente une ressource mise en commun entre le processus 1 et le processus 2. Le franchissement de la transition T_{17} lors de l'évolution du processus 1 entraîne la consommation du jeton présenté dans la place P_0. La ressource que constitue ce jeton n'est alors plus disponible pour l´évolution du processus 2. Lorsque la transition T_{18} est franchie, un jeton est alors est placé dans la place P_0 : la ressource devient alors disponible pour l'évolution des deux processus.

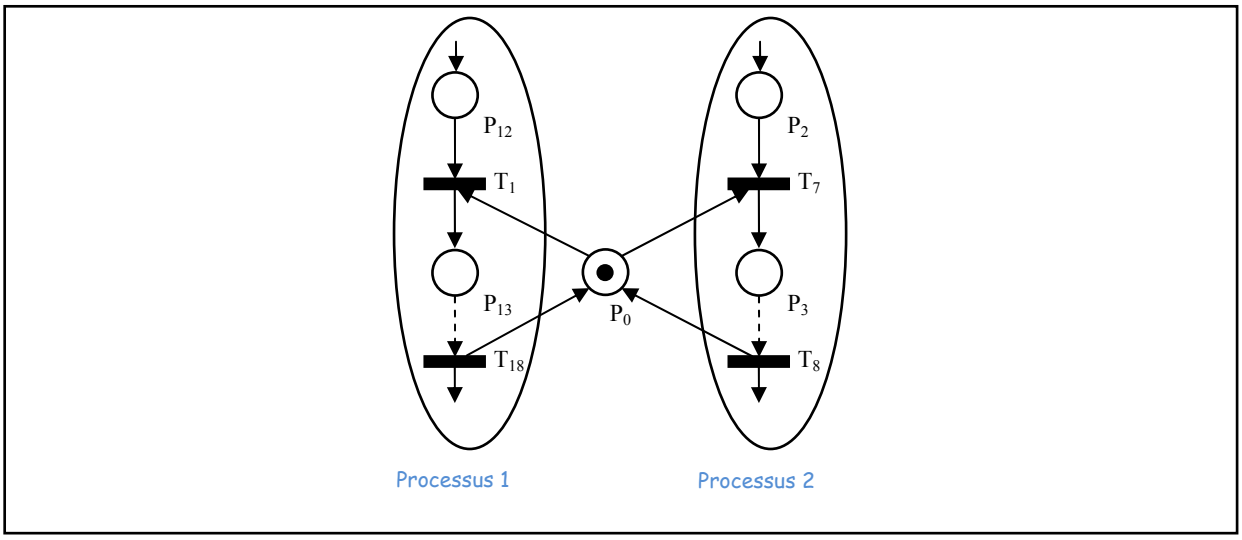


Figure 4.9: Synchronisation par Partage de Ressources

4.5.4 Mémorisation

Dans tous les modèles, on peut compter le nombre de tirs d'une transition en utilisant une place sans sortie. Dans La Figure 4.10 les places *Attente* et *Compteur* peuvent servir à indiquer combien d'instances d'un processus sont en attente.

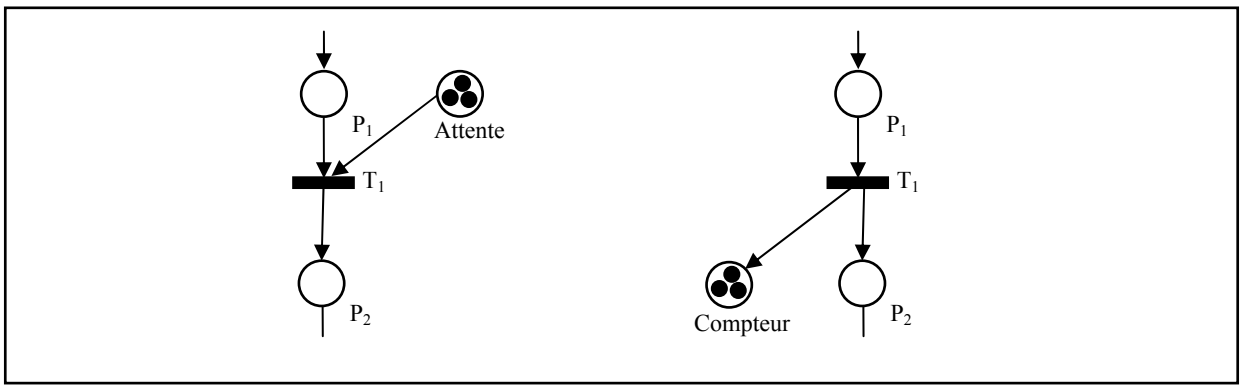
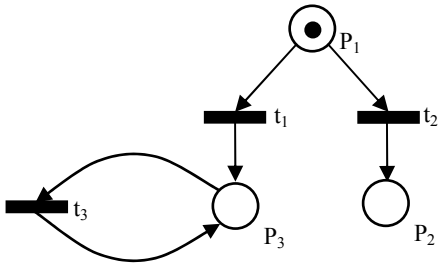


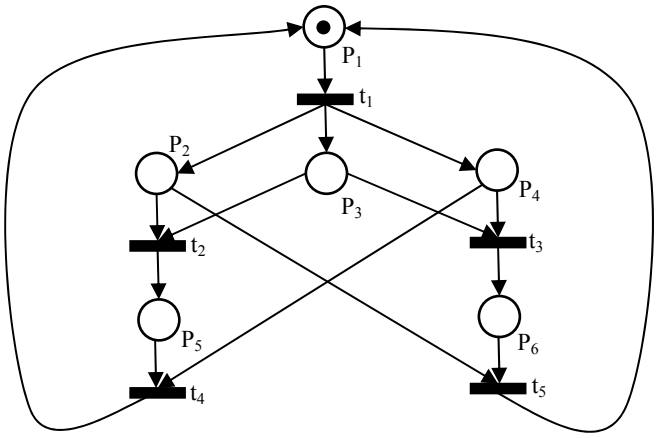
Figure 4.10: Synchronisation par Mémorisation

4.6 Exercices

Exercice N° 1

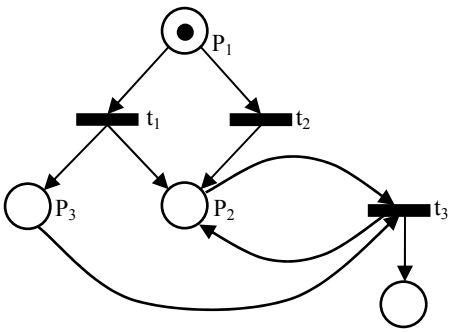
Dessiner les graphes des marquages accessibles pour les réseaux de Petri suivants ?

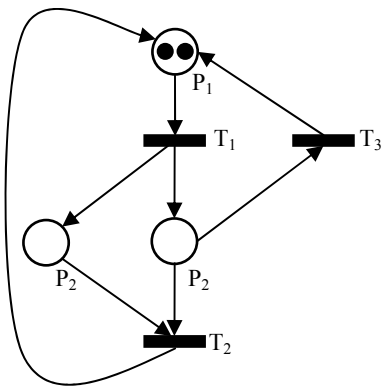




Exercice N° 2

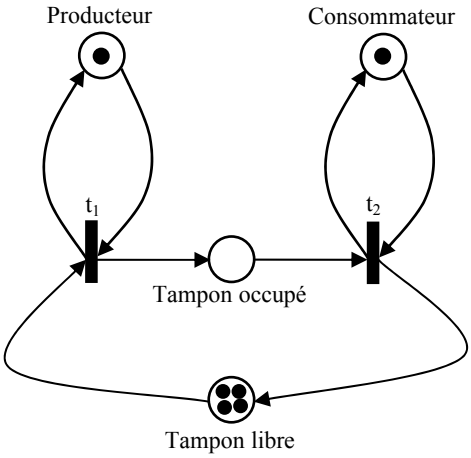
Trouver les graphes de couvertures pour les Réseaux de Petri suivants ?





