République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A/Mira de Bejaia
Faculté des Sciences Exactes
Département d'Informatique

Support de cours

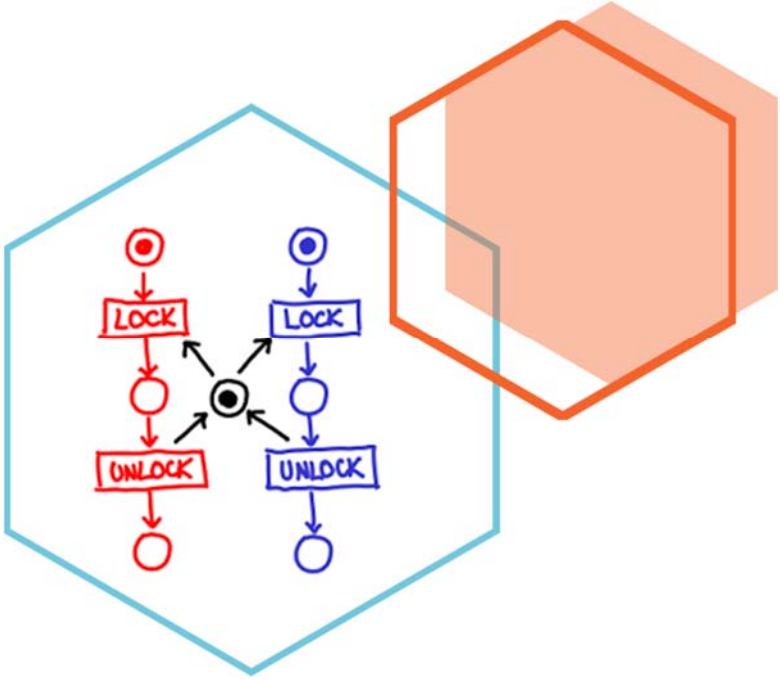
Préparé par :

Dr. Djamila Boukredera épouse Boulahrouz

Les Réseaux de Petri

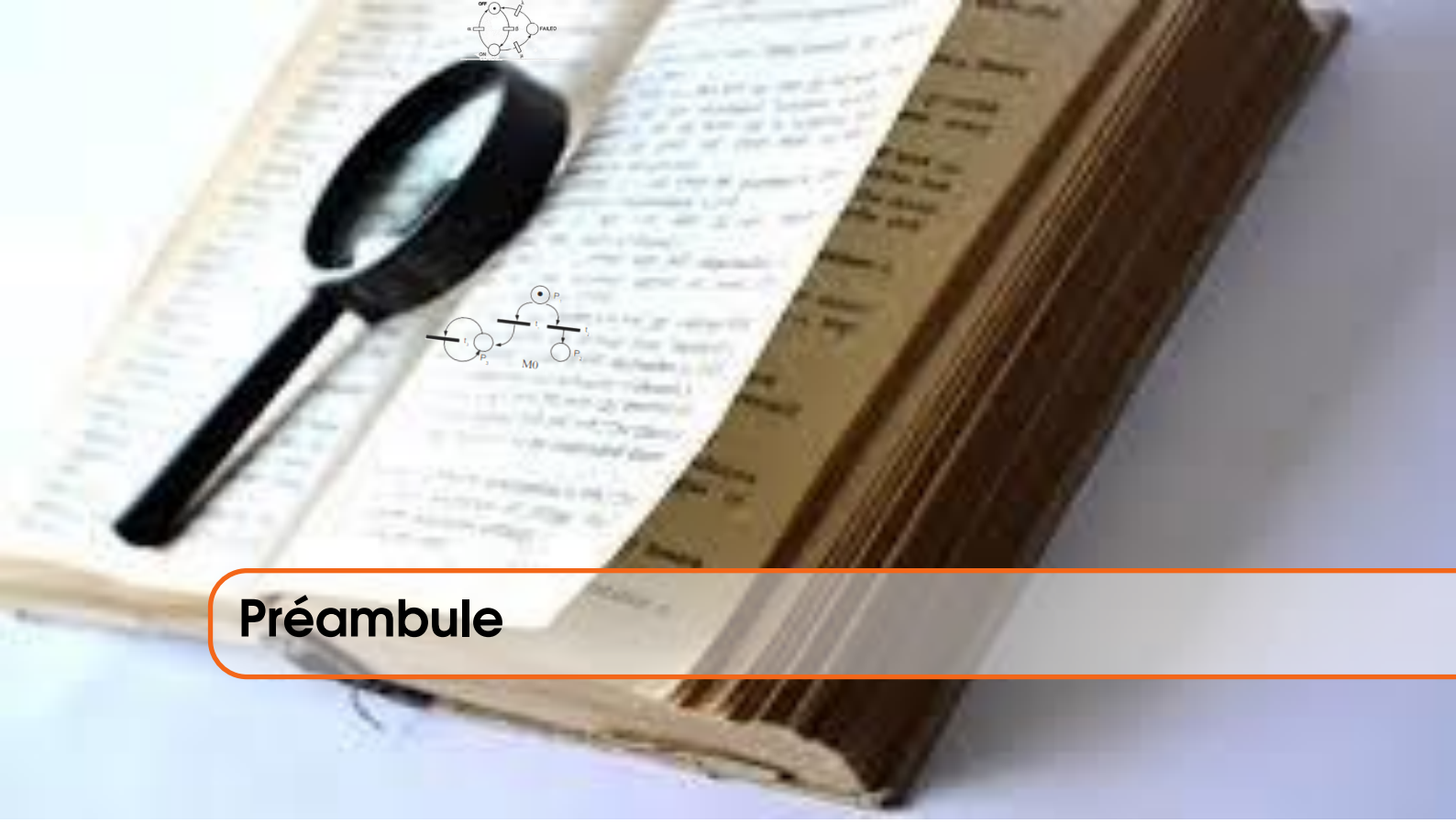
Un outil de Modélisation et d'Analyse des Systèmes Dynamiques à Evènements Discrets

Cours destiné aux étudiants de Master 2 Informatique option: Intelligence Artificielle



2019-2020

Table des Matières



Préambule

Avec les grands progrès technologiques, les systèmes informatiques actuels reposent sur des architectures de plus en plus complexes et de plus en plus exigeantes en terme de qualité. De tels systèmes nécessiteraient une spécification plus précise et une meilleure maîtrise de leur processus de conception. Les concepteurs devront alors aborder plusieurs problèmes importants : comment réussir à appréhender le comportement de ces systèmes afin de les concevoir, les réaliser et/ou les commander ? Répondre à ces interrogations impliquerait le recours à des méthodes de conception et de réalisation qui soient particulièrement efficaces. Au centre de ces méthodes, se trouve en général la **modélisation**. Les approches de modélisation doivent acquérir une connaissance approfondie du système, y compris de ses comportements locaux et de ses communications, sur la base d'une représentation bien définie de l'architecture conçue. Cette représentation doit être utilisée le plus tôt possible pour analyser et valider la conception.

L'**objectif de ce cours** est de présenter **la modélisation par les réseaux de Petri** et ses versions étendues. En effet, les réseaux de Petri, inventés par *Carl Adam Petri* au début des années soixante, sont une représentation mathématique graphique définis par une sémantique simple et claire. Ils permettent une modélisation facile des mécanismes clés de classes importantes de systèmes dynamiques à évènements discrets qui recouvrent des classes de systèmes de production, de systèmes automatisés, de systèmes informatiques et de systèmes de communication, etc., afin de permettre leur conception, leur évaluation et leur dimensionnement. Ils sont soutenus par diverses méthodes et outils informatiques d'analyse solides.

Les réseaux de Petri ordinaires et de haut niveau sont présentés dans ce document de façon détaillée et étayée par de nombreux exemples et exercices pratiques qui aident à bien cerner le formalisme. Enfin, une brève description de la plateforme CPN Tools est présentée pour permettre aux étudiants d'implémenter leurs modèles et d'évaluer leurs systèmes.

Ce support de cours est destiné, en premier lieu, aux étudiants de niveau *Master 2 Informatique option Intelligence Artificielle*, mais peut aussi servir pour des lecteurs désireux d'apprendre les principes essentiels et les démarches nécessaires à l'élaboration de modèles notamment de réseaux de Petri. En second lieu, Ce document peut être utile pour des étudiants de niveau *Master 2 option « Réseaux et Systèmes Distribués »*, *Master 2 « Automatisme »* de la faculté de Technologie ainsi

que pour le niveau *Master en Recherche Opérationnelle*. L'assimilation de ce cours permet à l'étudiant d'acquérir une bonne maîtrise des réseaux de Petri et de développer une méthodologie pour l'élaboration de modèles dédiés aux systèmes particuliers auxquels il sera confronté.

Ce document est structuré autour de quatre chapitres.

Chapitre 1- Systèmes Dynamiques à Evènements Discrets

Dans ce chapitre, nous commençons par présenter une brève introduction à la théorie des systèmes et la nécessité de leur modélisation pour enfin se focaliser sur la classe des systèmes dynamiques à évènements discrets (SED). Ces derniers font l'objet de la modélisation par réseaux de Petri décrite dans ce cours. C'est pour cette raison qu'une partie importante de ce premier chapitre leur est consacrée. En effet, nous donnons tout d'abord une définition informelle des SED suivie d'une définition formelle moyennant les automates à états finis. Ensuite, nous mettons en évidence leurs utilités, leurs nécessités et les modèles mathématiques et graphiques. La dernière partie de ce chapitre est entièrement dédiée à l'évaluation qualitative et quantitative des modèles conçus où nous mettons l'accent sur les méthodes analytiques et l'approche par simulation.

Chapitre 2- Notions de Base sur les Réseaux de Petri

L'objet de ce deuxième chapitre est d'introduire les réseaux de Petri Places/Transitions communément appelés réseaux de Petri (RdP). Après avoir clarifié leur intérêt, nous présentons les concepts de base, le vocabulaire ainsi que les définitions fondamentales des RdP. Ensuite, nous abordons le fonctionnement dynamique de ces réseaux, le graphe de marquages accessibles ainsi que le calcul du graphe de couverture dans le cas d'un graphe infini. Un autre volet de ce chapitre est consacré à la représentation mathématique des RdP. Ainsi, la représentation matricielle et l'utilisation de l'équation fondamentale qui gouverne la dynamique du modèle sont exposées et expliquées par l'exemple. Enfin, une série d'exercices, en plus des exemples et des exercices d'application, termine ce chapitre.

Chapitre 3- Eléments de Modélisation et Propriétés des RdP

Dans ce chapitre, nous commençons d'abord par présenter les éléments nécessaires à la modélisation de différents comportements tels que le parallélisme, le conflit, l'exclusion mutuelle, la contrainte de capacité limitée, etc. Ensuite, nous exposons les propriétés structurelles et comportementales qui peuvent être vérifiées à l'aide des RdP, à savoir les blocages, la vivacité, la bornitude, l'accessibilité, etc. Nous définissons également la notion d'invariant de marquage de places et les composantes conservatives qui sont très importantes pour l'analyse du modèle de manière algébrique.

Chapitre 4- Réseaux de Petri de Haut Niveau

Ce chapitre a pour objectif de présenter les RdP de haut niveau, notamment les RdP colorés, hiérarchiques, temporisés et stochastiques. En premier lieu, nous exposons les différents problèmes qui ont motivé chacune des extensions proposées dans la littérature. Ensuite, nous détaillons chaque type de RdP en définissant les nouveaux concepts qu'il introduit et en décrivant son fonctionnement dynamique (comportement). Une attention particulière sera donnée aux RdP temporisés où nous présentons les différentes techniques d'introduction du temps dans ces réseaux. Enfin, nous présentons l'outil CPN Tools, qui permet de manipuler les réseaux de Petri de haut niveau et que les étudiants pourraient utiliser pour implémenter leurs modèle RdP.

1. Systèmes Dynamiques à Evènements Discrets

1.1 Introduction à la Théorie des Systèmes

Depuis plusieurs décennies la science et l'ingénierie s'intéressent à la notion de systèmes aussi bien pour mieux les comprendre que pour mieux les construire.

1.1.1 Qu'est ce qu'un Système ?

Par définition, un système est un ensemble de composants, élémentaires ou pas, en interaction entre eux et avec l'environnement pour la réalisation d'une fonction (ou d'une tâche) qui ne peut être effectuée par aucun de ces composants pris individuellement. Un système est connecté au monde extérieur à travers (voir Figure 1.1):

- **Ses entrées :**
  - Signaux d'excitation ou stimuli représentant les actions envoyées au système;
  - Des perturbations qui sont en général imprévisibles;
- **Ses sorties :** Réponses du système aux signaux d'entrée;

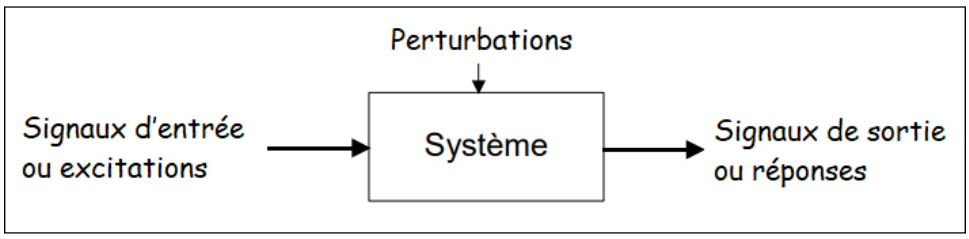


Figure 1.1: Interaction d'un système avec son environnement.

On peut, par exemple, citer les systèmes de production (ateliers flexibles, lignes d'assemblage, etc.), les réseaux informatiques et de télécommunications (systèmes multi-processeurs, réseaux d'ordinateurs, etc.), les réseaux de transport (routier, ferroviaire ou aérien), les systèmes informatiques, etc. L'enchaînement dynamique des tâches provient essentiellement de phénomènes telles que la synchronisation et l'exclusion mutuelle ou la compétition dans l'utilisation de ressources

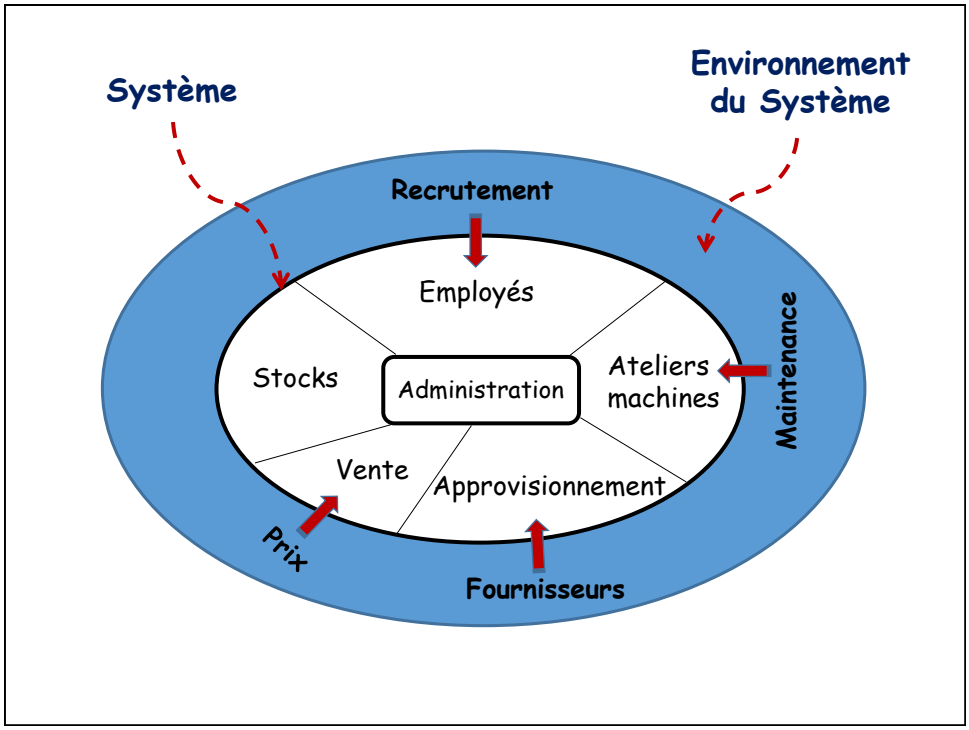


Figure 1.2: Exemple d'un système de production.

communes où une politique d'arbitrage ou de priorité s'avère nécessaire.
Un système est par conséquent déterminé par:

- **Sa frontière :** définie par le critère d'appartenance au système déterminant si une entité appartient au système ou à son environnement.
- **Sa mission :** la fonction ou la tâche pour laquelle il a été construit.
- **La définition des lois d'interaction** entre ses composants ainsi que leurs propriétés et les valeurs qu'ils peuvent prendre.

■ **Exemple 1.1** Dans cet exemple, nous considérons un système de production composé des parties suivantes :

- Administration
- Employés
- Service production (Ateliers, Machines, etc.)
- Service approvisionnement
- Service ventes
- Service de gestion de stocks
- etc.

Les parties du système de production illustré par la Figure 1.2 sont partiellement indépendantes mais interagissent entre elles afin d'atteindre l'objectif tracé. ■

1.1.2 Modélisation des Systèmes

Avec les progrès de la technologie, les systèmes informatiques actuels sont de plus en plus distribuées et reposent sur des architectures de plus en plus complexes. Cette complexité sans cesse grandissante est due principalement à la multiplicité des éléments qui composent ces systèmes, au fort degré d'interaction entre les différents composants qui le constituent ainsi qu'à l'incertitude concernant l'évolution globale du système. Ces systèmes se caractérisent essentiellement par leur

propension à avoir un très grand nombre de comportements possibles. Ainsi, pour maîtriser la complexité de tels systèmes et appréhender leurs comportements, les concepteurs sollicitent de plus en plus des activités de modélisation, de vérification et de validation.

La modélisation est la représentation d'un système par un autre, plus facile à appréhender. Il s'agit de concevoir un modèle qui est une approximation de la réalité par un formalisme mathématique. Ceci nous permet d'avoir un niveau d'abstraction élevé et une représentation du système, souvent simplifiée, qui le décrit sans équivoque tout en préservant les propriétés désirées.
La modélisation permet ainsi de visualiser le système, de capturer ses propriétés essentielles, de le simuler, d'analyser son comportement et de vérifier ses propriétés. Elle peut se substituer à l'expérimentation lorsqu'elle celle-ci est coûteuse ou trop difficile à réaliser. Elle permet ainsi un gain en temps, en coût et en complexité de développement et de maintenance des systèmes complexes.

En résumé, la modélisation est la conception d'un modèle qui doit représenter de manière plus ou moins fidèle au système réel tout en étant assez complet mais simple pour faciliter son analyse.

1.1.3 Classification des Systèmes Dynamiques

Il existe plusieurs types de systèmes tels que les systèmes statiques, dont la réponse à une excitation est instantanée, et les systèmes dynamiques dont la réponse est fonction de l'excitation et des réponses antérieures. Dans ce cours, on s'intéressera uniquement aux systèmes dynamiques.

1.1.3.1 Définition d'un Système Dynamique

Définition 1.1 En mathématiques, en chimie ou en physique, un système dynamique est la donnée d'un système et d'une loi décrivant l'évolution de ce système. C'est un système qui évolue dans le temps de manière :
- Causale : c'est à dire que son avenir ne dépend que de phénomènes du passé ou du présent; il dépend des conditions antérieures et de son évolution dans le temps;
- Déterministe : c'est-à-dire qu'à une « condition initiale » donnée à l'instant « présent » va correspondre à chaque instant ultérieur un et un seul état « futur » possible. Mais cela ne signifie pas qu'il soit nécessairement prédictible ni que l'avenir du système découle simplement des conditions du passé.

La dynamique d'un système traduit ainsi une abstraction du système et indique qu'il peut se trouver dans différents états stables. Elle est définie par les réactions du système qui dépendent de son état courant et des stimuli qui peuvent le faire passer d'un état à un autre. Notons que l'état d'un système est défini par un certain nombre de variables que l'on appelle **variables d'état** permettant de décrire son évolution à chaque instant.

1.1.3.2 Classification des Systèmes Dynamiques

Dans les systèmes dynamiques, le temps est un facteur essentiel dans l'évolution de l'état d'un système, c'est-à-dire ses variables d'état changent avec le temps. On distingue 3 types de systèmes dynamiques: continus, discrets et hybrides (voir Figure 1.3).

- **Systèmes dynamiques continus :** Ce sont des systèmes où les variables qui définissent leurs états sont continues, dans le sens que leurs valeurs évoluent de façon continue dans le temps (voir Figure 1.4). Par exemple, la pression atmosphérique, la position d'une planète dans son orbite, la vitesse, l'accélération, la température, etc. Un système dynamique continu est généralement décrit par un système d'équations algébriques ou différentielles.

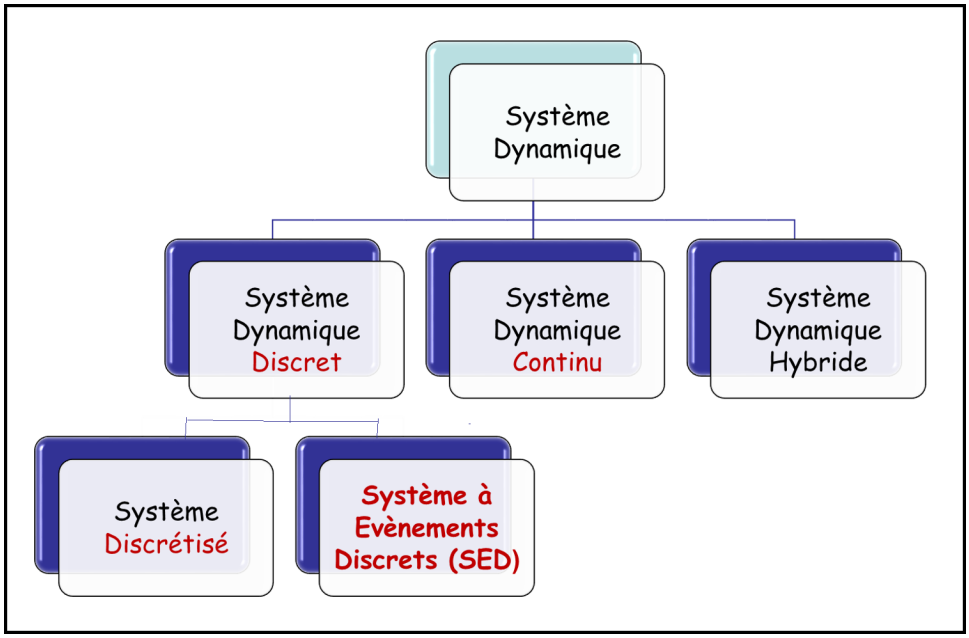


Figure 1.3: Classification des systèmes dynamiques.

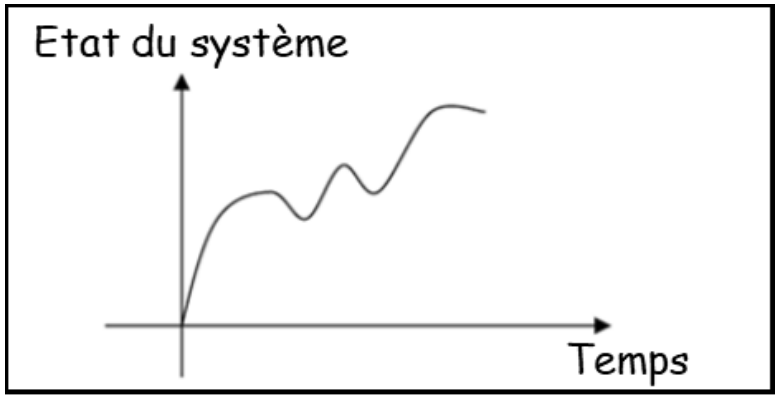


Figure 1.4: Système dynamique continu.

- **Systèmes dynamiques discrets :** Dans ce type de systèmes, l'état ne change qu'à certains instants. Le mot « discret » ne signifie ni « temps discret », ni « état discret », mais réfère au fait que l'espace d'états est décrit par un ensemble discret. Les systèmes discrets évoluent de manière discrète dans le temps, c'est à dire entre deux instants t_i et t_{i+1}, l'état du système n'évolue pas (voir Figure 1.5). Par exemple, dans un système de file d'attente, les variables d'état telle que la taille de la file d'attente changent de façon discrète. Les systèmes dynamiques discrets peuvent être classés en deux catégories:
  - **Systèmes discrétisés** : l'observation de l'état se fait à des instants réguliers. Ils correspondent à une représentation discrète des systèmes continus.
  - **Systèmes à événements discrets** : l'état du système n'est modifié que lors de l'occurrence de certains événements. C'est à ce type de systèmes qu'on s'intéressera dans la suite de ce cours.
- **Systèmes dynamiques hybrides :** Les systèmes réels sont souvent des systèmes complexes dont la dynamique est modélisée par des phénomènes discrets et continus. Un système dynamique hybride est composé de deux parties (partie continue, partie discrète). Les dynamiques

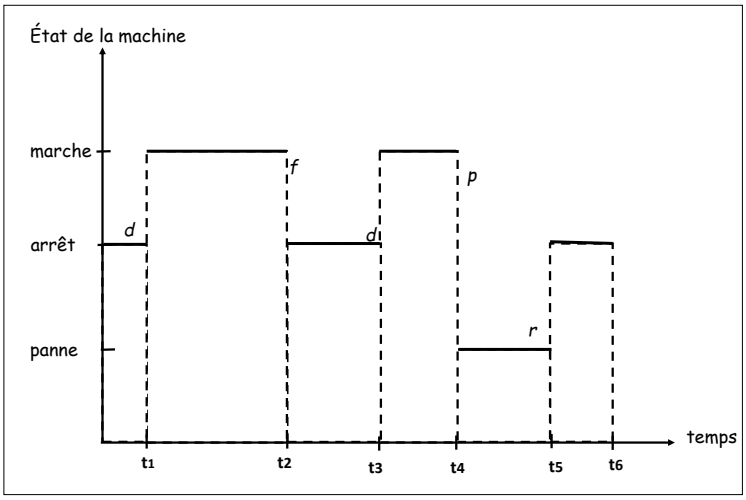


Figure 1.5: Système dynamique discret : cas de l'évolution de l'état d'une machine.

continues sont modélisées par des équations différentielles, alors que les phénomènes discrets par un automate d'états finis. Le modèle obtenu est appelé alors un automate hybride.

1.2 Systèmes à Evènements Discrets (SED)

Les systèmes à évènements discrets sont des systèmes logiques qui changent d'état à l'occurrence d'un évènement. Ils font partie de la famille des systèmes séquentiels dont les réponses vont dépendre du temps.

1.2.1 Définitions

Dans ce paragraphe, nous allons commencer par présenter de façon informelle une première définition des SED, puis nous exposerons sa définition formelle.

Définition 1.2 Les systèmes à événements discrets (SED) sont des systèmes dynamiques fondamentalement asynchrones qui ont un espace d'états discret, c'est à dire que les états sont dénombrables en nombre fini ou infini, et dont l'évolution est conforme à l'occurrence d'événements physiques discrets asynchrones à des intervalles de temps éventuellement irréguliers ou inconnus. La trajectoire des états est constante par morceaux [6, 11].

Un évènement peut être identifié par une action spécifique (une personne qui a pressé un bouton ou appuyé sur une touche du clavier), un début ou une fin de tâche (fin de cycle de nettoyage d'une imprimante) ou l'arrivée d'un problème inopiné vu comme une occurrence spontanée (impossibilité de sortir le train d'atterrissage, Bug d'un ordinateur pour cause indéterminée). Un évènement peut être aussi la conséquence de plusieurs conditions qui deviennent soudainement toutes satisfaites. Un événement peut être provoqué par un être humain, par une machine, par la nature ou par toute combinaison possible d'actions, de problèmes, etc. Un événement se produit instantanément et cause la transition de l'état d'une valeur (discrète) à une autre valeur.

Remarques:

R L'occurrence d'un évènement peut ne pas causer de changement d'états, mais si changement il y a alors il est causé par un évènement.

R Beaucoup de systèmes, en particulier des systèmes conçus par l'homme, sont des systèmes à états discrets. Même si ce n'est pas le cas, pour beaucoup d'applications une vue discrète d'un système complexe peut être nécessaire.

Formellement, un SED peut être défini comme un **automate à états** où l'espace d'états X et l'ensemble alphabet E sont dénombrables et qu'il n'existe pas un ensemble d'états finaux.

Définition 1.3 Un SED est un automate à états (E, X, Γ, f, x_0) où :
- E est un ensemble dénombrable d'événements,
- X est un espace d'états dénombrable,
- $\Gamma(x)$ est un ensemble d'événements possibles ou permis, défini pour tout $x \in X$ avec $\Gamma(x) \subseteq E$,
- f est une fonction de transition d'état, f: X x E $\rightarrow$ X, définie seulement pour e $\in \Gamma$ (x) quand l'état est x; f(x,e) n'est pas définie pour e $\notin \Gamma$ (x)
- x_0 est un état initial, $x \in X$.

L'état initial est essentiel dans la définition d'un SED. Cependant on est parfois intéressé par la description d'un système où tout état peut être pris comme point de départ. Ainsi on peut aussi définir un SED comme un quadruplet (E, X, Γ, f), omettant l'état initial.
Dans le diagramme des transitions d'états, $\Gamma(x)$ est représenté implicitement en incluant seulement les arcs émanant de x correspondant aux éléments de $\Gamma(x)$.

1.2.2 Exemples de Systèmes à Evènements Discrets

Les domaines d'applications des SED sont divers. Citons comme exemples les systèmes suivants:

- L'état d'une machine peut être sélectionné dans un ensemble tels que {marche, arrêt} ou {occupé, inactif, en panne};
- Un processus peut être dans 3 états {en exécution, prêt, en attente}. On peut vouloir décomposer les états de manière plus fine, par exemple incorporer dans l'état la valeur du compteur ordinal;
- La plupart des jeux peuvent être modélisés comme ayant un espace d'états discret. Par exemple, pour le jeu d'échecs chaque configuration de l'échiquier définit un état. L'espace résultant est donc discret; il est très grand mais pas infini.

D'autres domaines d'application plus complexes peuvent être représentés par des SED tels que la production manufacturière, la robotique, la circulation des véhicules, la logistique, les réseaux de communication et l'informatique.
Dans ce qui suit, on va présenter un exemple qui décrit le déplacement d'une souris dans un labyrinthe.

■ **Exemple 1.2 : Une souris dans un labyrinthe**
une souris se déplace de manière spontanée (sans intervention extérieure) dans un labyrinthe, tel que décrit par la Figure 1.6. Les salles S_i communiquent par deux portes unidirectionnelles P_1 et P_3 et une porte bidirectionnelle P_2.
On note par "p_i" l'événement : "*la souris passe par la porte* P_1".
Soit $\sum$ l'ensemble des événements : $\sum = \{ p_1, p_2, p_3 \}$.
Les situations sont:
$\Rightarrow$ *La souris est dans la salle* S_0
$\Rightarrow$ *La souris est dans la salle* S_1
$\Rightarrow$ *La souris est dans la salle* S_2
Ce qui définit trois états différents (A, B, C), relatifs aux possibilités d'occupation des salles. L'espace d'états s'écrit alors: $X = \{A, B, C\}$

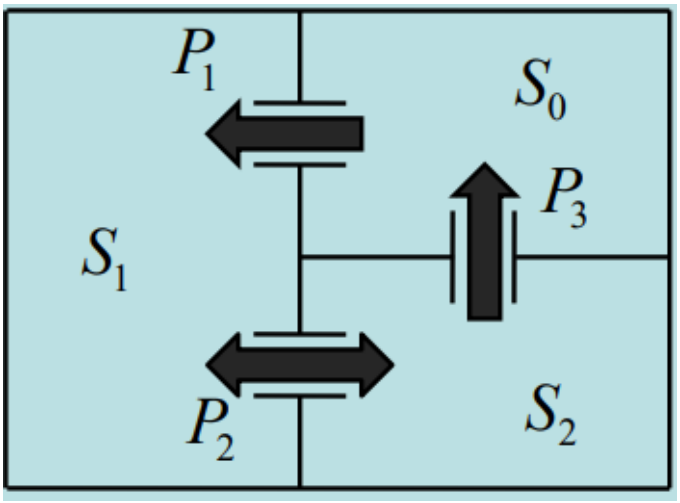


Figure 1.6: Exemple d'un SED: déplacement d'une souris dans un labyrinthe.

Soit E l'ensemble des évènements : $E = \{ p_1, p_2, p_3 \}$. Le passage de l'état *"souris en S_0"* (état A) à *"souris en S_1"* (état B) a lieu sur l'occurrence de l'événement p_1. On peut alors construire le diagramme d'états illustré par la Figure 1.7.

■

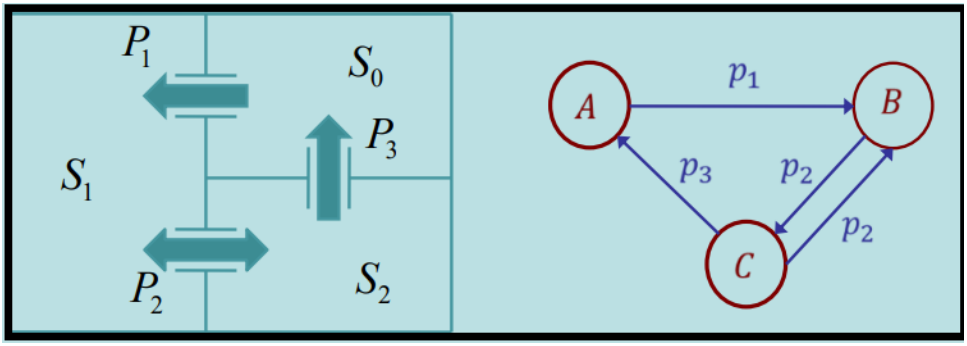


Figure 1.7: Diagramme d'états pour l'exemple de la Figure 1.6.

1.2.3 Propriétés des Systèmes à Evènements Discrets

Dans les SED, ce sont les évènements qui font progresser le système. Chaque évènement $e \in E$ définit un processus distinct et se produit à des instants déterminés. Les transitions d'états résultent de la combinaison de processus (évènements) asynchrones et concurrents. En outre, ces processus ne sont pas nécessairement indépendants entre eux.

Les SED sont donc conçus pour répondre à des évènements asynchrones non prédictibles tels que la fin d'exécution d'une tâche, le franchissement d'un seuil devant déclencher une action, l'arrivée d'un message, la défaillance d'un périphérique, etc.

La trajectoire illustrée par la Figure 1.8 met en évidence les propriétés caractéristiques des SED, à savoir la description de l'état par des grandeurs discrètes et l'évolution du système conditionnée par l'occurrence d'évènements discrets et asynchrones. La trajectoire (constante par morceaux) saute d'un état à un autre avec l'occurrence d'un évènement.

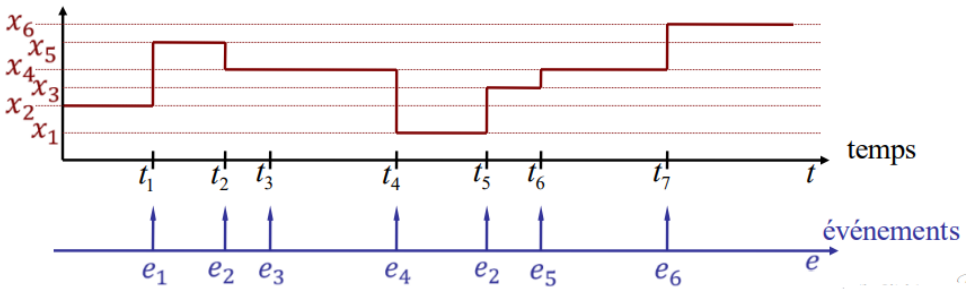


Figure 1.8: Trajectoire d'évolution d'un SED.

Remarques:

R D'après la Figure 1.8, nous déduisons ce qui suit:

- Un même évènement (e_2) peut conduire à des états différents (x_4 et x_3).
- Des évènements différents (e_2 et e_5) peuvent conduire à un même état (x_4).
- Des évènements (e_3) peuvent se produire sans provoquer de changement d'état; ils sont inactifs.

R La séquence des états et le temps de maintien associé à chaque état sont des caractéristiques de la trajectoire.

1.2.4 Systèmes à Evènements Discrets Temporisés et non-Temporisés

Dans un SED, on peut s'intéresser à une dynamique classique, résultant de l'écoulement du temps, ou à une dynamique engendrée par les événements sans prise compte des instants de leurs occurrences. On distingue alors les SED temporisés et les SED non-temporisés, respectivement.

► **Modèle non-temporisé :** Lorsqu'une séquence d'évènements (e_1, e_2, etc.) en entrée à un SED est spécifiée de manière déterministe sans information sur le temps auquel les occurrences de ces évènements se produisent, la trajectoire est décrite en terme de séquence d'états résultants. On parle alors d'un **modèle non-temporisé** du système. Reprenons l'exemple précédent, on spécifie les états mais on ne peut pas spécifier les instants où s'effectuent les transitions. La trajectoire devient comme le montre la Figure 1.9.
Un modèle **non-temporisé** décrit le comportement logique (et non temporel) du système.

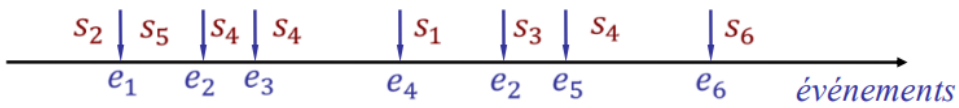


Figure 1.9: Trajectoire d'évolution d'un SED non-temporisé.

Il spécifie l'ordre des événements mais pas les instants d'occurrence; il permet de vérifier la correction du modèle et certaines propriétés structurelles et fonctionnelles du système. **Les modèles non-temporisés** sont en général une étape vers la construction des **modèles temporisés**.

