République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de fin d'études Master

Filière : Informatique

Option : système informatique

Thème :

Conception et implémentation d'un simulateur pour les réseaux de Pétri réversibles

Encadré Par :

Mr. Benamira Adel

President :

Mr. Chaoui Mohammed

Examinateur:

Mr. Brahimi Said

Présenté par :

Mr. Keita Namory

Juin 2023

Remerciements

Louange à Allah, de me donner la force et la patience de survivre, ainsi que le courage de surmonter toutes les difficultés pour atteindre la fin de mon master et compléter ce modeste travail.

J'exprime mes remerciements les plus sincères à mon encadreur, Monsieur Benamira Adel d'avoir eu la gentillesse de m'encadrer, pour l'assistance qu'il a témoignée envers moi pour sa disponibilité, ses orientations et ses précieux conseils.

J'aimerais aussi rendre hommage à tous mes professeurs, qui ont contribué à ma formation durant tout le long de mon cursus.

Je remercie aussi tous mes ami(e)s et collègues qui ont participé de près ou de loin à la réalisation de ce mémoire.

A toutes ces personnes qui ont un jour cru en moi, je vous remercie toutes et tous du fond de mon cœur.

Dédicaces

Je dédie ce modeste travail de mes années d'études :

A mes très chers parents, en leurs adressant mes plus vifs remerciements pour leur amour et soutient qu'ils m'ont apporté durant toutes mes années d'étude.

A mes tantes, Je mentionne particulièrement Mariama Keita

A mes oncles, Je mentionne particulièrement Moussa Abdoul Karim Haidara

Mes sœurs et mes frères, votre soutien et encouragement me marqueront à jamais.

Mes amis, Je mentionne particulièrement Hamadoun Cisse qui n'ont cessé d'être pour moi une source de motivation inépuisable. Et à tous ceux qui ont contribué de près ou de loin pour que ce projet soit possible, je vous dis merci

Résumé

La vérification formelle consiste à traduire le système, préalablement décrit dans un langage approprié, vers un modèle ”sémantique”, généralement un graphe d’états, sur lequel les propriétés de correction attendues sont vérifiées au moyen d’algorithmes spécifiques. Bien que limitée aux applications ayant un nombre fini d’états, cette approche est particulièrement utile dans les premières phases du processus de conception, permettant ainsi une détection rapide et économique des erreurs. La vérification formelle est fondée sur les trois éléments suivants : un langage de description du système (appelé modèle de spécification formelle), un modèle sémantique du parallélisme et un langage de spécification de propriétés. Les réseaux de Pétri réversibles ont été développés en tant que formalisme opérationnel pour la spécification des systèmes concurrents réversibles. Ils donnent une représentation dynamique d’un état du système par l’utilisation de jetons. Cette sous classe du réseau de Pétri a été utilisée avec succès pour modéliser une grande variété de systèmes industriels. L’objectif de ce présent travail, est de développer un outil qui nous permet de simuler un réseau de Pétri réversible avec des jetons distinctifs où chaque jeton est une ressource unique.

Mots Clés : Système réversible, Vérification formelle, Réseaux de Pétri récursives.

Sommaire

Listes des figures

Introduction générale

Les réseaux de Pétri sont un modèle mathématique largement utilisé pour la modélisation et l'analyse des systèmes concurrents. Ils offrent une représentation graphique puissante des interactions entre les différentes entités d'un système, permettant ainsi de comprendre et d'analyser son comportement.

Cependant, les réseaux de Pétri classiques présentent certaines limitations lorsqu'il s'agit de modéliser des systèmes réversibles, c'est-à-dire des systèmes où il est possible de revenir en arrière dans l'exécution. Afin de pallier ces limitations, il est nécessaire de développer un éditeur de réseau de Pétri réversible. Un tel éditeur permettrait de modéliser précisément les systèmes réversibles en prenant en compte les transitions réversibles.

L'objectif de ce mémoire est donc de concevoir et de réaliser un tel éditeur, en mettant l'accent sur la réversibilité des transitions. Cela permettra aux utilisateurs de créer, visualiser et analyser des réseaux de Pétri avancés, tout en tenant compte des comportements réversibles.

En fournissant un éditeur graphique avancé pour les réseaux de Pétri réversibles, cet effort de recherche vise à faciliter la modélisation précise des systèmes concurrents réversibles et à améliorer l'analyse des comportements complexes. Il offre également de nouvelles perspectives pour la simulation, la vérification et la compréhension des systèmes réversibles, ouvrant ainsi des possibilités d'application dans divers domaines tels que l'informatique, les télécommunications, la biologie et bien d'autres.

Ainsi, la réalisation de cet éditeur de réseau de Pétri réversible représente une contribution significative à la recherche en modélisation des systèmes concurrents et offre des outils précieux pour les praticiens cherchant à comprendre et à analyser les systèmes complexes.

Notre travail est organisé de la manière suivante :

Dans la première partie, nous allons décrire le calcul réversible et ses différentes formes. Ensuite nous allons décrire les méthodes formelles.

Dans la seconde partie nous parler des réseaux de pétri, du graphe de marquage, de la réversibilité dans les réseaux de pétri, de la sémantique de causalité et les réseaux de pétri réversible.

Dans la troisième partie, nous allons donner une conception d’un éditeur graphique de réseau de pétri. Ensuite l’implémentation de notre éditeur.

Chapitre 01 : Calcul réversible et Méthodes formelles

Introduction

Dans ce chapitre, nous allons parler du calcul réversible et des méthodes formelles.