Exercice N° 3

On considère un système producteur/consommateur dans lequel le tampon contient au maximum 4 produits. Le réseau de Petri correspondant sera le suivant :



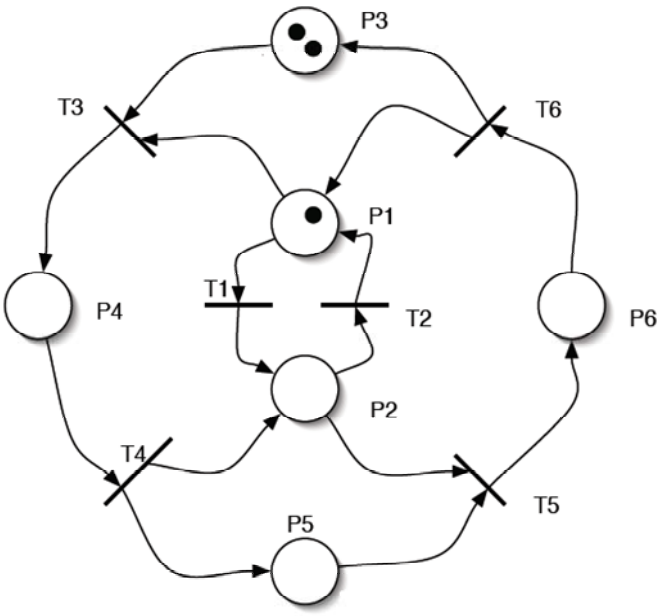
1. Donner les matrices $Pr\acute{e}$ et $Post$ pour ce réseau de Petri ?
2. Donner le graphe de marquages accessibles pour le marquage initial $M_0 = [1, 1, 0, 4]$?
3. Calculer le marquage résultat de la séquence

$$s = t_1\ t_1\ t_2\ t_1\ t_2\ t_2\ t_1\ t_1\ t_1\ t_1\ t_1\ t_2\ t_1\ t_1$$

sachant que le marquage initial est $M_0 = [1, 1, 2, 4]$? s est-t-il effectivement franchissable ?

Exercice N° 4

1. Dessiner le graphe de marquages accessibles pour le réseau de Petri ci-dessous ?
2. En déduire si ce réseau de Petri est borné, vivant, répétitif et sans blocage ? (justifier)
3. Donner les invariants de marquages ?



Chapitre 05 :

Chaînes de Markov

5.1 Introduction

Les systèmes que nous considérons dans ce chapitre sont des systèmes dynamiques faisant intervenir de l'aléatoire dans leurs évolutions au cours du temps. Les Chaînes de Markov sont un outil permettant de modéliser les processus dans lesquels une réalisation dépend de la réalisation précédente.

5.2 Rappels de Probabilités

On fait appel aux probabilités afin de modéliser une expérience (ou un système) sans disposer de l'information nécessaire pour la décrire exactement. Par exemple, on jette un dé et on lit le numéro apparu sur la face supérieure. Pour prédire si le résultat sera $1, 2, 3, 4, 5$ ou 6, il faut disposer d'une modélisation complète et parfaite du dé, de l'état de l'air, du numéro sur face supérieure au moment du jet de dé, de la vitesse à laquelle il est lancé, etc. Il est à note que dans ces expériences, le résultat est impossible à prévoir avec certitude, mais on connait l'ensemble des résultats possibles [Boccara 95].

5.2.1 Evénement

On considère un ensemble S qui représente l'ensemble des issues possibles d'une expérience. Dans l'exemple du jet d'un dé, l'ensemble des issues possible est $S = \{1, 2, 3, 4, 5, 6\}$. C'est un ensemble *discret*. Si, au contraire, l'expérience consiste à mesurer le temps jusqu'à ce que quelqu'un entre dans un bureau de Poste, l'ensemble S est continu : $S = \{x \in \mathcal{R} \mid x > 0\}$.

Un évènement A est un sous-ensemble de l'ensemble S des issues possibles pour l'expérience. Si le résultat de l'expérience appartient à A, on dit que l'évènement A s'est réalisé. Par exemple, on peut s'intéresser à l'évènement $A = \{2, 4, 6\} \subset S$ que le jet d'un dé donne un nombre pair.

Remarques

- Les évènements composés d'un seul élément de S sont appelés événements élémentaires.
- L'ensemble S tout entier est l'évènement certain.
- Le sous ensemble ϕ de S est appelé l'évènement impossible.
- Etant donnés deux évènements A et B :

– Si le résultat de l'expérience appartient à $A \cup B$, on dira que l'évènement A ou l'évènement B s'est produit.

– Si le résultat de l'expérience appartient à $A \cap B$, on dira que les deux évènements A et B se sont produits.

– Si $\cap B = \phi$, les deux évènements ne peuvent se produire simultanément : ils sont dits mutuellement exclusifs.

- Si le résultat de l'expérience n'appartient pas à A, c'est qu'il appartient à son complémentaire $\overline{\mathrm{A}} = \{\mathrm{e} \in \mathrm{S} \mid \mathrm{e} \notin \mathrm{A}\}$ dans S.

- Un ensemble d'évènements $A1, A2, \ldots$ forment une partition de S s'ils sont mutuellement exclusifs ($Ai \cap Aj = \phi \, si \, i \neq j$) et couvent l'ensemble des issues possibles de l'expérience ($\cup i \, Ai = S$).

5.2.2 Probabilité d'un Evènement

La probabilité $P[A]$ de l'évènement A est la fréquence relative à laquelle se produit cet évènement au cours d'une expérience répété un nombre infini de fois :

$$P[A] = \lim_{N \to \infty} N(A)/N$$

Quelques Propriétés

- $0 \leq P[A] \leq 1$
- $P[S] = 1$
- $P[\phi] = 0$
- $P[A \cup B] = P[A] + P[B] - P[A \cap B]$
- $P[A \cup B] = P[A] + P[B] \, si \, A \cap B = \phi$
- $P[\bar{A}] = 1 - P[A]$
- $P[A] \leq P[B] \, si \, A \subset B$

5.2.3 Probabilité Conditionnelle

L'occurrence d'un évènement A peut dépendre du fait que l'évènement B s'est produit. La probabilité de l'évènement A sachant que B s'est produit est donnée par :

$$P[A/B] = \frac{P[A \cap B]}{P[B]}$$

On a donc :

$$P[A \cap B] = P[A/B].P[B]$$

Exemple

Soit trois cartes, la première carte a ses deux faces rouges (RR). La deuxième carte a ses deux faces bleu (BB). Finalement, la troisième carte a une face rouge et l'autre bleu (RB).

On tire une carte et on ne regarde que la face dessus, elle est rouge. Quelle est la probabilité que l'autre face soit bleu ?

Réponse: $P[RB/R] = ⅓$

5.2.4 Indépendance

Deux évènements A et B sont dits indépendants si et seulement si $P[A \cap B] = P[A].P[B]$. Dans ce cas, on a :

$$P[A/B] = \frac{P[A \cap B]}{P[B]} = P[A]$$

C'est-à-dire, Savoir que B s'est produit n'apporte aucune information sur la probabilité que A se produit.