- **Modèle temporisé :** Lorsque l'entrée d'un SED est spécifiée de manière déterministe comme une séquence d'évènements auxquels sont associés les dates d'occurrence t_i, la trajectoire complète du **modèle temporisé** peut être construite. Le calcul de mesures quantitatives à différents instants est essentiel lors de l'analyse du comportement de tels SED pour évaluer leur performance.

1.3 Etude et Modélisation des Systèmes à Evènements Discrets

La conception, la réalisation, la gestion et l'optimisation de ces systèmes soulèvent une liste de problèmes-types tels que la spécification, la conception, la validation logique, l'évaluation de performances, etc. Il existe actuellement une panoplie de formalismes de modélisation permettant l'étude des systèmes à événements discrets [11], tels que:

- **Les chaînes de Markov**: C'est un processus stochastique possédant la propriété de Markov, c'est à dire que l'information utile pour la prédiction du futur est entièrement contenue dans l'état présent du processus et n'est pas dépendante des états antérieurs ; on dit que le système n'a pas de « mémoire ». Les processus de Markov portent le nom de leur inventeur, Andreï Markov.

- **Les réseaux de files d'attente**: C'est une théorie mathématique relevant du domaine des probabilités qui consiste en l'étude de systèmes où des clients se présentent à un dispositif de service, appelé serveur. Cette théorie permet de modéliser de nombreuses situations dans lesquelles les clients arrivent à intervalles aléatoires dans le système comportant un ou plusieurs serveurs auxquels ils vont adresser une demande ou une requête.

- **Les automates d'états finis**: C'est un outil abstrait de modélisation formelle basé sur la théorie des automates. Il est très souple et s'adapte à des domaines très différents en informatique où l'évolution du système est séquentielle. Ils servent à représenter les systèmes déterministes les plus simples.

- **Les StateCharts** : Extension de l'outil classique automates à états avec les notions de hiérarchie, parallélisme et diffusion.

- **Les réseaux de Petri** : Ils constituent une famille de formalismes mathématiquement fondés. Ils permettent la modélisation de divers systèmes plus complexes (informatiques, industriels, etc.) travaillant sur des variables discrètes et comportant à la fois des phénomènes de synchronisation, de concurrence et de parallélisme. Plusieurs version étendues de réseaux de Petri ont vu le jour afin de répondre aux différentes spécificités des systèmes à modéliser, tels que les contraintes temporelles et l'aspect aléatoire.
- ...

1.4 Evaluation de Modèles

Tout modèle abstrait doit être analysé et évalué pour déterminer s'il vérifie ou non certaines propriétés clés du système modélisé. Il existe deux types d'évaluation : l'évaluation qualitative et l'évaluation quantitative. Notons que bien que l'on oppose régulièrement qualitatif et quantitatif, ces deux formes d'évaluation sont très liées voire **complémentaires**.

1.4.1 Evaluation Qualitative

L'évaluation qualitative s'intéresse à l'étude et à l'analyse du comportement logique (et non temporel) du système. Elle permet la vérification d'une manière précise des propriétés structurelles et comportementales du système telles que:

√ Correction du modèle.
√ Absence de blocage et d'inter-blocage.
√ Propriétés de vivacité d'atteignabilité pour des états donnés du système.
√ Absence de code mort.
√ Terminaison correcte du modèle et absence ou présence de cycles.
√ Bornitude ou non des états du modèle.
√ L'équité.
√ Satisfaction de la propriété de concurrence et du non déterminisme, etc.

L'analyse qualitative permet en particulier de répondre à la question *"un état particulier peut-il être atteint ?"* ou encore *"un état non désiré peut-il être atteint ?"*. On ne s'intéresse pas aux questions : *quand ?* ou *pendant combien de temps ?*.

1.4.2 Evaluation Quantitative

L'évaluation quantitative consiste à calculer les valeurs des indices de performances du système tels que:

√ Taux de perte de messages.
√ Taux d'occupation ou d'inoccupation d'un serveur.
√ Nombre moyen de clients dans le système ainsi que leurs temps de séjour moyen.
√ Temps moyen d'attente d'un client dans le système.
√ Temps de réponse, etc.

L'évaluation quantitative s'opère sur des SED temporisés. Elle permet de répondre aux questions:

- **Quand** un état particulier du système sera t-il atteint ?
- **Combien de temps** le système passe t-il dans un état particulier ?
- **Combien de fois** un état particulier peut-il être atteint sur un intervalle de temps donné ?

Les réponses à ces questions sont essentielles lors de l'analyse du comportement des SED car elles fournissent des mesures quantitatives particulièrement utiles pour évaluer la performance du système. Pour calculer ces indices, on distingue deux grandes familles de méthodes d'analyse quantitative:

► **Méthodes Analytiques :** Ce sont des modèles mathématiques qui calculent de manière rigoureuse les indices de performance en résolvant analytiquement les équations qui modélisent le fonctionnement du système. les résultats obtenus sont exacts et très précis. Citons comme exemples de méthodes:

- Les réseaux de files d'attente (Queuing Networks);
- L'analyse opérationnelle (algèbre de dioïdes);
- Les réseaux de Petri;
- Les processus stochastiques (chaînes de Markov);

Cependant et malgré leur faible coût d'exploitation et leur exactitude, les modèles analytiques ne pourraient être utilisés avec succès que si le système est soumis à des hypothèses très restrictives. La taille, la complexité et la diversité des répartitions temporelles de la plupart des systèmes réalistes interdisent la génération de modèles analytiques. Ainsi, plus le système à modéliser est complexe, plus les méthodes analytiques sont **limitées**. C'est pour cette raison que la **simulation** devient le dernier recours pour évaluer des SED très complexes.

► **Simulation :** Simuler c'est développer des expériences sur un modèle en vue d'évaluer les

performances d'un système pour répondre à un problème donné en déroulant l'historique de son fonctionnement [16]. Une approche de modélisation basée sur la simulation peut être utilisée d'une manière itérative soit pour améliorer la qualité du modèle proposé, soit pour évaluer les effets de la variation de certaines valeurs de paramètres sur le comportement du modèle et sur les résultats obtenus.

1.4.3 Buts de la Modélisation et de la Simulation des SED

La simulation, devenue incontournable, est pratique et facile à mettre en oeuvre. Elle permet de modéliser aussi bien des systèmes existants déjà ou non (à concevoir) pour les analyser et valider des choix de solutions. Les buts de la modélisation par simulation peuvent être résumés comme suit:

√ Meilleure compréhension du système: *« We don't realize how little we know about a system until we attempt to simulate it »* (Donald Knuth).
√ Représenter la dynamique d'un système complexe.
√ Mesure de performance du système.
√ Identification des facteurs critiques.
√ Réponse aux questions *« Que se passera-t-il si l'évènement n arrive avant m mais après p et que... ? »*.
√ Valider la conception (design) en estimant les coûts et les délais :
- Erreur détectée « sur le papier », facteur 1
- Erreur détectée « en production », facteur 10
- Erreur détectée « chez le client », facteur 100

√ Outil d'aide à la décision en « temps réel » (durant exploitation).
√ Outil de diagnostic et dépannage (local et à distance).
√ Outil de formation du personnel (simulateurs de vol, simulateurs de choix...).

Cependant, La simulation n'est pas une méthode directe, elle est coûteuse et les résultats obtenus doivent être interprétés avec beaucoup d'attention. Si le modèle analytique existe alors les résultats de simulation peuvent être utilisés pour appuyer les résultats analytiques.

1.5 Exercice d'Application

Il s'agit de modéliser un système réel comme un SED.

Exercice d'application 1.1 **Description du système réel:** On a 2 bras robotiques A et B avec deux piles de pièces a et b d'un objet donné. L'objectif est de déplacer les pièces de la pile a vers la pile b. Pour ce faire, le robot A prend une pièce de a et la ramène vers le robot B où se fera le transfert. Le robot B récupère la pièce et va la déposer sur la pile b (voir Figure 1.10). Modéliser ce système en identifiant ses différents états ainsi que les transitions et les évènements déclencheurs permettant l'évolution du système. ■

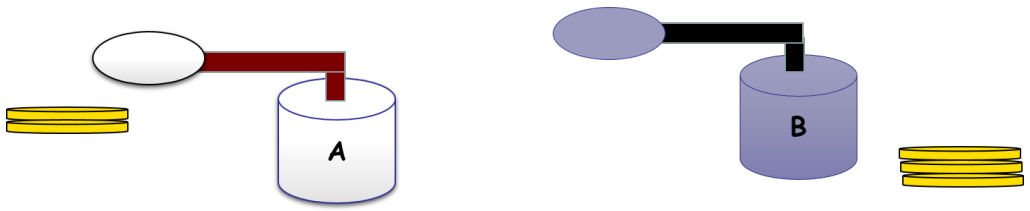


Figure 1.10: Exemple d'un système réel.

Solution : La solution est illustrée par la Figure 1.11.

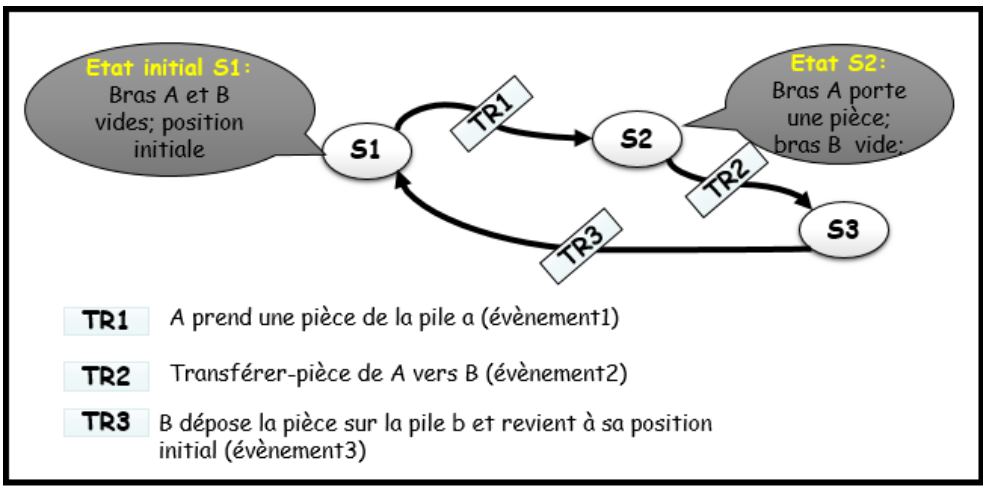


Figure 1.11: Modèle du système réel de l'exercice d'application.

Remarques:

R Dans cet exercice, nous avons opté pour certains choix de modélisation:

- Le déplacement de A vers B n'est pas modélisé comme un nouvel état dans ce modèle, c'est un choix d'abstraction. Mais si ce mouvement est important dans l'analyse du système alors il faut le modéliser par un état, une transition et un évènement.
- Le retour de A à sa position initiale et le déplacement de B vers la pile b ne sont pas modélisés, et ce n'est qu'un choix d'abstraction. Mais si ces mouvements sont importants dans l'analyse du système alors il faut les modéliser.
- Dans ce modèle, on a aussi ignoré la modélisation des piles ayant différents nombres de pièces.

1.6 Conclusion

Dans ce chapitre, nous avons présenté la définition d'un système dynamique, puis celle des systèmes dynamiques à évènements discrets ainsi que leurs types et leurs propriétés. Nous avons aussi soulevé les problèmes-types causés par la conception et le développement de ces systèmes nécessitant des activités de modélisation, de vérification et de validation. Ensuite, nous avons rappelé les outils existants pour les modéliser et analyser leur comportement. Nous avons, par la suite, exposé les méthodes d'évaluation qualitatives et quantitatives utilisées pour déterminer la correction du modèle et calculer ses indices de performances.

Dans le chapitre qui suit, nous allons nous intéresser au formalisme des réseaux de Petri et plus particulièrement aux concepts de base des RdP sous leur aspect graphique et algébrique.

1.7 Exercices

Exercice 1.1 : ***Marche aléatoire***

Une marche aléatoire est un modèle utile pour plusieurs processus intéressants, en particulier pour

quelques jeux de hasard.
Quand la marche aléatoire s'effectue dans deux dimensions, elle peut être visualisée comme une particule qui peut être déplacée d'une unité de distance (« un pas ») à la fois dans l 'une des quatre directions : nord, sud, ouest ou est. On suppose que la direction est choisie de manière aléatoire et indépendante de la position courante. L'état du système est défini par la position de la particule (x_1, x_2) mesurée dans un plan, avec x_1 et x_2 des entiers.

1. Déterminer l'espace d'états de ce système ansi que l'ensemble des évènements pouvant provoquer des changements d'états.
2. Tracer le chemin de progression (trajectoire) dans l'espace (x_1, x_2) résultant de l'état initial $(0,0)$ et de la séquence d'événements $(E,S,0,0,N,N,0)$.
3. Voyons maintenant comment les événements se situent dans le temps. Supposons qu'il y a quatre « joueurs » distincts, chacun responsable du mouvement de la particule dans une seule direction $(N,S,E,0)$. Chaque joueur agit en émettant occasionnellement un signal pour que la particule se déplace dans sa direction. Le système est donc actionné par les événements définis par ces quatre joueurs agissant de manière asynchrone.
   (a) En supposant que le joueur N émette des signaux aux instants discrets $(7,9)$, S le fasse à l'instant 2, O à $(4,6,10)$ et E à $(1,11)$, tracer la trajectoire dans l'espace d'états sous la forme d'un chronogramme où les événements causent les transitions d'état.
   (b) Que se passerait-il si E et S émettent un signal au même instant 1 ? que peut-on déduire?

Exercice 1.2 : ***Systèmes Informatiques (SI)***
Dans un S.I. typique, les travaux, tâches ou transactions sont les clients en compétition pour l'attention des serveurs, qui sont des processeurs (U.C., etc.), ou des périphériques (imprimantes, disques durs, etc.).
Quand un serveur est occupé au moment d'une demande d'un processus (ou tâche), ce processus est placé dans une file d'attente, qui est une partie essentielle du S.I. Il est souvent pratique de représenter un tel système avec un modèle de réseau de files d'attentes tel que décrit par la Figure 1.12. Dans ce système les processus arrivent dans la file d'attente de l'U.C. Une fois servis par

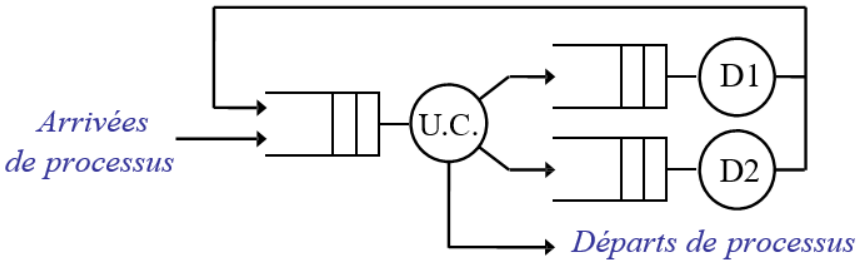


Figure 1.12: Un exemple de configuration de système informatique.

l'U.C., soit ils quittent le système, soit ils demandent l'accès à l'un des deux disques durs puis ils retournent ensuite dans la file d'attente pour l'U.C.

1. Déterminer l'espace d'états de ce système ansi que l'ensemble des évènements pouvant provoquer des changements d'états.
2. Donner une représentation possible de l'état du système en considérant ou non les tailles de files d'attente. Proposer un modèle états/transitions.

2. Notions de Base sur les Réseaux de Petri

2.1 Introduction

La modélisation, l'évaluation et l'analyse de performances des systèmes à événements discrets reste une préoccupation principale des différentes communautés scientifiques. Les recherches en modélisation et évaluation de performances sont deux tâches très liées du processus de conception d'un système. Plusieurs formalismes de modélisation, telles que les approches basées sur les réseaux de files d'attente, ont connu un développement considérable afin de pouvoir étudier des systèmes relativement complexes. Cependant, malgré la puissance des outils que ces travaux ont fourni, ils demeurent applicables à une classe relativement restreinte de systèmes. Ceci a motivé les chercheurs d'adopter un nouveau formalisme plus puissant introduit au début des années 60. Il s'agit des réseaux de Petri (RdP) [15] comme formalisme de base augmenté d'extensions qui vont de simples temporisations constantes [7] jusqu'à des mécanismes beaucoup plus sophistiqués tels que les réseaux de Petri stochastiques, les réseaux de Petri stochastiques généralisés [2, 4] et les réseaux de Petri colorés [12].

La littérature sur les réseaux de Petri est très riche et les résultats théoriques et pratiques les concernant sont très abondants. De part leur puissance d'expression, leur simplicité d'utilisation et plus particulièrement la diversité des techniques automatiques qui leur sont associées, les réseaux de Petri [17] se trouvent être l'outil ayant été le plus largement étudié et utilisé aussi bien pour l'analyse et l'évaluation analytique que dans la simulation du système étudié. Ils sont d'une aide précieuse pour le concepteur dans les différentes phases d'analyse. La Figure 2.1 montre la modélisation d'un système basée sur les réseaux de Petri.

2.2 Concepts de Base

Dans ce paragraphe, nous allons présenter les concepts de base des réseaux de Petri ordinaires.

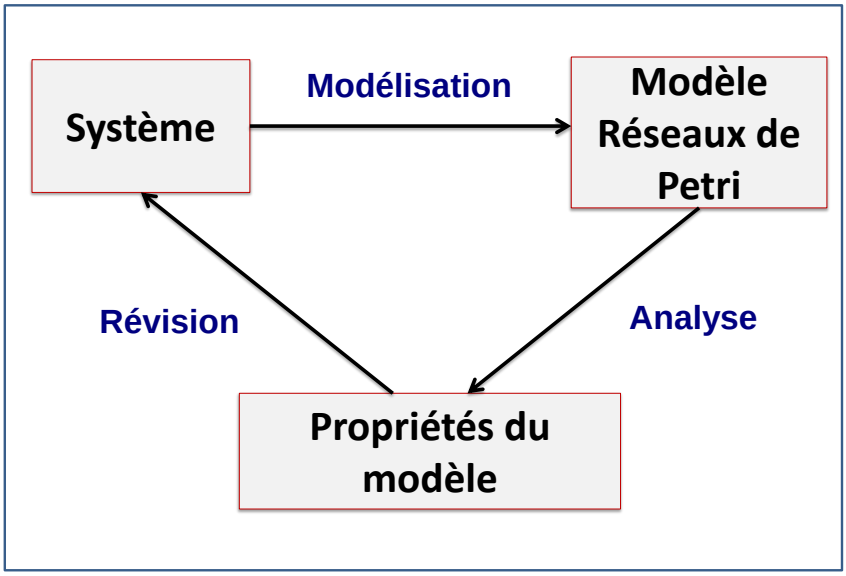


Figure 2.1: Modélisation par réseaux de Petri

2.2.1 Présentation Informelle

Les réseaux de Petri constituent une famille de formalismes mathématiquement fondés, adaptés à la modélisation de systèmes discrets parallèles et asynchrones [12, 15, 17]. Ils jouissent d'outils matures et éprouvés, tel que CPN Tools [8], qui permettent de créer, simuler et analyser des modèles RdP.

Historiquement, le concept de réseau de Petri a été développé pour la première fois (1960-1962) par le mathématicien Allemand Carl Adam Petri [17]. L'auteur a défini un outil graphique et mathématique permettant de modéliser le comportement dynamique des systèmes à événements discrets afin de permettre leur conception, leur évaluation et leur amélioration. Depuis, ils ont été adaptés, étendus et modifiés pour modéliser des systèmes à caractéristiques particulières qui ne pouvaient pas être représentés de façon satisfaisante avec le modèle original.

2.2.2 Elements de Base d'un Réseau de Petri

Concrètement, un réseau de Petri (RdP) est un **graphe biparti orienté** comprenant deux sortes de nœuds : les **places** et les **transitions**. Les **arcs** de ce graphe relient les transitions aux places ou les places aux transitions.
Le graphe biparti du RdP comporte (voir Figure 2.2):

- Un ensemble fini de **places** $P = \{P1, P2, P3, ..., Pn\}$ représentées et symbolisées par des cercles. elles représentent des **conditions** qui traduisent l'état d'une ressource du système (machine libre, stock vide, convoyeur à l'arrêt, etc.). Chaque place d'un RdP peut contenir un nombre entier de marques (on parle aussi de **jetons**).
- Un ensemble de **transitions** $T = \{T1, T2, T3, ..., Tm\}$ symbolisées par des traits (ou rectangles) et représentant l'ensemble des **événements** (les **actions** se déroulant dans le système) dont l'occurrence provoque la modification de l'état du système.
- Un ensemble fini d'**arcs orientés** représentés par des flèches et assurant la liaison d'une place vers une transition ou d'une transition vers une place. A chaque arc, on attribut un poids, appelé aussi valuation de l'arc (nombre entier positif). Par défaut ce nombre est égal à 1. Une

transition peut voir un ou plusieurs arcs en entrée et un ou plusieurs arcs en sortie. Chacun de ces arcs a son propre poids. Un graphe orienté est dit biparti, c'est-à-dire qu'un arc relie alternativement une place à une transition et une transition à une place. Ainsi, deux places (ou deux transitions) ne peuvent pas être reliées directement (voir Figure 2.3).

- Un ensemble de **jetons** ou marques de places qui, dans un RdP classique, sont indiscernables et permettent de déterminer si une condition est satisfaite ou non. Ils représentent, en général, les ressources disponibles.

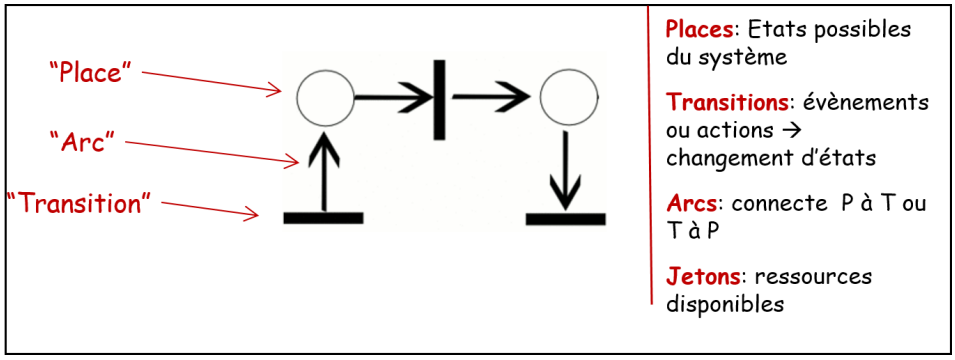


Figure 2.2: Elément de base d'un RdP.

Une transition sans place en entrée est une transition source; une transition sans place en sortie est une transition puits

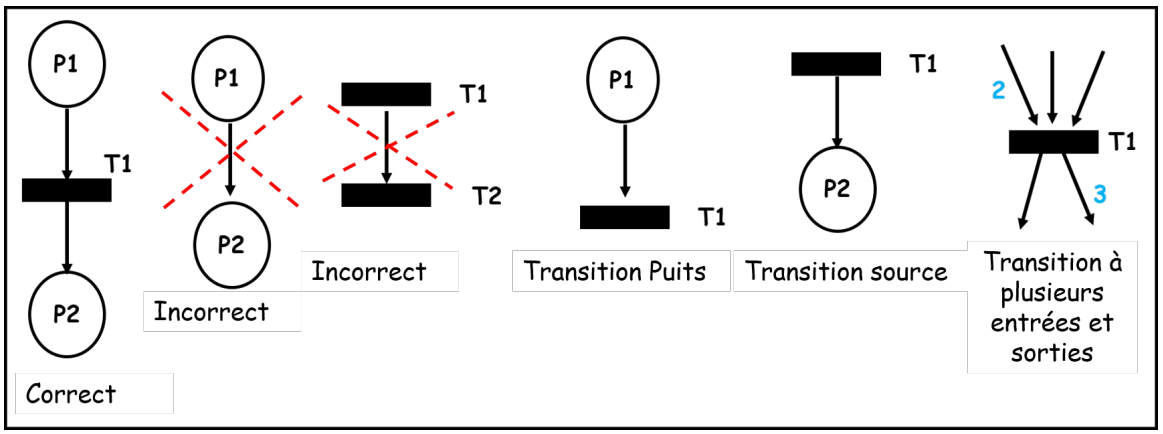


Figure 2.3: Arcs, places et transitions.

2.2.3 Le Vocabulaire

- ***Condition:***

C'est un prédicat ou une description d'un état du système; elle est vraie ou fausse (une machine est au repos, une machine est en réparation, une commande est en attente, etc.). Un état du système peut être décrit comme un ensemble de conditions. Une condition est représentée par une place.

- ***Evénement***:

Les événements représentent des actions se déroulant dans le système (début de traitement sur une machine, panne sur une machine, début de traitement d'une commande, etc.). Le déclenchement d'un événement dépend de l'état du système. Un événement est représenté par une transition.

- ***Déclenchement, pré-condition, post-condition***: Les conditions nécessaires au déclenchement d'un événement sont les pré-conditions de l'événement. Lorsqu'un événement se produit, certaines

de ses pré-conditions peuvent cesser d'être vraies alors que d'autres conditions, appelées post-conditions de l'événement deviennent vraies. La Figure 2.4 montre un exemple d'un RdP.
• ***Place en entrée (ou en amont)***: Lorsqu'une place est reliée à une transition par un arc : $p_i \longrightarrow t_j$, on parle de place en entrée (ou en amont) de t_j.
• ***Place en sortie (ou en aval)***: Lorsqu'une transition est reliée à une place par un arc $t_j \longrightarrow P_i$, on parle de place en sortie (ou en aval) de t_j.
• ***RdP pur / impur***: Si une place p est à la fois place d'entrée et de sortie d'une même transition t, alors elle est dite **impure**, sinon elle est pure. Un **RdP pur** est un RdP qui ne contient aucune place impure, sinon il est **impur**.
• ***RdP ordinaire / généralisé***: Si tous les arcs ont un poids égal à 1, on dit que le RdP est **ordinaire**. Un réseau de Petri **généralisé** est un réseau dans lequel les valuations des arcs ne sont pas forcément égales à 1.
• ***transition source***: Une transition sans place en entrée est une transition source(voir Figure 2.3).
• ***transition puits***: Une transition sans place en sortie est une transition puits (voir Figure 2.3).

2.2.4 Définition Formelle d'un RdP

Définition 2.1 Un réseau de Petri, appelé aussi réseau de Petri ordinaire ou encore réseau de Petri place/transition est défini par un quadruplet **R = (P, T, Pre, Post)** où [12]:

- P = ensemble fini de places qui modélisent les ressources utilisées dans le système **P ={P1, P2, P3,..., Pn}**
- T = ensemble fini de transitions qui modélisent les actions du système, **T={T1, T2, T3,..., Tm}** et $\mathbf{T} \cap \mathbf{P} = \varnothing$.
- ***Pré*** est l'application d'**incidence avant**: ***Pré*** $\subseteq$ **(P X T)**, ***Pré(p,t)*** est une valeur (un poids ≥ 0) associée à l'arc allant de la place ***p*** à la transition ***t***.
- *Post* est l'application d'**incidence arrière**: **Post** $\subseteq$ **(T X P)**, **Post(t,p)** est une valeur un poids ≥ 0) associée à l'arc allant de la transition **t** à la place ***p***.
- M_0 est une application de $P \longrightarrow \mathbb{N}$ appelée application du marquage initial. $M_0(p)$ est une valeur (≥ 0) représente le nombre initial de jetons dans la place ***p***.

■ **Exemple 2.1** : Un robot R ne cesse de prendre et de déposer des cubes. Définir les événements (actions, transitions) et les conditions (états) du système.
Les évènements (transitions)sont:
- Un cube arrive (évènement e1)
- Le robot saisit le cube (évènement e2)
- Le robot dépose le cube (évènement e3)

Les conditions (ou états) du système sont (places) :
- Le robot est au repos (condt c1)
- Un cube est en attente (condt c2)
- Un cube est en cours de déplacement (condt c3)
- Le cube a été déposé (condt c4) ■

Le RdP modélisant ce système est illustré par la Figure 2.5; il comporte un cycle pour représenter le fait que le robot ne cesse de prendre et de déposer le cube.

2.2.5 Marquage d'un RdP

Le marquage d'une place P_i est le nombre de jetons contenus dans cette place à un instant donné, il noté par $M(P_i)$. Une place peut être vide ou marquée.

Lorsque la place représente une condition logique (ex. : machine à l'arrêt, convoyeur en panne), la

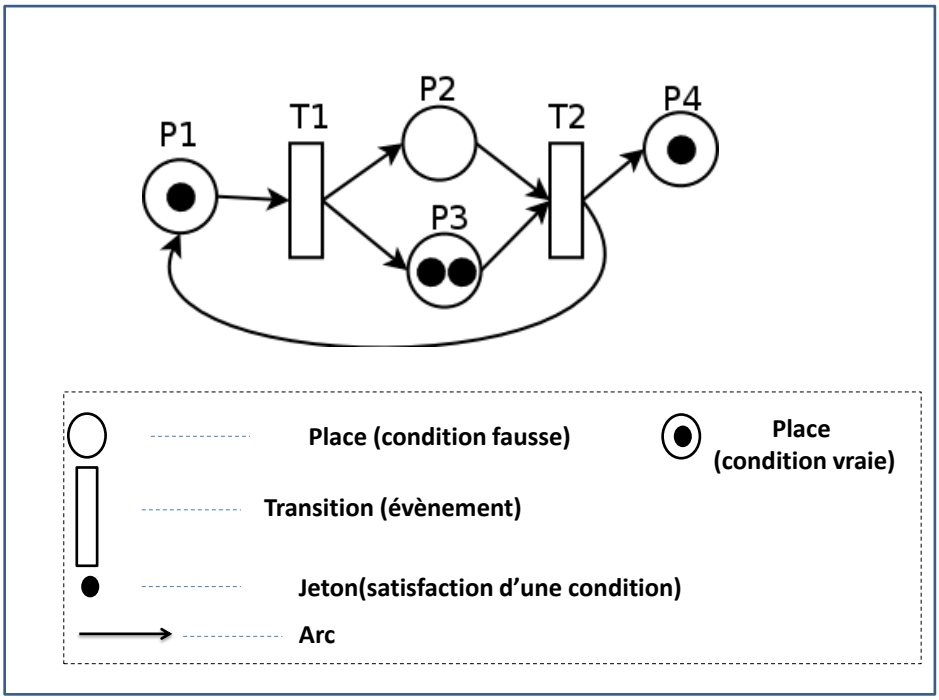


Figure 2.4: Exemple d'un réseau de Petri

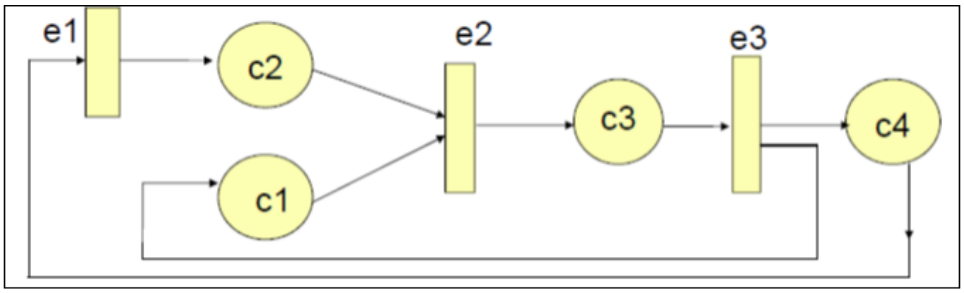


Figure 2.5: Modèle RdP du système robot.

présence **d'un jeton** indique que cette condition est vraie; elle est fausse dans le cas contraire.

Lorsqu'une place représente une ressource du système (un stock par exemple), elle peut contenir **plusieurs jetons** (dans l'exemple du stock, le nombre de jetons peut indiquer le nombre de pièces stockées).

Le marquage du RdP à un instant i est déterminé par la configuration complète du réseau, avec toutes les marques positionnées, c'est à dire la répartition des jetons dans les différentes places qui composent ce RdP.

Au cours de l'évolution du système, le marquage est susceptible d'être modifié. Le marquage du RdP à un instant donné est représenté par un vecteur M qui définit l'état du système modélisé à cet instant. Le marquage initial M_0 représente le marquage à l'instant $t = 0$ et correspond à la distribution initiale des jetons dans chacune des places du RdP. M_0 est alors défini par: $M_0 = [M_0(p_1), M_0(p_2), M_0(p_n)]$. Dans ce cas, on parle du **Rdp marqué** par opposition à un Rdp

non marqué, c'est-à-dire pour lequel le marquage initial n'est pas précisé.

L'évolution du marquage dénote, par conséquent, l'évolution de l'état du système à partir de M_0. Un **réseau marqué** est défini par $N =< R, M >$, où $R = (P, T, Pr, Post)$ et M = l'ensemble de Marquages.

■ **Exemple 2.2** : Soit le RdP de la Figure 2.6: Ce RdP possède 4 places, 4 transitions et 8 arcs

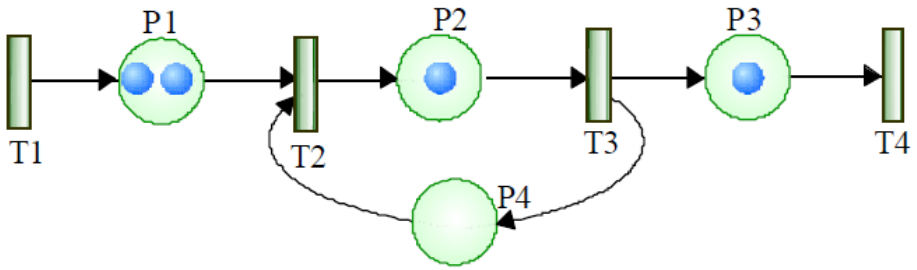


Figure 2.6: Marquage d'un réseau de Petri

orientés. Soit donc : $P = \{P1, P2, P3, P4\}$ et $T = \{T1, T2, T3, T4\}$; Le marquage initial est $M_0 = [2, 1, 1, 0]$; La place $P1$ est en amont (une entrée) de la transition $T2$ et elle est en aval (une sortie) de la transition $T1$;
$T1$ est une transition sans place d'entrée: transition source;
$T2$ est une transition sans place de sortie: transition puits. ■

2.3 Fonctionnement et Dynamique d'un RdP

Pour rendre compte de l'évolution du système modélisé, les RdP intègrent un formalisme permettant de passer d'un marquage à un autre : c'est le **franchissement** des transitions.

2.3.1 Sensibilisation et Franchissement d'une Transition

Le passage d'un marquage à un autre se fait par le franchissement des transitions. Une transition ne peut être franchie (ou tirée) que si elle est **sensibilisée**.

Définition 2.2 Une transition t_i est dite **sensibilisée** (ou franchissable ou validée ou tirable) si et seulement si toutes les places d'entrée (en amont) possèdent suffisamment de jetons (>= au poids de l'arc correspondant). La Figure 2.7 illustre différents cas de sensibilisation d'une transition.

Une transition peut être **franchie** (tirée) si et seulement si elle est **sensibilisée**. Les conséquences du franchissement (toujours supposé **instantané**) de la transition sont les suivantes :

√ On **prélève dans chacune des places d'entrée** un nombre de jetons égal au poids respectif de l'arc reliant la place à la transition.
√ On **dépose dans chacune des places** de sortie un nombre de jetons égal au poids respectif de l'arc reliant la transition à la place.

On passe ainsi de M_i à M_j par le tir de la transition t_j. On note $M_i[t_j > M_j$.

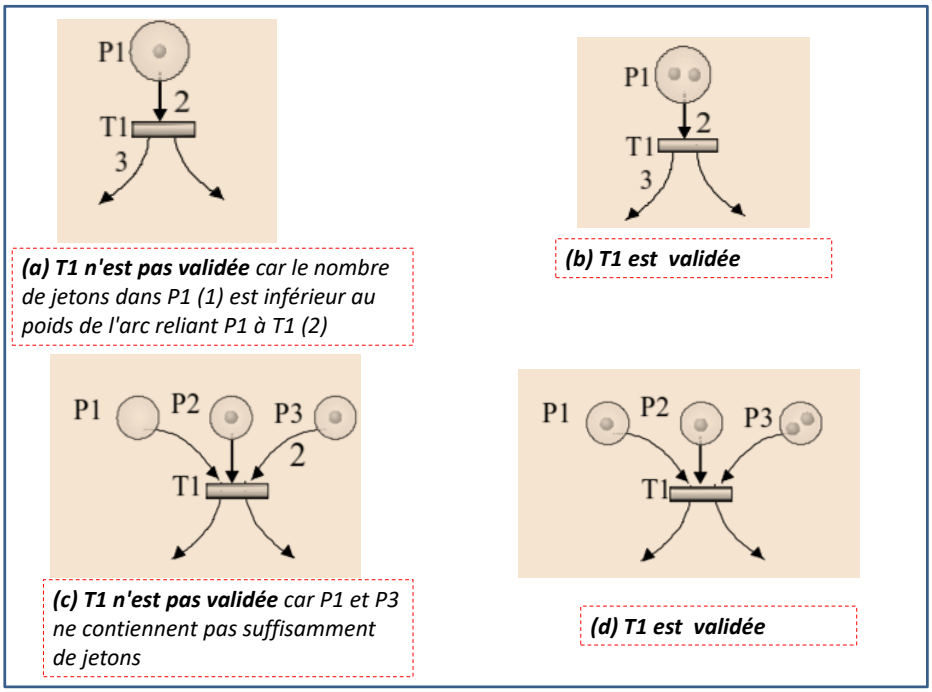


Figure 2.7: Exemple de sensibilisation des transitions

Remarques :

R Concernant la production et la consommation des jetons:

- Le nombre de jetons produits et celui des jetons consommés sont **indépendants**.
- Le **poids** de l'arc entrant à une transition représente le **nombre de jetons à consommer** (*pr*) par cette transition après franchissement.
- Le **poids** de l'arc sortant d'une transition représente le **nombre de jetons à produire** (*post*) par cette transition après franchissement.