Au début nous allons définir le calcul réversible et décrit ses différents formes. Ensuite nous allons définir les méthodes formelles et le type de spécification et ses avantages.

1. Calcul Réversible

1.1 Définition

Le calcul réversible est une forme de calcul dans laquelle le calcul peut être effectué en avant et en arrière [1].

La motivation pour l'informatique réversible a commencé avec l'observation de Landauer selon laquelle seul le calcul irréversible génère de la chaleur, ce qui a donné naissance à une forte ligne de recherche sur la création de porte logique et de circuits réversible. Par conséquent, le calcul réversible est favorable à l'informatique à faible consommation d'énergie. Par la suite, la motivation pour étudier la réversibilité est née d'une grande variété d'applications qui intègrent naturellement un comportement réversible. Il s'agit notamment de systèmes biologiques dans lesquels le calcul peut être effectué en avant ou en arrière, les systèmes chimiques, les systèmes de transactions, les systèmes de débogage, les problèmes d'exploration spatiale et les calculs quantiques. Dans ce type de système, chacun peut revenir automatiquement à un état spécifique en cas d'erreur ou de points de contrôle [1] [2].

1.2 Formes de réversibilité

Dans le calcul réversible, la stratégie à appliquer lors d'un retour en arrière est l'une des questions les plus importante qui se posent. Plusieurs approches ont été explore dans la littérature au cours de la dernière décennie, qui diffèrent par l'ordre dans lequel les étapes sont exécuté en arrière. Les plus importantes d'entre elles sont le retour en arrière (backtracking), la réversibilité causale (causal reversing) et la réversibilité hors de l'ordre causale (out-of-causal-order reversing) [1].

1.1.1. Le retour en arrière(backtracking)

Le retour en arrière est le processus dans lequel les évènements sont annulés dans l'ordre inverse de leur apparition [2]. Cette forme d'inversion garantit qu'a tout état d'un calcul, il existe au plus un état prédécesseur. Dans le contexte des systèmes concurrent, cette forme de réversibilité peut être considérée comme trop restrictive, car le fait de défaire les actions

uniquement dans l'ordre dans lequel elles ont été prise induit de fausses dépendances causales sur les séquences d'action en arrière : les actions, qui auraient pu être inversées dans n'importe quel ordre, sont forcées d'être défaites dans l'ordre précis dans lequel elles se sont produites [1].

1.1.1. La réversibilité causale (causal reversing)

La réversibilité causale relâche la rigidité du retour en arrière. Elle permet une forme plus flexible de réversibilité en autorisant les évènements à s'inverser dans un ordre arbitraire, en supposant qu'ils respectent les dépendances causales qui existent entre eux [1]. C'est-à-dire que l'inversion causale signifie que les évènements qui causent d'autres évènement ne peuvent être annulés qu'après que les évènements causés aient été annulés en premier ; ainsi, des évènements indépendants peuvent être annulés dans n'importe quel ordre, indépendamment de l'ordre dans lequel ils se sont effectivement produits [2].

1.1.2. La réversibilité hors de l'ordre causale (out-of-causal-order reversing)

Il existe cependant de nombreux exemple concret où le fait de défaire des choses dans un ordre non causal est inhérente ou pourraient être bénéfique. En fait, cette forme d'annulation joue un rôle essentiel dans les mécanismes qui régissent les transactions à long terme et les réactions biochimiques. Considérez que chaque état de l'exécution est le résultat d'une série d'actions qui ont contribué de manière causale à l'existence de l'état actuel. Si les actions devaient être inversées d'une manière respectueuse de la causalité, nous ne serions en mesure que d'aller et venir à travers des états précédemment visité. Par conséquent, ne peut souhaiter appliquer une réversibilité hors ordre afin de créer de nouvelles alternatives d'état actuels qui étaient auparavant inaccessible par un chemin d'exécution uniquement vers l'avant [1].

Puisque la réversibilité hors ordre causale contredit l'ordre du programme en violant les lois de la causalité, elle s'accompagne de ses propres particularités qui doivent être prise en considération lors de la conception de systèmes réversible[1].

2. Méthodes Formelles

2.1. Définition

Les méthodes formelles consistent à utiliser les mathématiques pour le développement de logiciels. Les principales activités sont [3]:

- L'écriture d'une spécification formelle ;
- La preuve de certaines propriétés de cette spécification ;
- La construction d'un programme en manipulant mathématiquement la spécification ;

- La vérification du programme à l‘aide de raisonnements mathématiques.

La spécification est le premier document décrivant les fonctionnalités requises du système à réaliser pour le client. Elle ne doit décrire que ce que le système doit faire, pas comment il doit le faire. Elle joue deux rôles importants : un document contractuel avec le client décrivant ses besoins et exigences, et un document normatif sur la base duquel le concepteur devra produire une solution [3].

A la fin de chaque étape du cycle de développement, un document appelé spécification est produit. Le terme spécification peut être utilisé à différents niveaux du développement du système : spécification des exigences, implémentation, module, propriété, tests, procédures de test, données, traitement. La spécification initiale représente le cahier des charges du produit et la spécification finale représente le produit lui-même.[3].

2.2. Type de spécification

Les spécifications peuvent être divisées en trois types selon leur degré de formalisation : les spécifications informelles, les spécifications semi-formelles et les spécifications formelles. Les spécifications informelles, qui sont décrites en langage naturel, manquent souvent de précision. La description ci-dessous n'est pas suffisamment précise et claire, elle n'explique pas si la zone désignée peut être dispersée ? [3].