5.2.5 Variable Aléatoires

Si on lance 10 fois une pièce de monnaie à pile ou face. L'ensemble des issues possible est :

$$S = \{pppppppppp, ppppppppppf, \dots, ffffffffff\}, |S| = 2^{10}$$

Généralement, on s'intéresse au nombre de fois où la pièce est tombée sur face que la séquence de piles et de faces obtenue. Par exemple, si on obtient le résultat $pppfppffpf$, ce qui nous intéresse c'est que la pièce est tombée 4 fois sur face.

De manière générale, on est intéressé par une mesure (c'est-à-dire un nombre) associé à l'expérience que par le résultat proprement dit. Cette mesure est appelée variable aléatoire [Feller 60].

Une variable aléatoire est une fonction :

$$\begin{aligned} X: \quad & S \longrightarrow E \\ & e \longrightarrow X(e) \end{aligned}$$

qui associée à chaque issue e possible un élément de E (espace d'état).

L'image d'une variable aléatoire est l'ensemble :

$$S_x = \{x \in E \,/\, \exists e \in S, X(e) = x\}$$

L'image d'une variable aléatoire est donc l'ensemble des valeurs qu'elle peut prendre. Si cet ensemble est fini ou dénombrable, on dit que cette variable aléatoire est *discrète*. Sinon, elle est *continue*.

Par exemple, on peut définir l'évènement $[X = x] = \{e \in S \,/\, X(e) = x\}$.

Exemple

On lance une pièce de monnaie trois fois. L'ensemble des issues possibles associées à cette expérience est :

$$S = \{ppp, ppf, pfp, pff, fpp, fpf, ffp, fff\}, |S| = 8$$

Si on suppose la pièce équilibrées, on peut considérer que ces huit issues sont équiprobables. Notions X le nombre de cotés face obtenus. X est une variable aléatoire qui prend les valeurs : $0, 1, 2$ ou 3. $X_S = E = \{0, 1, 2, 3\}$. On note, par exemple, $[X = 2]$ l'évènement "le coté face est sorti exactement deux": $[X = 2] = \{e \in S \,/\, X(e) = 2\}$. La Figure 5.1 illustre la fonction de la variable aléatoire X avec l'évènement $[X = 2]$.

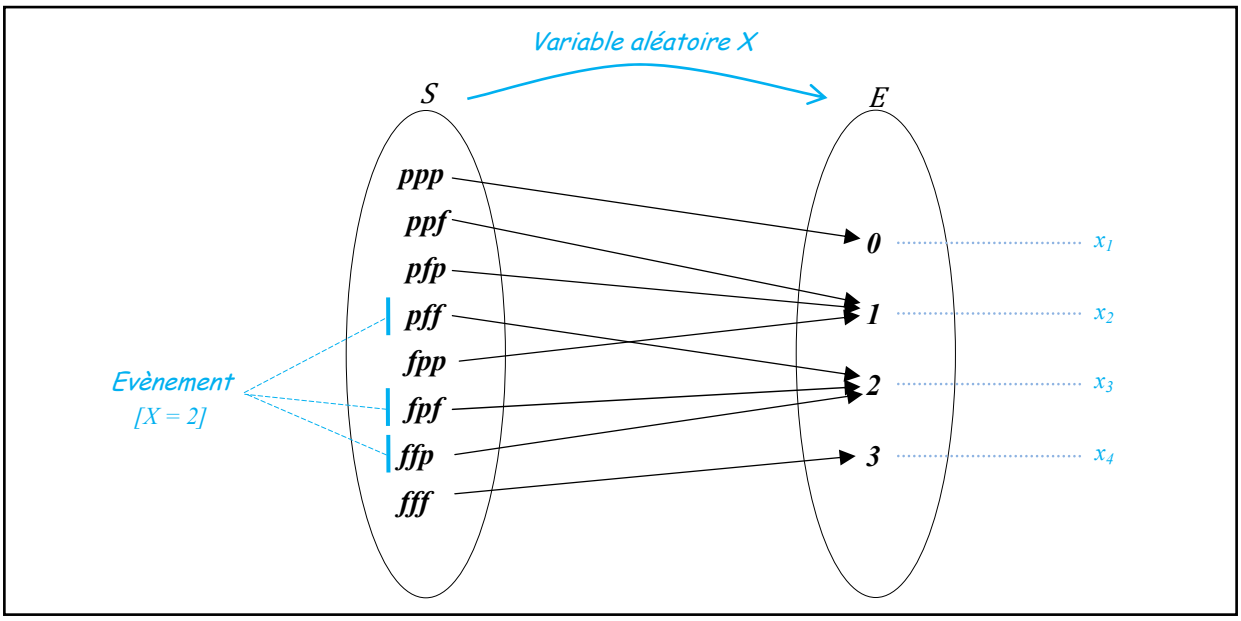


Figure 5.1: Variable Aléatoire .

Remarques

- On n'a pas besoin de probabilité pour définir une variable aléatoire.
- A chaque valeur x_i de X on associe les probabilités p_i de l'évènement$[X = x_i]$, on obtient ce qu'on appelle une *loi de probabilité* ou une *distribution de probabilité* de la variable aléatoire X.

Pour illustrer, sur l'exemple précédant à la valeur $X = 2$, on peut associer la probabilité $P = 1/8$ puisque on a 3 chances sur 8 d'obtenir exactement deux fois le coté face.

5.3 Processus Stochastique

Un processus stochastique ou aléatoire $(X_t)_{t\geq 0}$ est une séquence de variables aléatoires fondées sur le même ensemble des issues S, et indexées par le temps t. Pour t fixé, X_t est simplement une variable aléatoire prenant ses valeurs dans un certain espace d'états suivant une distribution qui peut dépendre de t. le temps t peut être *continu* ou *discret* (c'est-à-dire $t = 1, 2, \ldots$). De même, l'ensemble des états possibles pour le processus peut être *continu* ou *discret* (c'est-à-dire $X_t = 1, 2, \ldots$) [Karlin 68].

Si on tire aléatoirement une valeur x_i de X_i pour chaque valeur de t, la séquence de valeur obtenues est appelée une trajectoire du système. Il représente une succession d'états que le système peut prendre au cours du temps.

Les différentes variables aléatoires ne sont en général pas indépendantes les unes des autres. Ce qui fait réellement l'intérêt des processus stochastiques est la dépendance entre les variables aléatoires.

Pour spécifier entièrement un processus stochastique, il suffit de spécifier :

1. la loi de probabilité de la première variable aléatoire X_1, qui spécifie donc l'état du processus lors de la première observation.
2. pour toute valeur de $t > 1$ la probabilité conditionnelle : $P[X_t = j | X_1 = i_1, \ldots, X_{t-1} = i_{t-1}]$.

En d'autres termes, un processus stochastique représente une évolution dans le temps d'une variable aléatoire.

On peut s'intéresser au comportement transitoire du processus, c'est-à-dire à l'évolution des fonctions de *distribution de probabilité* de la variable aléatoire X au cours du temps t. généralement, on s'intéresse plutôt au comportement stationnaire (c'est-à-dire à l'équilibre) qui est obtenu si la *distribution de probabilité* converge vers une distribution de probabilité unique indépendante du temps quand $t \longrightarrow \infty$. Il est noter que le régime permanent peut n'exister que sous certains condition et il faut soigneusement établir ces conditions quand on étudie un processus stochastique [Guikhman 80].