R Concernant le franchissement des transitions:

- Lorsqu'une transition est validée cela n'implique pas qu'elle sera franchie immédiatement.
- Il y a **un seul franchissement** à la fois.
- Le franchissement d'une transition est **indivisible**.
- Le franchissement d'une transition à une durée nulle (sauf dans les RdP temporisés que nous verrons plus tard).

La Figure 2.8 montre un exemple de franchissement de transitions.

2.3.2 Séquence de Franchissement

Une séquence de franchissement S est une suite de transitions $\sigma = (T_1, T_2, ..., T_n)$ qui permet, à partir d'un marquage M, de passer au marquage M' par le franchissement successif des transitions définissant la séquence.

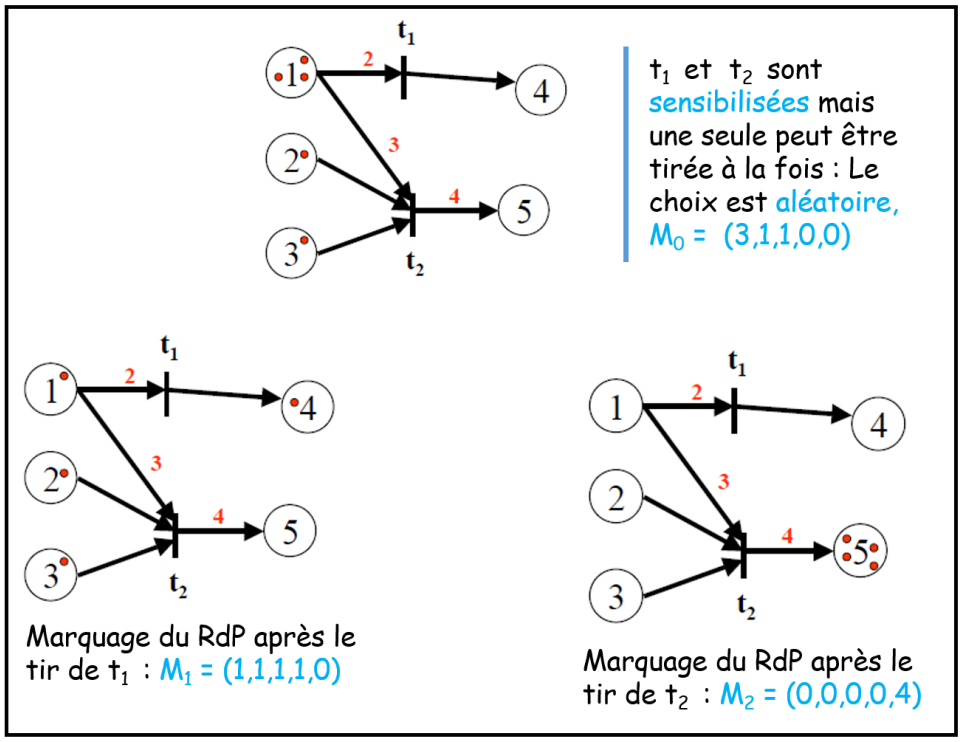


Figure 2.8: Exemple de franchissement de transitions

Notation 2.1. $M[S > M' ssi\ M[t_1 > M_1, M_1[t_2 > M_2, ..., M_{n-1}[t_n > M_n$.

Dans la Figure 2.9, l'ensemble des transitions est $t_1, t_2, t_3, t_4, t_5, t_6$. A partir de M_0 la séquence $\sigma = [t_1, t_2, t_3]$ est une séquence de tir. De même $\sigma = [t_1, t_1, t_5, t_6]$ est une séquence de tir. Par contre, la séquence $\sigma = [t_1, t_2, t_5]$ n'est pas une séquence de tir.

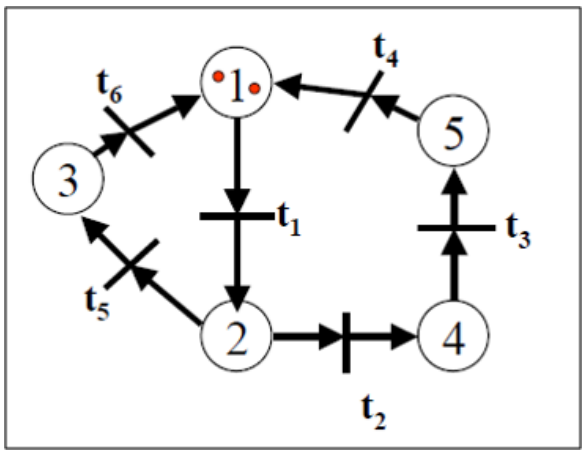


Figure 2.9: Exemple de séquence de franchissement de transitions

2.3.3 Graphe des Marquages Accessibles

Un graphe des marquages accessibles, appelé aussi **espace d'états**, est une représentation graphique de toutes les situations possibles du RdP au cours de son évolution à partir du marquage initial. C'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.

Pour construire le graphe de marquage d'un RdP, on démarre avec le seul marquage initial M_0, et on construit les différents arcs et nœuds progressivement. Pour chaque nouveau marquage M_i, on détermine l'ensemble des transitions franchissables et pour chaque transition de cet ensemble on ajoute un arc vers le nouveau marquage. Si un marquage est déjà présent dans le graphe, on se contente de tracer l'arc vers le marquage existant.

Notation 2.2. *: Ensemble des états accessibles ("Reachability set"):*

$$(E, M_0) = \{M \mid \exists\ \sigma\ telque\ M_0[\sigma > M\}.$$

La Figure 2.10 présente le graphe de marquage du RdP qui se trouve à droite de la figure.

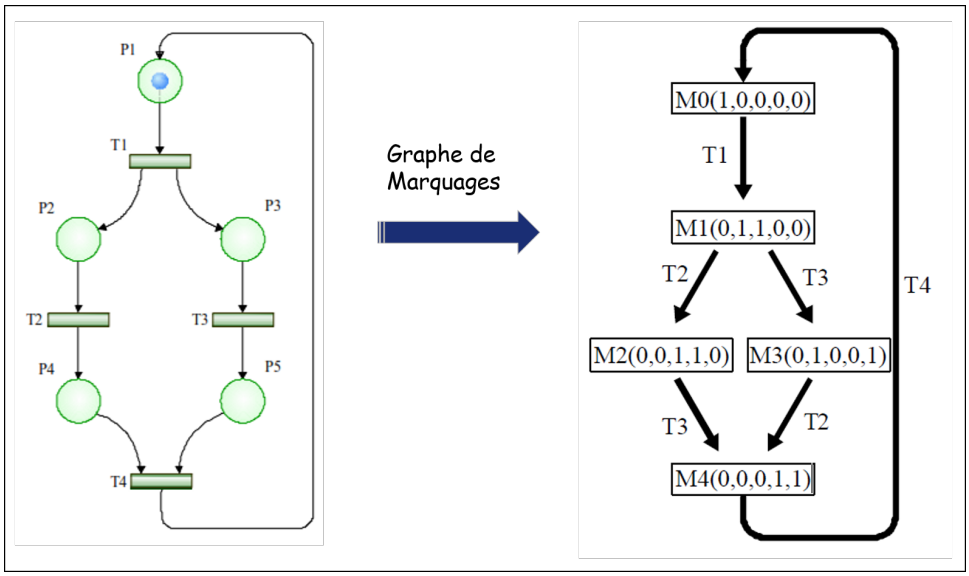


Figure 2.10: Exemple de graphe de marquages

Grâce au graphe de marquages, on peut étudier le comportement du RdP et déterminer plusieurs propriétés du modèle. Il nous permet de répondre à plusieurs questions tels que:

1. Combien de marquages sont accessibles (joignables)?
2. Quels sont les marquages accessibles?
3. Y a t-il des marquages terminaux accessibles?
4. Est-ce qu'un marquage non désiré (représentant un cas d'erreur) fait-il partie des marquages accessibles?, etc.

Notons que lors de la construction d'un graphe de marquages, deux situations peuvent se présenter:

1. **Le graphe est fini**: Il représente la situation la lus favorable car on peut déduire toutes les propriétés du RdP par une simple inspection de ce graphe.

2. **Le graphe est infini**: Dans ce cas, la construction exhaustive du graphe vue précédemment devient inapplicable. Comme alternative, on construit un autre graphe appelé **"graphe de couverture"** qui permet de déduire certaines propriétés.

2.3.4 Graphe de Couverture

Le graphe de couverture est une méthode alternative utilisée lorsque le graphe des marquages accessibles est infini. C'est un graphe où le nombre de marquages est fini. Avant de présenter la

méthode de construction d'un arbre de couverture, on définit la notion de couverture et couverture stricte:

Définition 2.3 **Couverture :**
Un vecteur M_2 couvre un vecteur M_1 (noté $M_2 = M_1$) si et seulement si chaque composante de M_2 est égale à sa composante dans M_1 :

$$M_2 = M_1 \Leftrightarrow M_2(p_i) = M_1(p_i)\ \forall i = 1,2,...,n$$

Définition 2.4 **Couverture stricte** :
Un vecteur M_2 couvre strictement un vecteur M_1 (noté $M_2 > M_1$) si et seulement si chaque composante de M_2 est égale à sa composante correspondante dans M_1 et au moins une composante de M_2 est supérieure à sa composante correspondante dans M_1 :

$$M_2 > M_1 \Leftrightarrow (M_2(p_i) \geq M_1(p_i)\ \forall i = 1,2,...,n\ et\ \exists\ i\ telque\ M_2(p_i) > M_1(p_i)\ )$$

Le mécanisme de construction du graphe de couverture peut être décrit par l'algorithme suivant:

Algorithme 1 — Construction du graphe de couverture.

Pas 1: *A partir du marquage initial M_0, déterminer toutes les transitions validées et les marquages 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 signifie que la place peut contenir autant de jetons que souhaité; elle est donc* ***non bornée***

Pas 2: *Pour chaque nouveau marquage M_i, faire étape 2.1 ou étape 2.2 :*

Pas 2.1 *: s'il* ***existe****, sur le chemin de M_0 à M_i (ce dernier exclu), un marquage $M_j = M_i$, alors M_i n'a pas de successeurs.*

Pas 2.2 *: s'il* ***n'existe pas*** *de marquage $M_j = M_i$ sur le chemin de M_0 à M_i, alors on prolonge l'arborescence 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 pour M_k,*
- ▷ *s'il existe un marquage M_j sur le chemin de M_0 à M_k tel que $M_k > M_j$ alors on met ω pour chacune des composantes supérieures aux composantes de M_i.*

L'arborescence ainsi obtenue est toujours et finie et peut être utilisée pour déduire un certain nombre de propriétés. La Figure 2.11 présente un exemple de graphe de couverture.

Remarques :

Limitations des graphes de couverture :

- Le symbole ω correspond à une perte d'information. D'une manière générale, il ne permet pas de répondre aux questions concernant:
  - **L'accessibilité d'un marquage**: Dans le cas d'un réseau non-borné , il est impossible de vérifier à l'aide d'un graphe de couverture si M est accessible. On peut « seulement » vérifier qu'il existe un marquage M' tel que $M' \supseteq M$.
  - **La vivacité du réseau**.
- A partir d'un graphe de couverture on peut déduire une séquence de franchissement valide partant de M_0 alors qu'en réalité elle ne peut être tirée depuis M_0 (voir Figure 2.12(a)).
- On peut avoir un même graphe de couverture pour des comportements différents (voir Figure 2.12(b))

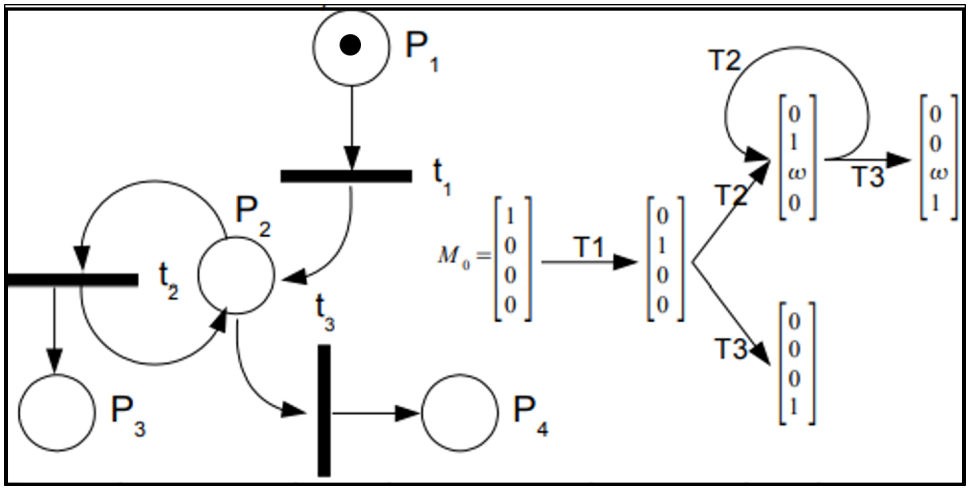


Figure 2.11: Exemple de graphe de couverture

Exercice d'application 2.1 : **Atelier de coupe de bois.**
Un atelier est constitué d'une machine de coupe et d'un stock. Quand une commande arrive et que la machine de coupe est disponible, la commande peut être traitée (découpe). Une fois le traitement terminé, la commande qui a été traitée est stockée. Sinon, la commande doit attendre que lamachine de coupe se libère avant de pouvoir être traitée.

1. Quels sont les différents états du système ?
2. Quels sont les évènements qui permettent la dynamique du système ?
3. Déterminer le modèle RdP correspondant à ce système.
4. Donner le marquage du RdP pour 6 commandes: 3 en attentes, une en traitement et 02 en stock.

■

Solution :
L'état du système est caractérisé par :
- Machine disponible ⟶ Place **P1**
- commande en attente ⟶ Place **P2**
- commande en cours de traitement ⟶ Place **P3**
- commande stockée ⟶ Place **P4**

L'état du système va évoluer quand se produisent les évènements suivants :
- Arrivée d'une commande ⟶ Transition **T1**
- Début découpe (traitement) ⟶ Transition **T2**
- commande Fin découpe (traitement)⟶ Transition **T3**

Le RdP et le marquage M sont présentés par la Figure 2.13.

Commentaires :
- Chaque place contient un nombre entier (positif ou nul) de jetons. Elle indique la valeur de la variable d'état associée à la place. Par exemple, la place $P2$ contient 3 jetons qui signifient 3 commandes en attente d'être traitées par la machine de coupe.
- La présence des jetons dans les places indiquent l'état des ressources du système : Par exemple, dans la place $P1$, un jeton indique que la machine est libre, alors que l'absence d'un jeton indique que la machine n'est pas disponible. De même, une marque dans la place $P3$ indique qu'une commande est en train d'être traitée par la machine de coupe. Le nombre de jetons dans la place $P4$ indique le nombre de commandes qui ont été traitées et stockées. La fin de la coupe par la machine se traduit par la suppression du jeton dans la place $P3$ et par la

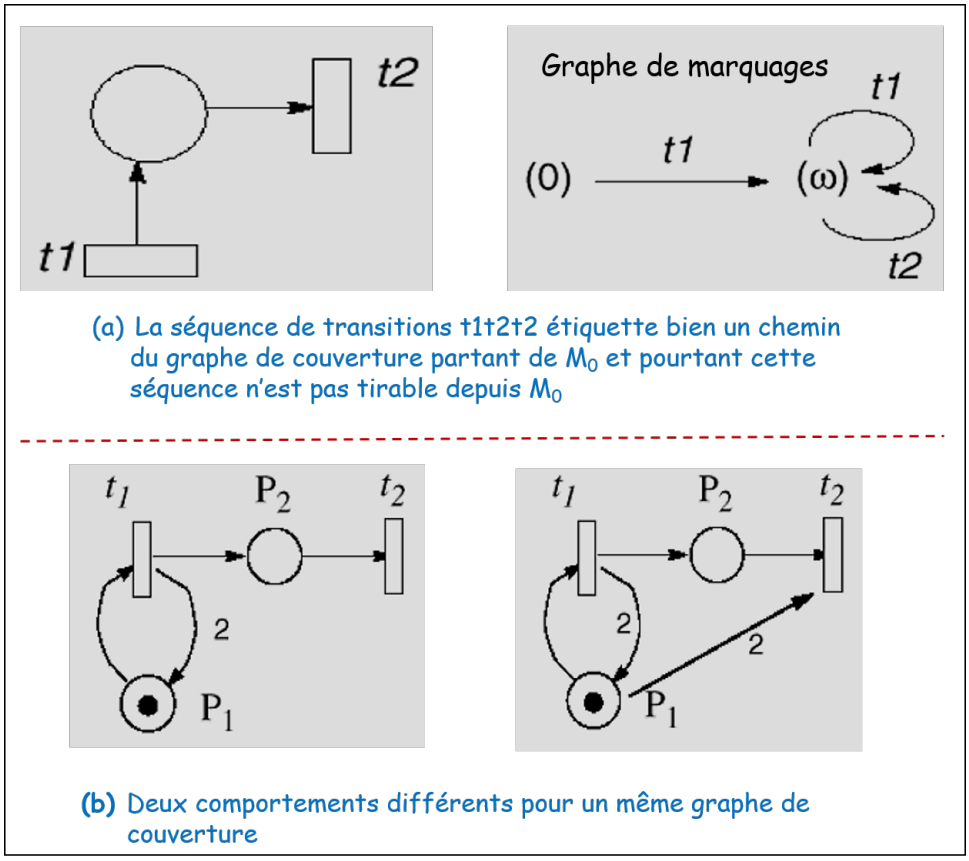


Figure 2.12: Limites des graphes de couverture

création d'un jeton dans la place $P1$ et dans la place $P4$.

- Les jetons d'une même place n'ont pas d'identité individuelle : elles sont indiscernables. Par exemple, chaque commande en attente dans la place $P2$ est traitée de la même façon que les autres.

2.4 Représentation Matricielle des réseaux de Petri

La présentation faite des réseaux de Petri est une représentation graphique, mais il est possible d'en faire une représentation mathématique grâce à l'algèbre linéaire. Cette notation permet de représenter un RdP de manière plus formelle mais moins lisible que la représentation graphique.

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. Ainsi, ***Pré*** et **Post** sont représentées par des matrices à ***m*** lignes (nombre de places), ***n*** colonnes (nombre de transitions).

2.4.1 Matrice d'incidence avant W^- (*Pré*)

C'est une matrice **m x n** (m places et n transitions) tel que $W^-_{ij} = Pré\ (Pi, Tj)$. Elle indique le nombre de jetons à enlever de la place Pi par le tir de la transition Tj (voir la Figure 2.14).

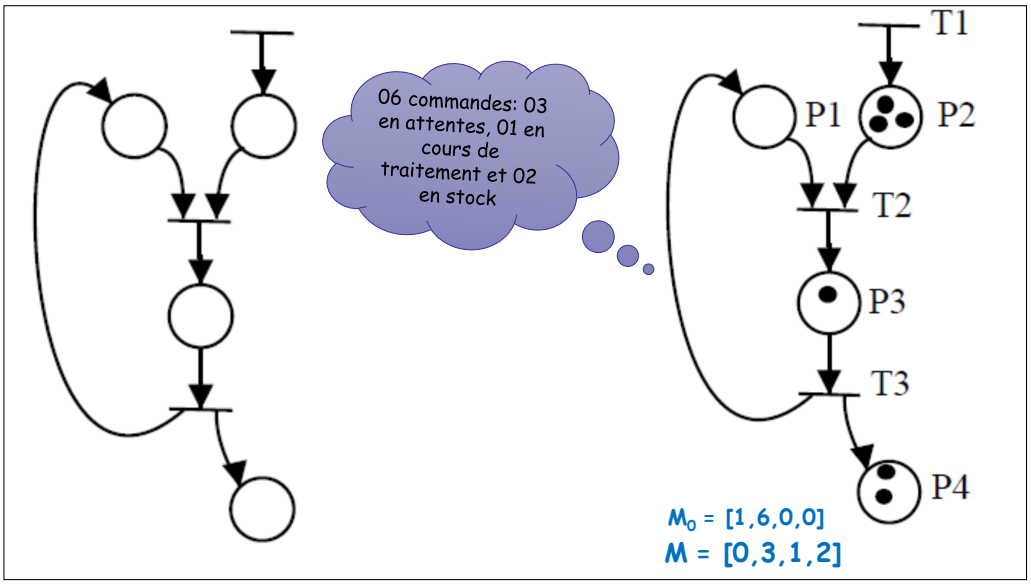


Figure 2.13: RdP de l'exercice : Atelier de coupe de bois.

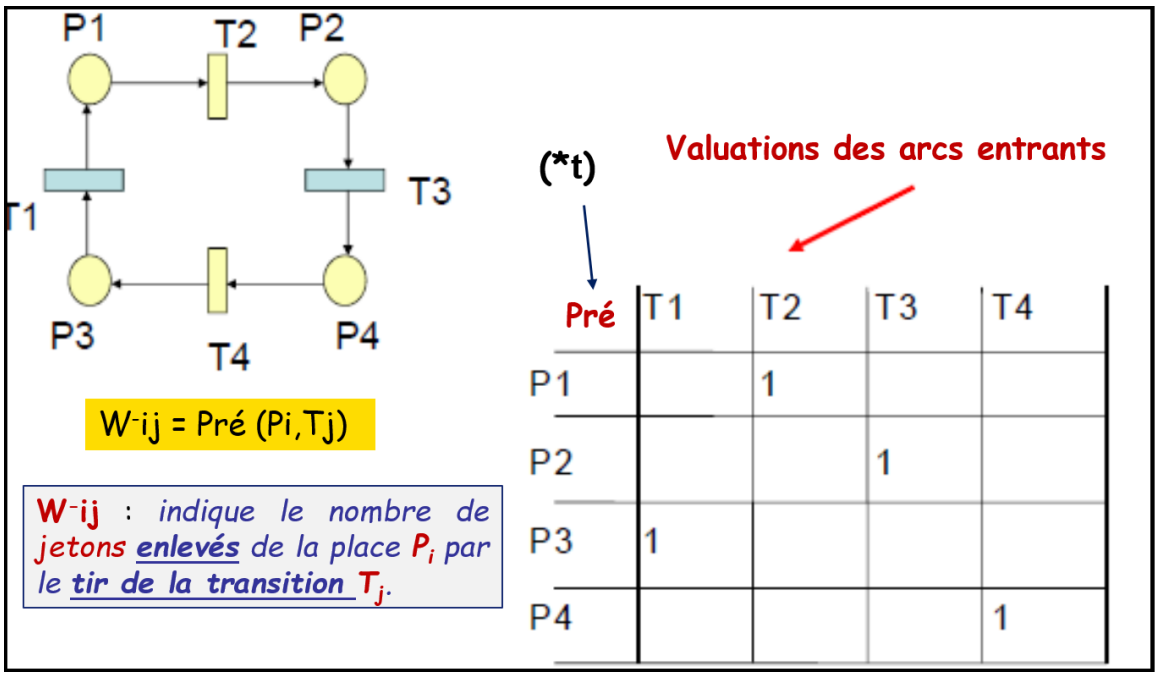


Figure 2.14: Matrice d'incidence avant

2.4.2 Matrice d'incidence arrière W^+ (*Post*)

C'est une matrice **m x n** (m places et n transitions) tel que $W^+_{ij} = Post(Pi, Tj)$. Elle indique le nombre de jetons à ajouter à la place Pi par le tir de la transition Tj (voir la Figure **??**).

2.4.3 Matrice d'incidence *W* du RdP

C'est une matrice **m x n** (m places et n transitions) tel que $W_{ij} = W^+_{ij} - W^-_{ij}$. Elle indique la modification du marquage de la place Pi par le tir de la transition Tj. La matrice d'incidence

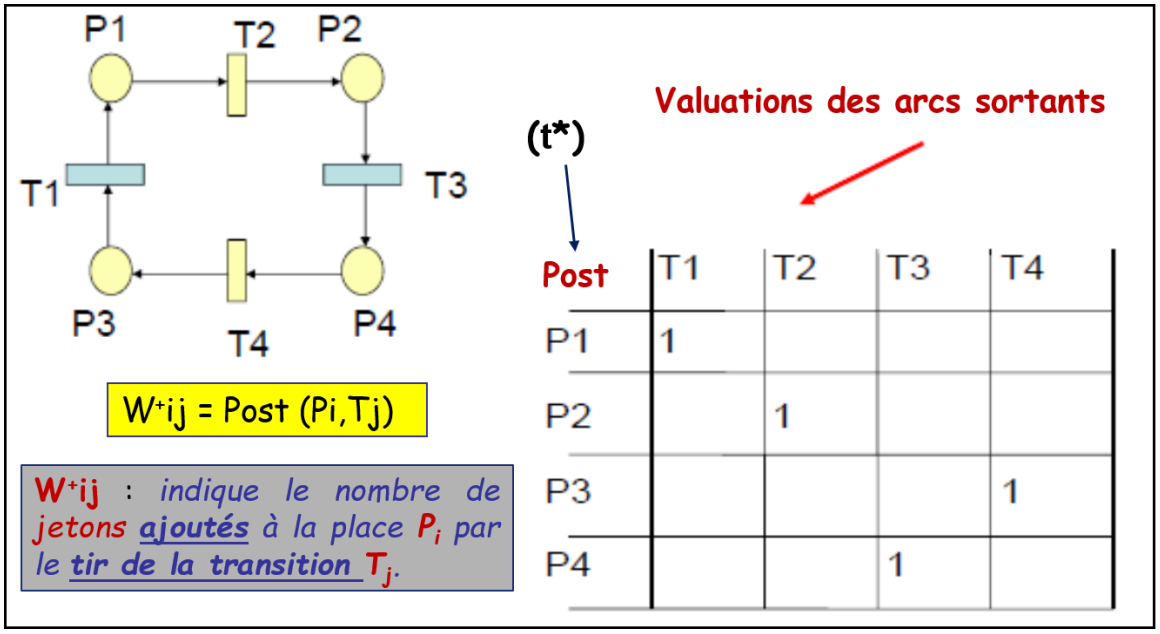


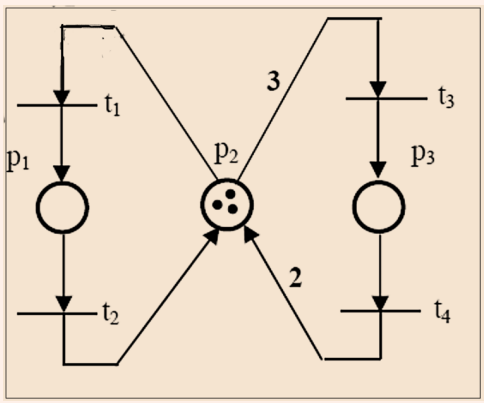
Figure 2.15: Matrice d'incidence arrière

W définit complètement le RdP; elle est indépendante du marquage, par contre elle est liée à la structure du RdP. La Figure 2.16 montre les matrices d'incidence du RdP de l'exercice 2.1 ("*Atelier de coupe de bois*").

Remarque Importante

R La matrice d'incidence n'a de sens que dans le cas des réseaux **purs**. En effet, dans les réseaux **non purs**, où une place est en entrée et en sortie de la même transition, le solde des marquages est **nul**, la matrice d'incidence perd donc une partie de l'information.

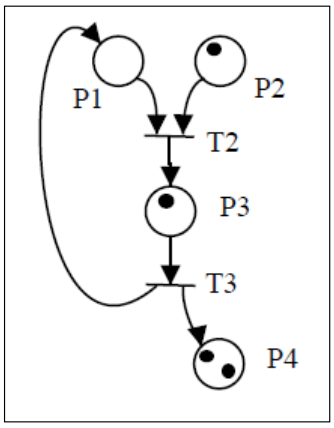
Exercice d'application 2.2 Soit le RdP de la figure suivante:



1. Indiquer le marquage initial
2. Déterminer la matrice d'entrée W^- et la matrice de sortie W^+
3. Déterminer la matrice d'incidence du RdP.

■

RdP Atelier de coupe simplifié



Matrice d'incidence avant

Pré : P × T → N ;

W⁻	T2	T3
P1	1	0
P2	1	0
P3	0	1
P4	0	0

Matrice d'incidence arrière

Post : P × T → N.

W⁺	T2	T3
P1	0	1
P2	0	0
P3	1	0
P4	0	1

W = W⁺ - W⁻

W	T2	T3
P1	-1	1
P2	-1	0
P3	1	-1
P4	0	1

Matrice d'incidence du RdP

Figure 2.16: Matrices d'incidence d'un RdP

2.5 Equation Fondamentale et Vecteur d'Occurrence

Les règles de fonctionnement mentionnées précédemment sont suivies par une **équation fondamentale** qui gouverne l'évolution de l'état du modèle.

2.5.1 Définition

Définition 2.5 Soit $S = t1, t2, t3, \ldots, tn$ une séquence de franchissement, telle que : $M_i[S > M_k$. On définit **S**, vecteur caractéristique de la séquence S en précisant pour chaque transition le nombre de fois où la transition est franchie dans la séquence.

Par exemple, si on a un graphe contenant les transitions *t1, t2* et *t3* et si on prend la séquence de transition : ***S = t1 t2 t1***, le **vecteur d'occurrence** $\underline{S}$ **= [**$\underline{S}$**(t1),** $\underline{S}$**(t2),** $\underline{S}$**(t3)] = [2, 1, 0]**.
Compte tenu de ces différentes définitions, l'**équation fondamentale** (appelée aussi équation de changement d'état) peut être décrite comme suit:

$$M_k = M_i + W * \underline{S}$$

Cette équation permet de calculer, en une opération et sans parcours du réseau, le marquage obtenu après le franchissement d'une séquence complète.
Remarque Importante :

R Les résultats de l'**équation fondamentale**, même s'ils sont toujours calculables, n'ont de sens que si la séquence S est effectivement franchissable.

2.5.2 Utilisation de l'équation fondamentale

L'equation fondamentale gouverne la dynamique du modèle; elle est donc utile pour l'analyse du système. Grâce à cette équation, on peut calculer tous les marquages de manière algébrique, comme le montre l'exemple suivant:

■ **Exemple 2.3** On considère le RdP et sa matrice d'incidence W de la Figure 2.16.
Le franchissement de la transition $T2$ conduit au marquage $M' = [1123]^T$ qui peut être obtenu algébriquement par la relation:

$$M' = M_0 + W * \underline{S}$$

Dans ce cas $S = T_2$ d'où $\underline{S}$ **= *[0 1]*** :

$$\overset{M_0}{\begin{bmatrix}0\\1\\1\\2\end{bmatrix}} + \overset{W}{\begin{bmatrix}-1 & 1\\-1 & 0\\1 & -1\\0 & 1\end{bmatrix}} * \overset{\underline{S}}{\begin{bmatrix}0\\1\end{bmatrix}} = \overset{M'}{\begin{bmatrix}1\\1\\0\\3\end{bmatrix}}$$

A partir du marquage M', le tir de T_1, donne le nouveau marquage ***M'' = [0 0 1 3]*** qui peut être également obtenu algébriquement par la relation :

$$M'' = M'_0 + W * \underline{S} \text{ où S} = T_1; \ \underline{S} = [1\ 0]:$$

$$\overset{M'}{\begin{bmatrix}1\\1\\0\\3\end{bmatrix}} + \overset{W}{\begin{bmatrix}-1 & 1\\-1 & 0\\1 & -1\\0 & 1\end{bmatrix}} * \overset{\underline{S}}{\begin{bmatrix}1\\0\end{bmatrix}} = \overset{M''}{\begin{bmatrix}0\\0\\1\\3\end{bmatrix}}$$

Enfin ce marquage peut être obtenu à partir du marquage initial par le franchissement de la séquence S = $T_2\ T_1$ de la manière suivante $\underline{S}$ **= *[1 1]*** :

$$\overset{M_0}{\begin{bmatrix}0\\1\\1\\2\end{bmatrix}} + \overset{W}{\begin{bmatrix}-1 & 1\\-1 & 0\\1 & -1\\0 & 1\end{bmatrix}} * \overset{\underline{S}}{\begin{bmatrix}1\\1\end{bmatrix}} = \overset{M''}{\begin{bmatrix}0\\0\\1\\3\end{bmatrix}}$$

■

Remarque importante :

R L'obtention d'un marquage à partir d'un autre ne signifie pas forcément l'unicité du vecteur caractéristique. En effet, si on reprend l'exemple précédent, on aboutit à partir du marquage initial M_0 au marquage M_4 par franchissement de la séquence $T_1T_2T_3$ **(S=[1 1 1 0])** ou par franchissement de la séquence $T_1T_3T_2T_4T_1T_2T_3$ **(S=[2 2 2 1]^T)**.

R Une séquence est dite **répétitive** si à partir d'un marquage initial, on aboutit à un marquage final identique au marquage initial comme le montre l'exemple de la Figure 2.17.

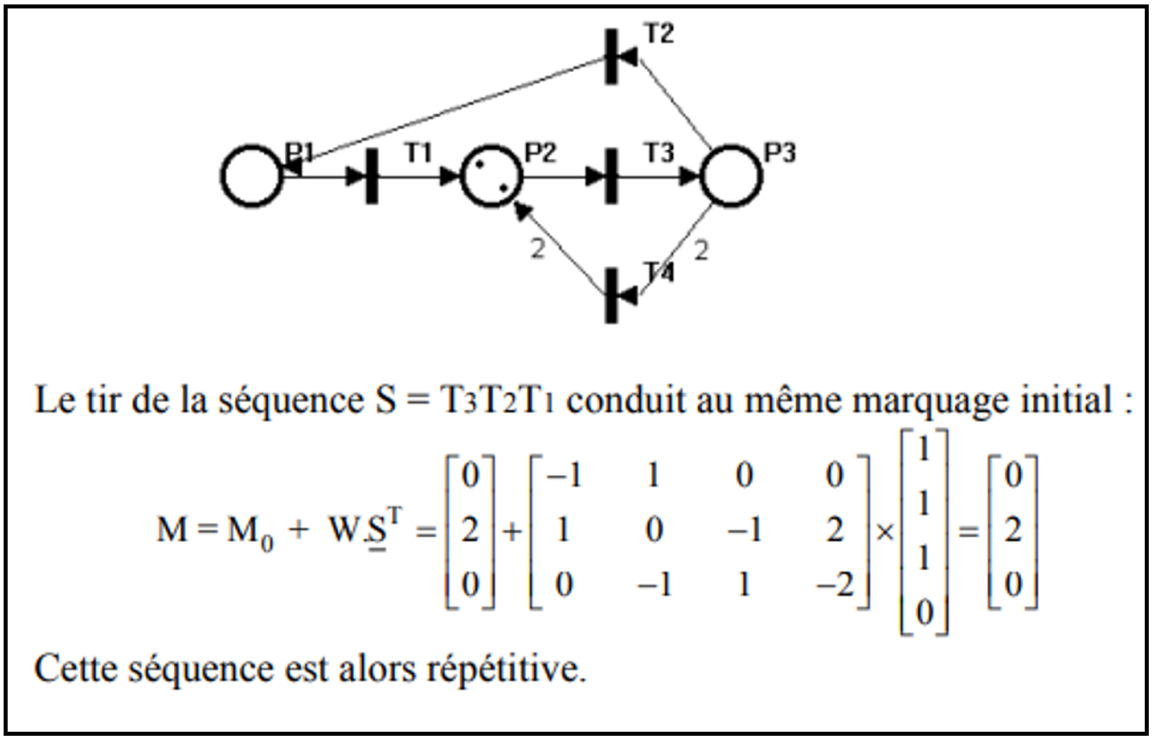


Figure 2.17: Exemple d'une séquence de franchissement répétitive.

Exercice d'application 2.3 Soit le RdP et son graphe de marquage présentés par la Figure 2.10:

1. Etablir la matrice d'incidence avant *(Pré)*
2. Etablir la matrice d'incidence arrière *(Post)*
3. Etablir la matrice d'incidence W
4. En utilisant l'algèbre linéaire, déterminer le marquage résultant de la séquence de franchissement $S = T_1T_3T_2T_4T_1$ puis celui de la séquence $S' = T_1T_4$. Comparer les marquages obtenus avec le graphe de marquage du RdP, que peut-on déduire?

■

2.6 Conclusion

Dans ce chapitre, nous avons présenté les concepts de base des réseaux de Petri. Par la suite, nous avons montré comment fonctionne un RdP en expliquant les différentes règles de franchissement des transitions et les principes de circulation des jetons dans le modèle. Nous avons aussi défini ce qu'est un graphe de marquages accessibles puis nous avons présenté l'algorithme de construction du graphe de couverture dans le cas où le graphe de marquages est infini. La représentation matricielle et l'utilisation de l'équation fondamentale ont été enfin exposées et expliquées par l'exemple.
Dans le chapitre qui suit, nous allons nous intéresser aux éléments de modélisation nécessaires à la description de tout type de comportement qu'un système pourrait exhiber.

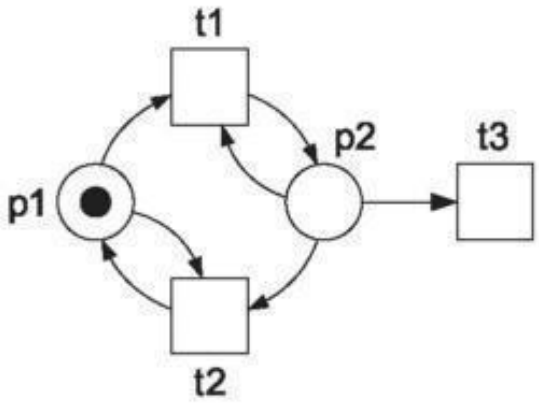
2.7 Exercices

Exercice 2.1

Dessiner le RdP défini par *P = p1, p2, T = t1, t2, et F = (pl, t1),(t1, p2), (p2, t2), (t2, p2), (t2, p1)*, où *F* représente l'ensemble des arcs du RdP.