Fragment de spécification d’un editeur de texte
La sélection est le processus de désignation des zones du document que vous souhaitez travailler dessus. La plupart des actions d’édition et de formatage nécessite deux étapes : d’abort vous sélectionnez le/les objets que vous voulez travailler dessus, tels que du texte ou des graphiques ; puis vous vous lancez l’action appropriée.

Figure 1.1:Exemple de spécification informelle

Les spécifications semi-formelles utilisent des langages de spécification textuels ou graphiques avec une syntaxe précise et une sémantique plutôt faible, comme Merise et UML.

Figure 1.2:Exemple de spécification semi-formelle

Les spécifications formelles sont exprimées dans un langage avec une syntaxe et une sémantique précise, elles sont construites sur une base théorique solide (mathématiques), et permettent une vérification automatique. Les réseaux de Pétri, les grammaires formelles, les automates à états finis, la logique formelle, l'algèbre des processus, la théorie des graphes sont des exemples de telles techniques [3].

```
Specification Distributeur_Client [MONNAIE , THE, CAFE, CHOCOLATE, CHANGE] : noexit
Behaviour
        Distributeur [MONNAIE , THE, CAFE, CHOCOLATE, CHANGE, CHOIX]
                | [MONNAIE , THE, CAFE, CHOCOLATE, CHANGE] |
        (Client [MONNAIE, THE, CAFE, CHOCOLATE, CHANGE, CHOIX]
        |||
        Client [MONNAIE, THE, CAFE, CHOCOLATE, CHANGE, CHOIX])
Where
.
.
.
Endspec.
```

Figure 1.3: Exemple de spécification formelle

Puisqu‘elles sont fondées sur les mathématiques, les spécifications formelles sont précises alors que les spécifications informelles et les spécifications semi-formelles sont ouvertes à la réinterprétation.

2.3. Les avantages des méthodes formelles

Le principal avantage des méthodes formelles est l'utilisation de concepts issus de la logique et de la technique mathématique. Ces concepts fournissent des outils efficaces pour organiser la pensée des concepteurs et faciliter la communication entre toutes les personnes impliquées dans la programmation. De plus, ils nous permettent de décrire de manière précise et sans ambiguïté

les demandes de l'utilisateur du logiciel à exécuter. Les concepts d'ensemble, de relations, de fonctions et leurs diverses propriétés et opérations, ainsi que la quantification universelle et existentielle, permettent d'établir une spécification de manière simple et claire et de démontrer mathématiquement les propriétés d'une spécification [4].

Les principaux avantages techniques d'une spécification formelle par rapport à une spécification informelle sont la précision et la clarté. Des inexactitudes et des ambiguïtés peuvent facilement se glisser dans des spécifications informelles. Cela pourrait ouvrir la voie à de multiples interprétations. En revanche, les termes de spécifications formelles n'ont qu'une seule interprétation [4].

Un autre avantage des spécifications formelles est que les questions peuvent être posées et répondues précisément de manière scientifique. De plus, les approches formelles fournissent des spécifications qui peuvent être rigoureusement vérifiées, analysées et testées tôt dans le cycle de développement, ce qui n'est pas le cas des approches informelles. Cela signifie que les défauts peuvent être détectés et corrigés à un stade précoce, réduisant ainsi les coûts et le temps de développement et améliorant la qualité du logiciel [4].

Les méthodes formelles nous permettent de spécifier ce qui est requis à un niveau particulier d'abstraction. Certains comportements et propriétés peuvent être intentionnellement exclus s'il est préférable de reporter leur spécification détaillée à une étape ultérieure du cycle de développement [4].

Selon Hoffman et Stooper, des spécifications strictes jouent un triple rôle dans le développement logiciel. Premièrement, une spécification documente avec précision les décisions de conception, est indépendante de la mise en œuvre et sert de base aux revues de conception. Lors de l'implémentation, la même spécification prend en charge le développement parallèle. D'une part, ils disent à l'utilisateur à quoi s'attendre, d'autre part, ils disent au programmeur quoi faire et servent de base à la phase de test Enfin, lors de la maintenance, ces mêmes spécifications supportent l'analyse des évolutions et contribuent à la formation du nouveau personnel [4].

Conclusion

Nous avons donné une présentation des concepts de calcul réversible et de méthodes formelle. Dans le chapitre suivant nous allons présenter les réseaux de pétri réversible.

Chapitre 02 : Réseaux de Pétri Réversibles

Introduction

Dans ce chapitre, nous allons évoquer les concept clé sur les réseaux de pétri qui sont le graphe de marquage, la réversibilité dans les réseaux de pétri, la sémantique de causalité et les réseaux de pétri réversible. Nous essayerons de donner une description de ses concepts.

1. Réseau de Pétri

Les réseaux de Pétri ont été introduits en 1962 par Carl Adam Pétri dans sa thèse "Kommunikation mit Automaten". Possédant une représentation graphique simple, les réseaux de Pétri permettent de modéliser des systèmes dynamiques échangeant des ressources. Ainsi, à l'aide des réseaux de Pétri on peut modéliser et analyser les systèmes discrets, particulièrement les systèmes concurrents, représenter les concepts de parallélisme, de synchronisation (rendez-vous, précédence...), de partage de ressources, de communication, de causalité [5]

De manière informelle, un réseau de Pétri est un formalisme mathématique qui peut être représentée par un graphe biparti avec deux types de nœuds. Des places (Pi) représentant l'état du système modélisé, elles sont représentées par des cercles. Les transitions (Ti) qui représentent des événements ou des actions qui entraînent des changements d'état, elles sont représentées par des rectangles. Les places et les transitions sont reliées par des arcs évalués. Il n'y a pas d'arcs entre les places ou entre les transitions. La valeur 1 est omise et aucun arc ne représente la valeur 0. Un réseau de Pétri est un graphe doté d'une sémantique opérationnelle, c'est-à-dire un comportement associé au graphe, ce qui permet de décrire la dynamique d'un système représenté. Pour cela, des jetons sont ajoutés à ces places. Ces jetons sont représentés par des points ou des chiffres à l'intérieur des places. Une répartition des jetons dans les places à un instant donné est appelée marquage du réseau de Pétri [3][6].