5.4 Processus Markovien

Soit un processus stochastique$(X_t)_{t\geq 0}$. Considérons des instants dans le temps $t_1 < t_2 < \cdots < t_n$. On suppose connaître l'état du processus à l'instant $t_i, i = 1, 2, \ldots, n$. l'état x_n est l'état courant et on s'intéresse à son état future x_{n+1} à l'instant t_{n+1} . Le processus $(X_t)_{t\geq 0}$ a la propriété Markovienne si :

$$P\left[X_{t_{n+1}} = x_{n+1} \middle| X_{t_n} = x_n, \ldots, X_{t_1} = x_1\right] = P\left[X_{t_{n+1}} = x_{n+1} \middle| X_{t_n} = x_n\right]$$

et ce quelque soit n et les instants considérés [Bouleau 88].

Autrement dit, cette équation signifie que le processus est Markovien si son état dans le future ne dépend que de l'état présent et pas des ses états passés. L'état courant du processus contient toute l'information permettant de caractériser son évolution.

Un processus Markovien à espace d'état discret est appelée une chaine de Markov.

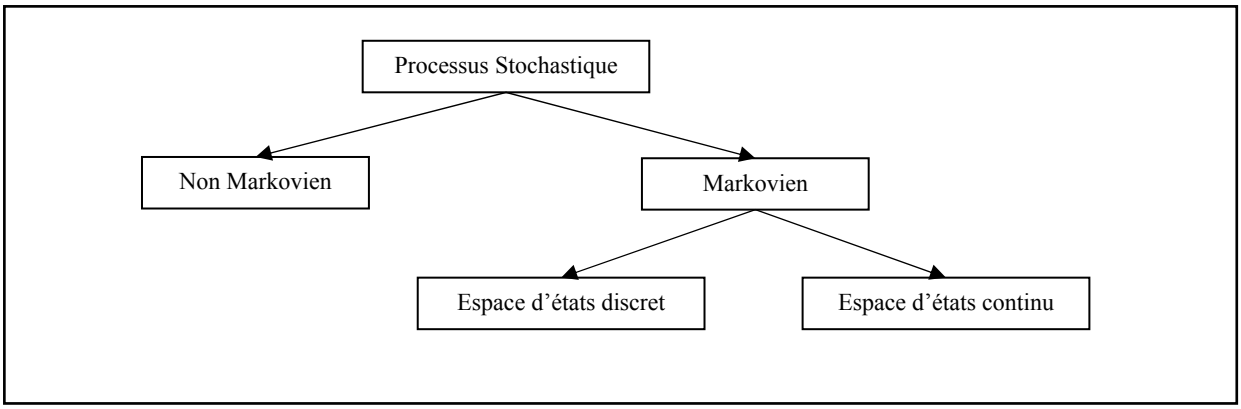


Figure 5.2 Table de Transition.

On s'intéresse dans la suite de ce chapitre particulièrement aux chaines de Markov à temps discret.

5.5 Chaine de Markov à temps discret

Les Chaînes de Markov sont un outil fondamental pour modéliser les systèmes dynamique qui évoluent dans le temps. Ces systèmes admettent un certain nombre d'états différents. L'état change au

cours du temps discret. A chaque changement, le nouvel état est choisi avec une distribution de probabilité fixée au préalable, et ne dépendant que de l'état présent.

5.5.1 Exemple introductif

Une souris se déplace dans le labyrinthe de la Figure 5.3 qui comporte 5 cases numérotées de 1 à 5. Initialement, elle se trouve dans la case N°1. A chaque minute, elle change de case en choisissant l'une des cases adjacentes de manière équiprobable. Dès qu'elle atteint soit la nourriture (la case N°4), soit sa tanière (la case N°5), elle y reste [Berglund 14].

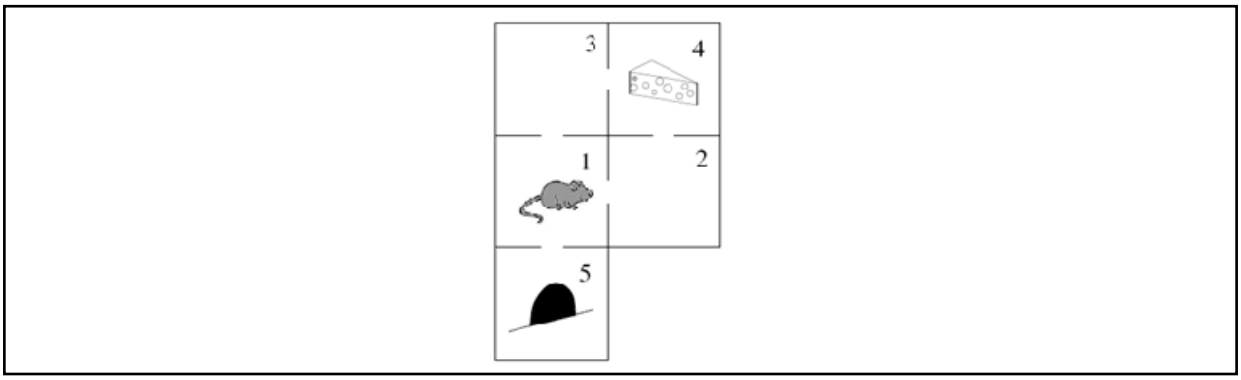


Figure 5.3 : La Souris dans le Labyrinthe.

On se pose alors les questions suivantes :

1. Avec quelle probabilité la souris atteint-elle la nourriture plutôt que sa tanière?
2. Au bout de combien de temps atteint-elle sa tanière ou la nourriture?

On peut essayer de répondre à ces questions en construisant un arbre décrivant les chemins possibles (voir la Figure 5.4).

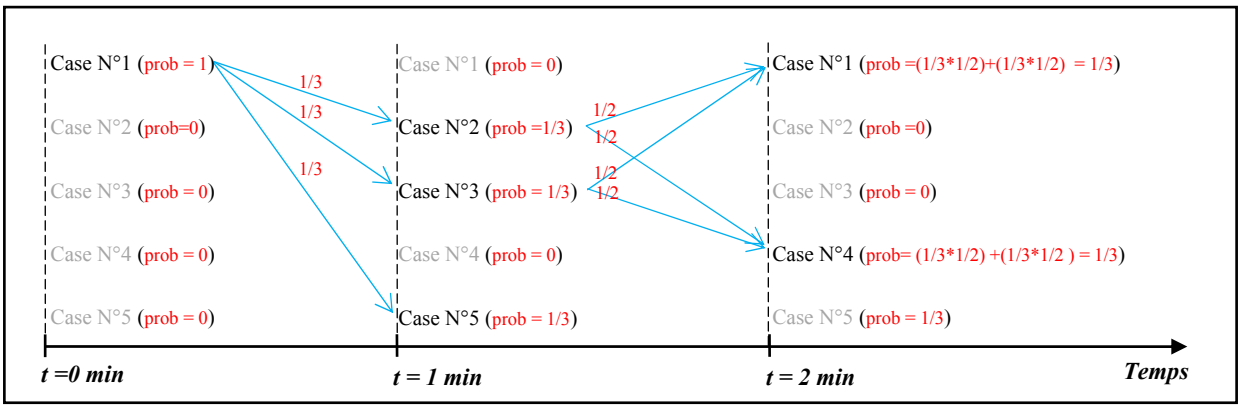


Figure 5.4 : L'Arbre de Probabilité de tous les Chemins Possibles.