Exercice 2.2

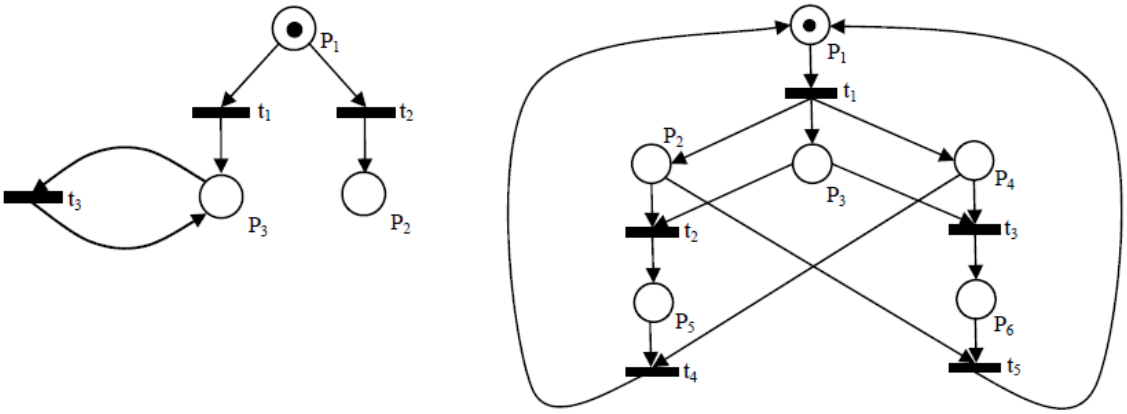
Considérons le RdP de la figure suivante :



1. Définir formellement le RdP comme un triple (P,T,F).
2. Dans quelles conditions les transitions de ce RdP seront-elles sensibilisées ?
3. Déterminer le graphe de Marquage de ce RdP.

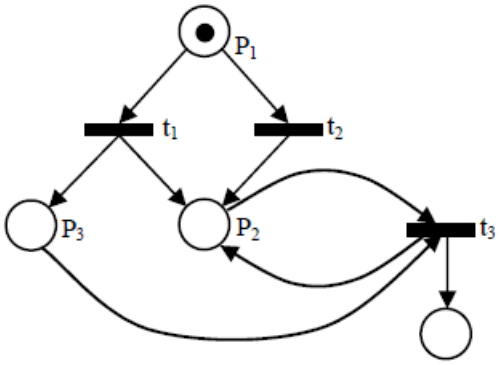
Exercice 2.3

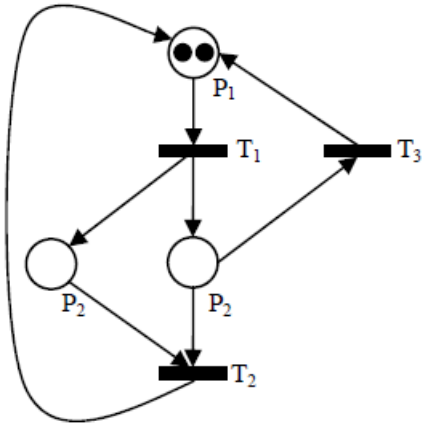
Déterminer les graphes de marquages accessibles des graphes suivants:



Exercice 2.4

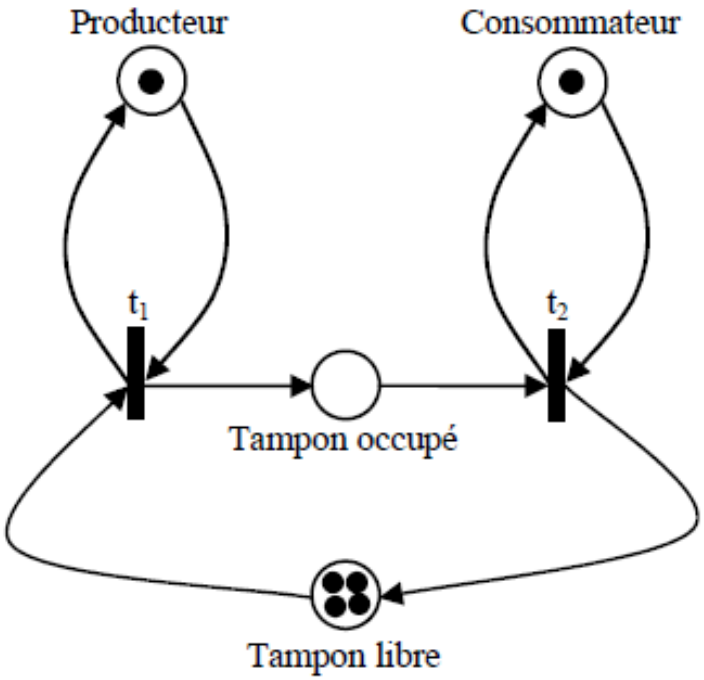
Trouver les graphes de couvertures pour les Réseaux de Petri suivants ?





Exercice 2.5

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 *Pré 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 = $t_1t_1t_2t_1t_2t_2t_1t_1t_1t_1t_1t_2t_1t_1$
4. sachant que le marquage initial est M_0 [1,1,2,4]? est-t-il effectivement franchissable ?

3. Eléments de Modélisation et Propriétés des RdP

3.1 Introduction

Comme nous l'avons déjà mentionné, les RdP sont un outil intégrant un fort potentiel pour la modélisation des SED à comportements parallèles, concurrents et distribués. Ils s'appuient sur un support mathématique très élaboré leur permettant d'effectuer une analyse formelle des différentes propriétés comportementales et structurelles. C'est un outil très puissant pour l'analyse et l'optimisation des performances des systèmes modélisés. Dans ce qui suit, nous allons présenter les différents éléments de modélisation nécessaires à la description de divers comportements, puis nous exposerons les propriétés qu'un modèle devrait vérifier.

3.2 Eléments de Modélisation des Réseaux de Petri

Une des caractéristiques importantes des réseaux de Petri est de pouvoir 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. [11, 12, 17]. Dans cette section, nous allons définir les éléments de modélisation qui sont des concepts associés à la sémantique opérationnelle (la dynamique, le comportement) des RdP.

3.2.1 Parallélisme et Concurrence

Le parallélisme représente la possibilité que plusieurs processus évoluent simultanément au sein du même système. Sur un RdP, ceci est représenté par une transition ayant plusieurs places de sortie, comme le montre la Figure 3.1. Il s'agit aussi d'une exécution concurrente vu que plusieurs transitions peuvent être franchies et le franchissement de l'une n'empêche pas celui des autres. Cependant, une seule transition peut être titrée à la fois. Le choix de la transition à franchir est non-déterministe.

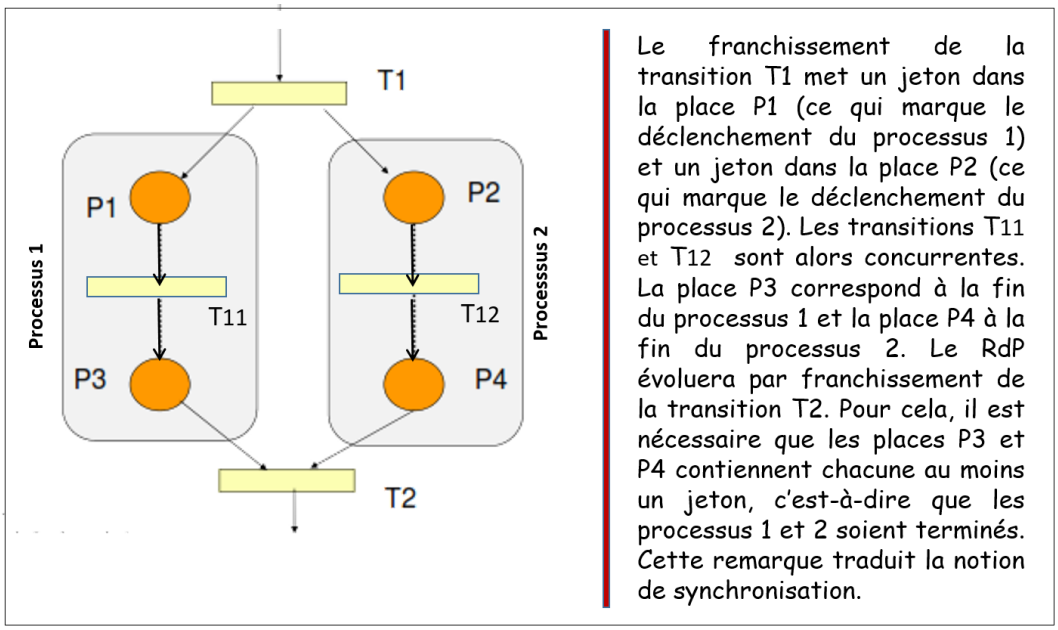


Figure 3.1: Modélisation du parallélisme dans un RdP.

3.2.2 Synchronisation Mutuelle

La synchronisation mutuelle, ou synchronisation par rendez-vous, permet de synchroniser les opérations de deux ou plusieurs processus. Elle est représentée, sur un RdP, par une transition ayant plusieurs places en entrée, comme le montre la Figure 3.2.

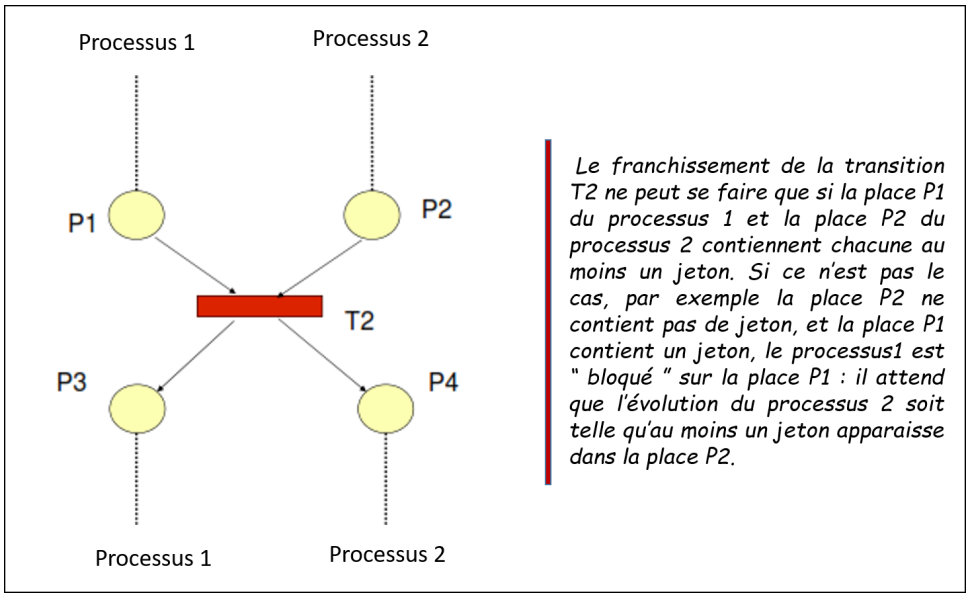


Figure 3.2: Modélisation de la synchronisation mutuelle dans un RDP.

3.2.3 Synchronisation par Signal (Sémaphore)

Dans ce type de synchronisation, l'évolution d'un processus est conditionnée par celle d'un autre, c'est à dire il ne peut avancer que si l'autre a déjà franchi une transition donnée. Par

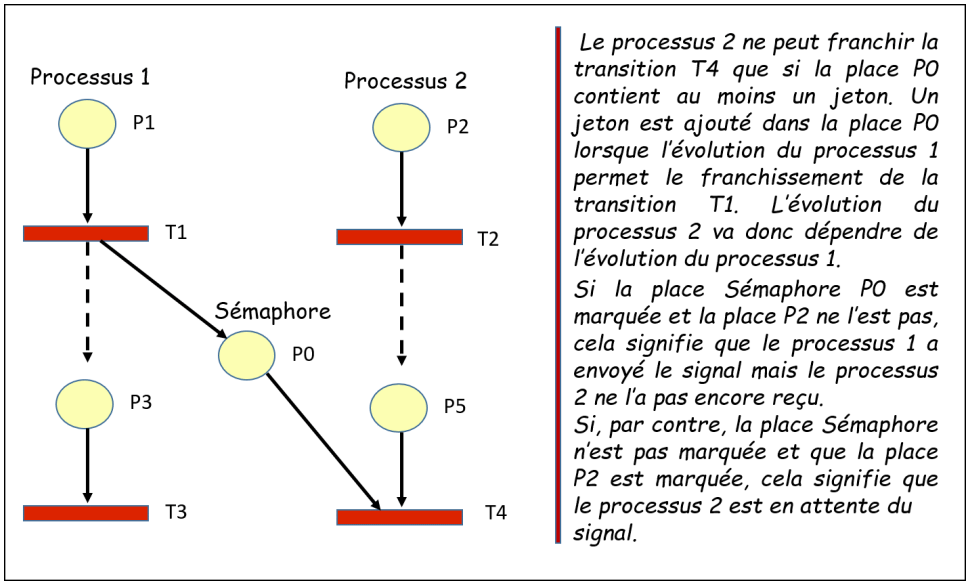


Figure 3.3: Modélisation de la synchronisation par sémaphore dans un RDP.

contre, l'avancement des opérations du premier processus ne dépend pas de l'avancement des opérations du second processus. On parle dans ce cs d'une communication asynchrone. La Figure 3.3 illustre ce cas de figure.
Des exemples de cas de séquence, parallélisme, concurrence et synchronisation sont montrés par la Figure 3.4.

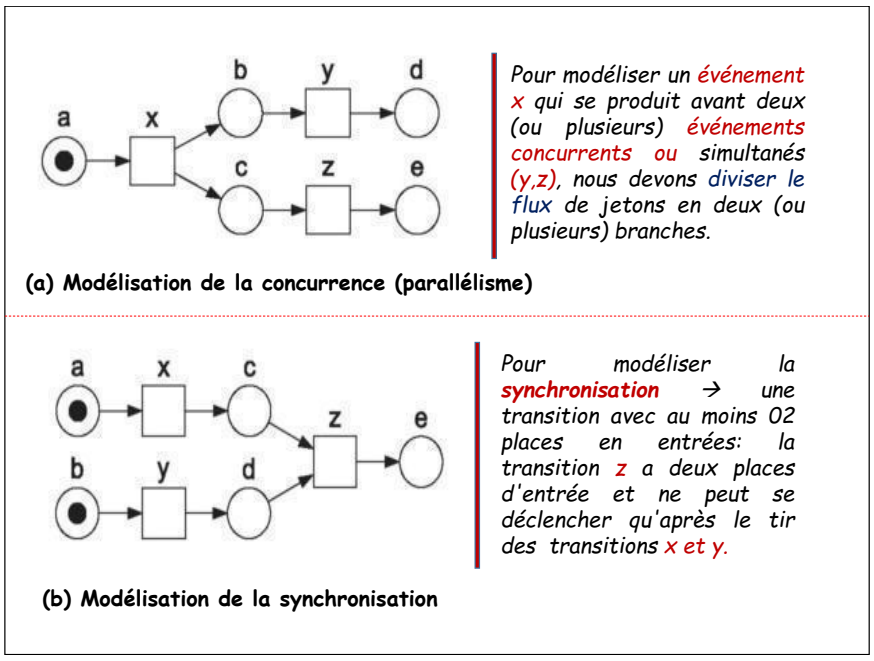


Figure 3.4: Exemple de modélisation du parallélisme et de la synchronisation.

3.2.4 Partage de Ressources

C'est un type de modélisation lié à un système au sein duquel plusieurs processus partagent une même ressource en utilisant le principe de l'**exclusion mutuelle**. Dans ce cas, deux places sont mutuellement exclusives si pour un marquage donné M, elles ne peuvent pas être simultanément marquées quelque soit le marquage M' atteint à partir de M. Dans l'exemple de la Figure 3.5, les places $P4$ et $P5$ sont mutuellement exclusives. Dans l'exemple de la Figure 3.4 (a), il y a exclusion mutuelle entre les transitions y et z.

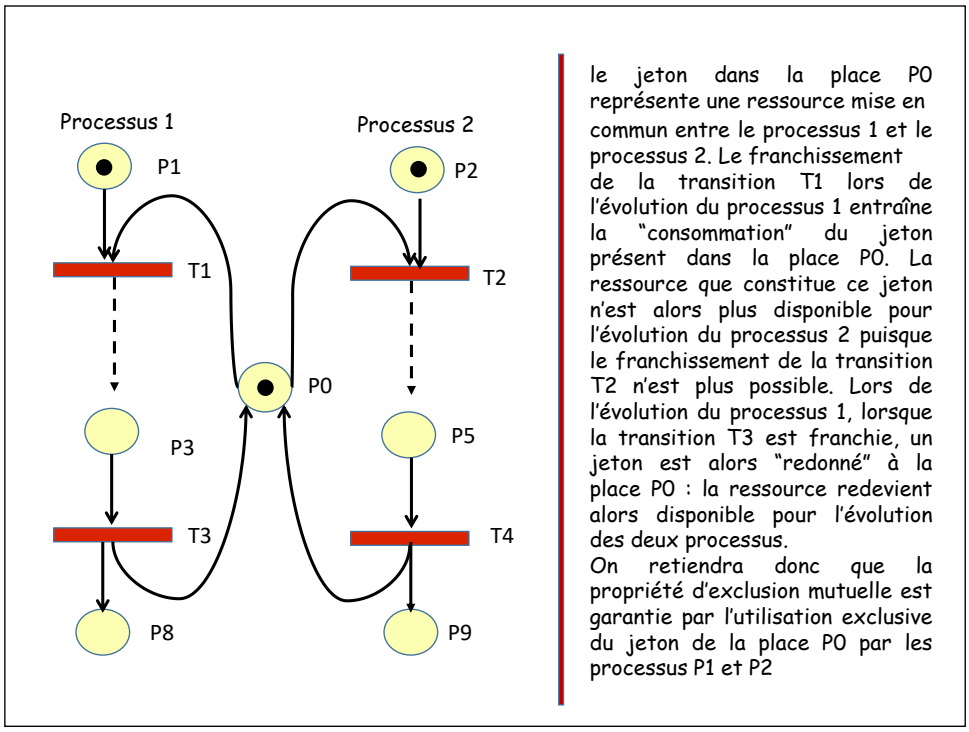


Figure 3.5: Modélisation de la synchronisation par partage de ressources.

3.2.5 Conflits Structurels et Conflits Effectifs

Un réseau de Petri est dit sans conflit si et seulement si toute place a au plus une transition de sortie. Dans le cas contraire, le RdP peut présenter un conflit structurel et/ou effectif.

Définition 3.1 : Conflit structurel
Deux transitions sont en **conflit structurel** lorsqu'elles possèdent une place d'entrée commune. Cette situation de conflit correspond à la concurrence de la consommation des jetons d'une place.

Définition 3.2 : Conflit effectif
On parlera d'un **conflit effectif** entre deux transitions en **conflit structurel** s'il existe un marquage qui sensibilise les deux transitions et que le franchissement d'une transition empêche le franchissement de l'autre: une seule transition sera franchie, mais rien dans le réseau ne permet de prévoir laquelle. Un conflit effectif signifie qu'il y a indéterminisme du réseau, c'est à dire que l'évolution du système décrit présente une **partie aléatoire**.

Les Figures 3.6 (a) et (b) illustrent respectivement le cas avec conflit effectif et le cas sans conflit effectif.

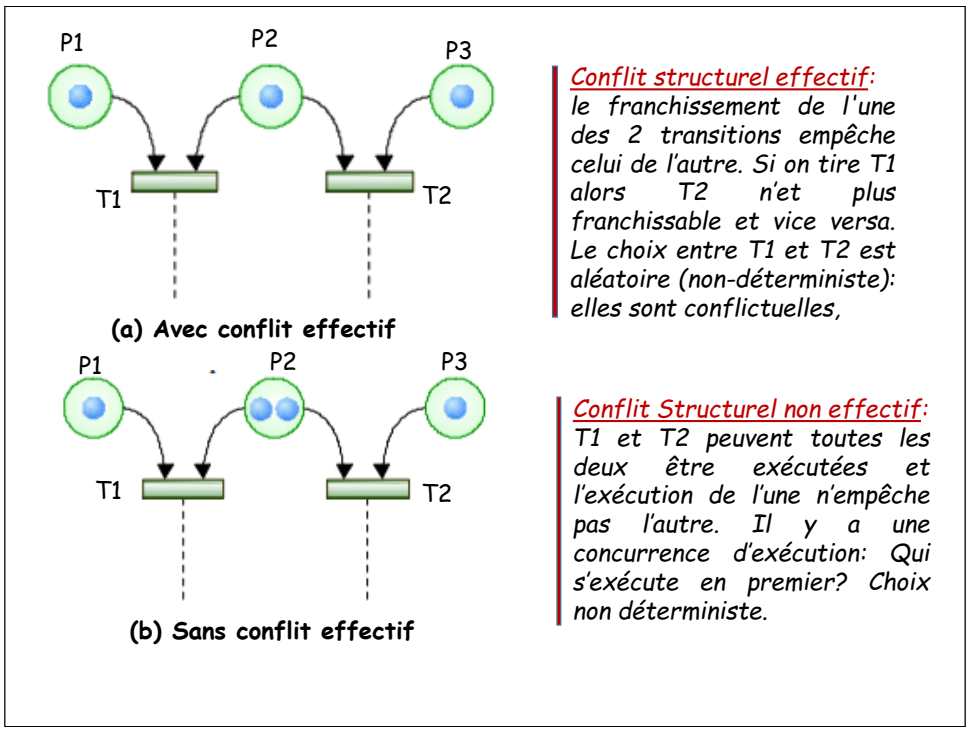


Figure 3.6: Modélisation du conflit structurel effectif et non effectif.

3.2.6 Capacité Limitée

Souvent nous sommes appelés à modéliser des situations où une limite de capacité est imposée. Citons comme exemples la limite du tampon dans le modèle producteur-consommateur ou la limite d'un stock de ressources. Le modèle RdP de la Figure 3.7 illustre un exemple de modélisation d'un stock de capacité totale égale à 3 (ou un tampon de taille limitée à 3): d'où le nom **capacité limitée**. Le marquage de $P5$ indique le nombre d'emplacements libres dans le stock (ou le tampon); le marquage de la place $P4$ indique le nombre d'emplacements occupés.

▪ **Exemple 3.1** : Un constructeur d'automobile achète des pompes à injection auprès d'un fournisseur. Pour réduire l'espace à l'entrepôt, au plus trois pompes sont en stock. Du point de vue des pompes, deux événements sont importants: une pompe est livrée (« deliver ») ou intégrée dans une voiture. Cela conduit à trois états: *not-delivered, in-stock et built-in*. ▪

Le modèle de la Figure 3.8 propose une solution à ce problème.

3.2.7 Mémorisation

Dans tous les modèles, on peut avoir besoin de compter le nombre de tirs d'une transition en utilisant une place sans sortie. On peut aussi modéliser le nombre d'instances d'un processus qui sont en attente, etc. La Figure 3.9 illustre des cas de mémorisation. Les RdP de base n'imposent pas de politiques pour résoudre les conflits et ordonnancer le franchissement des transitions. Dans ces cas, le choix est souvent aléatoire (non déterministe)

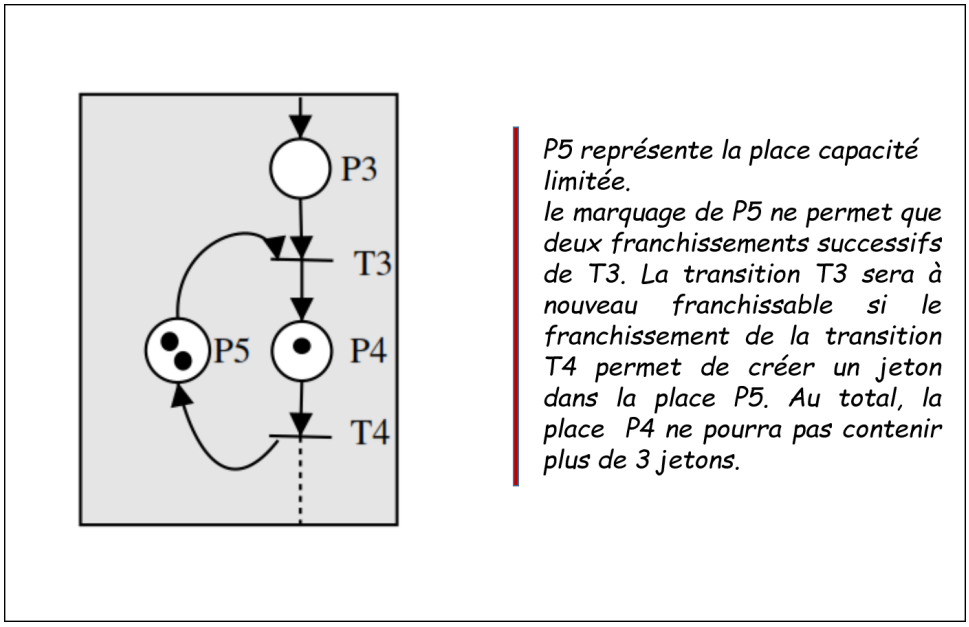


Figure 3.7: Modélisation de la capacité limitée.

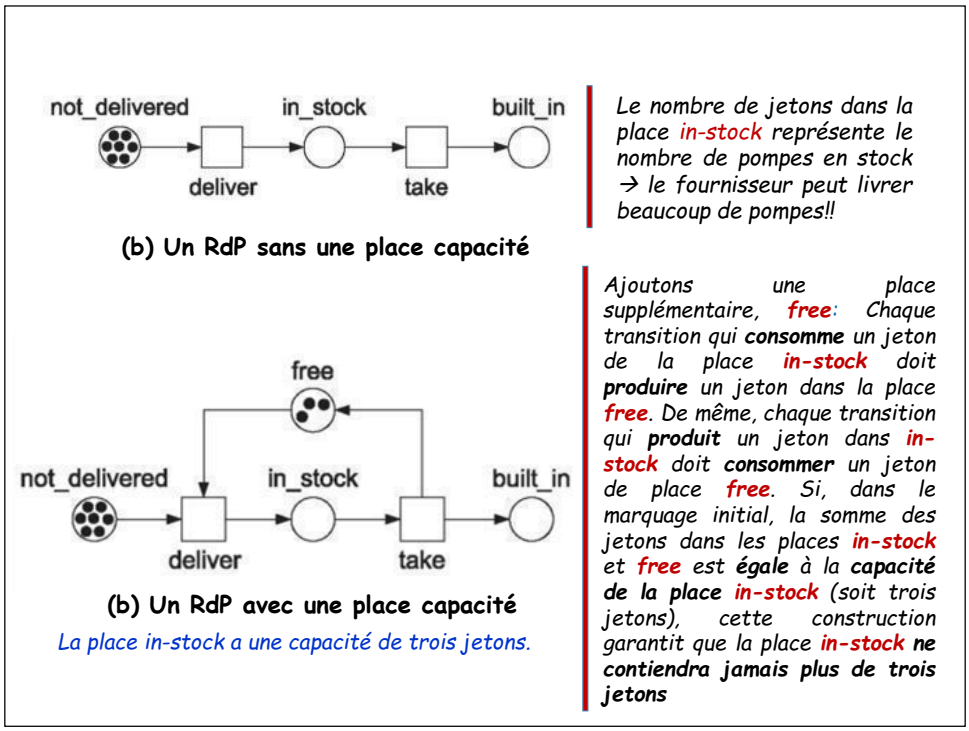


Figure 3.8: Exemple de modélisation avec capacité limitée.

ce qui peut engendrer des situations non désirables tels que:

- **Situation de famine (starvation)**: Un processus (une partie du réseau) est en situation de **famine** s'il se voit refuser l'accès à une ressource pendant un temps indéfini. En effet, il se peut qu'un processus soit privé, pendant une période indéfinie, d'une ressource, car la séquence des transitions franchies ne lui permet jamais de consom-

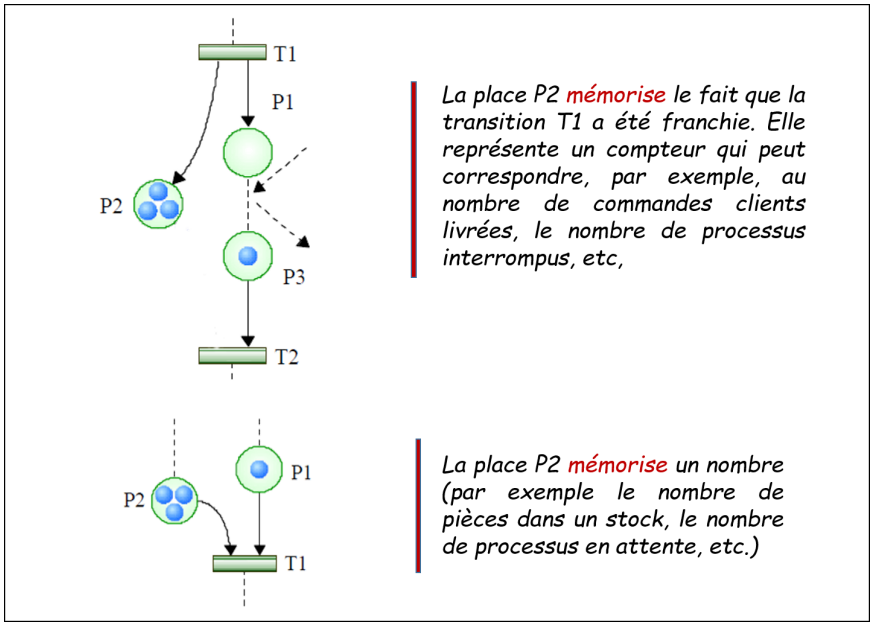


Figure 3.9: Modélisation de la mémorisation.

mer cette ressource. La Figure 3.10 montre un cas de famine.

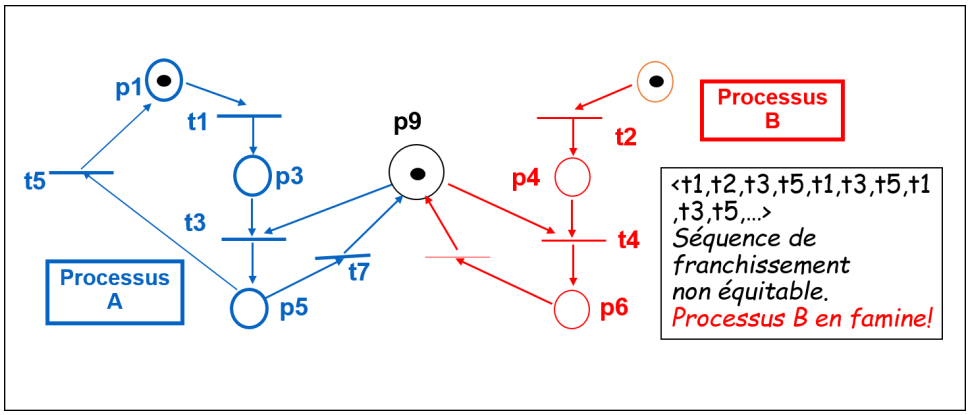


Figure 3.10: Situation de famine.

► **Situation d'interblocage (deadlock)**: un RdP est dit être en situation d'interblocage lorsque, dans un marquage donné M, aucune transition n'est franchissable, c'est à dire aucune action possible à partir d'un nœud accessible. La Figure 3.11 illustre un cas d'interblocage que nous pouvons déceler grâce à l'analyse du RdP.

3.3 Propriétés des Réseaux de Petri

L'analyse d'une fonction ou d'un système passe par l'étude des propriétés du réseau de Petri qui les représente. Citons, par exemple, les propriétés qui permettent d'affirmer que les spécifications incluses dans le modèle réseau de Petri sont correctes. Nous pouvons ainsi démontrer qu'un système modélisé par un RdP est sans blocage, que le nombre d'états pouvant être atteints est fini ou encore déterminer si une partie ou l'ensemble du réseau peut ou non évoluer. Nous pouvons aussi affirmer si le modèle comporte du code

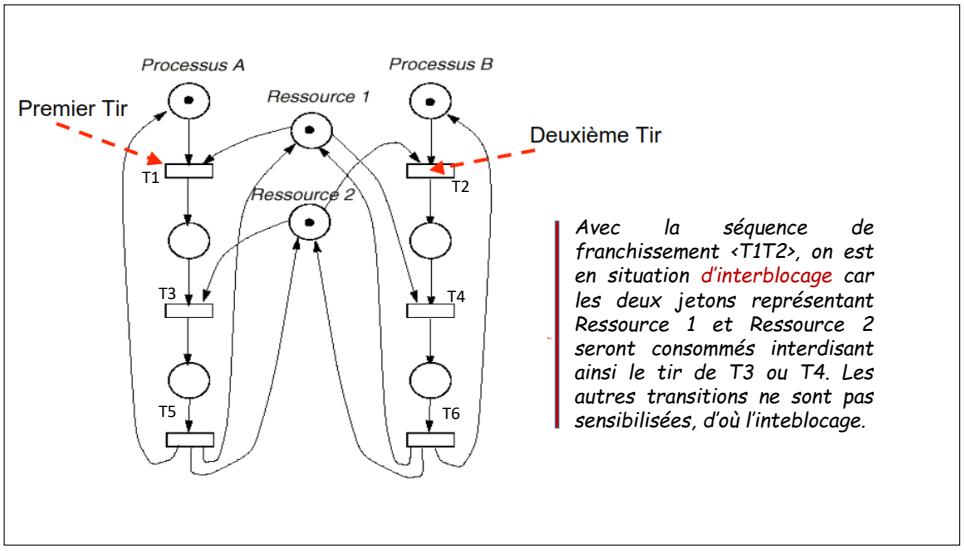


Figure 3.11: Situation d'interblocage.

mort, des boucles infinies, etc.

De nombreuses propriétés des RdP ont été mises en évidence [12]. Nous distinguons deux catégories, selon qu'elles concernent la structure du réseau (on parle alors de **propriétés structurelles**) ou bien son comportement (**propriétés comportementales**). Les propriétés structurelles ne dépendent que de la structure du RdP et non pas de la manière dont les jetons évoluent dans le réseau.

3.3.1 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 généralement indépendantes du marquage [15]. Ainsi, les propriétés structurelles d'un RdP consistent à déterminer:

- Si le RdP est pur ou non,
- S'il est généralisé ou ordinire,
- S'il présente des situations de conflits, etc.

3.3.2 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. Dans ce qui suit, nous allons décrire les propriétés les plus étudiées lors de l'analyse du système.

► ***Bornitude (Boundedness)***: Cette propriété définit et caractérise la possibilité pour une place d'accumuler une quantité bornée ou non de jetons au cours de l'évolution d'un réseau. L'absence d'une borne limite pour le nombre de jetons dans une place est généralement une source de problème. Un **RdP borné** a toujours un nombre fini de marquages.

> **Définition 3.3** :Une place P_i est **bornée** pour un marquage initial M_0 si pour tout marquage accessible a partir de M_0, le nombre de marques dans P_i reste borné.

Elle est dite **k-bornée** si le nombre de marques dans P_i est toujours inférieur ou égal à k. Un RdP marqué est **k-borné** si toutes ses places sont k-bornées.

$$P_j \text{ est k-bornée} \Leftrightarrow \forall\, M_i \text{ accessible depuis } M_0, \text{ et } P_j \in P, M_i(P_j) \leq k$$

Il faut parfois s'assurer qu'une place dans un RdP peut avoir une capacité limitée k; c'est-à-dire qu'à tout stade de l'exécution, cette place ne contient pas plus de k jetons (*upper Bound*). **Remarque** : Si **k=1** alors le RdP est dit **sauf ou binaire**. Dans le cas où il existe une place P_j non bornée alors tout le RdP est qualifié de non borné. La Figure 3.12 montre plusieurs exemples de RdP bornés et non-bornés.

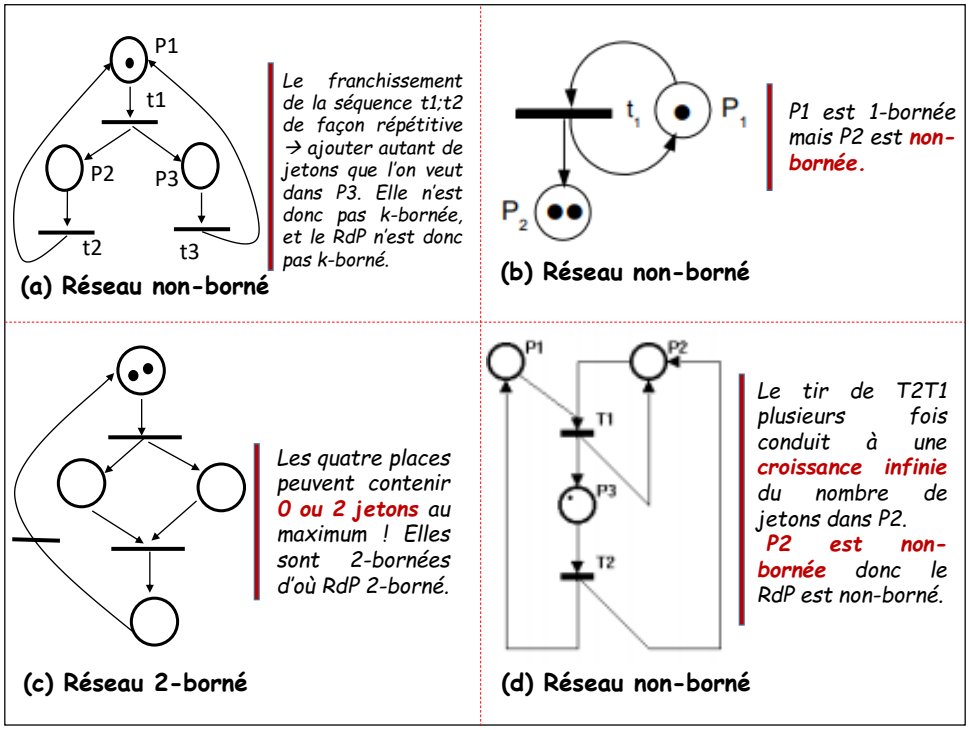


Figure 3.12: Exemple de réseaux bornés et non-bornés.