1.1.Définition

Formellement, le réseau de pétri est un tuple (S, T, F, I, L) avec : [7]

- S et T deux ensemble disjoint de place et de transition
- $F : (S \times T \cup T \times S) \rightarrow IN$, la relation de flux
- $I: S \rightarrow IN$, le marquage initial
- $L : T \rightarrow A$, pour A un ensemble d'action, la fonction d'étiquetage

Les réseaux de pétri sont représentés en dessinant les places sous forme de cercles et les transitions sous formes de rectangles, contenant leur étiquette. Pour $x, y \in S \cup T$, il y a F(s,t)

arcs de x à y. lorsqu'un réseau de pétri représente un système concurrent, un état global de ce système est donné sous forme de marquage, une fonction M : S → IN. Un tel état est représenté en plaçant M(s) points (tokens) à chaque place s. L'état initial est donné par le marquage I. Afin de décrire le comportement du réseau, on définit la relation de transition de pas entre les marquages [7].

2. Graphe de marquage

Pour définir l'état d'un système modélisé par un réseau de Pétri, il est nécessaire de compléter le réseau de Pétri par un marquage. Ce marquage consiste à disposer un nombre entier (positif ou nul) de jetons dans chaque place du réseau de Pétri [5].

Un graphe de marquage, également appelé espace d'états, est une représentation graphique de toutes les situations possibles dans l'évolution d'un RdP à partir d'un marquage initial. Le marquage initial noté M0 est la distribution initiale des jetons dans le réseau à l'instant initial. Un graphe de marquage est un graphe dont chaque sommet correspond à un marquage accessible et chaque arc correspond au franchissement d'une transition permettant de passer d'un marquage à l'autre [6] [8].

On note le marquage initial, M0, le marquage à l'instant initial (t=0). L'ensemble des marquages accessibles, A (R ; M0), pour le RdP ci-dessous est : A (R ; M0) = {M0, M1, M2, M3, M4}

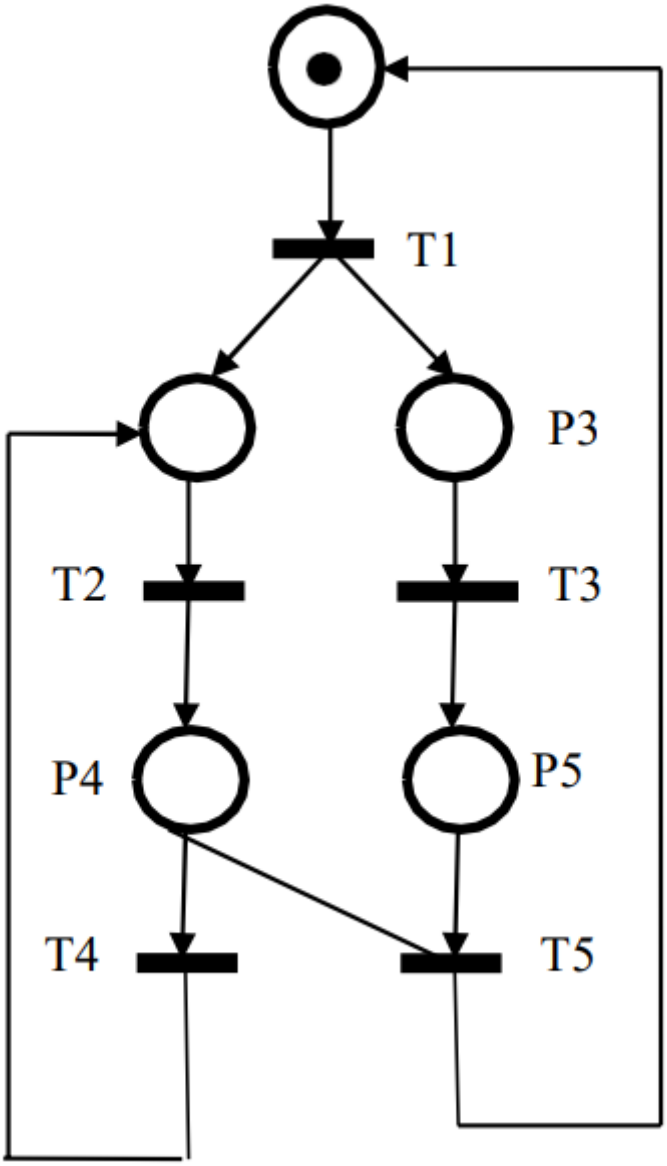


Figure 2.1 : Un réseau de pétri

L'évolution du RdP est représentée ci-dessous avec les marquages représentés sous la forme de vecteurs colonnes.