Par exemple, il est clair que la souris se retrouve dans sa tanière au bout d'une minute avec probabilité 1/3. Sinon, elle passe soit dans la case 2, soit dans la case 3, et depuis chacune de ces cases elle a une chance sur deux de trouver la nourriture. Il y a donc une probabilité de 2/6 que la souris trouve la nourriture au bout de deux minutes. Dans les autres cas, elle se retrouve dans la case de départ, ce qui permet d'établir une formule de récurrence pour les probabilités cherchées.

Cette manière de faire est très compliquée, et devient rapidement impossible à mettre en œuvre quand la taille du labyrinthe augmente. Dans la suite, on va développer l'outil de Chaîne de Markov qui permet de résoudre ces types de problèmes.

5.5.2 Définition

Une Chaîne de Markov à temps discret un processus Markovien $(X_n)_{n=1,2,\dots}$ à temps discret ($n \in \mathbb{N}$) dont l'espace d'états est discret ($X_n \in \{0, 1, 2, \dots\}$) [Norris 97].

Remarque

- Le nombre d'états peut être fini.

5.5.3 Probabilités de Transition

Une Chaîne de Markov peut être caractérisée par ses probabilités de transition en une seule étape.

$$p_{ij} = P[X_{n+1} = j \,/\, X_n = i]$$

p_{ij} est la probabilité de passer de i à j en un pas de temps [Berglund 14]. On note que ces probabilités ne dépendent pas de l'instant n considéré. On dit dans ce cas que la Chaîne de Markov est *homogène* dans le temps.

5.5.4 Représentation

Une Chaîne de Markov à temps discret peut être représentée de deux façons :

5.5.4.1 Un Graphe Orienté

Les nœuds du graphe représentent les états que peut prendre le processus, et les arcs représentent les transitions entre états. Les poids des arcs correspondent aux probabilités de réaliser ces transitions. La Figure 5.5 présente le graphe de transition d'une Chaîne de Markov de l'exemple de la section 5.5.1.

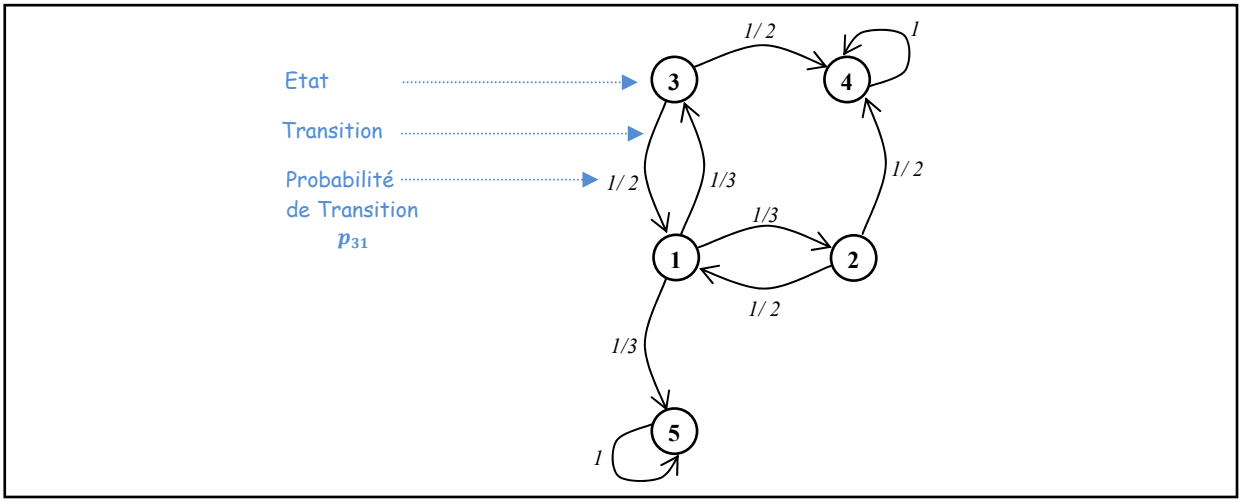


Figure 5.5: Graphe de Transition d'une Chaîne de Markov de l'Exemple Introductif.

Remarque

- La somme des poids des arcs sortant d'un nœud vaut 1.

5.5.4.2 Une Matrice des Probabilités de Transition

On appelle matrice des probabilités de transition la matrice carrée $P = [p_{ij}]$ d'ordre n (où n est le nombre d'éléments de l'espace d'états E),

$$\boldsymbol{P} = \begin{pmatrix} p_{00} & p_{01} & \cdots & p_{0n} \\ p_{10} & p_{11} & \cdots & p_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ p_{n0} & p_{n1} & \cdots & p_{nn} \end{pmatrix}$$

Les éléments de la matrice correspondent aux probabilités de transition entre états. La ligne i de cette matrice donne la probabilité de passer de l'état i vers tout autre état j (y compris l'état i lui-même) en un pas de temps. La somme des éléments d'une ligne est égale à 1 ($\sum_{j=1}^{n} p_{ij} = 1$). On parle alors de matrice *stochastique*.

Reprenons l'exemple de section 5.5.1, la matrice P est donnée par :

$$P = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 1/3 & 1/3 & 0 & 1/3 \\ 1/2 & 0 & 0 & 1/2 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Si le processus passe de l'état i à l'état j en deux étapes, c'est qu'il est passé par un état k (qui peut être i ou j). On a donc la probabilité $p_{ij}^{(2)}$ de passer de l'état i à l'état j en deux étapes :

$$p_{ij}^{(2)} = \sum_{k}^{n} p_{ik} p_{kj}$$

Il claire que $p_{ij}^{(2)}$ est l'élément (i, j) de la matrice $P^2 = PP$. On peut ainsi établir que les probabilités de transitions $p_{ij}^{(n)}$ en n étapes s'obtiennent avec $P^{(n)}$, c'est-à-dire en mettant la matrice P à la puissance n. Puisque on a $P^n = P^m P^{n-m}$, il vient :

$$p_{ij}^{(n)} = \sum_{k}^{n} p_{ik}^{m} p_{ik}^{n-m}, \quad 0 \leq m \leq n$$

Cette équation est connue sous le nom d'équation de *Chapman-Kolmogrov* [Norris 97]. Elle exprime que pour passer de i à j en n étapes, il faut passer par un état k à l'étape m.

5.5.5 Probabilités d'Etats

On note $\pi_i^{(n)} = P[X_n = i]$ la probabilité que la Chaîne de Markov soit dans l'état i à l'instant. Pour manipuler ces probabilités d'états, on utilise le vecteur $\pi^{(n)} = [\pi_0^{(n)}, \pi_1^{(n)}, \dots]$ dont la somme des termes vaut 1. Chaque Chaîne de Markov a un état initial. L'état initial d'une Chaîne de Markov est défini par le vecteur de probabilités : $\pi^{(0)} = [\pi_0^{(0)}, \pi_1^{(0)}, \dots]$ appelé la distribution de l'état initial.

Puisque la matrice P donne les probabilités de transition en une étape, on a :

$$\pi^{(n)} = \pi^{(n-1)} P$$

On peut déduire récursivement que :

$$\pi^{(n)} = \pi^{(n-1)} P = \pi^{(n-2)} P^2 = \cdots = \pi^{(0)} P^n$$

5.6 Classification des Etats

Il est évident que les probabilités de transition associées aux états jouent un rôle important dans l'étude des Chaînes de Markov. Pour décrire les propriétés des Chaînes de Markov, il est nécessaire de présenter quelques concepts et définitions concernant ces états. Les états se classifient en fonction de la possibilité que le processus les atteindre les uns à partir des autres [Norris 97].