► ***Vivacité (Liveness)*** : Comme nous l'avons mentionné précédemment, l'évolution du marquage d'un RdP se fait par franchissement de transitions. Lorsqu'au cours de l'évolution du RdP, certaines transitions ne sont jamais franchies, cela signifie que les événements associés à ces transitions ne se produisent pas et que le marquage d'une partie du RdP n'évolue pas. Nous déduisons alors que le sous-système modélisé par cette partie ne fonctionne pas. La propriété de **vivacité** porte sur les transitions et permet d'examiner si une partie ou l'ensemble du réseau peut ou non **évoluer**.

Définition 3.4 : *Une transition T_i est **vivante** pour un marquage initial M_0 si pour tout marquage accessible M_k, il existe une séquence de franchissements à partir de M_k contenant T_i. Autrement dit, il sera toujours possible de franchir à nouveau T_i, peu importe les transitions déjà franchies. La vivacité est donc une propriété très forte.*
*Un **réseau est vivant** si toutes ses transitions sont vivantes.*

■ **Exemple 3.2** Pour illustrer l'intérêt de la vérification de cette propriété, considérons l'exemple d'un système dynamique destiné à fonctionner en permanence, tels que les systèmes de signalisation routière ou les systèmes d'information d'une organisation. Si le modèle RdP qui représente ce type de système comporte une transition qui n'est pas vivante et si une fonction du système est associée au franchissement de cette transition, cela veut dire qu'a partir d'un certain instant, cette fonction ne sera plus disponible dans le futur, ce qui peut traduire une erreur ou une panne. La Figure 3.13 (a) et (b) montre un exemple d'un réseau non vivant et d'un réseau vivant, respectivement. ■

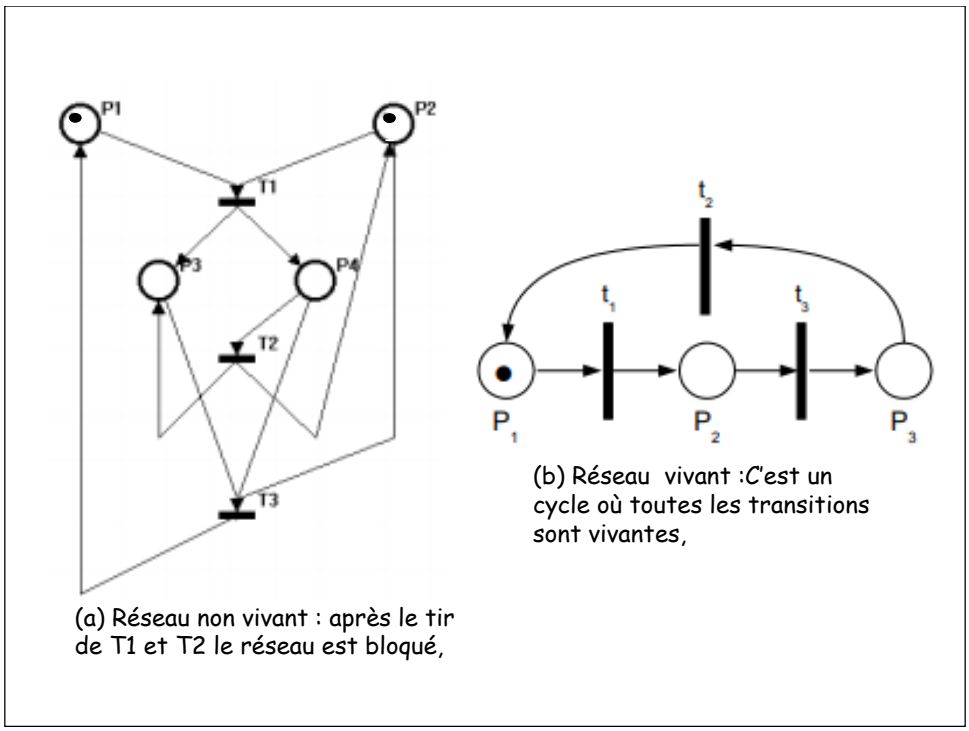


Figure 3.13: Exemple d'un réseau vivant.

► ***Quasi-vivacité***: Une transition T_j est quasi-vivante pour un marquage initial M_0 s'il existe une séquence de franchissements à partir de M_0 qui contient T_j. Il s'en suit qu'un **RdP est quasi-vivant** pour un marquage initial M_0 si toutes ses transitions sont quasi-vivantes pour ce marquage initial. Par conséquent, une transition qui n'est pas quasi-vivante est **inutile**!

La quasi-vivacité définit une propriété moins contraignante que la vivacité. Là où la vivacité exige que la transition soit franchissable à partir de tout marquage, la quasi-vivacité impose juste l'existence d'une séquence de transition permettant de franchir T_j , depuis le seul marquage initial. On peut donc dire de manière simple qu'un tel réseau quasi-vivant ne comporte pas de branches mortes pour le marquage initial, il existe toujours au moins un moyen de franchir chaque transition en partant de M_0. La Figure 3.14 montre un exemple d'une transition quasi-vivant.

► ***Blocage (deadlock)***: Un blocage (ou **état puits**) est un marquage terminal tel qu'aucune

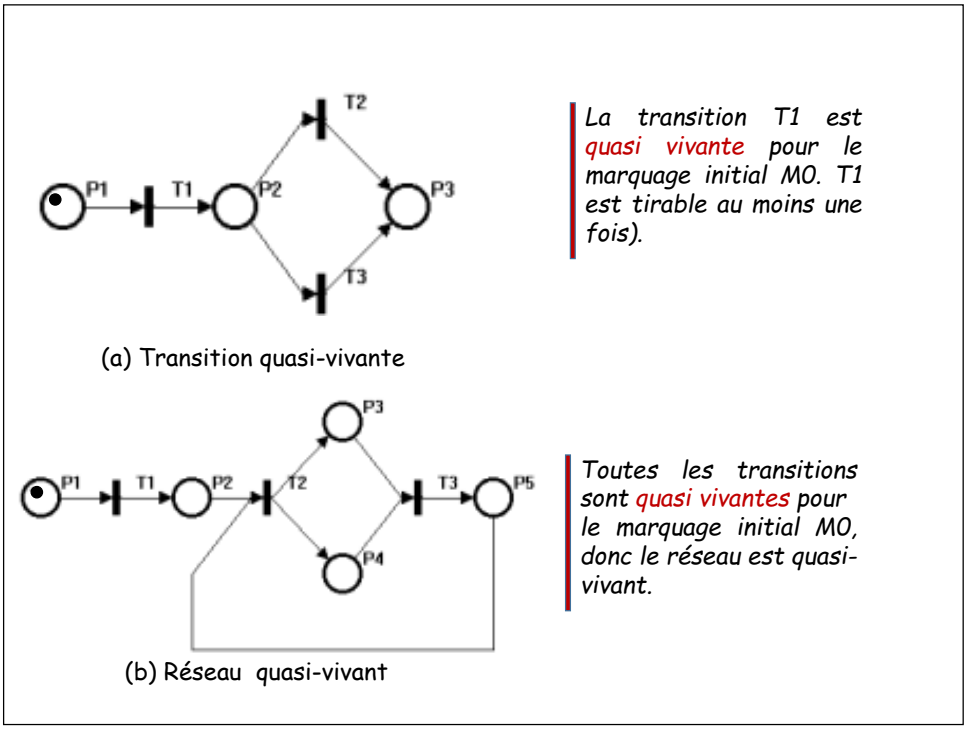


Figure 3.14: Exemple d'une transition quasi-vivante.

transition n'est franchissable. Un RdP est dit donc sans blocage (**sans deadlock, pas de deadmarkings**) pour un marquage initial M_0 si aucun marquage accessible M_i n'est un blocage.
Garantir l'absence de blocage d'un RdP est un résultat intéressant pour celui qui modélise. C'est une **propriété de correction** pour les systèmes qui sont supposés ne jamais terminer. La liaison entre les deux propriétés de vivacité et de blocage existe : **un RdP vivant est sans blocage**.

R Un réseau non vivant peut contenir ou non des blocages. La propriété d'absence de blocage est plus faible que celle de vivacité. Elle implique seulement que le réseau a toujours la possibilité d'évoluer.

La Figure 3.15 montre deux exemples de RdP avec Blocage.

- ***Transition morte (Dead transition)*** : Lorsqu'au cours de l'évolution du RdP, certaines transitions ne sont jamais franchies, cela signifie que les événements associés à ces transitions (mortes) ne se produisent pas et que le marquage d'une partie du RdP n'évolue pas. Nous déduisons alors que le sous-système modélisé par cette partie ne fonctionne pas; c'est un **code mort**.

Définition 3.5 Une transition T d'un réseau de Petri est **morte** si et seulement si **t** n'est sensibilisée à aucun marquage accessible. Ainsi, si un réseau de Petri est **vivant**, il n'a pas de transition **morte**.

- ***Accessibilité***: Un marquage M_i est **accessible** à partir d'un marquage initial M_0 s'il existe une séquence correcte de franchissements de transitions ***S*** telle que l'exécution

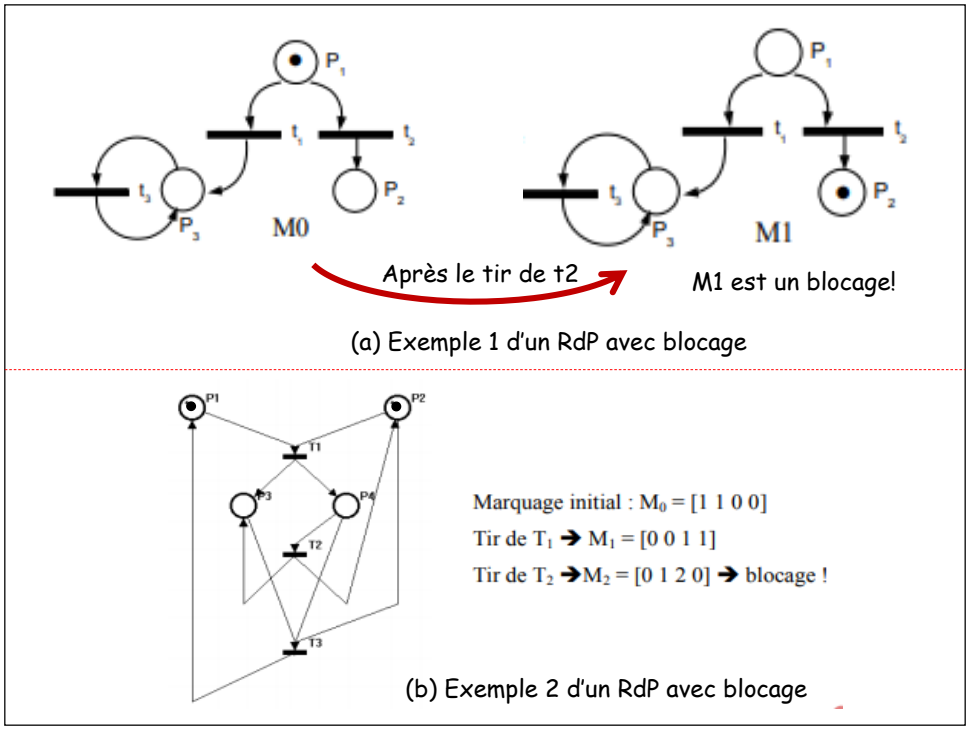


Figure 3.15: Exemple de RdP avec blocage.

de cette séquence à partir de M_0 produit le marquage M_i.
Cette propriété permet de vérifier si un état non désiré risque de se produire.

► ***Etat d'accueil (Home-marking) et réseaux ré-initialisables***: Un RdP marqué a un **état d'accueil** M_a pour un marquage initial M_0 si pour tout marquage accessible M_k à partir de M_0, il existe une séquence de franchissements permettant d'atteindre le marquage M_a. Si un RdP présente un **état d'accueil**, il est facile de vérifier s'il est sans blocage et d'étudier sa vivacité.

Un RdP est dit **réversible** ou **ré-initialisable** si son état initial M_0 est un état d'accueil, c'est-à-dire qu'il existe une évolution lui permettant de revenir au marquage initial. Cette propriété est très importante dans le cas de fonctionnements répétitifs comme la plupart des processus industriels. Il est donc important de vérifier si les réseaux qui les représentent sont réinitialisable (réversible). La Figure 3.16 montre un exemple d'un RdP ré-initialisable.

• ***Terminaison***: Un RdP **termine** s'il atteint toujours un **marquage terminal** à partir duquel aucune transition ne peut être tirée (marquage puits ou deadmarking). Un RdP se termine si et seulement si chaque exécution est finie. (voir Figure 3.17)
Un RdP avec un graphe d'accessibilité fini et acyclique termine, ce qui signifie que les boucles et les blocages empêchent les processus de terminer. Notons cependant que la vivacité et la terminaison s'excluent. Le blocage et l'inter-blocage sont généralement des terminaisons non désirées.

Remarques:

R Toutes les propriétés sus-citées sont définies pour un marquage M_0 donné.

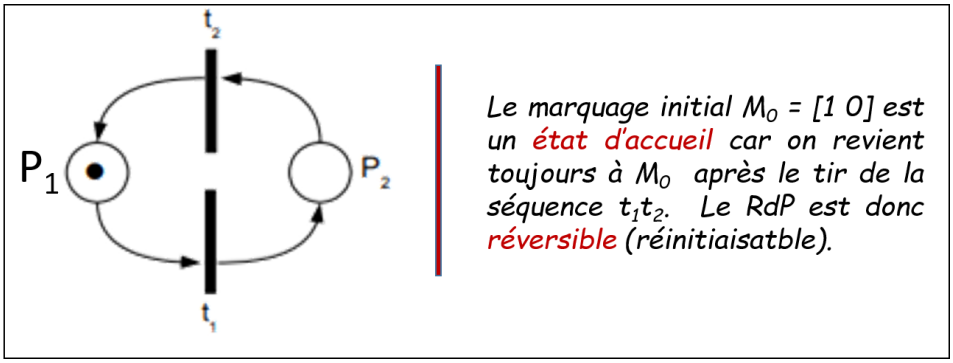


Figure 3.16: Exemple de RdP ré-initialisable.

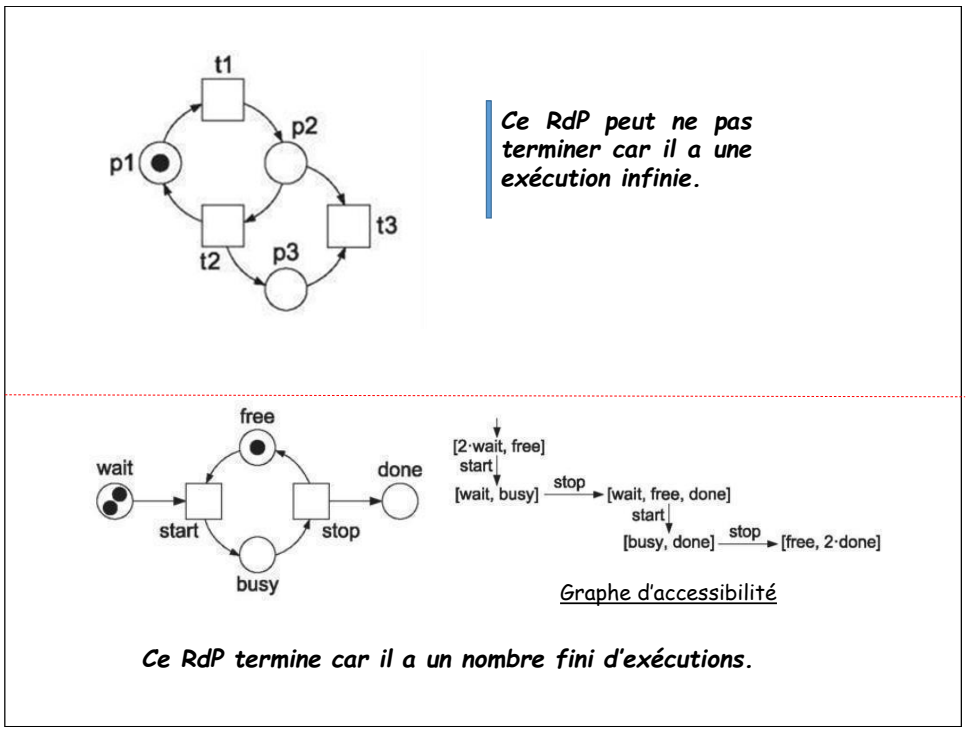


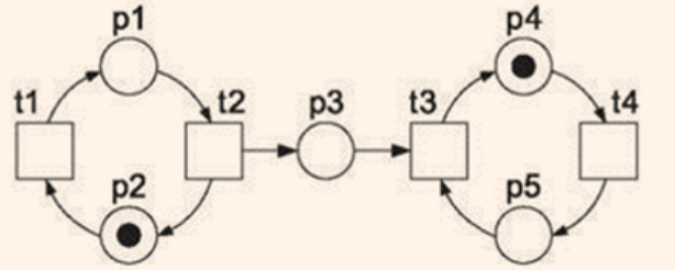
Figure 3.17: Exemple de terminaison de RdP.

Leur validité pour un autre marquage n'est en rien garantie.

R En plus des RdP purs (ou impurs), ordinaires (ou généralisé), sauf ou binaire, d'autres réseaux particuliers peuvent être à partir de la présence combinée de plusieurs propriétés tels que:

- RdP **conforme**: C'est un RdP qui est à la fois sauf et vivant (1 seul jeton par place, toutes les transitions sont franchissable).
- Rdp **propre**: C'est un RdP réinitialisable.
- RdP **bien formé**: C'est un RdP qui est à la fois : borné, vivant, propre.

Exercice d'application 3.1 Analyser si le système modélisé par le réseau de Petri suivant est borné, termine, vivant, sans blocage, réversible, ou a des transitions mortes ou des états d'accueil (home-markings).



Solution

- L'exécution **infinie** témoigne que la place **p3** et, par conséquent, le réseau est **non borné**
- **Absence de deadlock (blocage)**: L'exécution infinie permet de conclure que le système ne termine jamais ⇒ pas de deadmarkings (états terminaux).
- Le réseau est donc **vivant** et il n' y a pas de **transitions mortes**
- Chaque marquage accessible est un **home-marking (état d'accueil)**, le réseau est donc **réversible**.

3.3.3 Composantes Conservatives et Invariants

Outre la vérification des propriétés génériques telles que celles exposées précédemment, il existe d'autres propriétés plus spécifiques telle que l'existence d'**invariants** qu'il serait aussi intéressant de vérifier pour certains systèmes.

Dans ce qui suit, on va s'intéresser aux invariants de places dont l'existence permet de déterminer si les contraintes du système modélisé ont été respectées. Citons, par exemple, le cas du problème des lecteurs-rédacteurs où certaines contraintes doivent être respectées telles que: le nombre de lecteurs accédant simultanément à la ressource est limité à 3, le nombre de rédacteurs accédant simultanément à la même ressource est limité à 1, et il ne peut pas y avoir simultanément un lecteur et un rédacteur accédant à cette ressource.
Considérons le RdP de la Figure 3.18. On remarque que pour le sous-réseau, *SR*, formé des places (1, 2) et des transitions (*a*, *b*), on a: $M(1)+M(2)=1$, quel que soit la

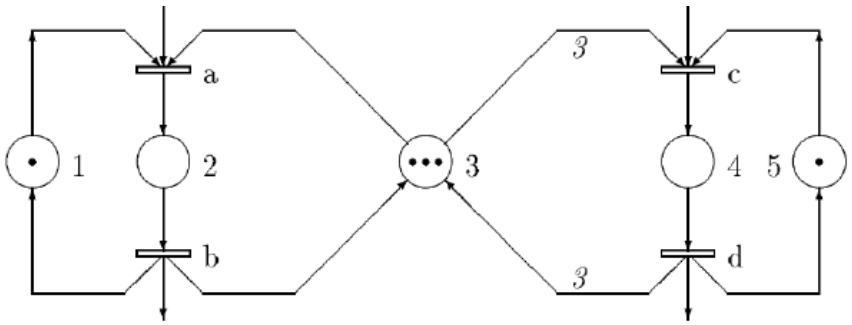


Figure 3.18: Exemple de p-invariant.

séquence de franchissement à partir de M_0; Ainsi, $\forall$ le marquage initial M_0, on a : $M(1)+M(2)=M_0(1)+M_0(2)$

Le sous réseau *SR* est appelé **composante conservative** et La forme linéaire **M(1) + M(2) = 1** étant appelé **invariant linéaire de place**.

Un invariant du marquage de places (ou **p-invariant**) est une propriété du marquage d'un ensemble de places du réseau. On est en présence d'un invariant de marquage si, pour un ensemble donné de places, il existe une combinaison linéaire des marques présentes dans les places de valeur constante indépendamment de M_0 quelque et soit l'évolution du réseau (c'est à dire pour tous les marquages accessibles).

La Figure 3.19 montre un exemple d'un RdP avec deux invariants de marquage de places. Cependant, il est difficile de déterminer "visuellement" l'existence d'**invariants** dans un

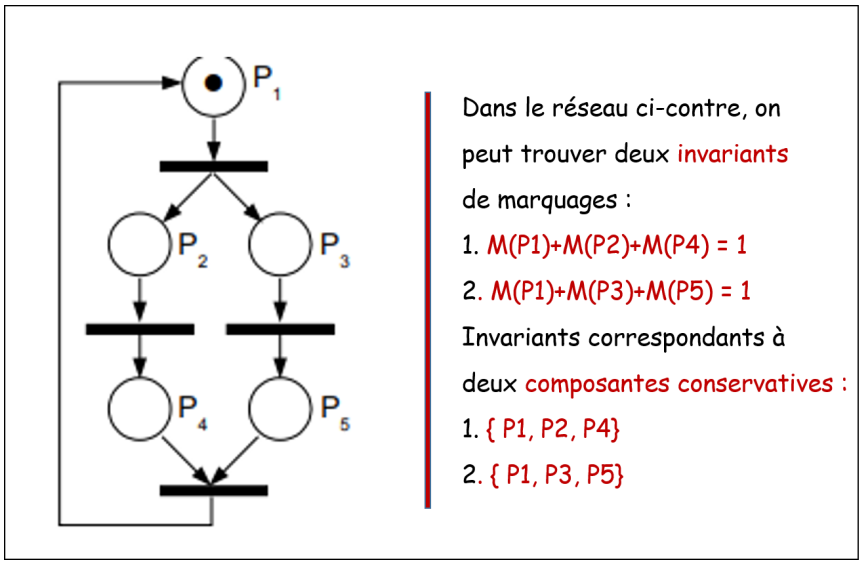


Figure 3.19: Exemple d'invariants de marquage.

RdP complexe. C'est pour cette raison qu'on a recours à la méthode algébrique. Pour cela, nous définissons les concepts suivants:

Définition 3.6 :
- Soit ***F*** un vecteur colonne de pondération des places; $F^T x\, M(P)$ représente une combinaison linéaire des places:

$$F^T\ x\ M(P) = f1 * M(P1) + f2 * M(P2) + \ldots + fn * M(Pn)$$

- Rappelons que l'équation d'état est : $M' = M + W * \underline{S}$;

- En multipliant chaque terme par F^T: F^T x $M' = F^T$ x $M + F^T$ x W x $\underline{S}$

- S'il existe un F tel que : F^T x $W = 0$ alors

- F^T x $M' = F^T$ x M et F^T x $M' = K_0$, indépendamment de la séquence S.

On définit:
- **F** est l'**invariant** avec $F \neq 0$,

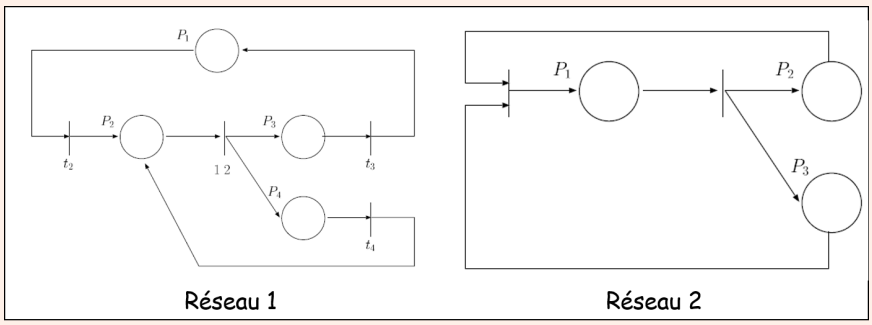
- F^T **x** W est la **composante conservative**,
- F^T **x** $M' = F^T$ **x** M correspond à l'invariant linéaire de place, **P-invariant**.

Remarques :

R Si une **composante conservative** est telle que $K_0 = 0$ alors :
- Aucune des ses places n'est marquée dans M_0,
- Elles restent vides quelque soit le marquage,
- C'est anormal; C'est probablement une erreur de modélisation.

R Si une **composante conservative** est telle que $K_0 = 1$ alors:
- Il y a toujours une de ces places marquées d'un jeton, et pas plus d'un jeton,
- Ces places sont booléenne (1-bornée),
- Si un RdP est tel que toutes ses places appartiennent à des invariants de place avec $K_0 = 1$ alors le RDP est **sauf**

Exercice d'application 3.2 Considérons *Réseau 1* et *Réseau 2* de la figure ci-dessous. Déterminer si ces réseaux possèdent des $p-invariants$?



Solution

► **Réseau 1** : Soient $W1$ la matrice d'incidence du *Réseau 1*, et $F1 = (x_1, x_2, x_3, x_4)$ le vecteur de pondération de places. Existe t-il un $F1$ **tel que** $F1$ ***** $W1$ **= 0?**

$$\underset{F1}{\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}} * \overset{W}{\begin{bmatrix} -1 & 0 & 1 & 0 \\ 2 & -1 & 0 & 1 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -2 \end{bmatrix}} = \begin{bmatrix} -x_1 + 2x_2 = 0 \\ -x_2 + x_3 + x_4 = 0 \\ x_1 - x_3 = 0 \\ x_2 + 2x_4 = 0 \end{bmatrix}$$

D'où $F1 = (0,0,0,0) \Rightarrow$ pas de p-invariant pour le *Réseau 1*.

► **Réseau 2**: On procède de la même manière que pour *Réseau 1* avec $W2$ et $F2$: Existe t-il un $F2$ **tel que** $F2$ ***** $W2$ **= 0?**

$$\underset{F2}{\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}} * \overset{W}{\begin{bmatrix} 1 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix}} = \begin{bmatrix} x_1 - x_2 - x_3 = 0 \\ -x_1 + x_2 + x_3 = 0 \end{bmatrix}$$

$\Rightarrow x_1 = x_2 + x_3 \Rightarrow$ **(1 1 0) et (2 1 1)** sont des **p-invariants** et il y en a une infinité.

Propriétés :

1. Si un RdP possède un p-invariant alors une combinaison linéaire des nombres de marques dans ses places est **constante**.
2. Si un RdP possède un p-invariant dont toutes les composantes sont strictement positives alors il est **borné**.
3. Si un RdP possède un p-invariant dont toutes les composantes sont égales à 1, alors le nombre de marques dans ce réseau est **constant**.

3.4 Conclusion

Dans ce chapitre, nous avons commencé par présenter les éléments nécessaires à la modélisation de différents comportements tels que le parallélisme, le conflit, l'exclusion mutuelle, la contrainte de capacité limitée, etc. Ensuite, nous avons mis l'accent sur les propriétés structurelles et comportementales qu'un système modélisé devrait satisfaire. Nous avons aussi défini la notion d'invariant de marquage de places et la composante conservative qui sont très importantes pour l'analyse du modèle de manière algébrique.
Dans le chapitre suivant, nous allons exposer les différentes extensions qu'a connu le formalisme des RdP, à savoir les réseaux de haut niveau.

3.5 Exercices

Exercice 3.1

Un ordinateur (PC) est composé d'une unité centrale (UC) et d'un écran. L'écran et l'UC peuvent être allumés et éteints à l'aide d'un switch (interrupteur).

1. Modéliser le processus allumer/ éteindre de l'écran et de la tour par un réseau de Petri.
2. Ajustez ce modèle de manière à ce que l'écran ne puisse être allumé qu'après la mise en marche de la tour.

Exercice 3.2

Un hôpital a un système d'information périmé pour soutenir l'administration financière. Pour remplacer les informations actuelles, l'hôpital lance un grand projet d'automatisation. Au début du projet, les exigences fonctionnelles sont collectées. Ensuite, l'équipement requis (matériel) et, en même temps, le logiciel requis sont achetés. Une fois le matériel et le logiciel acquis, le système d'information peut être intégré. Après avoir testé le système d'information, le projet est terminé.

1. Identifier à partir de cette description les différentes actions du processus d'automatisation.
2. Déterminer les liens de causalité (séquence), de concurrence et de synchronisation entre les différentes activités.
3. Donner le RdP du processus d'automatisation.

Exercice 3.3

Un hôpital emploie quatre cardiologues qui peuvent être soit dans l'état actif (« active) soit dans l'état en attente (« stand-by »). Un cardiologue en attente peut être appelé à tout moment et devient actif. Après un certain temps, ce cardiologue retourne à l'état stand-by. Dans l'état initial, il y a deux cardiologues actifs et deux en attente.

1. Modéliser le processus autour de ces quatre cardiologues par un réseau de Petri.

2. Comment le modèle devrait-il être ajusté si au plus deux cardiologues peuvent être actifs ?

Exercice 3.4

1. Modéliser par un RdP les feux de circulation.
2. Considérons la situation dans laquelle il y a deux feux de circulation au croisement de deux routes à sens unique. Ils doivent travailler de telle sorte qu'il y ait toujours au moins un feu rouge ; autrement dit, au plus un feu de circulation est vert ou orange. Modéliser cette situation par un RdP.
3. Déterminer le graphe d'accessibilité du RdP de la question 2.

Exercice 3.5

Dans une clinique, les patients consultent des spécialistes. Chaque patient a un rendez-vous avec un certain spécialiste. Nous décrivons le cours des affaires autour d'un spécialiste en tant que modèle de processus. Nous utilisons comme formalisme un réseau de Petri. Le spécialiste reçoit des patients. À chaque moment, le spécialiste est dans l'une des trois situations suivantes:

- le spécialiste est libre et attend le patient suivant,
- le spécialiste est occupé à traiter un patient,
- le spécialiste documente le résultat du traitement,

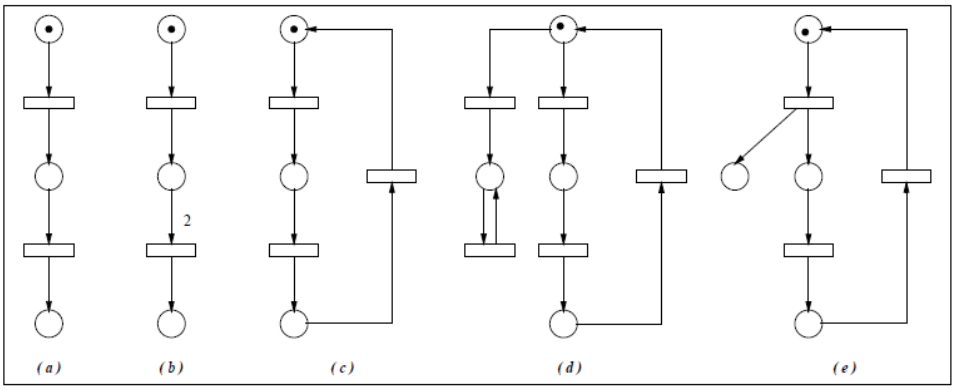
Chaque patient qui visite un spécialiste est dans l'une des trois situations suivantes:

- le patient attend,
- le patient est en cours de traitement par le spécialiste,
- le patient a été traité par le spécialiste

1. Proposer un RdP pour modéliser ce système.
2. On considère qu'à un instant donné, il y a 3 patients en attente, un patient qui fini sa visite médicale libérant ainsi le spécialiste. Quel est le marquage M correspondant à cet instant ? Quels sont les marquages accessibles depuis M en tirant une transition à la fois?
3. Quels sont les marquages terminaux (finaux) accessibles à partir de M ?
4. Comment devrions-nous ajuster la structure ou le marquage du réseau de Petri proposé dans le cas de trois spécialistes?

Exercice 3.6

Soient les réseaux représentés par la figure suivante:



Déterminer les RdP bornés, vivants et quasi-vivants.

Exercice 3.7

Proposer un réseau de Petri qui modélise une chaîne automatisée de peinture de portes d'automobile. Les portes et la peinture sont supposés être en quantité infinie et sont donc non modélisés. Pour peindre une porte il faut :

- Acquérir un robot pour déplacer la porte vers la zone de peinture.
- Acquérir une machine à peindre : on libère alors le robot qui n'a plus d'utilité. Une fois la porte peinte il faut :
- Acquérir de nouveau un robot pour déplacer la porte peinte : on libère alors la machine à peindre
- Délivrer la porte peinte : on libère alors le robot.

Initialement, on suppose qu'on a x portes ($x \geq 0$), r Robots ($r \geq 0$) et m machines ($m \geq 0$). Le RdP proposé présente t-il des invariants?

Exercice 3.8

On désire modéliser un ascenseur qui se déplace entre le 1^{er} et le 2^{eme} étage ainsi que les passagers qui arrivent au 1^{er} et se rendent au 2^{eme}. On ne représente pas l'appel de l'ascenseur ni le choix de l'étage de destination du moment qu'il n'y a qu'un seul.

1. Proposer un RdP qui modélise le fonctionnement de cet ascenseur en considérant ce qui suit:
   - L'ascenseur peut monter et descendre, qu'il soit vide ou non;
   - Un passager ne peut entrer que si l'ascenseur est au 1^{er} et il ne peut sortir que si l'ascenseur est au 2^{eme};
   - L'ascenseur peut monter avant que tous les passagers ne soient entrés.
2. Etendre le RdP pour représenter également des passagers qui arrivent au 2^{eme} et veulent se rendre au 1^{er} .
3. Modifier le RdP pour modéliser la capacité limitée de l'ascenseur et qui est de 10 passagers au maximum.

Exercice 3.9

On considère le protocole suivant de gestion des cabines et des paniers d'une piscine. A l'entrée, un client qui a trouvé une cabine libre y entre et se change en posant ses vêtements dans la cabine. Il demande ensuite un panier qu'il remplit pour libérer la cabine. Après la baignade le client rentre dans une cabine avec son panier, le vide et le libère. Ensuite il se rhabille et libère la cabine.

Soient Nc le nombre de cabines et Np le nombre de panier.

1. Décrire ce protocole par un RdP avec $Nc = 3$ et $Np = 5$.
2. Montrer qu'il y a un état de blocage. Y-a-t-il blocage pour toute valeurs de Nc et de Np?
3. Définir un protocole tel qu'il n'y ait pas de blocage et donner le RdP correspondant.
4. Modifier le RdP de 2. pour modéliser le nombre de clients qui attendent une cabine pour entrer à la piscine.

Conseil : Avant même d'établir le RdP, analyser ce protocole et détecter le blocage qui peut apparaître et pourquoi.

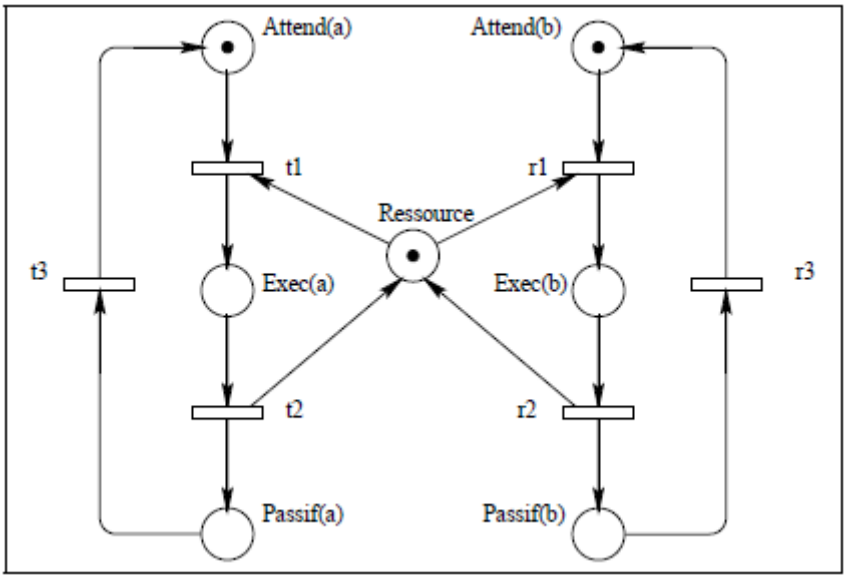
Exercice 3.10

Une administration fait entrer des clients puis ferme la porte d'entrée avant de commencer le service. Au fur et à mesure qu'ils sont servis les clients sortent par une autre porte. La porte d'entrée ne sera rouverte que lorsque tous les clients qui étaient entrés seront sortis. Donner le RdP correspondant.

Conseil : Utiliser la notion d'arc inhibiteur.

Exercice 3.11

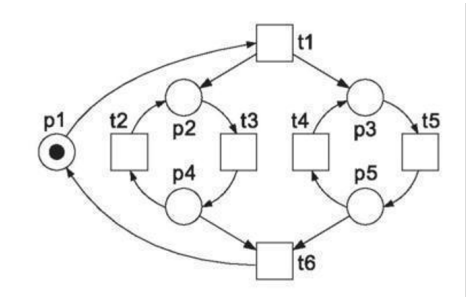
Soit le RdP de la figure suivante modélisant l'exclusion mutuelle de deux processus sur une ressource:

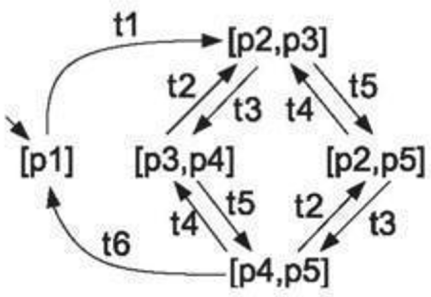


1. Prouver qu'il y a bien une exclusion mutuelle entre les processus a et b en construisant le graphe des marquages accessibles.
2. Même question en utilisant l'équation fondamentale pour trouver le vecteur $p-invariant$.