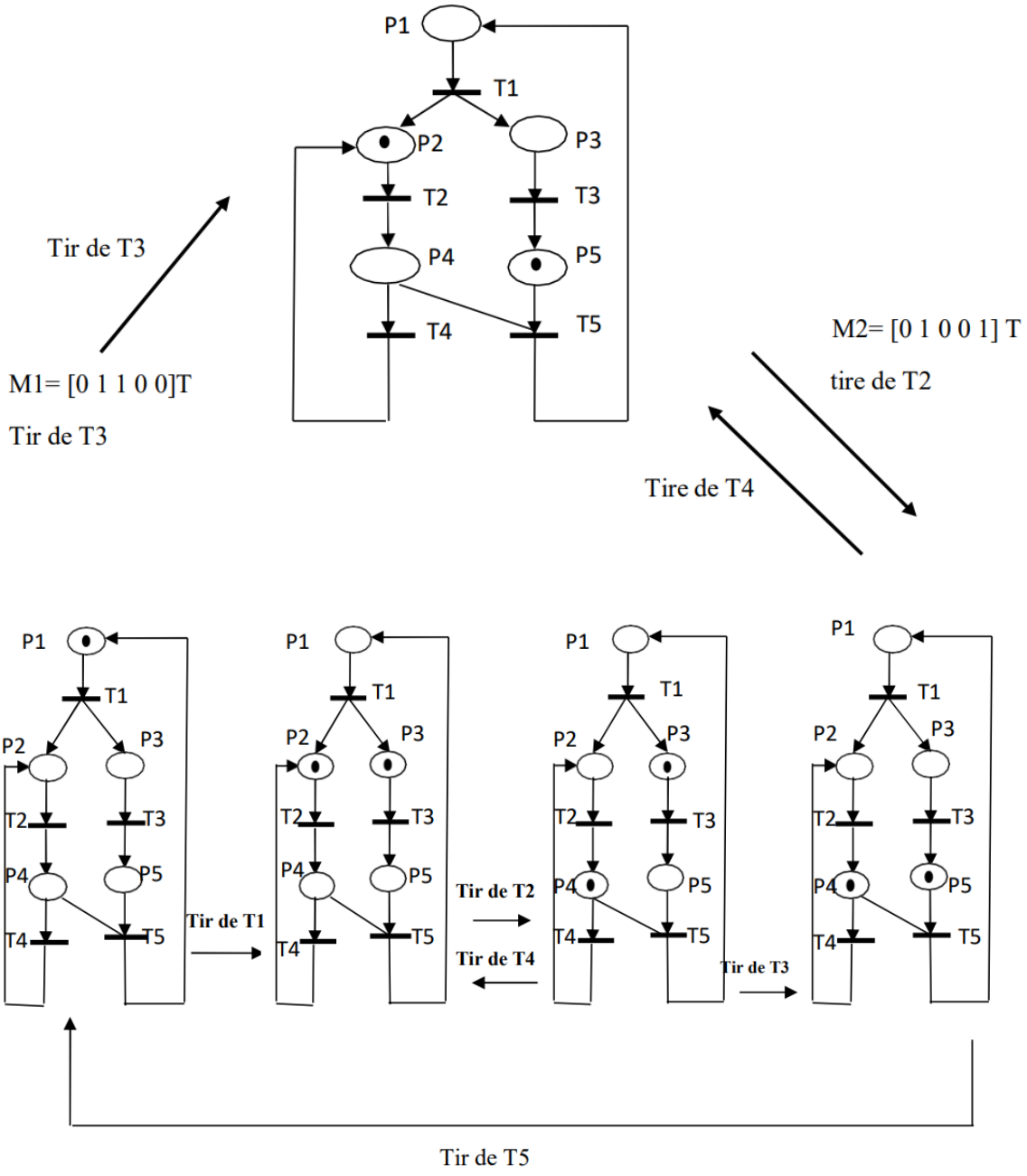


Figure 2.2 : Ordre de tir des transitions

L'évolution du RdP peut être représentée sous la forme d'un graphe des marquages GA (R ; M0), dont les sommets correspondent aux marquages accessibles.

Enfin, il est possible de représenter un graphe de marquage sous forme d'un « organigramme ». Ainsi pour l'exemple ci-dessous, on aura :