État Accessible depuis un autre état

Soient i et j deux états de E. On dit que j est *accessible* depuis i si et seulement si il existe un entier $n \in \mathbb{N}$ et une suite d'états $k_0 = i, k_1, \dots, k_n = j$ tels que $p_{ik_1}\, p_{k_1k_2} \dots p_{k_{n-1}j} > 0$.

En d'autres termes, j est accessible depuis i si il existe un chemin dans le graphe de transition, partant de i et arrivant en j. Ceci se traduit également en termes des probabilités de transition en n pas et des probabilités de premier passage.

De façon générale, l'état j est accessible depuis i si et seulement si il existe n tel que $\exists\, n, p_{ij}^{(n)} > 0$.

États Communiquent

On dit que deux états i et j *communiquent* si l'état j est accessible de l'état i et l'état i est accessible de l'état j. La relation de communication est transitive, si l'état i communique avec l'état j et l'état j communique avec l'état k, alors l'état i communique avec l'état k.

Classe des états Communiquent

Les états peuvent être partitionnés en un ou plusieurs classes ou sous-ensemble d'états communiquent entre eux. Ces classes sont appelées les classes *irréductibles*.

Chaîne de Markov Irréductible

Si tous les états de E communiquent entre eux, E tout entier est la seule classe irréductible. On dit alors que la haîne est *irréductible*.

État Absorbant

Un état est dit *absorbant* si le processus ne peut pas sortir de cet état. Autrement, l'état i est absorbant si et seulement si $p_{ii} = 1$.

État transitoire

On dit que i est un état *transitoire* si après y avoir accéder, le processus peut ne plus y revenir. Autrement dit, un état i est transitoire si et seulement si il existe un état j ($j \neq i$) accessible de l'état i et j n'est pas accessible de l'état i. Par conséquent, un état transitoire est un état dans lequel on ne passe qu'un nombre fini de fois.

État récurent

On dit que i est un état *récurrent* si après y avoir accédé, le processus y reviendra. Autrement dit, un état i est récurrent si et seulement s'il n'est pas transitoire.

Remarque

- Tous les états d'une chaine de Markov irréductible sont récurent.

Période d'un état

On dit que l'état i est de période d_i si d_i égale au plus grand commun diviseur (P.G.C.D) de tous les n pour lesquels $p_{ii}^{(n)} > 0$.

$$d_i = PGCD\ (n \geq 1, p_{ii}^{(n)} > 0)$$

Si $d_i = d > 1$, on dit que i est *périodique* de période d.

Apériodique

L'état i est *apériodique* lorsque $d_i = d = 1$.

Ergodique

Un état récurrent et apériodique est dit *ergodique*. Une Chaîne de Markov est dite *ergodique* si tous ses états sont ergodiques.

Toutes ces définitions sont illustrées sur la Figure 5.6.

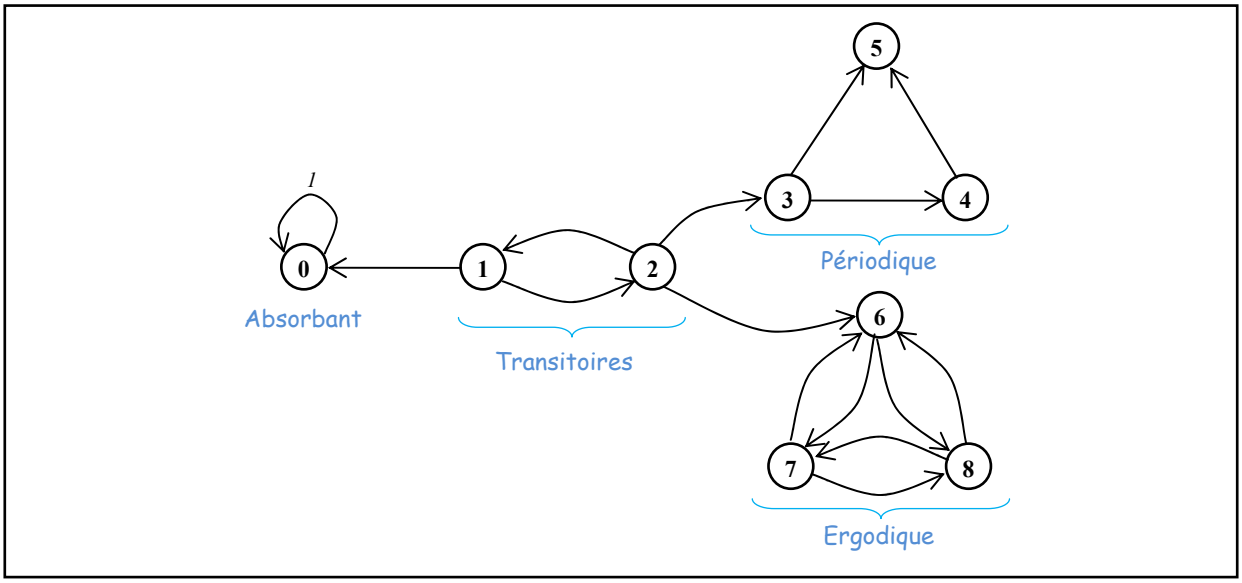


Figure 5.6: Classification des Etats d'une Chaîne de Markov.

5.7 Sémantique & Exécution

5.7.1 Régime Transitoire

L'analyse d'une chaine de Markov au régime transitoire consiste à déterminer le vecteur des probabilités d'états $\pi^{(n)}$ à l'instant n. Il est à noter que l'évolution de distribution $\pi^{(n)}$ dépend de la distribution initiale. Rappelons que :

$$\pi^{(n)} = \pi^{(n-1)}P = \pi^{(n-2)}P^2 = \cdots = \pi^{(0)}P^n$$

Dans l'exemple de la section 4.2.1, La matrice de transition à 2 pas est donnée par :

$$P^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 1/3 & 0 & 0 & 1/3 & 1/3 \\ 0 & 1/6 & 1/6 & 1/2 & 1/6 \\ 0 & 1/6 & 1/6 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Partant de l'état i à l'instant initial, les probabilités de transition en 2 étapes à partir de cet état est donnée par la ligne i de P^2. Ainsi, si à l'instant initial la chaine de Markov est dans l'état 1 de façon certaine ($\pi^{(0)} = [1, 0, 0, 0, 0]$), alors on sait que la distribution de probabilités des états à l'instant 2 est donnée par :

$$\pi^{(2)} = \pi^{(0)}p^2 = [\tfrac{1}{3}, 0, 0, \tfrac{1}{3}, \tfrac{1}{3}] \ .$$

La matrice de transition à 3 pas est donnée par :

$$P^3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 1/9 & 1/9 & 1/3 & 4/9 \\ 1/6 & 0 & 0 & 2/3 & 1/6 \\ 1/6 & 0 & 0 & 2/3 & 1/6 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

La distribution de probabilités des états à l'instant 3 est donnée par :

$$\pi^{(3)} = \pi^{(0)} p^3 = [0, \tfrac{1}{9}, \tfrac{1}{9}, \tfrac{1}{3}, \tfrac{4}{9}] \ .$$

Si on continue le calcule jusqu'à l'ordre 8 (8 transitions), on a

$$P^8 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 1/81 & 0 & 0 & 40/81 & 40/81 \\ 0 & 13/1296 & 13/576 & 85/108 & 85/324 \\ 0 & 13/1296 & 13/576 & 85/108 & 85/324 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