Exercice 3.12

Soit le RdP et son graphe des marquages accessibles de la figure suivante:





Déterminer les différentes propriétés de ce RdP.

4. Réseaux de Petri de Haut Niveau

4.1 Introduction

A la différence des autres formalismes de modélisation, notamment les files d'attentes et les chaînes de Markov, les RdP semblent bien appropriés pour décrire les systèmes dans lesquels interviennent les problèmes de concurrences, de synchronisation, de partage de ressources et de parallélisme. Néanmoins, les RdP ordinaires tels qu'ils ont été décrit dans les chapitres précédents montrent des insuffisances critiques quand les systèmes à modéliser sont très complexes, larges ou exhibant des caractéristiques temporelles ou aléatoires. On distingue trois problèmes sérieux des RdP ordinaires, à savoir:

Problème 1: Grande taille du réseau

Modéliser un système industriel réel par des RdP ordinaires peut générer des modèles de **taille trop importante** rendant leur manipulation et leur analyse difficile voire impossible. Pour illustrer ce problème, considérons l'exemple suivant:

■ **Exemple 4.1** : Modélisation d'un système de gestion d'inventaire

Il s'agit de modéliser un système d'information pour maintenir à jour la quantité stock de chaque produit. Pour ce faire, nous proposons le modèle RdP suivant:

- Modéliser chaque type de produit par une place. S'il y a cinq produits, nous avons besoin d'au moins cinq places.
- Il existe deux types d'événements que nous modélisons par des transitions: le stock d'un certain produit peut être augmenté ou diminué. Pour chaque produit, il y a donc une transition pour augmenter le stock et une pour diminuer le stock. Par conséquent, pour 5 produits nous avons besoin d'au moins 5*2 = 10 transitions. Le RdP de la Figure 4.1 modélise une partie d'un système d'inventaire.

■

- ***Constat:*** Pour 05 produits ⇒ 15 Places et 10 transitions. Pour 100,000 Produits en stock ⇒ Le RdP doit contenir 100 000.3.300.000 Places et 200,000 Transitions!

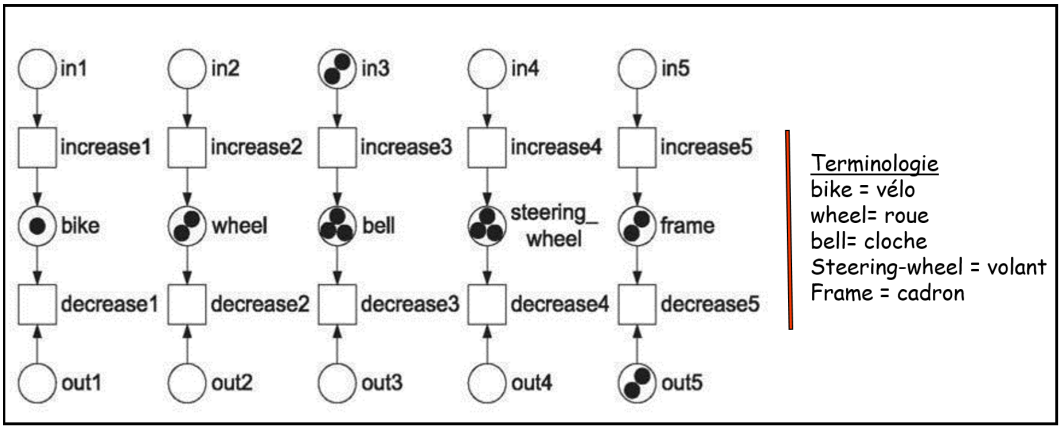


Figure 4.1: Exemple d'un RdP pour la gestion d'inventaire.

Il est clair que nous ne pouvons pas facilement modéliser de telles situations comme un réseau de Petri, car les réseaux résultants sont **trop grands** pour être manipulés!

Problème 2: Puissance expressive limitée

Dans les systèmes industriels, il est fréquent que des conditions telle que $x < y$ se produisent. Pour modéliser de tels systèmes, nous avons donc besoin d'un moyen pour exprimer des conditions. Or, ceci ne peut être exprimé par les RdP ordinaire à cause de jetons indiscernables. La valeur représentée par un jeton ne peut être déterminée que par la place qui le contient.

Problème 3: Aucune modélisation explicite du temps

Il est facile d'exprimer la causalité entre les événements dans un réseau de Petri ordinaire, mais il est difficile d'indiquer quand un événement a lieu et combien de temps cela prend. Or, et comme les transitions sont instantanées et donc intemporelles, les réseaux de Petri ordinaires ne peuvent pas décrire les aspects temporels d'un système, il n'est donc pas possible d'évaluer la performance du système modélisé. Pour mieux illustrer ce problème, considérons l'exemple suivant:

▪ **Exemple 4.2** Sur la base d'un modèle de système de production d'une usine automobile, nous voudrions déterminer combien de voitures l'usine peut produire chaque semaine. ▪

La performance d'un système est souvent exprimée en termes de temps d'écoulement, de temps d'attente, d'utilisation, de coûts et d'articles pour chaque unité de temps.

En résumé, les RdP ordinaires peuvent modéliser les relations causales entre différents événements mais ils peuvent devenir trop grands, ils ne sont pas assez expressifs et ils ne modélisent pas explicitement le temps. Les deux premières faiblesses sont causées par l'utilisation de jetons noirs indiscernables, quant à la troisième faiblesse, elle est causée par le fait que le tir des transitions ne tient pas compte du temps.

Pour pallier à ces problèmes, de nombreuses variantes et extensions des réseaux de Petri qui introduisent la notion de couleur de jeton (une couleur est un type de données) pour un réseau plus compacte, la notion du temps, de prédicats, sans parler des notions stochastiques, temporelles, objets, flous, structures algébriques, continus, hybrides, etc.

Dans ce qui suit, nous présentons les extensions les plus répandues, à savoir les RdP colorés, temporisés, hiérarchiques et stochastiques. Toutes ces versions de RdP sont qualifiées de RdP de haut niveau.

4.2 Les Réseaux de Petri Colorés (RDPC)

Il s'agit de passer des jetons **indiscernables** à des jetons **reconnaissables** en affectant à chaque jeton du RdP une certaine valeur représentant la nature de la donnée stockée, appelée **couleur**. Cette couleur est une valeur typée qui a un rôle similaire au type d'une variable dans un programme informatique. Cette sémantique de différenciation des jetons amène une grande expressivité en plus d'une compacité du modèle. Les RdP étendus avec la couleur sont les **RdP colorés**.

4.2.1 Description Informelle des RdP Colorés

Ce type de RdP a été introduit afin de permettre la modélisation de systèmes ayant des comportements symétriques avec des réseaux de taille réduite et sans perte d'informations. L'exemple illustratif suivant permet de mettre en exergue l'utilité des RdP colorés.

■ **Exemple 4.3** : *Confection des cartes perforées dans un hôpital*
Dans un hôpital, il y a un bureau de service où les patients peuvent se faire une carte perforée. Cette carte perforée contient des informations sur le nom, l'adresse, la date de naissance et le sexe du patient. De plus, chaque patient reçoit un identifiant. Il y a un employé qui travaille au bureau de service. A tout moment, l'employé peut aider au plus un patient. Si l'employé sert un patient, alors le bureau est occupé; sinon, il est libre. Nous supposons que le temps qu'il faut pour faire une carte perforée dépend de l'expérience (exprimée en années) de l'employé. Pour cette raison, le numéro et le nombre d'années d'expérience de l'employé du centre de service sont enregistrés.

Tout d'abord, nous allons modéliser ce processus par un **RdP ordinaire**. Nous distinguons (voir Figure 4.2):
- Etats du Patient: En attente (Wait), En cours (inHandling) et Terminé (done),
- Etats de l'employé de bureau: Libre (free) et Occupé (busy),
- Transitions: Début (Start) et Fin (Stop).

■

Commentaires :
- Dans ce RdP, plusieurs aspects ne sont pas encore considérés. Premièrement, nous ne pouvons pas retrouver des informations sur les patients et les employés. Pour inclure cette information, nous avons besoin d'un grand nombre de places et de transitions.
- Dans le RdP de la Figure 4.2, un jeton dans la place *Wait* ou *Done* modélise un patient. Or dans cet exemple, nous voulons distinguer entre les différents patients en incluant des informations sur l'identifiant, le nom, l'adresse, la date de naissance et le sexe du patient.

Cependant, dans un réseau de Petri ordinaire, il n'est pas possible de décrire les **attributs d'un jeton**. Il est donc naturel d'étendre les réseaux de Petri de telle manière que chaque jeton ait une valeur. Un jeton qui représente un certain patient a alors une valeur à partir de laquelle nous pouvons dériver les cinq attributs sus-cités. La Figure 4.3 présente un exemple d'une telle valeur. La place *Wait* contient un seul jeton. Ce jeton a la valeur:

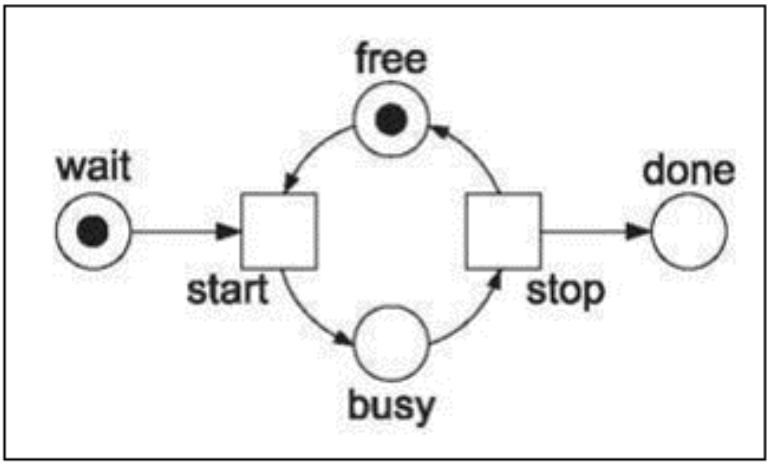


Figure 4.2: Modèle RdP ordinaire du bureau de service des cartes perforées.

(12345, Peter, "Kerkstraat 10, Amesterdam", 13-Dec-1962, male)

Il représente la requête du patient *Peter* pour une carte perforée. *Peter* est de sexe masculin qui vit à $Kerkstraat 10, Amesterdam$ et il est né le 13 décembre 1962. Son identifiant est 12345.

De même, un jeton dans la place *free* modélise un employé caractérisé par un numéro-employé et un nombre d'années d'expérience. Le jeton dans la Figure 4.3 a la valeur: *(641112, 7)* et représente un employé avec un numéro 641112 et une expérience de 7 années.

Dans un RdP coloré, une place contient des jetons similaires, c'est-à-dire des jetons

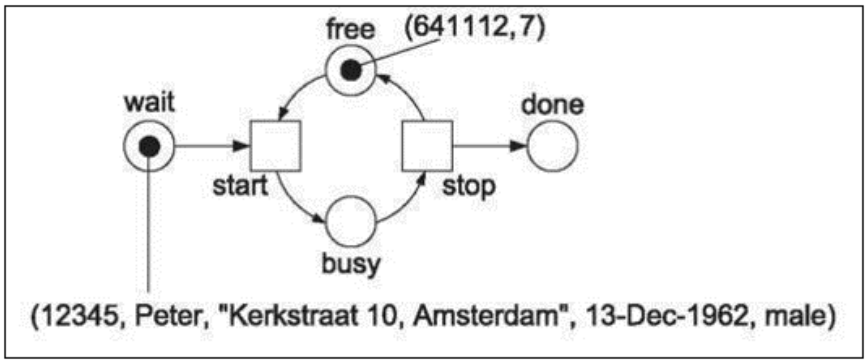


Figure 4.3: Modèle RdP coloré du bureau de service des cartes perforées.

représentant le même type d'objet. Par exemple, la place *wait* ne contient que des jetons qui représentent les patients. La place *free*, en revanche, contient seulement des jetons qui représentent des employés.

Chaque place est donc associée à un certain **type**. Ce type de place, appelé **ensemble de couleurs**, indique quel type de jetons cette place peut contenir. La valeur d'un jeton doit être conforme au type de la place correspondante. Dans l'exemple précédent, nous distinguons les types de places suivants:

- Le type de la place *wait* est le produit cartésien:
  Patient = Id x Nom x Adresse x Date-naissance x sexe
- Le type de la place *free* est le produit cartésien: **Employé = NumEmp x Expérience**
- Un jeton dans la place *busy* représente un employé occupé à traiter les données d'un patient. La place est donc de type **(Employé, Patient)**. Une valeur possible est:
  (*(641112, 7)*, *(12345, Peter, "Kerkstraat 10, Amesterdam", 13-Dec-1962, male)*)
- Un jeton dans la place *done* représente un patient avec une carte perforée; par conséquent, le type de cette place est **Patient.**

Comme nous l'avons déjà mentionné, le marquage d'un RdP ordinaire est déterminé en donnant le nombre de jetons pour chaque place. Cependant, pour décrire le marquage d'un réseau de Petri coloré, nous devons également décrire **les valeurs** de ces jetons, comme le montre la Figure 4.4.

Place	Valeur
wait	(12345, Peter, "Kerkstraat 10, Amsterdam", 13-Dec-1962, male)
free	(641112, 7)
busy	
done	

Figure 4.4: Marquage du RdP coloré de la Figure 4.3.

4.2.2 Comportement d'un RdP Coloré

Le comportement des RdP colorés est similaire à celui des RdP ordinaires. Une transition est franchissable et peut être tirée s'il y a assez de jetons dans ses places en entrée.

Comme exemple, considérons le RdP de la Figure 4.5. La transition *start* est sensibilisée car il y a un patient dans la place *wait* et un employé dans la place *free* (voir Figure 4.5 (a)). Le tir de cette transition va consommer un patient de la place *wait*, un employé de la place *free* et va produire un jeton dans la place *busy* (voir Figure 4.5 (b)). La valeur de ce jeton contient des informations sur l'employé et le patient pour lequel la carte perforée est confectionnée. Pour le RdP coloré de la Figure 4.5 (b), le jeton produit dans la place *busy* a pour valeur (couleur):
(*(641112, 7)*, *(12345, Peter, "Kerkstraat 10, Amesterdam", 13-Dec-1962, male)*)

La Figure 4.5 (c) montre le marquage du RdP coloré après le tir de la transition *stop*.

Cependant, dans certains cas le franchissement d'une transition dans un RdP coloré peut nécessiter une règle additionnelle que nous allons illustrer par un exemple.

▪ **Exemple 4.4** : *Vérification de la qualité du produit dans une chaîne de fabrication*.
Supposons que trois produits *A*, *B* et *C* sont fabriqués. Une machine analyse chaque produit. Avec ce balayage, la machine reconnaît si la qualité du produit est faible **low** ou élevée

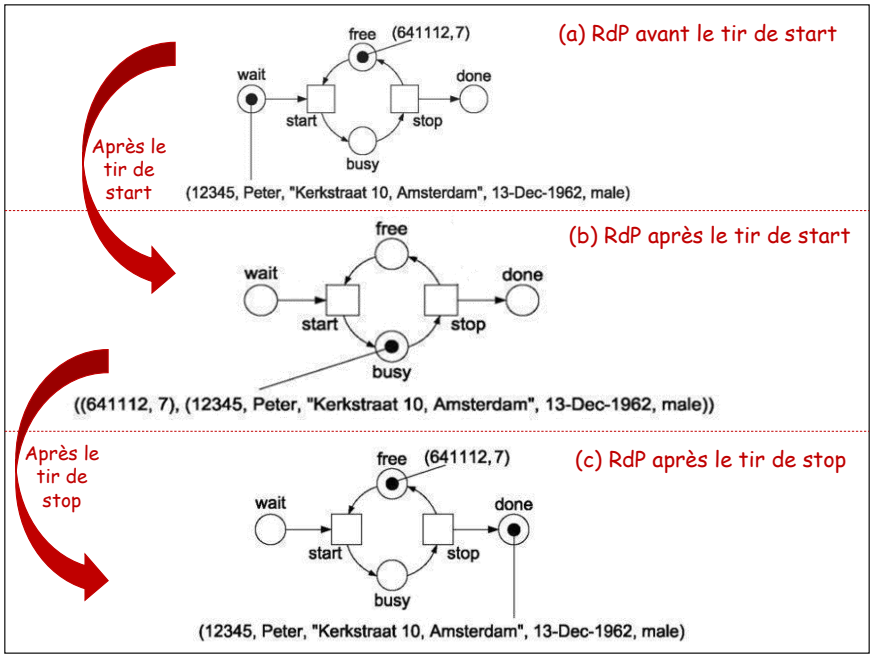


Figure 4.5: Comportement d'un RdP coloré.

high. Les produits de qualité inférieure sont placés dans un seul contenant et les produits de haute qualité sont placés dans un autre.

- Un produit peut être modélisé par **3 états** : Non-contrôlé , Qualité faible et Haute qualité. Ces trois états sont modélisés respectivement par les places *unchecked*, *low*, et *high*.
- Les transitions $check - low$ (c'est-à-dire que le produit est de qualité médiocre) et $check - high$ (c'est-à-dire que le produit est de haute qualité) modélisent le **scan**.

■

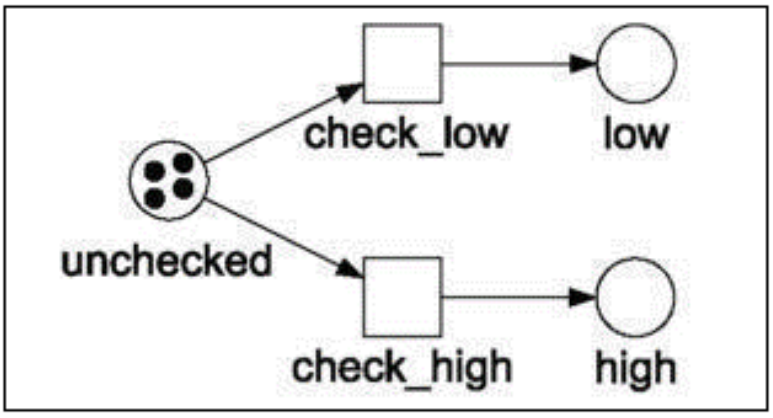


Figure 4.6: Un RdP ordinaire modélisant le contrôle de qualité d'un produit.

La Figure 4.6 présente un RdP ordinaire modélisant le contrôle de qualité d'un produit.
- Constat:

- Ce modèle ne spécifie pas le type d'un produit: s'agit-il du produit A, B ou C? Tous les produits sont similaires!

- La décision sur la qualité des produits est modélisée par un choix non déterministe.

Pour pallier à ces problèmes, nous étendons le modèle RdP en ajoutant de la **couleur**:

- Un produit peut être défini par son identité et sa qualité. De ce fait, on déclare le type ***Product*** comme suit:
  Product = Prodtype x Quality ; où ***ProdType*** et ***Quality*** sont des types énumérés dont les valeurs sont respectivement $[A, B, C]$ et $[High, Low]$.
- La place *unchecked* est alors de type *Product*. Par exemple, un jeton dans la place *unchecked* peut avoir la valeur (A, low), (B, low), ou $(C, high)$.

La Figure 4.7 montre le RdP coloré modélisant le processus du contrôle de qualité. Les

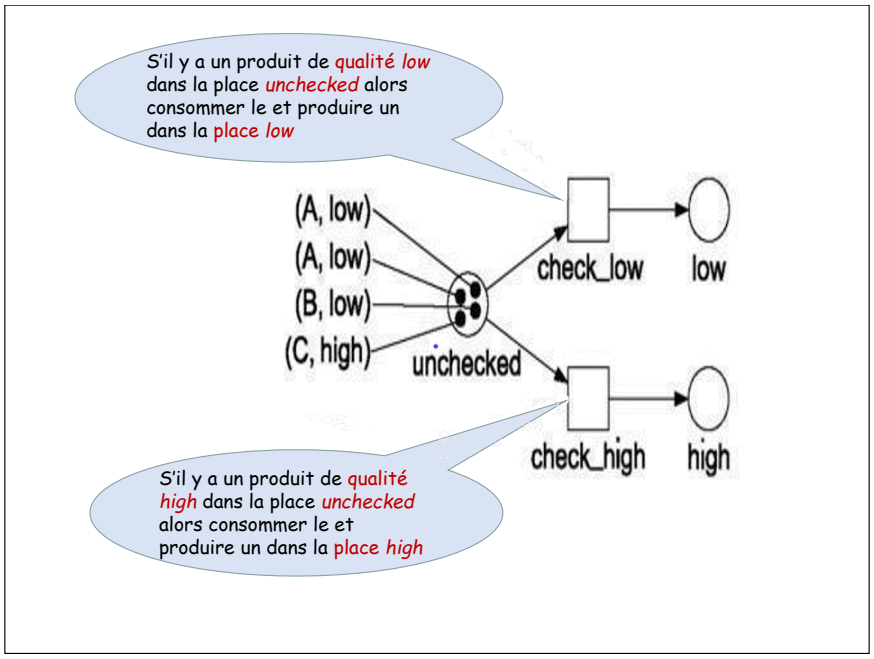


Figure 4.7: Un RdP coloré modélisant le contrôle de qualité d'un produit.

règles de franchissement des transitions de ce RdP sont définies comme suit:

- La transition $check - low$ n'est sensibilisée que s'il y a un produit de qualité low dans la place *unchecked*. Le tir de cette transition consomme ce produit de la place *unchecked* et le reproduit dans la place low.
- La transition $check - high$ n'est sensibilisée que s'il y a un produit de qualité $high$ dans la place *unchecked*. Le tir de cette transition consomme ce produit de la place *unchecked* et le reproduit dans la place $high$.
- Le jeton avec la valeur (B, low) permet la sensibilisation de la transition $check - low$ mais pas la transition $check - high$. De même, le jeton avec valeur $(C, high)$ permet la sensibilisation de la transition $check - high$ mais pas la transition $check - low$.
- Contrairement à un RdP ordinaire, une transition dans un RdP coloré n'est pas nécessairement sensibilisée si toutes les places d'entrée contiennent suffisamment de jetons. La sensibilisation d'une transition peut également dépendre de la **valeur** d'un jeton.
- Dans un RdP coloré, une transition sensibilisée peut être tirée. Le tir d'une transition permet de consommer un jeton de chaque place en entrée et de produire un jeton

dans chaque place de sortie. La **valeur** du jeton produit est **calculée** à partir des **jetons consommés**.

4.2.3 Description Formelle des RdP Colorés

Dans cette section, nous allons présenter les sémantiques formelles des RdP colorés.

4.2.3.1 Types de place, Valeurs de jetons et Marquages

- Chaque place dans un RdP coloré a un certain type, tel que **Patient**, **Employé** ou **Product**. Un jeton dans une place doit être de même type que cette place; il possède une valeur.
- Comme dans un RdP ordinaire, une place dans un RdP coloré peut contenir des jetons de même valeur mais aussi des jetons de valeurs différentes. Par exemple, dans la figure 4.7 la place *unchecked* a 2 jetons avec la valeur (A, low), 1 jeton de valeur (B, low) et 1 jeton de valeur $(C, high)$.
- Pour référencer des jetons multiples dans une même expression, il faut utiliser les **multi-ensembles** (multisets) où le même élément peut apparaître plusieurs fois. Ainsi le marquage des places de la Figure 4.7 est décrit par les 3 multi-ensembles suvants:
  M(*unchecked*) = [(A, low), (A, low), (B, low), $(C, high)$]
  M(low) = [ ]
  M($high$) = [ ], où [ ] représente le multi-ensemble vide.

Formellement, nous avons:

- Pour chaque type de place, un ensemble **U** contenant toutes les valeurs possibles de ce type de place.
- Un multi-ensemble sur **U** est alors une application qui assigne à chaque élément $u \in U$ un nombre naturel $f(u)$. Chaque élément $u \in U$ spécifie une valeur du type de place représenté par **U**.
- Soit **M(U)** l'ensemble de tous les multisets possibles sur **U**. Le marquage d'un RdP coloré peut être défini comme le mapping $m : P \rightarrow M$ où P représente l'ensemble des places du RdPC, et m affecte un multiset $M(U)$ à toute place $p \in P$, avec **U** représentant le type de place p.
  Comme exemple, l'ensemble U du type *Product* de l'exemple précédent est défini par:
  $$U = \{ (A, low), (A, high), (B, low), (B, high), (C, low), (C, high)\}$$
  Des multisets possibles sur **U** pourraient être, par exemple:
  $$[2.(A, low), (B, low), (C, high)] \text{ ou } [(B, low), (B, high), (C, high)].$$

4.2.3.2 Expressions des arcs

Dans un RdPC, le tir des transitions ne modifie pas seulement la distribution des jetons dans les places, mais peut également modifier les couleurs (c'est-à-dire les valeurs) des jetons qui circulent sur le réseau. Cela nécessite une spécification de la sensibilisation et du déclenchement des transitions. Pour cela, nous introduisons des **expressions d'arcs**.

Les **arcs** d'un RdPC doivent porter des **expressions** (inscriptions) qui permettent de spécifier quels sont les jetons qui seront considérés lors du tir de la transition et quels sont les jetons qui seront produits.

Comme exemple, considérons l'exemple de la confection des cartes perforées. La Figure 4.8 montre le RdPC correspondant avec des expressions d'arcs. Par exemple, les arcs reliant la transition *start* à ses places d'entrée et de sortie portent des expressions contenant deux variables x et y. La variable x est de type *Employ* et la variable y de type *Patient*.

La transition *start* est sensibilisée s'il existe un jeton dans la place $free$ dont la valeur peut être affectée à la variable x, et un jeton dans la place *wait* dont la valeur peut être affectée à la variable y.
Le tir de la transition *start* affecte un jeton de la place $free$ à la variable x et un jeton

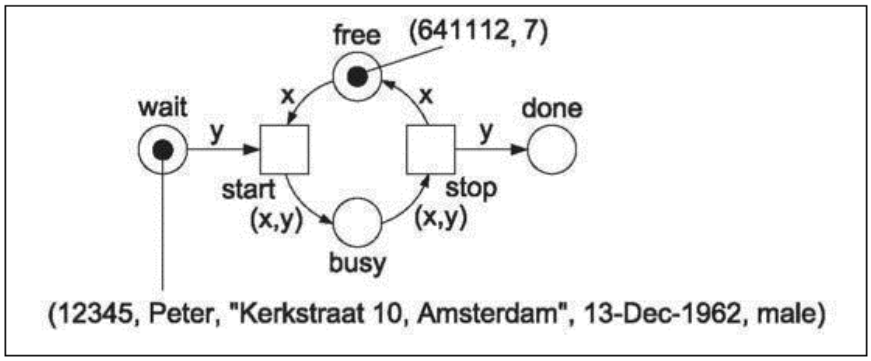


Figure 4.8: RdP coloré étendu avec des expressions d'arcs.

de la place *wait* à la variable y. Ces 02 jetons sont consommés et un jeton avec la valeur (x,y) est produit dans la place *busy*. Avec ce nouveau marquage, la transition *stop* est sensibilisée et peut être tirée. Elle va consommer le jeton avec la valeur (x,y) de la place *busy* et produit un jeton de valeur x dans la place *done* et un jeton de valeur y dans la place $free$. Si plusieurs valeurs peuvent être affectées à une variable alors nous avons plusieurs correspondances (**bindings**).

Cependant, les inscriptions d'arc ne sont pas suffisantes pour formaliser le comportement des RdP colorés, en général. Parfois, il est nécessaire qu'une transition ne soit sensibilisée qu'à un certain **binding** qui satisfait certaines **conditions**. Pour modéliser de telles conditions, on peut utiliser un **garde de transition** **(« guard »)**.

4.2.3.3 Gardes de Transition

Un garde définit une contrainte supplémentaire qui doit être remplie avant qu'une transition soit sensibilisée.

■ **Exemple 4.5** Considérons le RdPC de la Figure 4.7 (contrôle de qualité d'un produit) , la transition $check-low$ n'est sensibilisée que dans le cas d'un produit de faible qualité et la transition $check-high$ est validée uniquement dans le cas d'un produit de haute qualité. Pour modéliser ces conditions, nous pouvons utiliser un **garde de transition**.
Pour ce faire, nous étendons l'exemple de contrôle de qualité d'un produit par ce concept de garde (voir Figure 4.9). ■

Le comportement du RdPC de la Figure 4.9 est défini comme suit:

- Il existe trois bindings potentiels pour la transition $check-low$:
$$(check-low, (x = (A, low)))$$
$$(check-low, (x = (B, low)))$$

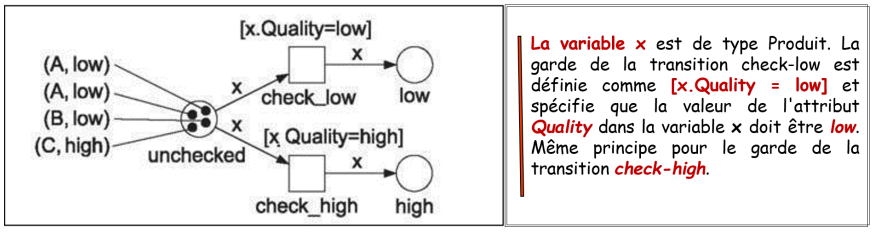


Figure 4.9: RdP coloré étendu avec des gardes de transition.

$$(check-low,(x=(C,low)))$$

Mais seuls les deux premiers bindings permettent de sensibiliser la transition $check-low$ car le garde de cette transition évalue à vrai (true).

- Il existe trois bindings potentiels pour la transition $check-high$:

$$(check-high,(x=(A,low)))$$
$$(check-high,(x=(B,low)))$$
$$(check-high,(x=(C,low)))$$

Mais seul le troisième binding permet d'évaluer le garde de la transition à true et donc de la sensibiliser. Les deux autres bindings évaluent le garde de la transition à faux (false).

Formellement, les inscriptions d'arc et les gardes sont des **expressions**. Une expression peut contenir des constantes et des variables.

La différence entre une inscription d'arc et un garde est que la valeur d'une inscription d'arc est toujours un **multiset**, alors que la valeur d'un garde est un **booléen**.

Récapitulons...

l'idée des RDPC est de différencier les jetons au lieu de différencier les places. Dans un RDPC:

- Chaque place a **un type**, et chaque jeton a une **valeur** (c'est-à-dire, une couleur) qui est conforme au type de la place.
- Un **arc** peut porter une inscription. Une **inscription** d'arc est une expression avec quelques variables qui évaluent à un multiset.
- Une transition peut avoir un **garde**. Un garde est une expression booléenne, et il peut y avoir des variables exactement de la même manière que les inscriptions d'arc (des arcs entourant cette transition).
- Une transition T est sensibilisée par un **binding** si des jetons correspondent aux valeurs des inscriptions d'arc et si le garde de T évalue à vrai. Une transition sensibilisée est tirée tout en consommant et en produisant les jetons correspondants.

4.2.3.4 Définition Formelle

Définition 4.1 un réseau de Petri coloré est un n-uplet RDPC = **(P, T, C, Type, Pré, Post,** M_0**)** où :

- **P** est un ensemble fini de places,
- **T** est un ensemble fini de transitions disjoint de P, T ∩ P = ∅,
- **C** est un ensemble fini non vide de domaines où chaque élément de C est appelé Type qui est ici une couleur,

- **Type**: $P \cup T \rightarrow C$ est une fonction qui associe des types (couleurs) aux places et détermine les modes de franchissement,
- **Pré, Post**: fonctions des arcs. Chaque arc porte une fonction ***f(C)***. Ces fonctions relatives aux couleurs de franchissement associées aux arcs, définissent les règles de franchissements,
- M_0: un ensemble multiple appelé marquage initial du réseau.

Exercice d'application 4.1 Modéliser le système de gestion des inventaires de la Figure 4.1 par un RdP coloré. ■

Solution: Elle est présentée par la Figure 4.10.

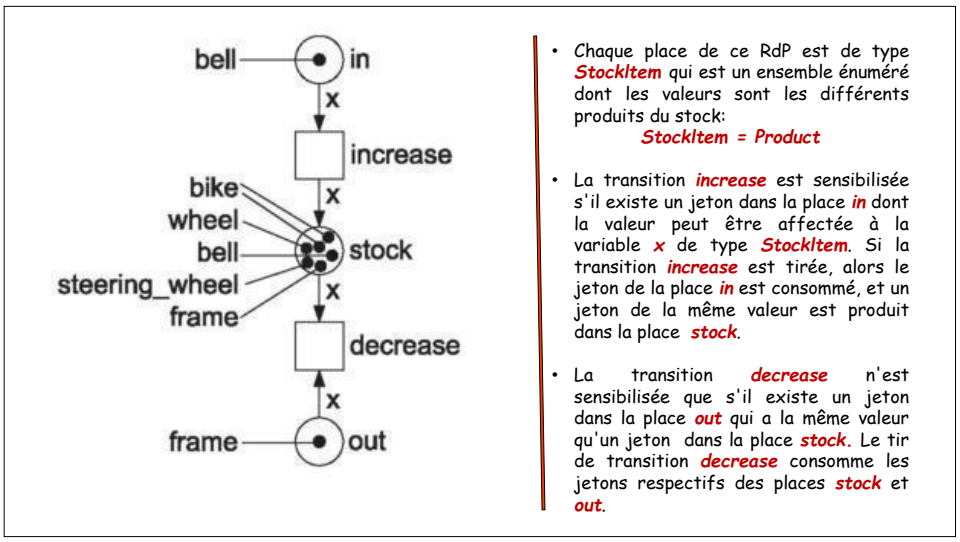


Figure 4.10: RdP coloré du problème de gestion des inventaires.

Exercice d'application 4.2 Étendre le modèle de l'exercice précèdent de sorte que chaque article en stock soit une paire constituée d'un produit et de la quantité du produit en stock. En outre, il devrait être possible d'augmenter et de diminuer la quantité d'un article en stock par un nombre n à chaque tir de ces transitions. Utilisez le marquage suivant à la comme marquage initial.
■

Solution: Elle est présentée par la Figure 4.11

4.2.4 Résumé

Un réseau de Petri **coloré** est un réseau de Petri étendu par l'**ajout de données**. La modélisation avec des réseaux de Petri colorés permet d'obtenir des modèles de processus plus compacts, comme l'ont montré les exemples de cette section. Alors que la complexité d'un réseau de Petri est exprimée dans la structure du réseau, la complexité d'un réseau de Petri coloré se trouve dans les types de place et les différentes expressions.

L'expressivité accrue des RdP colorés par rapport aux RdP ordinaires nous permet de modéliser des systèmes industriels qui ne peuvent être modélisés par des RdP ordinaires. De plus, nous n'avons mis aucune restriction sur les expressions dans un RdP coloré.

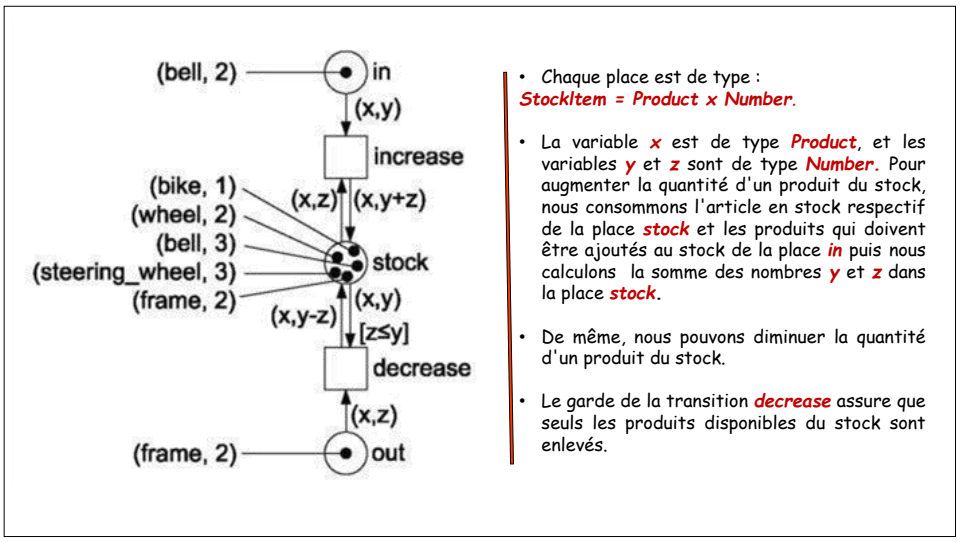


Figure 4.11: RdP coloré du problème de gestion des inventaires étendu.

Cependant malgré l'avantage de cette expressivité, l'analyse des RdP colorés est beaucoup plus compliquée, car certaines questions, telles que "*Peut-on atteindre un marquage m à partir du marquage mp ? initial*", ne peuvent être décidées en général.

4.3 Les Réseaux de Petri Temporisés

Etendre les RdP avec le **temps** permet de décrire adéquatement les aspects temporels d'un système dans le modèle. Les réseaux étendus par des contraintes de temps sont des réseaux de Petri paramétrés par des **horloges**, ce qui permet de modéliser des contraintes temporelles en divers points de l'exécution. Cette classe de RdP est très utilisée dans la modélisation des systèmes temps réel où l'importance du traitement du temps est très importante et souvent critique. Ils permettent d'évaluer les **performances temporelles** de ce type de systèmes telles que par exemple l'évaluation du temps d'exécution d'une séquence d'opérations en fonction de la capacité des machines et du stock.

Les RdP peuvent inclure les aspects temporels en associant un événement à l'instant dans lequel les résultats sont survenus. Cette fonction peut être associée à une transition, une place ou un arc [7, 14, 19]. On distingue plusieurs techniques pour introduire le temps dans un RdP.