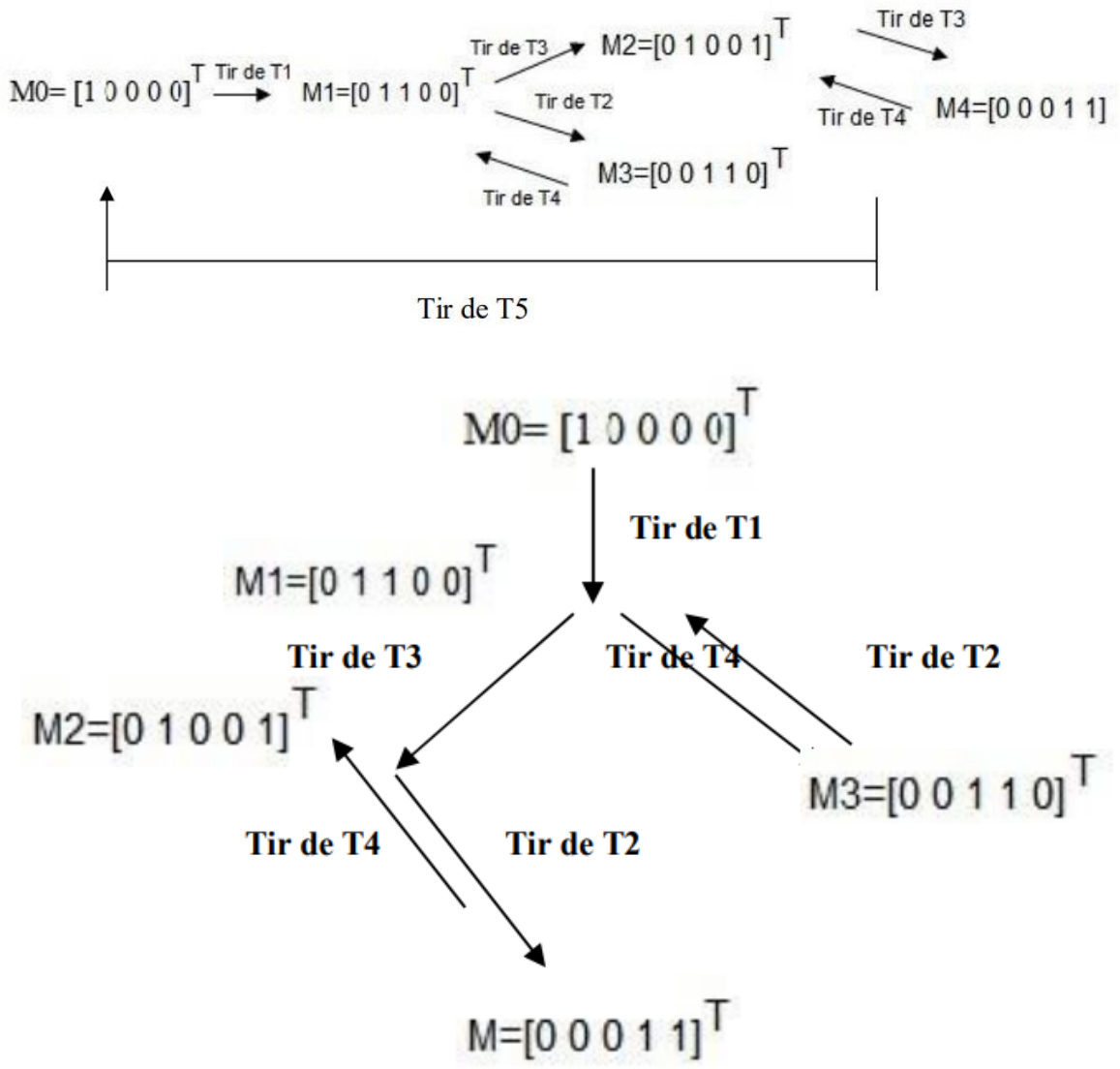


Figure 2.3 : Graphe de marquage

Le marquage dit initial décrit l'état initial du système. Ainsi, l'ensemble des marquages accessibles à partir du marquage initial par franchissement d'une séquence de transition correspond à l'ensemble des états du système. Contrairement aux graphes d'état, une place d'un RdP ne correspond pas à un état du système mais participe, à travers son marquage, à la description d'un ou de plusieurs états du système. L'ensemble des marquages accessibles est équivalent au graphe d'état représentant le comportement du système [5].

Remarque : [5]

- le graphe de marquages est utilisé quand le nombre de marquages accessibles est limiter.

• La représentation graphique d'un graphe de marquage permet de déterminer certaines propriétés du graphe. Par exemple si le graphe présente une zone non bouclée, cette partie du marquage une fois atteinte constitue un arrêt de l'évolution du RdP et celui-ci sera déclaré avec blocage.

3. La réversibilité dans les Réseaux de Pétri

La réversibilité étant la capacité de revenir en arrière [Larousse]. La réversibilité dans les réseaux de pétri est la propriété des transitions de pouvoir être franchie en avant et en arrière autrement dit c'est le fait d'annuler les transitions déjà franchie. Une façon assez naturelle d'implémenter l'annulation des transitions exécutées est d'introduire des inversions de celles-ci, comme le montre la figure 2.4 [9].

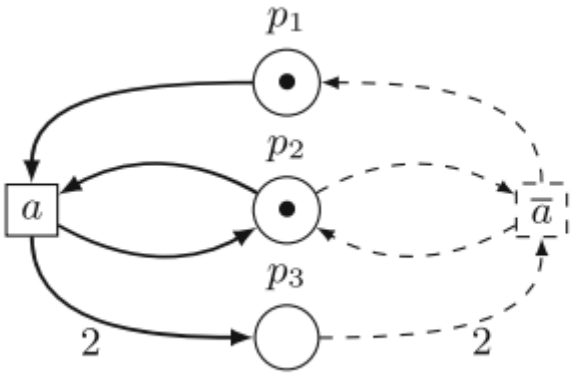


Figure 2.4 : Une transition a et son inverse

Sur la figure 2.5, les lignes pleines représentent un réseau de pétri avec son graphique d'accessibilité. De plus, en utilisant les lignes en pointillés, le diagramme montre la transition inverse ajoutée au réseau d'origine et l'élargissement résultant du graphe d'accessibilité d'origine. Nous observons que le réseau de pétri d'origine n'était pas réversible, mais celui modifié est réversible et son ensemble de marquages atteignables est le même que pour le réseau de pétri d'origine. Ainsi, dans ce cas, l'inversion des transitions a amélioré le

comportement global [9].