La distribution de probabilités des états à l'instant 8 est donnée par :

$$\pi^{(8)} = \pi^{(0)} p^8 = [\tfrac{1}{81}, 0, 0, \tfrac{40}{81}, \tfrac{40}{81}]$$

5.7.2 Régime Stationnaire

Après un grand nombre de transitions ($n \longrightarrow \infty$), la probabilité que la Chaîne de Markov soit dans un état donné est constante et ne dépend pas de l'état initial. On dit que la Chaîne de Markov atteint le régime stationnaire ou un état d'équilibre. La distribution des états dans ce régime est la loi stationnaire ou loi limite. On note cette loi par π tel que :

$$\pi = \lim_{n \to \infty} \pi^{(n)}$$

, et les probabilités d'états sont constantes d'une transition à l'autre, donc :

$$\pi = \pi \ P$$

Il est noter que le régime stationnaire n'est pas atteint par tous les systèmes, il y a des systèmes qui n'atteindront jamais l'état stationnaire. Dans ce cas, soit y a un problème quelque part dans le système, soit le système en question dépend d'autre facteurs qui changent a chaque fois.

Théorème de Kolmogrov

Pour une Chaîne de Markov irréductible et (apériodique) ergodique, la distribution stationnaire

$$\pi = \lim_{n \to \infty} \pi^{(n)}$$

existe et est indépendante de l'état initial. La distribution stationnaire π peut être calculée en résolvant le système d'équations suivant [Norris 97] :

$$\begin{cases} \pi P = \pi \\ \sum_{i \in E} \pi_i = 1 \end{cases}$$

Remarque

- Si au moins une des puissances de la matrice P de la chaîne de Markov n'a que des termes strictement positifs, alors la les conditions du théorème de Kolmogrov sont les mêmes.

Il est clair que la Chaîne de Markov de l'exemple de la section 4.2.1 ne vérifie pas les conditions du théorème de Kolmogrov. La Chaîne de Markov n'est pas irréductible, les états 4 et 5 sont Absorbants.

Exemple

Considérons une Chaîne Markov définie par la matrice de transition P et le graphe de transition de la Figure 5.7 :

$$P = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} 1/2 & 0 & 1/2 \\ 1/4 & 1/2 & 1/4 \\ 1/3 & 1/3 & 1/3 \end{pmatrix} \end{array}$$

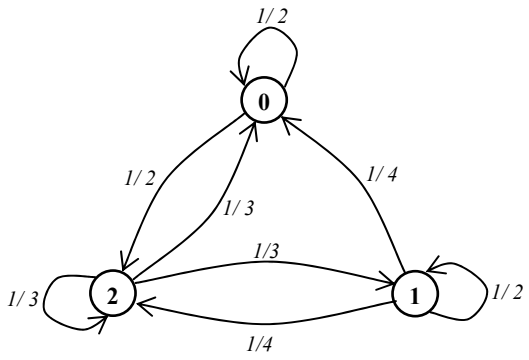


Figure 5.7 : Graphe de Transition d'une Chaîne de Markov.

Tous les états communiquent, donc la Chaîne est irréductible. Tous les états sont de manière évidente de période 1 , donc la chaine est apériodique. D'après le théorème de Kolmogrov, on peut dire qu'il existe une distribution stationnaire π. π est l'unique solution du système :

$$\begin{cases} \pi P = \pi \\ \sum_{i \in E} \pi_i = 1 \end{cases}$$

$$\begin{cases} \frac{1}{2}\pi_0 + \frac{1}{4}\pi_1 + \frac{1}{3}\pi_2 = \pi_0 \\ 0\,\pi_0 + \frac{1}{2}\pi_1 + \frac{1}{3}\pi_2 = \pi_1 \\ \frac{1}{2}\pi_0 + \frac{1}{4}\pi_1 + \frac{1}{3}\pi_2 = \pi_2 \\ \pi_0 + \pi_1 + \pi_2 = 1 \end{cases} \Rightarrow \begin{cases} -\frac{1}{2}\pi_0 + \frac{1}{4}\pi_1 + \frac{1}{3}\pi_2 = 0 & \text{..... (1)} \\ 0\,\pi_0 - \frac{1}{2}\pi_1 + \frac{1}{3}\pi_2 = 0 & \text{..... (2)} \\ \frac{1}{2}\pi_0 + \frac{1}{4}\pi_1 - \frac{2}{3}\pi_2 = 0 & \text{..... (3)} \\ \pi_0 + \pi_1 + \pi_2 = 1 & \text{..... (4)} \end{cases}$$

On a :

$$(3) - (1) \Rightarrow \pi_0 - \pi_2 = 0 \Rightarrow \pi_2 = \pi_0 \qquad \text{..... (5)}$$

On remplace *(5)* dans *(3)*, on obtient:

$$\frac{1}{2}\pi_0 + \frac{1}{4}\pi_1 - \frac{2}{3}\pi_0 = 0 \Rightarrow \frac{1}{4}\pi_1 = \frac{1}{6}\pi_0 \Rightarrow \pi_1 = \frac{4}{6}\pi_0 \qquad \text{..... (6)}$$

On remplace *(5)* et *(6)* dans *(4)*, on obtient:

$$\pi_0 + \frac{4}{6}\pi_0 + \pi_0 = 1 \Rightarrow \frac{16}{6}\pi_0 = 1 \Rightarrow \pi_0 = \pi_2 = \frac{6}{16} = \frac{3}{8}$$

On a donc :

$$\pi_1 = \frac{4}{6}\frac{3}{8} = \frac{1}{4}$$

La distribution stationnaire $\pi = [\frac{3}{8}, \frac{1}{4}, \frac{3}{8}]$.

5.8 Chaîne de Markov Absorbante

Une Chaîne de Markov est dite absorbante s'il existe au moins un état absorbant et s'il on peut passer de n'importe quel état à un état absorbant.

$$\forall\, i \in E, \exists\, k \text{ absorbant}, k \text{ est accessible depuis } i$$

Lorsqu'on a affaire à une Chaîne de Markov absorbante, on est généralement intéressé par les deux questions suivantes :

- ➔ Combien de temps faudra-t-il pour que le processus soit absorbé, étant donné son état initial ? On appelle n_i le temps moyen jusqu'à l'absorption en partant de i.
- ➔ S'il existe plusieurs états absorbants, quelle est la probabilité pour un processus d'être absorbé par un état donné. On appelle b_{ij} la probabilité que le processus soit absorbé dans j si son état initial est i.

Théorème 1

Les quantités n_i sont solution du système d'équations:

$$n_i = 1 + \sum_{k \in S'} p_{ik}\, n_k$$

où i est un état non absorbant et S' l'ensemble de tous les états non absorbants.

Théorème 2

Soit j un état absorbant et S' l'ensemble de tous les états non absorbants. Alors les probabilités $b_{ij}(i \in S')$ sont solution du système d'équations :

$$b_{ij} = p_{ij} + \sum_{k \in S'} p_{ik}\, b_{kj}$$

Exemple

Une certaine pièce d'équipement électrique peut se trouver dans 3 états :

1: *bonne*, 2 : *condition marginale*, 3 : *défaillante*.

A la fin de chaque jour de service, l'état de la pièce est enregistré. La matrice de transition obtenue est :

$$P = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 0.7 & 0.2 & 0.1 \\ 0.5 & 0.3 & 0.2 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

Calculer la durée de vie moyenne d'une pièce se trouvant initialement en bonne condition ?