4.3.1 Techniques d'Introduction du Temps dans les RdP

Comme décrit dans [5], nous distinguons trois techniques de base pour représenter le temps dans les RdP: **Firing Durations** (FD), **Holding Duration** (HD) et **Enabling Duration** (ED).

Firing Durations (FD)**:** C'est l'approche la plus intuitive pour introduire le temps dans les RdP. On affecte à chaque transition un certain délai spécifiant combien de temps prend le tir de cette transition (**RdP T-temporisés**). Ce délai peut, par exemple, correspondre à la durée d'exécution de la tâche modélisée par cette transition. Le tir d'une telle transition retire immédiatement tous les jetons des places d'entrée mais ne

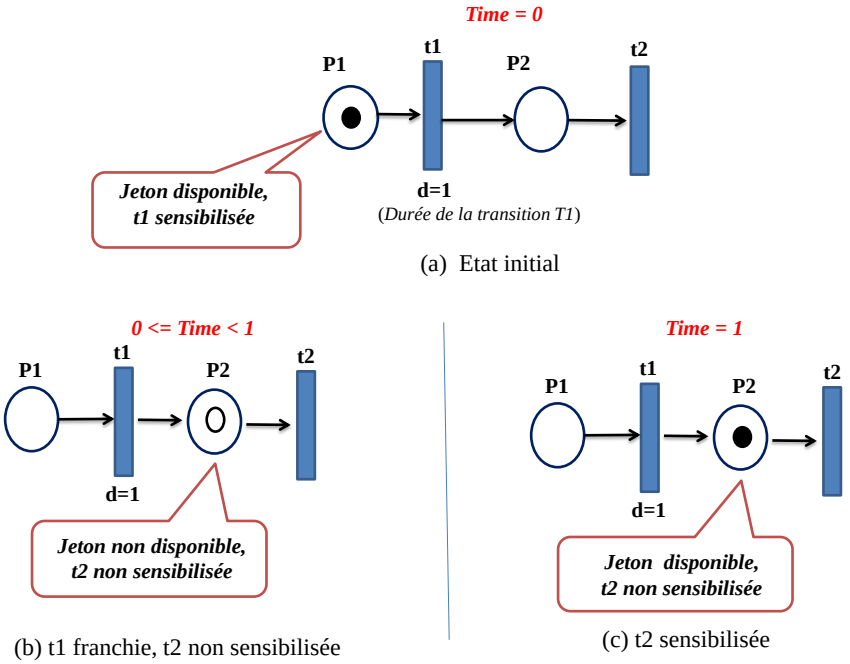
crée aucun jeton dans les places de sortie jusqu'à ce que le délai de franchissement représenté par la temporisation de la transition soit écoulé. Ainsi, entre la consommation et la production des jetons on compte une durée égale à la temporisation de la transition.
Cette approche semble évidente pour introduire le temps, mais le **marquage** complet du RdP n'**est pas visible** à tout moment, car certains jetons peuvent être **cachés** à l'intérieur d'une transition. De plus, plusieurs techniques d'analyse ne sont plus applicables du moment que le tir d'une transition n'est plus atomique.
Notons qu'il existe une autre manière d'introduire le temps en affectant la temporisation aux places (**RdP P-temporisés**). Dans ce cas, le délai correspond au temps de séjour des jetons dans la place. Les RdP T-temporisés et les RdP P-temporisés sont **équivalents**.

Holding Durations (*HD*)**:** Cette approche vient résoudre les problèmes de la précédente. On étend les RdP en ajoutant du temps aux jetons et aux transitions. Un jeton peut porter un **timbre temporel** appelé **timestamp** et une transition peut produire des jetons avec un délai. Cette technique **HD** repose sur la classification des jetons en deux types: **disponible**, et **non-disponible**. Les jetons **disponibles** peuvent sensibiliser une transition, par contre les **non-disponibles** ne peuvent pas. Le principe de **HD** est que lorsqu'une transition avec temporisation est tirée alors l'action de retrait et de création de jetons est faite **instantanément**. Cependant, les jetons créés ne sont pas disponibles pour sensibiliser de nouvelles transitions jusqu'à ce qu'ils séjournent dans leurs places de sortie pendant la durée spécifiée par la temporisation de la transition qui les a créé.

La Figure 4.12 illustre le principe de **HD**. La Figure 4.12 (a) montre l'état initial d'un RdP où à l'instant $t = 0$ la transition $t1$ est sensibilisée et franchie immédiatement. Le tir de $t1$ provoque la consommation du jeton (disponible) de la place d'entrée $P1$ et la création d'un jeton dans la place de sortie $P2$ (voir Figure 4.12 (b)). Cependant, le jeton crée dans $P2$ n'est pas disponible pour sensibiliser la transition $t2$ jusqu'à ce que la durée d de la transition $t1$ qui l'a crée soit écoulée. Ainsi, la transition $t2$ n'est sensibilisée et franchie qu'à l'instant $t = 1$ (temps initial + d) comme le montre la Figure 4.12 (c).

Comme nous pouvons le remarquer, **HD** et **FD** représentent le temps de la même manière. La seule différence réside dans le fait que dans **FD**, les jetons sont temporairement retenus dans les transitions alors que dans **HD**, ils sont retenus dans les places. De ce point de vue, aucune distinction n'est faite entre les deux techniques. Mais sur le plan sémantique, **HD** est plus proche des RdP classiques et le graphe d'accessibilité est plus réduit.

Enabling Durations (*ED*)**:** La dernière manière d'intégrer le temps dans les RdP est la technique *ED*. Cette technique permet le tir immédiat de la transition; le retrait et la création des jetons s'effectue au même instant et les délais représentés par les temporisations sont représentés en forçant les transitions à être sensibilisées pour une période de temps spécifiée avant d'être franchies. Cette technique est préconisée pour la modélisation des interruptions dans les systèmes.



(a) Etat initial

(b) t1 franchie, t2 non sensibilisée

(c) t2 sensibilisée

Figure 4.12: Réseaux de Petri temporels avec des durées HD

La différence principale entre **HD** et **ED** apparaît lorsqu'il y a confusion ou conflit dans le RdP. Considérons l'exemple de la Figure 4.13 (a). Si la technique **HD** est appliquée alors les transitions $t1$ et $t3$ sont tirées concurremment à l'instant $t = 0$, plaçant un jeton dans la place $P2$ et un autre dans la place $P5$. Les jetons dans $P2$ et $P5$ deviennent disponibles à $t = 1$ et à $t = 5$ respectivement. Aucune autre transition n'est alors sensibilisée. La Figure 4.13 (b) montre l'état final du RdP avec **HD**.

■ **Exemple 4.6** Considérons maintenant le même RdP de la Figure 4.13 (a) mais avec l'application de l'approche **ED**. Initialement, les transition $t1$ et $t3$ sont sensibilisées mais elles ne seront franchies qu'à $t = 1$ et $t = 5$ respectivement. Ainsi à $t = 1$, le tir de $t1$ permet de créer un jeton dans la place $P2$ sensibilisant ainsi la transition $t2$ qui pourra être franchie dans deux unités de temps, soit à $t = 3$. Or, $t3$ est toujours sensibilisée mais son franchissement n'aura lieu qu'à $t = 5$. Par conséquent, $t2$ sera tirée avant $t3$ et consommera les jetons des places $P2$ et $P3$; un jeton est donc crée dans $P4$ et $t3$ n'est plus sensibilisée. La Figure 4.13 (c) montre l'état final dur RdP avec **ED**. En conclusion, le changement de la politique temporelle d'un RdP peut causer un changement dramatique de la façon dont il s'exécute. ■

Le choix de l'une de ses techniques dépend fortement du système à modéliser et de ses spécifications. Il convient de noter cependant, qu'il est naturel d'opter pour la technique **HD** dans la modélisation des processus du moment que les transitions représentent des opérations qui, une fois lancées, ne peuvent pas être interrompues. C'est pour cette raison que nous nous intéresserons à cette technique dans la suite de ce cours.

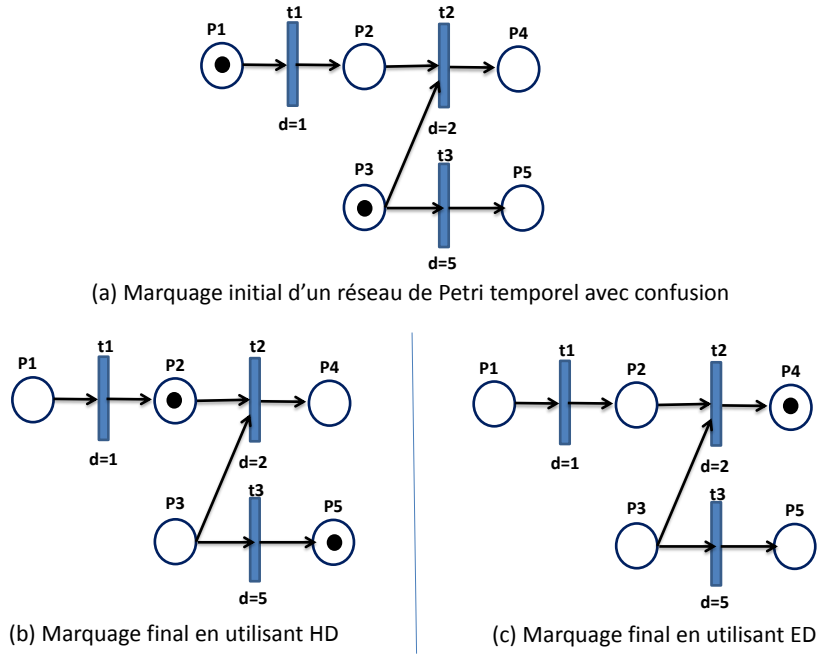


Figure 4.13: Réseaux de Petri temporels montrant la différence entre HD et ED

4.3.2 Concept de Timestamps des Jetons

Quand on considère le temps, on suppose une **horloge globale** représentant le temps du modèle. Le temps passe en unités de temps dont la valeur concrète - qu'elle représente une seconde, deux minutes ou cinq heures - dépend de la spécification du système.
Pour chaque jeton, on affecte un **timestamp** qui indique à partir de quel moment, par rapport à l'horloge globale, le jeton peut être consommé par une transition.

■ **Exemple 4.7** Reprenons l'exemple de la confection des cartes perforées de la Figure 4.3. Dans ce modèle, on ne représente pas le temps nécessaire pour faire une carte perforée. On étend alors le modèle avec le temps comme le montre la Figure 4.14. Le jeton dans la

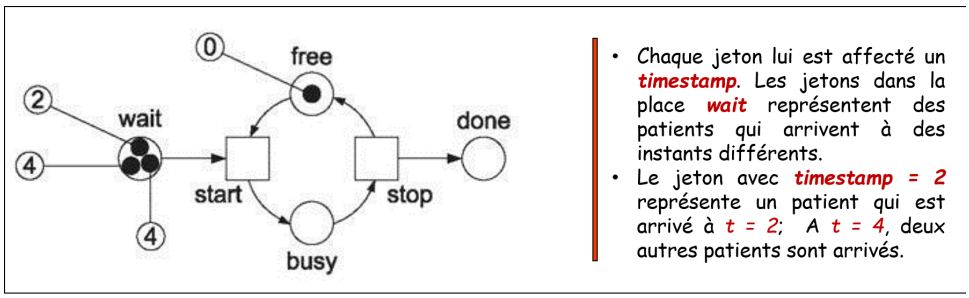


Figure 4.14: Modèle RdP temporisé du bureau de service des cartes perforées.

place $free$ à un $timestamp = 0$, il spécifie que l'employé est disponible à partir de $t = 0$. A l'instant $t = 0$, seul l'employé est disponible; à l'instant $t = 2$, l'employé et un patient sont disponibles; et au moment $t = 4$, l'employé et les trois patients sont disponibles. ■

Le **marquage** d'un RdP temporisé est décrit par le **nombre** et les **timestamps** des jetons dans chaque place. Un jeton n'est disponible pour la consommation qu'à partir du moment où son timestamp est inférieur ou égal à l'heure actuelle.
Une transition ne peut être déclenchée que si tous les jetons à consommer sont disponibles.

4.3.3 Temps de Sensibilisation d'une Transition

Le temps de sensibilisation d'une transition T est le point temporel auquel ses places d'entrée contiennent suffisamment de jetons disponibles pour que T soit sensibilisée.
La transition *start* du RdP de la Figure 4.14 a un temps de sensibilisation de 2, car c'est l'instant le plus tôt où les places *free* et *wait* contiennent un jeton disponible.
La Figure 4.15 présente un autre exemple de la sensibilisation des transitions temporisées.

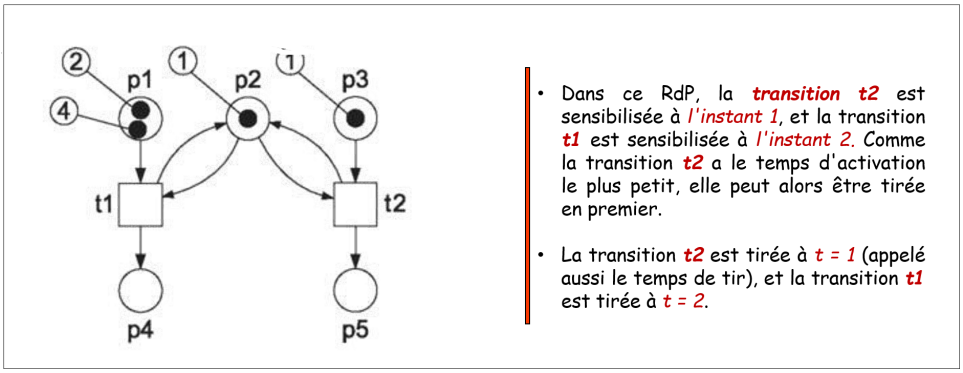


Figure 4.15: Modèle RdP temporisé avec un choix non-déterministe.

Exercice d'application 4.3
Quels sont les marquages accessibles à partir du marquage donné sur la Figure 4.15? Quels sont les temps de sensibilisation et de tir? à supposer que les jetons sont produits sans délai.
■

Solution: Elle est présentée par la Figure 4.16

4.3.4 Temporisation des Jetons Produits

Comment déterminer les timestamps des jetons produits?

Le **timestamp** d'un jeton produit est égal au temps de tir augmenté d'un éventuel délai. Ce délai est déterminé par la transition qui est tirée.
Le tir lui-même est instantané et représente donc une action atomique. Cependant, le délai garantit que les jetons produits ne peuvent pas être consommés avant que ce délai ne soit écoulé. Par conséquent, ce délai a le même effet que si on spécifie que le tir d'une transition prend du temps.

■ **Exemple 4.8** Considérons le RdP temporisé de la Figure 4.17. Le délai est exprimé par un @+ suivi du délai. Dans ce modèle, le jeton produit par la transition *start* dans la place *busy* obtient le **timestamp** de 3 unités de temps. Ainsi, la transition *stop* ne peut être tirée qu'après 3 unités de temps. Dans ce cas, 3 unités de temps modélisent le temps nécessaire

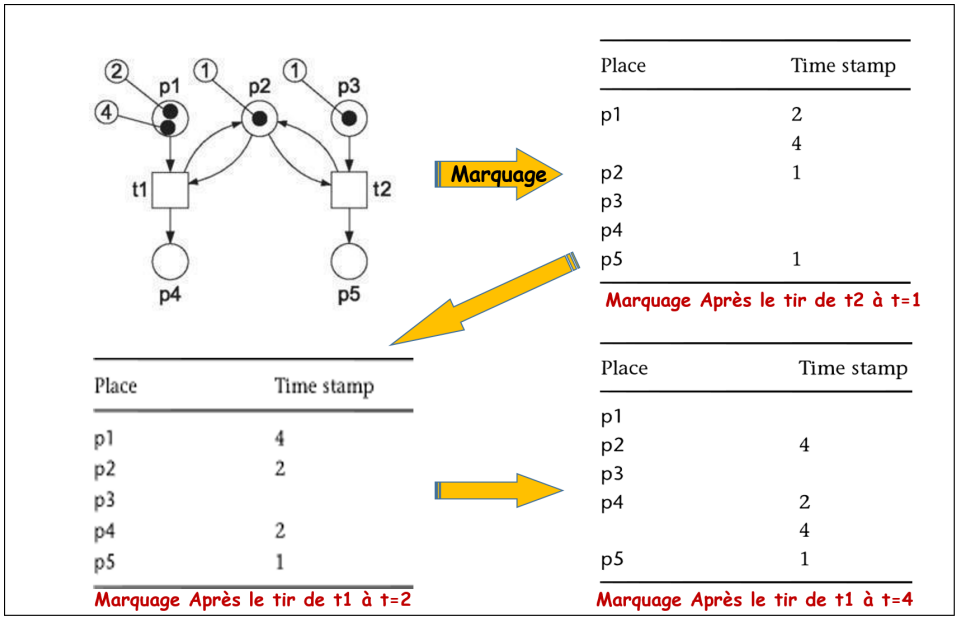


Figure 4.16: Marquages accessibles du RdP temporisé de la Figure 4.15.

pour confectionner une carte perforée.

Les jetons produits par la transition *stop* dans les places *free* et *done* n'ont pas de **délai** (**timestamp = 0**). Nous modélisons le fait que l'employé peut immédiatement commencer à aider le prochain patient.
Après le tir de la transition *start*, le jeton produit dans la place *busy* a un **timestamp de 5=2+3** unités de temps. Ce qui représente le temps de tir de la transition *stop* (voir Figure 4.18). ∎

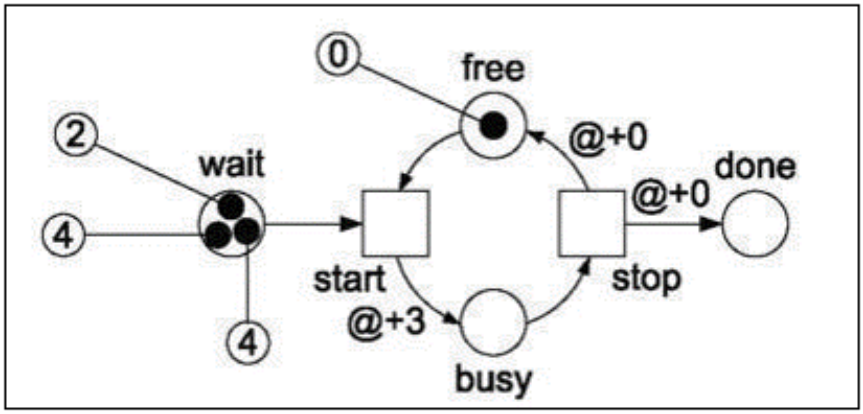


Figure 4.17: Exemple d'un RdP temporisé.

La séquence de marquages accessibles à partir du Marquage de la Figure 4.18 est présentée par la Figure 4.19.
Grâce au marquage final de la Figure 4.19, on peut répondre à la question: *Combien de temps les patients doivent-ils attendre?*

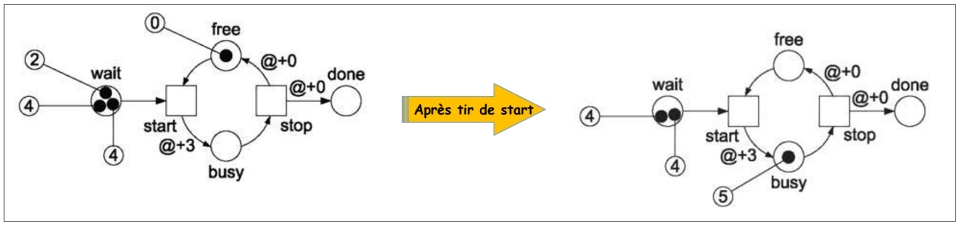


Figure 4.18: Marquage d'un RdP temporisé.

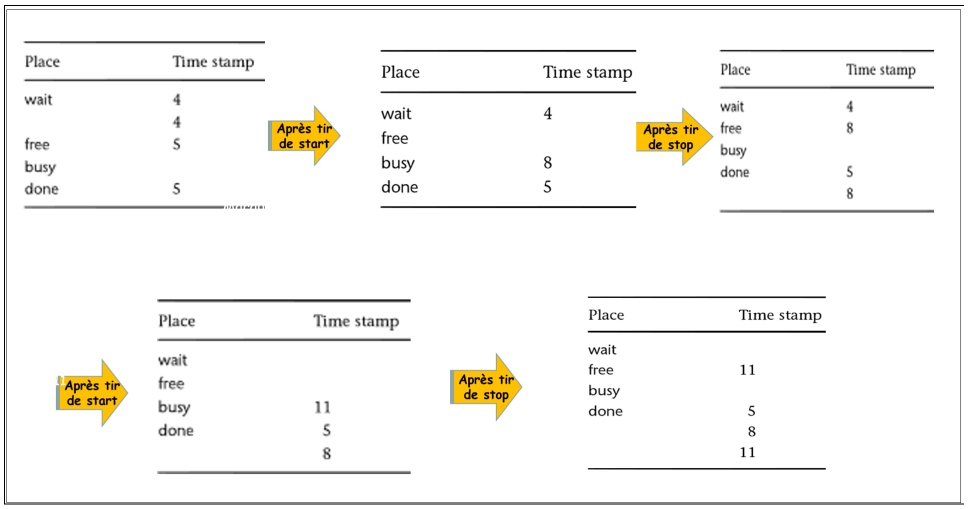


Figure 4.19: Séquence de marquages d'un RdP temporisé.

En effet, les **timestamps** des jetons dans la place *done* du marquage final montrent quand la carte perforée de chaque patient est faite. Le jeton avec le **timestamp 11** indique que le dernier patient a dû attendre **11 - 4 = 7** unités de temps.

Notons que quand les délais sont fixes alors nous utilisons les RdP temporisés. Dans certains cas, les délais sont estimés par des **lois de probabilités**, nous parlons alors de **RdP stochastiques**.

4.3.5 RdP Colorés et Temporisés (RdPCT)

Les réseau de Petri colorés peuvent aussi être étendus avec le temps. Comme exemple, considérons le RdP coloré de la Figure 4.3 (*confection des cartes perforées*) qu'on désire étendre avec le temps pour modéliser la durée que nécessite la confection d'une carte. Cette durée dépend de l'expérience de l'employé du bureau: Si l'employé a une expérience de plus de 5 ans alors il lui faut 3 unités de temps pour confectionner une carte, sinon il faut 4 unités de temps. La Figure 4.20 montre le RdPC augmenté avec le temps. Dans un RdPCT, Chaque jeton a une valeur (couleur) et un timestamp.

Dans le RdPCT de la Figure 4.20, le délai pour confectionner une carte perforée est calculé par une expression. Dans ce cas, nous avons défini une fonction d qui prend la variable x comme argument. Cette fonction calcule le délai, compte tenu de la valeur de l'attribut $x.Experience$.
L'inscription d'arc **(x, y) @ + d (x)** définit qu'un délai de $d(x)$ unités de temps est ajouté

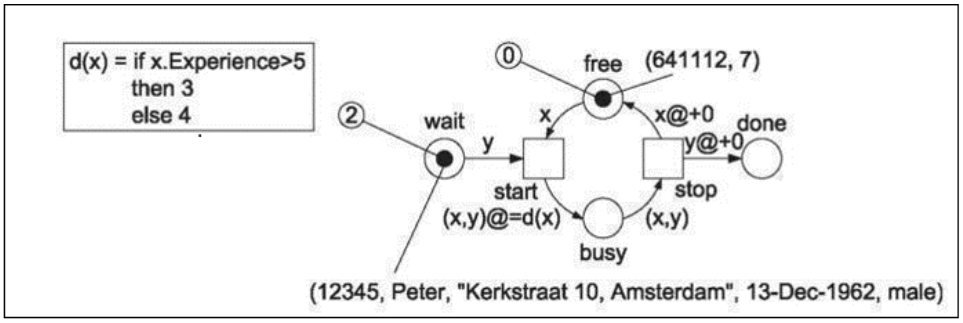


Figure 4.20: RdP coloré et temporisé du bureau de service des cartes perforées.

au jeton (x, y) produit dans la place *busy*.
Dans un RdPCT, une transition T est sensibilisée par un binding si aucun autre binding (de T ou de toute autre transition) n'existe avec un **temps de sensibilisation plus petit**.

Pour décrire le **marquage** d'un RdPCT, nous devons spécifier pour chaque place la valeur et le timestamp de chaque jeton. La Figure 4.21 montre le marquage du RdPCT de la Figure 4.20.

Place	Time stamp	Valeur
wait	2	(12345, Peter, "Kerkstraat 10, Amsterdam", 13-Dec-1962, male)
free	0	(641112, 7)
busy		
done		

Figure 4.21: Marquage du RdPCT de la Figure 4.20.

4.3.6 Définition Formelle des RdPCT avec la Technique HD

Pour représenter les jetons avec **timestamps**, nous adoptons la notation donnée par [12]. Chaque jeton porte un timestamp précédé du symbole @. Par exemple, deux jetons avec un timestamp égal à 10 sont notés 2@10. Le timestamp spécifie l'instant où le jeton est prêt à être retiré par une transition sensibilisée.

Les timestamps sont des valeurs d'un ensemble de temps, TS, qui est égal à l'ensemble des entiers positifs $\mathbb{N}^+$. Les marquages temporels sont représentés comme une collection de timestamps et qui sont des multi-ensembles de support $TS : TS_{MS}$. La définition formelle des RdPCT utilisant la technique **HD** est comme suit:

Définition 4.2 RdPCT = (Σ, f, M_0)où Σ est un réseau de Petri coloré comme décrit dans [12, 13]:

- Σ= **(S, P, T, A, N, C, G, E)** où:
  - **S** est un ensemble fini de types non vides, appelés ensembles des couleurs.
  - **P** est un ensemble fini de places.

  - **T** est un ensemble fini de transitions.
  - **A** est un ensemble fini d'arcs tels que: P ∩ T = P ∩ A = T ∩ A = ∅.
  - **N** est une fonction de nœuds. Elle est définie de A vers $P \times T \cup T \times P$. Elle correspond aux **applications d'incidence** avant et arrière.
  - **C** est une fonction de **couleur**. Elle est définie de P vers S. A chaque place est associé un type.
  - **G** est une fonction de **garde**. Elle est définie de T vers l'ensemble des expressions tels que: ∀ t ∈ T: [Type(G(t)) = Bool ∧ Type (Var(G(t))) ⊆ S].
  - **E** est une fonction d'**expression des arcs**. Elle est définie de A vers l'ensemble des expressions. Elle associe à chaque arc une expression qui servira pour la vérification ou la création de nouvelles valeurs de jetons. De chaque évaluation d'une expression d'un arc devrait résulter un ensemble de jetons (un multi-ensemble sur les différents types permis par la place). E contient des expressions d'entrée ainsi que les actions sortantes.
- **f** : $T \rightarrow TS$ représente la fonction de transition, qui associe à chaque transition $t \in T$ une durée déterministe non négative.
- **M** : $P \rightarrow TS_{MS}$ est le marquage temporel, M_0 représente le marquage initial du RdPCT.

Pour déterminer si des jetons sont disponibles ou non disponibles, nous définissons des fonctions sur l'ensemble de marquage M. Ainsi, pour un marquage M et un modèle donné de temps (en supposant une horloge globale), nous avons:

- m : $P \times M \times TS \rightarrow N$, qui définit le nombre de jetons disponibles;
- n : $P \times M \times TS \rightarrow N$, qui définit le nombre de jetons non disponibles pour chaque place du modèle RdPCT à un instant donné k, où k et le modèle de temps appartiennent à TS.

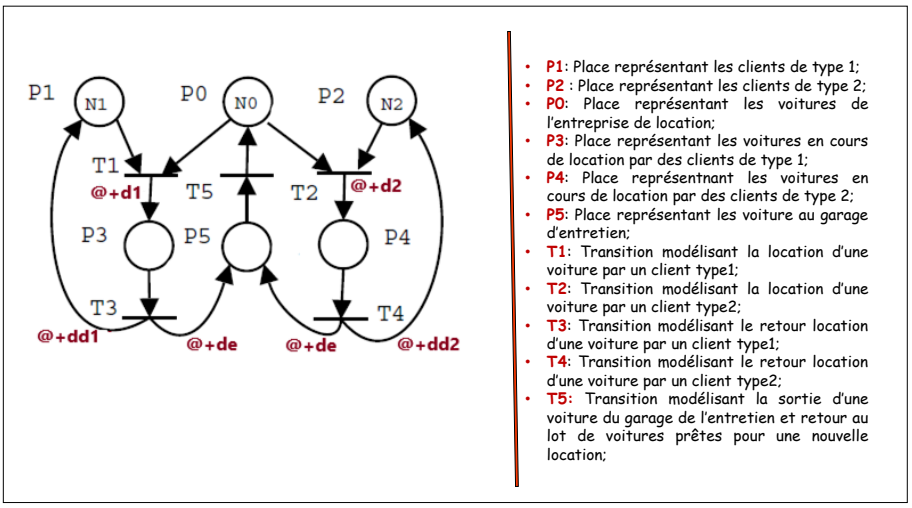
Pour ajouter la temporisation stochastique, les transitions concernées sont augmentées de variables aléatoires définies par des fonctions de distribution aléatoires.

Exercice d'application 4.4 On considère le fonctionnement d'une entreprise de location de voitures. Cette entreprise a deux types de clients :

- Chacun des N_1 clients de type 1 utilise une voiture pendant une durée d_1 ; le temps moyen entre deux demandes est de d_{d_1} ;
- Chacun des N_2 clients de type 2 utilisent une voiture pendant une durée $d2$; le temps moyen entre deux demandes est de d_{d_2}.
  N_0 représente le nombre de voitures de l'entreprise. Après chaque retour de location, la voiture passe au garage pour entretien pour une durée d_e.

Question: Modéliser ce processus par un RdP temporisé. ■

Solution: Le fonctionnement de cette entreprise est représentée par le RdP temporisé de la figure suivante:



4.4 Extension des Réseaux de Petri par des Arcs "Spéciaux"

Les RdP ordinaires ne permettent pas d'exprimer des conditions très riches. Pour tenir compte de certaines spécificités et mécanismes, il est souvent plus simple de les modéliser avec les arcs. On distingue les arcs inhibiteurs, les arcs de lecture et les arcs de vidange. En contrepartie, avec ces grands pouvoirs d'expressions, de nombreux problèmes deviennent indécidables. Les RdP faisant paraître ce type d'arcs échappent aux possibilités d'analyse formelle de ses propriétés. Ils ne se trouvent, par contre, qu'à des fins de confort de la visualisation et de l'implémentation automatisée.

4.4.1 Arcs Inhibiteurs

C'est une extension des RdP qui permet de tester l'**absence de jetons dans une place**, alors que lors d'un franchissement classique, on vérifie au contraire la présence d'une marque qui sera consommée. En effet, dans le cas général, il est impossible de tester si le contenu d'une **place est vide**; autrement dit, il est impossible de définir un RdP pour lequel une transition est tirable si une place donnée en entrée ne contient pas de jeton. Intuitivement, ce « **test à zero** » est en contradiction avec le principe de monotonie dans les RdP traditionnels.

▪ **Exemple 4.9** Comme exemple illustratif, considérons la modélisation d'un système dans lequel un producteur élabore n objets puis s'arrête jusqu'à ce qu'un consommateur utilise tous les n objets; le producteur reprend alors son activité.
Ainsi, après la production de n objets, la présence de jetons dans la place $objetsProduits$ doit **inhiber** l'exécution de la transition $produire$. ▪

Par définition, un **arc inhibiteur** ne peut relier qu'une place à une transition (et jamais une transition à une place). Il se termine (du coté de la transition) par un petit cercle au lieu d'une flèche, comme le montre la Figure 4.22. Un arc inhibiteur peut être de poids n.

► **Franchissement d'une transition avec arc inhibiteur:** Une transition t_i est sensibilisée si et seulement si, en plus des règles habituelles, il n'existe aucune marque dans des places d'entrée reliées à t_i par un **arc inhibiteur** (places vides). Les marquages obtenus après franchissement de la transition t_i sont identiques au cas des réseaux de

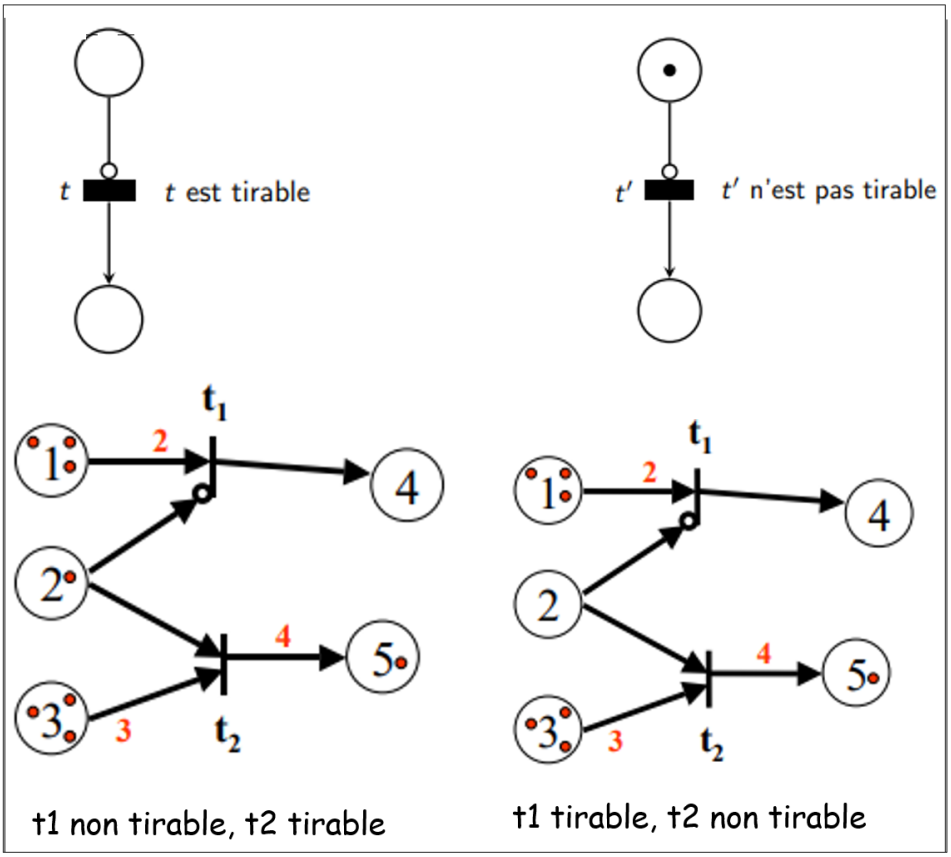


Figure 4.22: RdP avec arc inhibiteur.

Petri généralisés. L'opération de franchissement n'a alors aucune incidence sur le marquage de la place de départ de l'**arc inhibiteur**.
Pour un arc inhibiteur de poids '1', la place qui est vide reste vide (voir Figure 4.23). Si l'arc inhibiteur a un poids n, alors la transition à laquelle aboutit un arc inhibiteur est sensibilisée, en plus des règles habituelles, la place de départ de cet arc n'a pas n jetons.

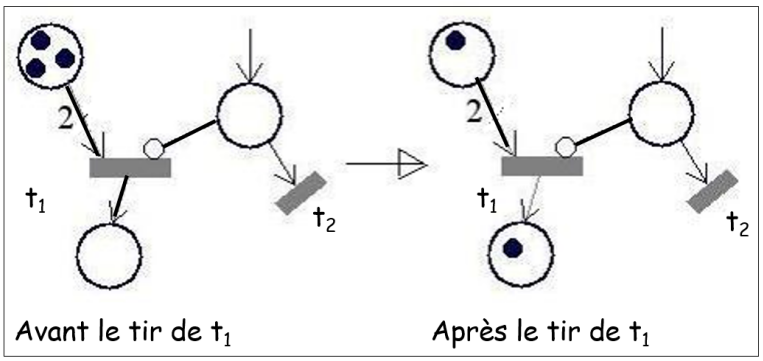


Figure 4.23: Tir d'une transition avec arc inhibiteur.

4.4.2 Arcs de Lecture

Avec cet arc, on teste le contenu d'une place sans enlever des jetons pour autant.
Une transition $t \in T$ est validée pour un marquage m ssi $\forall p \in P$,

$$m(p) \geq Pre(p,t) + Read(p,t).$$

Le marquage successeur est inchangé : $m'(p) = m(p) - Pre(p,t) + Post(t,p)$, comme le montre la Figure 4.24.

Notons qu'un arc de lecture peut toujours être simulé par autant de lectures et d'écritures.

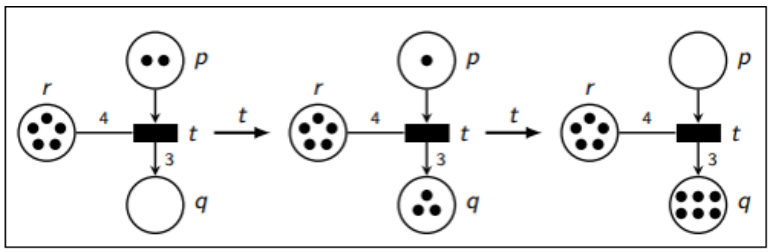


Figure 4.24: RdP avec arc de lecture.

Il s'agit donc plus d'une facilité d'écriture que d'un nouveau modèle (voir Figure 4.25).

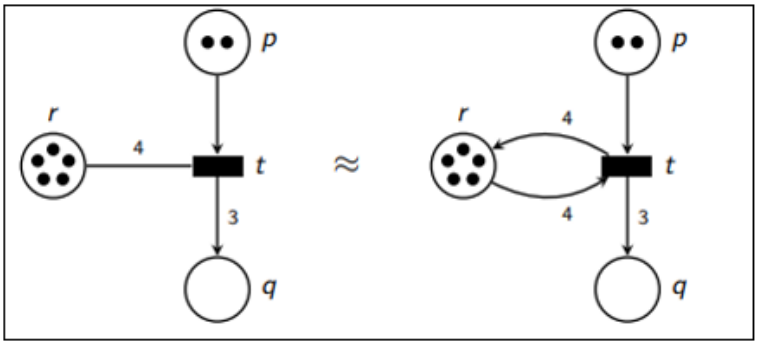


Figure 4.25: Transformation d'un arc de lecture.