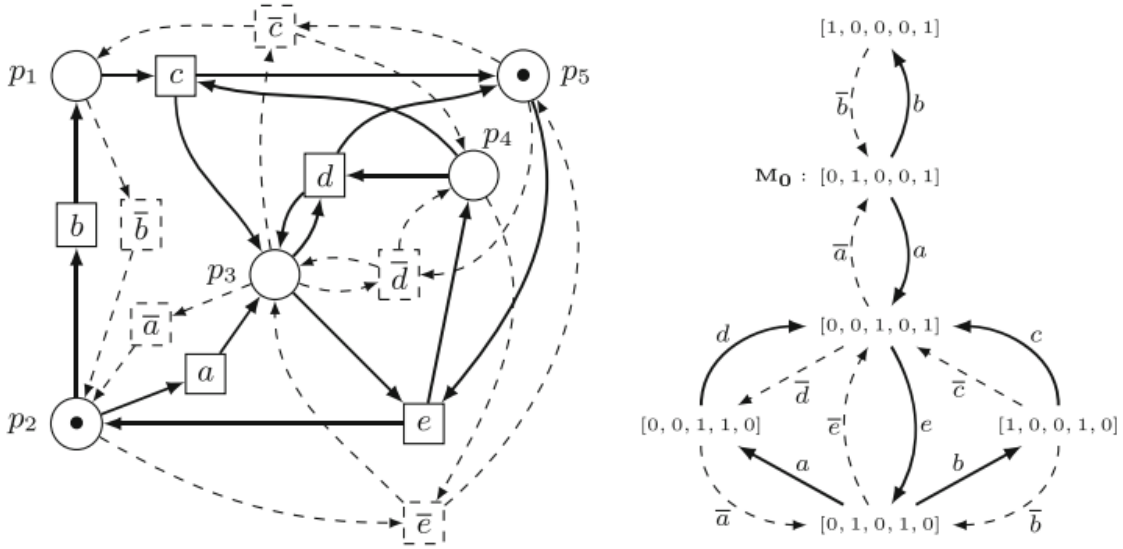


Figure 2.5 : un réseau de pétri avec inverse pour toutes les transitions et son graphe d'accessibilité.

Les transitions inverser permettent de rendre le réseau de pétri réversible.

La figure 2.6 montre un réseau de pétri qui a un état d'origine [0, 0, 1, 1]. Dans ce cas, il suffit d'ajouter un inverse b de la transition b pour obtenir un réseau de pétri réversible. De plus, l'ensemble des marquages accessibles reste inchangé [9].

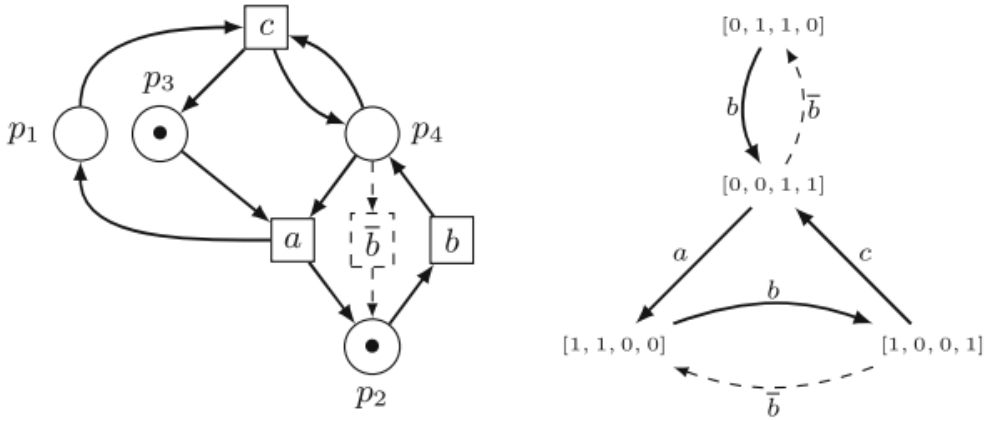


Figure 2.6 : un réseau de pétri avec une seule transition inverse et son graphe d'accessibilité

Les deux premiers exemples ont démontré que l'ajout de transitions inverses peut parfois améliorer le comportement du réseau d'origine. En général, l'ajout d'une transition inverse modifie l'ensemble d'accessibilité et permet également des calculs basés sur les transitions

d'origine qui n'étaient pas activées dans le réseau d'origine. Cela peut arriver même si nous nous limitons à inverser une seule transition [9].

La figure 2.7 montre un réseau de pétri avec un ensemble fini de marquages atteignables pour lesquels l'ajout d'un seul inverse change l'ensemble d'accessibilité en un ensemble infini. Notez que l'exécution de la transition inverse est activée avant la première exécution. En conséquence, ce réseau de pétri modéliserait un système dans lequel une action peut être annulée avant qu'elle ne soit effectuée, ce qui est contraire à notre intuition derrière l'inversion d'un calcul [9].

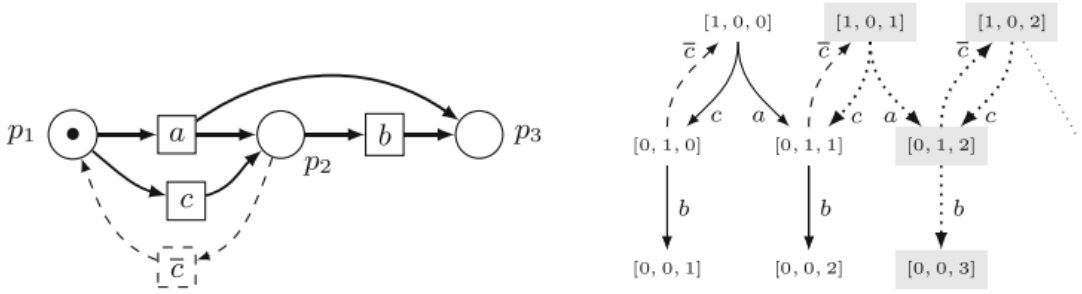


Figure 2.7 : un réseau de pétri avec une seule transition inverse et son graphe d'accessibilité

Commencer avec un réseau possédant un état d'origine n'aide pas non plus, comme le montre la figure 2.8. Le réseau a un état d'origine, mais encore une fois, il suffit d'ajouter la transition inverse de a pour obtenir un réseau avec un plus grand nombre de marquages atteignables [9].