Réponse

La durée de vie moyenne d'une pièce est quantité n_1 de la solution du système :

$$n_i = 1 + \sum_{k \in S'} p_{ik}\, n_k$$

$$\begin{cases} n_1 = 1 + 0.7\, n_1 + 0.2\, n_2 \\ n_2 = 1 + 0.5\, n_1 + 0.3\, n_2 \end{cases} \Rightarrow \begin{cases} 0.3\, n_1 - 0.2\, n_2 = 1 & \text{..... (1)} \\ -0.5\, n_1 + 0.7\, n_2 = 1 & \text{..... (2)} \end{cases}$$

On a :

équation (1) $*$ 0.7 $+$ équation (2) $*$ 0.2 .

On obtient $0.11\, n_1 = 0.9$

On a donc la durée de vie moyenne d'une pièce $n_1 =$ **8.18** Jours.

5.9 Exercices

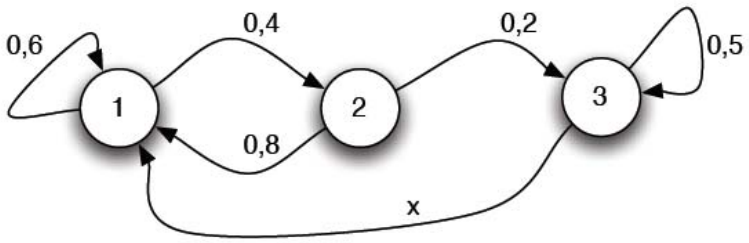
Exercice N° 1 (fiabilité de deux éléments en parallèle)

Soit un dispositif technique comprenant deux éléments montés en parallèle et fonctionnant indépendamment l'un de l'autre. Chaque élément a une fiabilité égale à p au cours d'une journée (c'est-à-dire. qu'il a une probabilité de $1-p$ de tomber en panne). Il n'y a pas de possibilité de réparation. Si X_n est le nombre d'éléments en panne au début de la $n^{ième}$ journée.

1. Décrire la Chaîne de Markov correspondante, sa matrice et son graphe de transition.
2. Calculer la probabilité p en fonction de n ? $AN : p = 0{,}9$.
3. Modifier la Chaîne précédente en stipulant qu'une machine en panne sera réparée au cours de la journée suivante ?

Exercice N° 2

Soit la Chaîne de Markov définie par son graphe de transition :



1. Dans le graphe de transition de la Chaîne de Markov, une des probabilités a été remplacée par x. Quelle est la valeur de x ?
2. Donner la matrice de transition de cette Chaîne de Markov ?
3. Si l'état initial est $\pi^{(0)} = [\,1,\ 0,\ 0\,]$, donner les probabilités de présence dans chaque état au pas 1 et au pas 2 ?
4. Si elle existe, calculer la distribution stationnaire de cette Chaîne de Markov ?
5. Calculer le nombre de pas moyen pour aller pour la première fois dans l'état 3 ?

Exercice N° 3

Les météorologistes d'un pays ont établi un tableau qui donne la probabilité du temps du lendemain en fonction de la météo du jour.

météo du lendemain / météo du jour	beau	couvert	pluie	orageux
beau	0.7	0.2	0.0	0.1
couvert	0.2		0.4	0.0
pluie	0.6	0.0	0.4	
orageux	0.6	0.0	0.3	

1. Quelques valeurs manquent. Compléter ce tableau ?
2. Si on suppose que ce tableau reflète ce qui se passe vraiment, expliquer pourquoi on peut modéliser la météo par une Chaîne de Markov à temps discret ?
3. Faire un schéma de cette Chaîne de Markov ?
4. Si elle existe, calculer la distribution stationnaire de cette Chaîne de Markov ?
5. Sur 365 jours, donner le nombre moyen de jours de beau temps, de jours de temps couvert, de jour de pluie et de jours orageux ?
6. Combien de jours en moyenne pour avoir une journée orageuse la première fois, sachant qu'il fait beau initialement?

Exercice N° 4

Un fumeur décide d'arrêter de fumer. Le premier jour suivant cette décision (jour 0), il ne fume pas. On suppose que la probabilité qu'il fume le jour $j+1$ s'il n'a pas fumé le jour j est α, et que la probabilité qu'il ne fume pas le jour $j+1$ s'il a fumé le jour j est β, Avec : $0 < \alpha, \beta \leq 1$.

1. Expliquer pourquoi on peut modéliser le système par une Chaîne de Markov à temps discret ?
2. Commenter la modélisation de ce problème par une Chaîne de Markov; donner son graphe et sa matrice de transition ?
3. Calculer la probabilité qu'il ne fume pas les jours 3 et 4 ?
4. Si elle existe, calculer la distribution stationnaire de cette Chaîne de Markov ?

Bibliographie

Bibliographie

[AFCET 77] Association française de cybernétique économique et technique, "Modélisation et maîtrise des systèmes techniques économiques sociaux", Paris, Ed. Hommes et Techniques, 1977.

[Aissani 07] A. Aissani, "Modélisation et Simulation", Office des Publications Universitaires, USTHB Alger, Algérie, 2007.

[Alur 94] R. Alur et D. Dill, "A theory of automata", Theoretical Computer Science Journal, Vol. 2 N°126, 1994.

[Arnold 92] A. Arnold, "Systèmes de transitions finis et sémantique des processus communicants", Editions Masson, 1992.

[Bank 98] J. Bank, "Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice" , Wiley Interscience, 1998.

[Berglund 14] N. Berglund, "Processus aléatoires et applications", Master 2 Pro de Mathématiques, Université d'Orléans, 2014.

[Bouleau 88] N. Bouleau, "Processus Stochastiques et Applications", Hermann, Paris, 1988.

[Brams 83] G.W. Brams, "Réseaux de Petri : Théorie et pratique", Masson, 1983.

[David 92] R. David et H. Alla, "Du Grafcet aux reseaux de Petri", Hermes, 1992.

[Feller 60] W. Feller, "An introduction to probability theory and its applications", John Wiley and sons, 1960.

[Guikhman 80] I. I. Guikhman et A. V. Skorokhod, "Introduction à la Théorie des Processus Aléatoires", Editions MIR, Moscou, 1980.

[Harel 87] D. Harel, "Statecharts : a Visual Formalism for Complex Systems", Science of Computer Programming, 1987.

[Hopcroft 01] J. Hopcroft, R. Motwani et J. Ullman, "Introduction to Automata Theory, Languages and Computation", 2nd edition, Addison-Wesley, 2001.

[Karlin 68] S. Karlin , "Initiation aux processus stochastique", Dunod, 1968.

[Law 91] A.M. Law, W.D. Kelton, "Simulation Modeling and Analysis" Editions McGraw-Hill, 2nd edition, 1991 .

[Murata 89] T. Murata, "Petri Nets: Properties, Analysis and Applications", IEEE, Vol. 77, 1989.

[Norris 97] J. R. Norris, "Markov Chains", Cambridge University Press, 1997.

[Pritsker 87] A.A.B. Pritsker, "Compilation of definitions of Simulation", Simulation, Vol.40, n°8, 1987.

[Vidal 92] G. Vidal-Naquet et A. Choquet-Geniet, "Reseaux de Petri et Systèmes Parallèles", Armand Colin, 1992.

[Wolper 91] P. Wolper. "Introduction à la calculabilité", InterEditions, 1991.