4.4.3 Arcs de Vidange

Avec un arc de vidange, on ne teste pas la place, mais on enlève tous les jetons qui s'y trouvent à la fois (**vider** la place).
La condition de tir de la transition à laquelle aboutit un **arc de vidange** reste inchangé comme le montre la Figure 4.26.

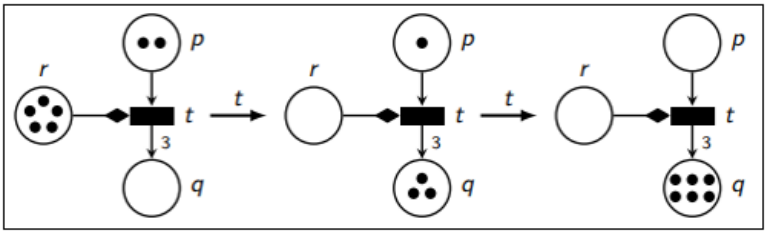


Figure 4.26: RdP avec un arc de vidange.

4.5 Les Réseaux de Petri Hiérarchiques

Un RdP de taille large peut être subdivisé en plusieurs modules. Les **modules** d'un RdP, tous types confondus, jouent un rôle similaire à celui des modules dans les langages de programmation ordinaires. Ils permettent de diviser le modèle en parties gérables avec des interfaces bien définies.

Les RdP avec modules sont également appelés RdP **hiérarchiques**. Un réseau de Petri hiérarchique est un modèle dans lequel une partie peut être représentée par une transition de substitution qui est une **abstraction** d'un autre modèle, c'est à dire un autre RdP [12, 13]. La hiérarchisation permet la modélisation modulaire en subdivisant le modèle en plusieurs modules.

■ **Exemple 4.10** Pour mieux illustrer les notions de ce type de RdP, nous considérons l'exemple présenté dans la Figure 4.27. Dans cet exemple, la transition $T1$ représente une transition de substitution à laquelle est associée un sous-réseau de même nom. Ce sous-réseau représente un RdP coloré où il est nécessaire que le domaine de couleur de la place $P1$ (respectivement $P2$) corresponde à celui de la place IN (respectivement Out).

- Une place IN représente un port d'entrée (*input port*) est utilisée pour importer des jetons de l'environnement.
- Une place Out représente un port de sortie (*output port*) est utilisée pour exporter des jetons vers l'environnement.
- Une place IN/Out représente un port d'entrée/sortie (*input/output port*) est utilisée pour importer et exporter des jetons.
- Les places **A**, **B**, **C** et **D** sont des places internes (locales) au sous-réseau $T1$.
- Les places $P1$ et IN (respectivement $P2$ et Out) doivent avoir aussi le même marquage, elles sont appelées des places de fusion.
- Les places $P1$ et $P2$ représentent les sockets d'entrées et de sorties de la transition de substitution $T1$. Elles sont, de ce fait, étiquetées par IN et Out respectivement.
- A chaque place port doit correspondre une place socket de même type et même marquage.

■

4.6 Les Réseaux de Petri Stochastiques (SPN)

Les RdP stochastiques représentent l'une des extensions les plus évoluées des RdP. Ils ont été introduits par Florin dès 1978 [9] pour répondre à certains problèmes faisant intervenir des phénomènes aléatoires. Les SPN ajoutent de l'indéterminisme aux RdP classiques. Ils ont été proposés dans le domaine de l'évaluation des performances.

Les réseaux de Petri stochastiques (SPN) sont des réseaux de Petri étendus où les transitions sont tirées après un délai **probabiliste** déterminé par une variable aléatoire.

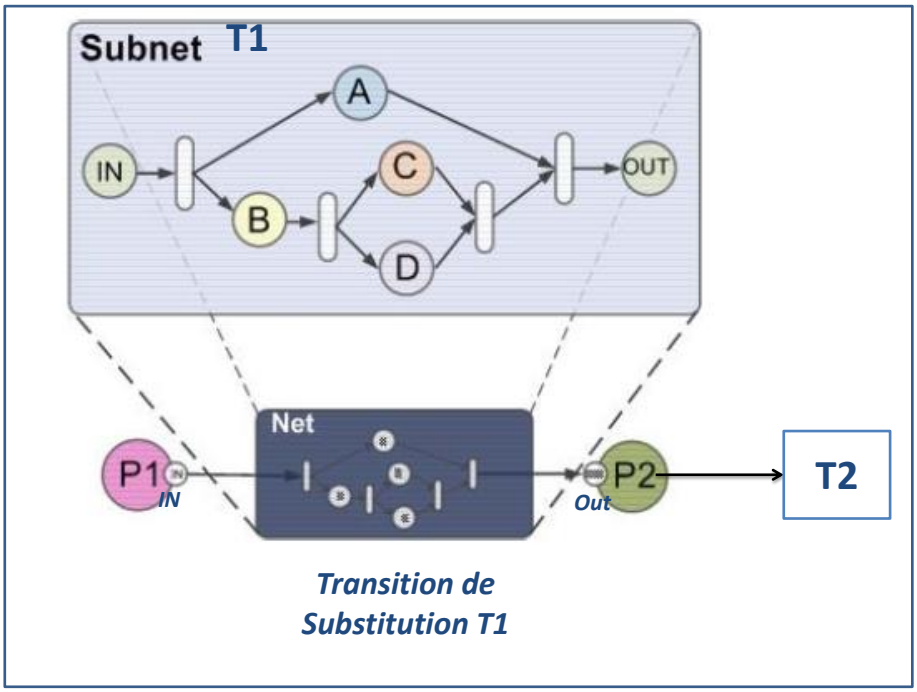


Figure 4.27: Exemple d'un RdP coloré hiérarchique

Les transitions comportent des temps de franchissement aléatoires, distribués par une loi de probabilité définie. Dans ce type de RdP, si une seule transition est franchissable, elle sera nécessairement franchie, mais si plusieurs transitions sont **concurremment franchissables** alors la **probabilité** permet de choisir laquelle tirer en premier. Ainsi, certains états du système ne sont atteignables que sous une certaine **probabilité**.

Le couple (marquage, instant d'entrée dans le marquage) constitue ainsi un **processus stochastique**. La théorie des graphes et la théorie des processus stochastiques sont alors exploitées pour l'évaluation des performances. Rappelons qu'un processus stochastique est un modèle mathématique utile pour la description de phénomènes de nature probabiliste en tant que fonction d'un paramètre qui a habituellement la signification du temps.
Pour certains types de SPN, les processus stochastiques sous-jacents sont des processus de Markov.

▪ **Exemple 4.11** Prenons l'exemple d'une lampe équipée d'une ampoule. La lampe peut être allumée et éteinte, et l'ampoule peut tomber en panne lorsque la lampe est allumée. Les ampoules défectueuses sont remplacées par des ampoules neuves, et avant d'effectuer l'opération de remplacement, l'interrupteur de la lampe est mis en position éteinte. Nous pouvons facilement identifier trois états dans notre système :

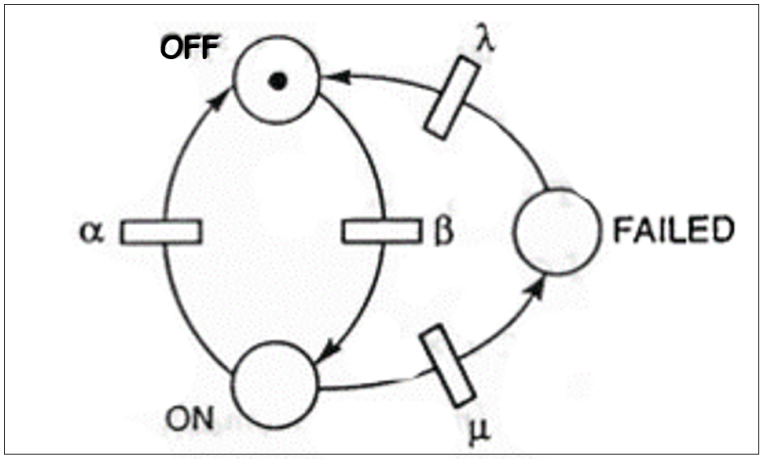
1. **Off** (éteinte)
2. **On** (allumée)
3. **Failed** (en panne)

Les transitions d'état obéissent aux règles suivantes :

- Lorsque la lampe est éteinte, elle peut être allumée,
- Lorsque la lampe est allumée, soit elle peut être éteinte, soit l'ampoule peut tomber en panne,

- Lorsque l'ampoule tombe en panne, elle est remplacée par une nouvelle, après avoir éteint la lampe.
  Supposons que :
- Les durées d'extinction de la lampe sont réparties de façon **exponentielle** avec paramètre β,
- Les périodes de temps pendant lesquelles la lampe est allumée sont réparties de façon **exponentielle** avec paramètre α,
- La durée de vie de l'ampoule (somme des durées des périodes d'allumage avant un défaut) est **exponentielle** distribuée avec le paramètre μ,
- le temps de réparation de la lampe est réparti de façon **exponentielle** avec le paramètre λ.

Le modèle SPN décrivant ce système est illustré par la figure suivante:



■

Initialement, un RdP stochastique a toutes ses transitions temporisées avec un temps aléatoire qui est distribué selon une loi probabiliste définie. Cependant et afin d'obtenir une représentation réelle du système, le RdP stochastique doit comporter, outre les transitions stochastiques, d'autres types de transitions notamment les transitions avec une temporisation nulle (immédiates) et les transitions avec une temporisation déterministe. De ce fait, nous distinguons différentes classes de RdP stochastiques à savoir:

RdP stochastiques généralisés (GSPN) Dans cette classe, introduite par Ajmone Marsan [1], le réseau comporte des transitions immédiates et des transitions stochastiques avec une temporisation aléatoire distribuée exponentiellement.

RdP stochastiques et déterministes (DSPN) Cette classe, également introduite par Ajmone Marsan [1], est une extension des *GSPN*. Le réseau contient des transitions immédiates, des transitions à temporisations déterministes et des transitions à temporisations stochastiques distribuées suivant une loi exponentielle (voir Figure 4.28).

RdP stochastiques étendus (ESPN) Dans cette classe, toutes les transitions sont des transitions temporisées aléatoires. Les temporisations stochastiques sont distribuées selon une loi de probabilité quelconque.

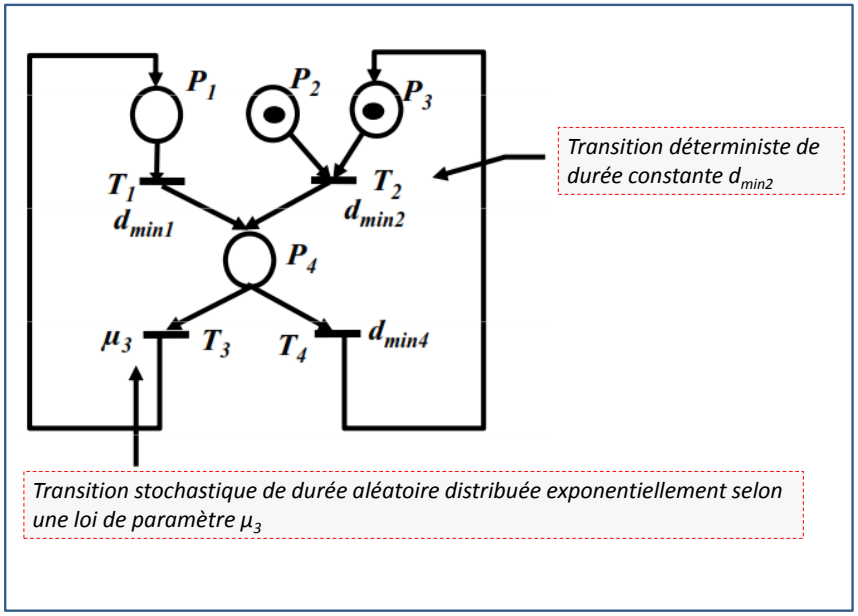


Figure 4.28: Exemple d'un RdP temporisé stochastique (DSPN)

4.7 Outils de Modélisation des Réseaux de Petri

Il existe plusieurs outils de simulation et de vérification des RdP selon la technique de vérification du modèle. Le tableau suivant présente une liste non exhaustive des plateformes de simulation des RdP et les classe par type de RdP qu'elles supportent [18].

Outils de modélisation	**RdP pris en charge**
LoLA; JFern; JARP; IOPT-Tools; Income Suite; INA; HISIm; GreatSPN; Geist3D; GHENeSys; GDTToolkit; FLOWer; ARP; Artifex; CPN-AMI; Disc Software Platform; ePNK; ExSpect	RdP places-transitions
JFern; Income Suite; INA; HISIm; GreatSPN; Geist3D; GHENeSys; F-net; ALPHA/Sim; ARP; **CPN Tools**; ExSpect	RdP temporisés
LoLA; JFern; Income Suite; INA; HISIm; HiQPN-Tool; Helena; GreatSPN; Geist3D; GDTToolkit; ALPHA/Sim; ALPiNA; Artifex; CoopnBuilder; COSA BPM; CPN-AMI; **CPN Tools**; ePNK; ExSpect; GDTToolkit; GHENeSys	RdP de haut niveau
QPME; PROTOS; Predator; PACE; ORIS; MISS-RdP; HISIm; HiQPN-Tool; GreatSPN; F-net; ExSpect	RdP sochastiques

En complément des plateformes du tableau ci-dessus, une liste encore plus détaillée est donnée sur le site [18]. Le choix d'une plateforme de simulation par rapport à une autre dépend, hormis le coût, d'autres facteurs tel que :

- Le **type de réseau** pris en charge : selon le type du système à modéliser, on peut être amené à utiliser des réseaux de Petri ordinaires ou encore des réseaux de Petri de haut niveau. Il faut dans ce cas choisir une plateforme qui peut prendre en charge le type de réseau que l'on va utiliser.
- La **capacité d'analyse** : il s'agit de la capacité de la plateforme de modélisation à déterminer les propriétés du réseau. Les outils qui permettent une extraction plus facile des résultats et un affichage plus clair sont souvent plus utilisés.
- Le **contrôle** du réseau de Petri lors de la **simulation** : une des problématiques des RdP est le tir simultané des transitions. Une plateforme de simulation de réseau devrait permettre, dans ce type de situation, de mettre en place une stratégie afin de définir la priorité de déclenchement lorsque deux transitions sont franchissables.
- **Une communauté d'utilisateurs et de développeurs active** : cela permet d'avoir d'une part un logiciel souvent mis à jour, des versions stables, et aussi de pouvoir trouver de l'aide en ligne en cas de difficulté à travers des tutoriels, vidéos et forum.

Au vu de tous ces critères, le choix de CPN Tools semble être très judicieux. Dans les paragraphes suivants, nous allons présenter brièvement la plateforme CPN Tools et ses caractéristiques.

4.7.1 CPNTools: un Outil de Spécification et de Vérification des RdP de Haut Niveau

La définition trop formelle des RdP convient plus aux mathématiciens qu'aux praticiens. Pour pallier à ce problème, **CPN Tools** a été conçu pour éviter les symboles formels. Développé par le CPN Group à l'université d'Aarhus au Danemark entre 2000 et 2010, CPN Tools [8] regroupe les trois des extensions présentées précédemment (RdP temporisé, RdP coloré et RdP hiérarchique) sous l'abréviation CPN (pour Coloured Petri Nets) quelque peu trompeuse car réduite à la seule extension colorée. C'est l'un des logiciels de modélisation les plus utilisés dans son domaine et qui est dédié à l'édition, la simulation et l'analyse des réseaux de Petri de haut niveau [3, 12, 13, 20].
CPN Tools est un outil très exploité par une large communauté d'universitaires et d'industriels ce qui a permis son développement. C'est un outil très puissant, dont l'ergonomie est assez originale et ludique. Le système ainsi que ses propriétés peuvent être spécifiés de manière visuelle et conviviale grâce à son éditeur graphique. L'outil offre deux modes de simulation: la simulation pas-à-pas et la simulation automatique. Dans chacun des modes, l'interface graphique assiste l'utilisateur dans la compréhension du comportement spécifié par un système d'activation par coloriages des composants comme le montre la Figure 4.29.

CPN Tools se compose de:

- Un éditeur (*CPN Editor*): permet d'éditer les RdP colorés (temporisés ou non) et de vérifier leurs syntaxes,
- Un simulateur intégré (CPN Simulator): permet de simuler le comportement de réseau en pas à pas ou automatiquement avec pour critère d'arrêt une date ou un nombre de tirs de transitions,
- Outil espace d'états (*State Space tool*): permet de construire le graphe d'occurrences et de l'exploiter pour la vérification des propriétés, telles que la vivacité et la bornitude, à l'aide d'un ensemble de primitives.

CPN Tools comporte des ensembles de couleurs simples (unité, entier, booléen, string

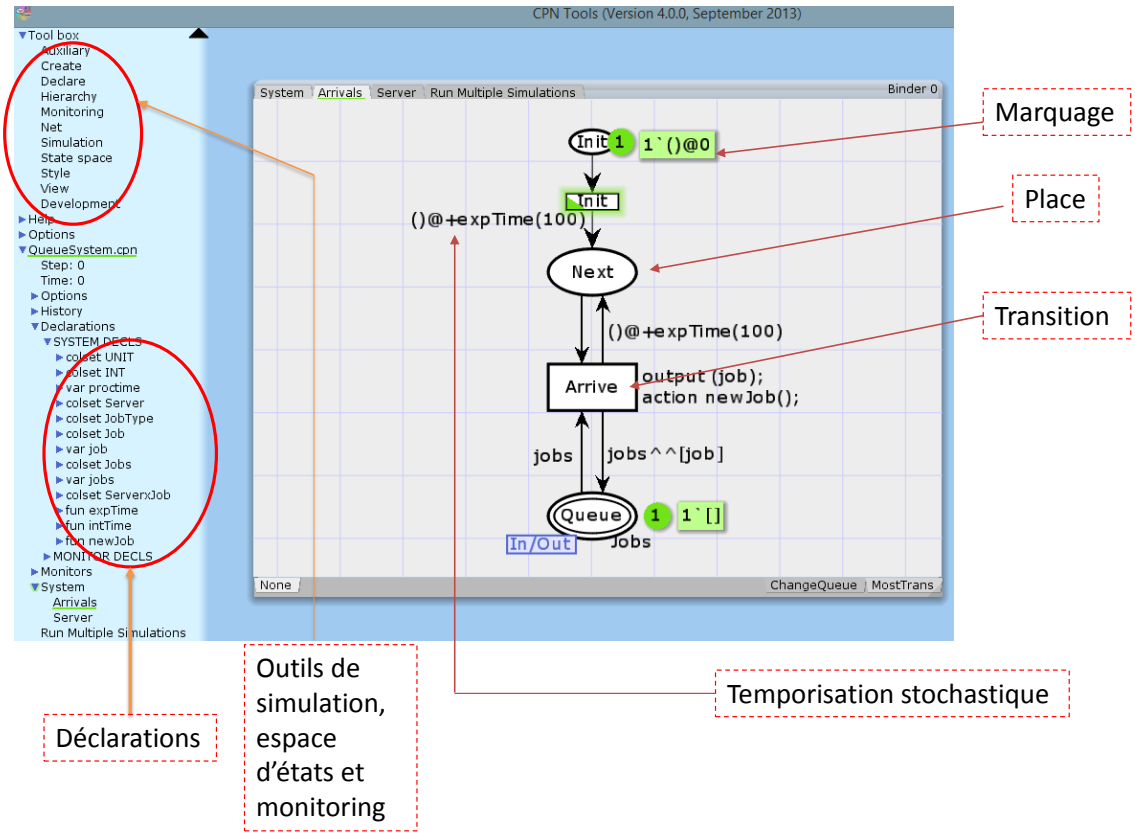


Figure 4.29: L'interface graphique de l'outil CPN Tools

(caractère), énuméré, indice) mais permet la définition de couleurs plus complexes. en plus de cela, grâce à un ensemble de structures et de syntaxes, des fonctions peuvent être construites permettant ainsi la possibilité d'élaborer un RdP paramétrable que nous pouvons instancier en changeant seulement les déclarations.

4.7.2 Caractéristiques de CPN Tools

Les moteurs de calcul et les traitements sont écrits en **CPN ML** qui est une variante du langage de programmation **ML** [10]. Un des atouts majeurs de CPN Tools est la combinaison des fonctionnalités des réseaux de Petri colorés et celles du langage fonctionnel Standard ML [10]. Les RdP colorés fournissent les primitives pour décrire les processus, tandis que le langage fournit les primitives pour définir des types de données (ensemble de couleurs) et des manipulations des données (expression d'arcs, gardes, etc.) (voir Figure 4.29).

Ce couplage des réseaux CPN à un langage de modélisation fonctionnelle (ML) permet une manipulation plus aisée des trois extensions précédentes et la définition de structures complexes telles que les listes par exemple.

Par ailleurs, cet outil a permis l'introduction de variables aléatoires, définies par une distribution de probabilité quelconque, pour caractériser la durée associée aux temporisations des places.

Ces deux notions complémentaires contribuent, de manière indiscutable, à accroître le pouvoir d'expression des RdP colorés de Jensen [12, 13] mais peuvent réduire, en contrepartie, les capacités de raisonnement formel associés aux RdP et ses extensions temporisés, colorés et hiérarchiques. En effet, l'analyse formelle d'un graphe de marquage peut être difficile, voire impossible dans certains cas. Le recours à la simulation de Monte-Carlo est donc d'un usage très fréquent dans les applications développées à l'aide de CPN Tools, notamment pour l'évaluation de performances.

4.7.3 Evaluation des Performances Moyennant CPN Tools

Face à la complexité et la taille des modèles, l'approche par simulation est très largement privilégiée dans la littérature, et notamment la simulation de Monte-Carlo. Cette dernière a pour essence l'utilisation d'expériences répétées pour évaluer une quantité donnée.

le simulateur 'Server CPN' [12, 13] est utilisé pour effectuer les différentes simulations de Monte-Carlo et enregistrer les résultats obtenus par les différents moniteurs pour l'évaluation de performances.

Définition 4.3
Un **moniteur** est chargé de collecter des données statistiques lors de la simulation Il constitue un mécanisme utilisé pour observer, inspecter, contrôler les simulations dans le but de collecter des données, mesurer la performance, définir des points de ruptures de simulation spécifiques au modèle. Il supporte aussi l'exécution de plusieurs réplications de simulation. Ils peuvent aussi modifier une simulation d'un réseau de Petri coloré et temporisé. De nombreux moniteurs différents peuvent être définis pour un réseau de Petri donné.

Sur la base des informations collectées, les moniteurs peuvent également effectuer des traitements ayant un intérêt vis à vis de la performance à évaluer. Un moniteur peut réaliser plusieurs traitements tels que:

- Arrêter une simulation lorsqu'un endroit particulier est vide,
- Compter le nombre de fois qu'une transition se produit,
- Mettre à jour un fichier lors du franchissement d'une transition avec une variable liée à une valeur spécifique à observer,
- Calculer le nombre moyen de jetons dans une place.

Pour plus de détails et une bonne prise en main de la plateforme CPN Tools, le lecteur peut se référer à une documentation très riche offerte sur le site officiel de la paleforme [8] ou aux ouvrages de Jensen [12, 13].

4.8 Conclusion

Dans ce chapitre, nous avons présenté la famille des formalismes basés sur les RdP. Nous avons décrit de manière formelle et informelle les extensions les plus utilisées dans la construction de différents types de modèle.
Enfin, nous avons présenté l'outil CPN Tools qui est un outil pour la modélisation, l'analyse et la simulation des RdP de haut niveau.

4.9 Exercices

Exercice 4.1 : un système bancaire simple

Considérons un système bancaire simple. Le système gère les comptes. Chaque compte a un numéro de compte, un titulaire de compte et un solde. Les titulaires de compte peuvent déposer ou retirer de l'argent. Seuls les montants inférieurs à 5 000 euros peuvent être retirés. En outre, le système n'accepte pas les transactions qui entraînent un solde négatif.

1. Modéliser le système par un RdP coloré.
2. Déterminer les différents types des différentes places de ce RDPC.
3. Implémenter le modèle avec CPN Tools.

Exercice 4.2 : Production d'un jouet automobile

Le processus de production d'un jouet automobile comprend quatre étapes : assemblage, peinture, séchage et emballage. Lors de l'étape d'assemblage, un travailleur de la production assemble quatre roues et un châssis. Cette étape prend cinq minutes. Ensuite, un peintre peint le produit. Cela prend huit minutes. Après la peinture, le produit sèche pendant au moins vingt minutes avant de pouvoir être emballé. La pièce dans laquelle les produits peuvent sécher est de taille limitée : au plus dix produits peuvent sécher en même temps. Enfin, un travailleur de la production fait l'emballage de la voiture-jouet. Cela prend dix minutes. Cela prend dix minutes. L'entreprise emploie trois ouvriers de production et deux peintres. La capacité de chaque employé est 1.

1. Modéliser le processus de production comme un réseau de Petri étendu avec le temps. Montrez le marquage initial et la façon dont le temps est modélisé.
2. Les voitures peuvent être peintes en deux couleurs (bleu et rouge). Les voitures rouges prennent sept minutes pour être peintes et vingt-cinq minutes pour sécher, et les voitures bleues prennent neuf minutes pour être peintes et dix-huit minutes pour sécher. Changer la solution dans (1) de telle sorte que cela soit possible.

Exercice 4.3 : Problème des philosophes

Considérons le problème du « dîner des philosophes » qui est un cas classique sur le partage de ressources en informatique système. La situation est la suivante : "Cinq philosophes, numérotés de 1 à 5, vivent dans une maison où ils se trouvent autour d'une table pour manger. Chaque philosophe a sa propre place à la table : Leur seul problème - outre ceux de la philosophie - est que le plat servi est un plat de spaghetti particulier, qui doit être mangé avec deux fourchettes. Il y a deux fourchettes à côté de chaque assiette, de sorte que cela ne présente aucune difficulté. En conséquence, cependant, deux voisins ne peuvent pas manger en même temps. "

1. Modéliser le système des philosophes par un réseau de Petri ordinaire. Il est supposé que chaque philosophe prend simultanément (et indivisiblement) sa paire de fourchettes. De manière analogue, il les pose simultanément (et indivisiblement). Chaque philosophe peut être représenté par deux places (penser et manger) et deux transitions (prendre et poser des fourchettes). Chaque fourchette peut être représentée par une seule place qui contient un jeton lorsque la fourchette n'est pas utilisée.
2. Modélisez le même système par un réseau de Petri coloré qui contient les deux ensembles de couleurs Phils = ph1, ..., ph5 et Forks = f1, ..., f5 représentant respectivement les philosophes et les fourchettes. (Indice : considérer que les deux ensembles de couleurs soient de type Nombre, c'est-à-dire, représenter chaque philosophe et chaque fourchette par un entier.)
3. Modifiez le modèle créé en (2) de telle sorte que chaque philosophe prenne d'abord sa fourchette droite et ensuite la gauche. De manière analogue, le philosophe dépose

d'abord sa fourchette gauche puis sa droite. Cette modification modifie-t-elle le comportement global du modèle ?

Exercice 4.4 : **Stationnement de voitures dans un aéroport**

Nous désirons modéliser le stationnement des voitures dans le parking d'un aéroport. Le stationnement dans cet aéroport suit certaines règles : Les voitures qui arrivent peuvent choisir entre deux parkings A et B. Les parkings A et B ont respectivement une capacité de 50 et 100 voitures. Pour entrer dans un parking, la voiture doit passer une barrière. La barrière s'ouvre seulement s'il y a au moins une place de stationnement disponible. Une lumière montre si c'est le cas. Passer la barrière prend une minute. Si aucune place de stationnement n'est disponible, les voitures doivent circuler. Les voitures entrant dans le parking A ont besoin d'une minute pour trouver une place de stationnement et quitter le parking après dix minutes.
En raison de la taille du parking B, il est plus difficile de trouver une place de parking. Si plus de deux tiers de la capacité de B est atteinte, les conducteurs ont besoin de cinq minutes pour trouver une place de stationnement ; sinon, cela prend deux minutes. Les voitures se garant au parking B partiront après un certain temps défini par une fonction $f()$.

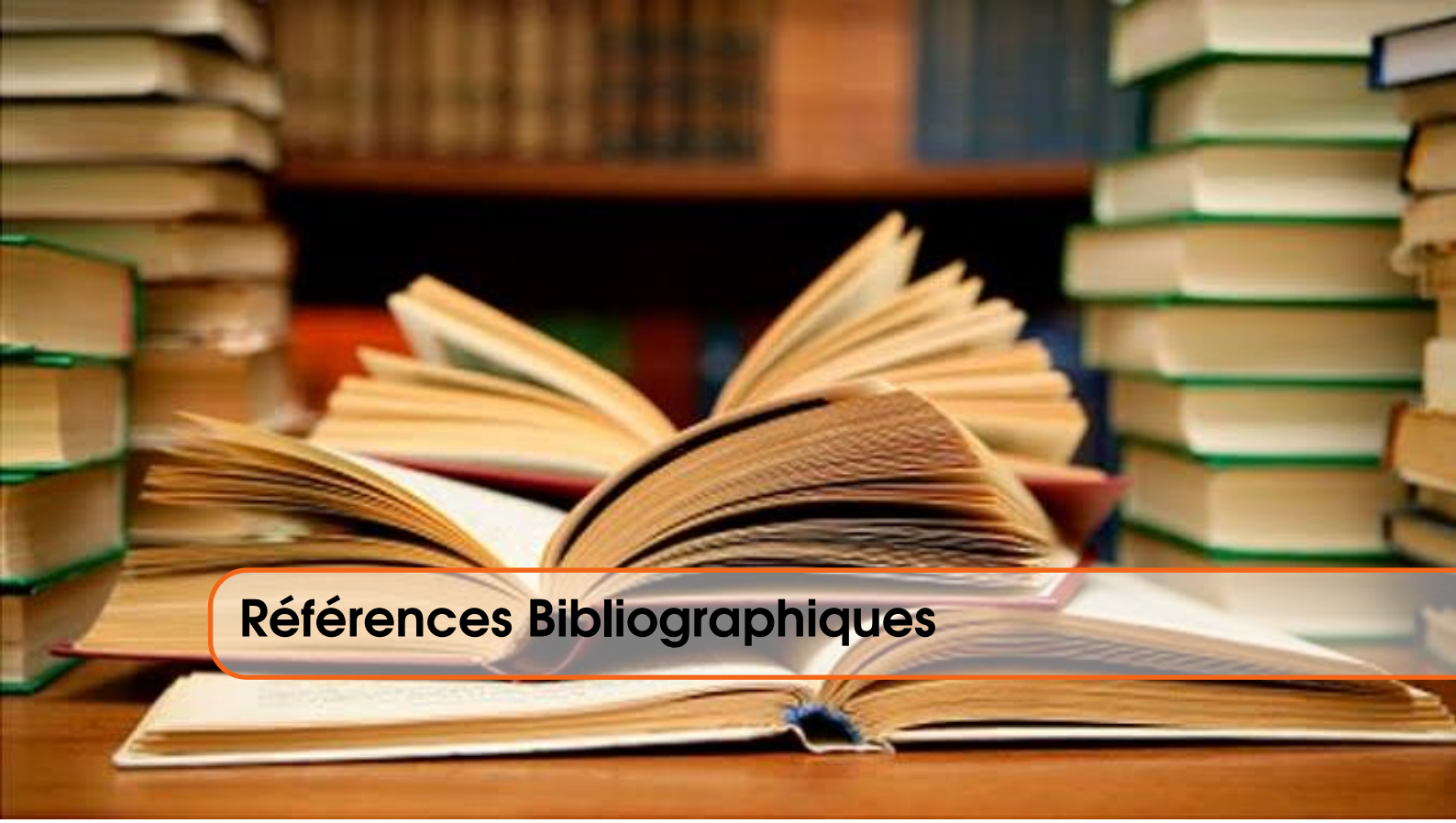
1. Modéliser le processus de stationnement, de l'arrivée d'une voiture jusqu'à son départ, par un RdP temporisé.
2. Étendre le modèle en (1) de sorte que seules les voitures d'une hauteur inférieure à deux mètres puissent entrer dans la barrière du parking B.

Exercice 4.5 : **Un service d'établissement de connexion**

On considère un système constitué de 02 processus qui peuvent communiquer entre eux selon un protocole de connexion/déconnexion décrit comme suit:

- Le processus appelant se considère connecté dès qu'il envoie une requête de connexion *CR* ; le processus appelé est connecté dès qu'il reçoit un *CR*.
- De même, le processus appelant se considère déconnecté dès qu'il envoie une requête de déconnexion *DR* ; le processus appelé est déconnecté dès qu'il reçoit un *DR*.

1. Modéliser ce protocole par un RdP coloré. Montrer que le modèle obtenu n'est pas borné; justifier pourquoi?
   conseil: modéliser chacun des deux processus et le canal de communication par un sous-RdP.
2. On désire corriger ce protocole en ajoutant un mécanisme d'acquittement. Chaque demande de connexion et de déconnexion doit être acquittée avant d'être considérée comme effective. Dans ce cas, l'appelant doit attendre l'acquittement (*Ack*) avant de poursuivre son déroulement. Modifier le modèle RdP précédent pour prendre en compte ce mécanisme de synchronisation. Est ce que le modèle modifié est borné? si oui, quelles sont les composantes conservatives?
3. En supposant que le canal de communication peut être défaillant provoquant de temps à autre une perte de messages. Dans ce cas le processus en attente du message perdu risque de se retrouver en état d'attente infinie! Pour pallier à ce problème, on propose que l'attente d'un processus ne doit pas dépasser un délai (d) (ou $d \geq$ temps de transmission d'un message + temps de réception d'un *Ack*). En outre, on suppose que les durées d'une de transmission et de réception de messages dépendent de l'état du canal de communication et sont donc aléatoires (entre [dmin, dmax]). Proposer un RdP coloré et temporisé pour modéliser cette version du protocole. En fixant $d = 100$, $dmin = 25$ et $dmax = 75$, implémenter puis simuler le modèle en utilisant la plateforme CPN Tools.



Références Bibliographiques

[1] Marco Ajmone Marsan, Gianni Conte, and Gianfranco Balbo. “A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems”. In: *ACM Transactions on Computer Systems (TOCS)* 2.2 (1984), pages 93–122 (cited on page 88).

[2] Gianfranco Balbo. “Introduction to stochastic Petri nets”. In: *Lectures on Formal Methods and PerformanceAnalysis*. Edited by chastic. Springer, 2001, pages 84–155 (cited on page 23).

[3] Michel Beaudouin-Lafon et al. “CPN/Tools: A post-WIMP interface for editing and simulating coloured Petri nets”. In: *International Conference on Application and Theory of Petri Nets*. Springer. 2001, pages 71–80 (cited on page 90).

[4] Yue Ben. “A Brief Study on Stochastic Petri Net”. In: (2013) (cited on page 23).

[5] Fred DJ Bowden. “A brief survey and synthesis of the roles of time in Petri nets”. In: *Mathematical and Computer Modelling* 31.10 (2000), pages 55–68 (cited on page 74).

[6] Christos G Cassandras and Stephane Lafortune. *Introduction to discrete event systems*. Springer Science & Business Media, 2009 (cited on page 13).

[7] Franck Cassez and Olivier H Roux. “Traduction structurelle des réseaux de petri temporels vers les automates temporisés”. In: *Hermès Science, éditeur: 4ème Colloque Francophone sur la modélisation des Systèmes Réactifs (MSR 03)*. Volume 54. 2003 (cited on pages 23, 74).

[8] CPN Tools. *CPN tools homepage*. `http://www.cpntools.org`. Retrieved January, 2015 (cited on pages 24, 90, 92).

[9] Gerard Florin, Céline Fraize, and Stéphane Natkin. “Stochastic Petri nets: Properties, applications and tools”. In: *Microelectronics Reliability* 31.4 (1991), pages 669–697 (cited on page 86).

[10] Robert Harper. “Programming in standard ML”. In: (2001) (cited on page 91).

[11] Branislav Hrúz and MengChu Zhou. *Modeling and control of discrete-event dynamic systems: With petri nets and other tools*. Springer Science & Business Media, 2007 (cited on pages 13, 17, 43).

[12] Kurt Jensen and Lars M Kristensen. *Coloured Petri nets: modelling and validation of concurrent systems*. Springer Science & Business Media, 2009 (cited on pages 23, 24, 26, 43, 50, 81, 86, 90, 92).

[13] Kurt Jensen, Lars Michael Kristensen, and Lisa Wells. "Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems". In: *International Journal on Software Tools for Technology Transfer* 9.3-4 (2007), pages 213–254 (cited on pages 81, 86, 90, 92).

[14] Philip Meir Merlin. "A study of the recoverability of computing systems." In: (1974) (cited on page 74).

[15] Tadao Murata. "Petri nets: Properties, analysis and applications". In: *Proceedings of the IEEE* 77.4 (1989), pages 541–580 (cited on pages 23, 24, 50).

[16] Tuncer Ören. "The many facets of simulation through a collection of about 100 definitions". In: *SCS M&S Magazine* 2.2 (2011), pages 82–92 (cited on page 19).

[17] Carl Adam Petri. "Introduction to general net theory". In: *Net theory and applications*. Springer, 1980, pages 1–19 (cited on pages 23, 24, 43).

[18] Petri Nets Tools Database. *Petri Nets Tools Database homepage*. `https://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html`. Retrieved October, 2019 (cited on page 89).

[19] Chander Ramchandani. "Analysis of asynchronous concurrent systems by timed Petri nets". In: (1974) (cited on page 74).

[20] Anne Vinter Ratzer et al. "CPN tools for editing, simulating, and analysing coloured Petri nets". In: *International Conference on Application and Theory of Petri Nets*. Springer. 2003, pages 450–462 (cited on page 90).