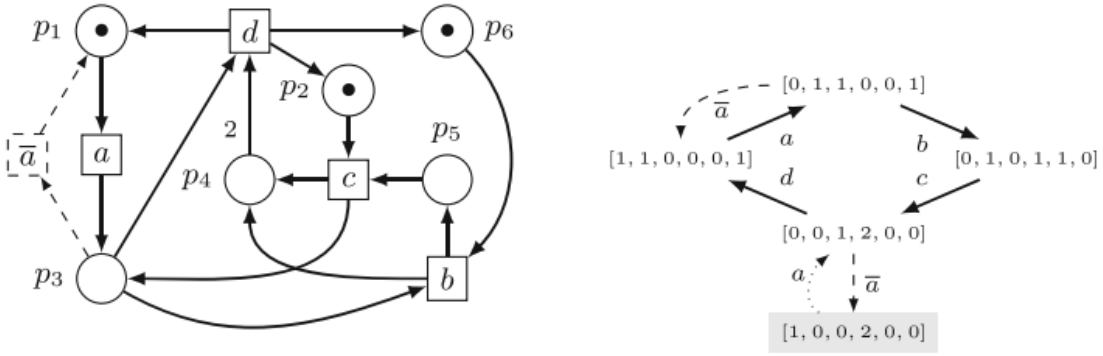


Figure 2.8 : un réseau de pétri réversible avec une seule transition inverse et son graphe d'accessibilité

Les exemples ci-dessus suggèrent qu'il n'est pas évident de pouvoir ajouter des transitions réversibles à un réseau de pétri sans changer radicalement son comportement. Nous pourrions également voir que l'ajout d'une seule de ces transitions peut entraîner de grands changements

dans le comportement du réseau. Ainsi, il est crucial de pouvoir décider si un inverse particulier peut être ajouté à un réseau de pétri sans trop changer son ensemble d'accessibilité [9].

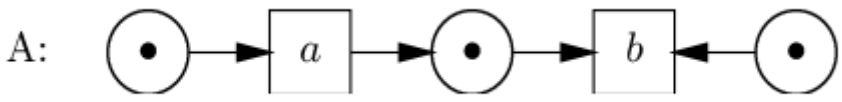
4. Sémantique de Causalité

L'un des aspects fondamentaux de l'analyse des réseaux de pétri est l'étude de la causalité, c'est-à-dire les relations de cause à effet entre les transitions. La causalité permet de comprendre comment l'activation d'une transition peut influencer ou empêcher l'activation d'autres transitions dans le réseau de pétri. Cependant, pour saisir la causalité dans les réseaux de pétri réversible, il est essentiel d'examiner les interprétations individuelles et collectives des jetons. Les jetons, représentant les unités d'information ou les ressources, peuvent être interprétés de différents manières en fonction du contexte du système. Les interprétations individuelles se réfèrent aux signification local des jetons, spécifique à chaque place, tandis que les interprétations collectives décrivent les relations entre les places et les transitions.

4.1.Les interprétations individuelles et collectives des jetons

Dans l'interprétation des jetons individuels des réseaux de Pétri, on distingue différents jetons résidant au même endroit, en gardant une trace de leur provenance. Si une transition se déclenche en utilisant un jeton qui a été produit par une autre transition, il existe un lien de causalité entre les deux. Par conséquent, les relations causales entre les transitions dans un réseau de pétri peuvent toujours être décrites au moyen d'un ordre partiel. Dans l'interprétation collective des jetons, en revanche, les jetons ne peuvent pas être distingués : s'il y a deux jetons dans un lieu, tout ce qui y est présent est le nombre 2. Cela donne lieu à des relations causales plus subtiles entre les transitions dans un réseau de pétri, qui ne peut être exprimé par des commandes partielles [7].

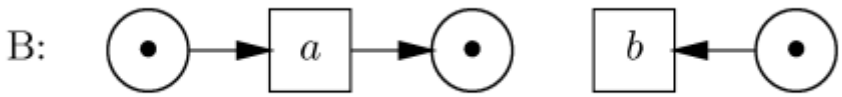
L'exemple suivant illustre la différence entre les deux interprétations.



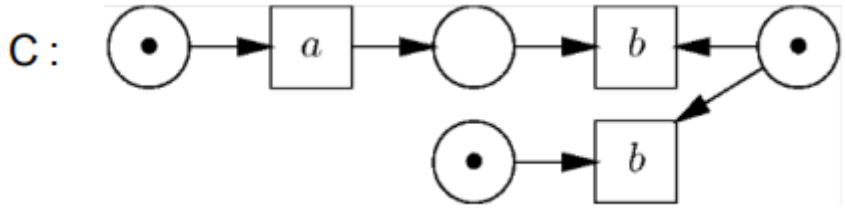
Dans ce réseau, les transitions étiquetées a et b peuvent se déclencher une fois chacune. Après que a a tiré, il y a deux jetons à la place du milieu. Selon la philosophie des jetons individuels, cela fait une différence lequel de ces jetons est utilisé pour tirer b. Si le jeton qui s'y trouvait déjà est utilisé (ce qui doit certainement être le cas si b se produit avant que le jeton de a n'arrive), les transitions a et b sont causalement indépendantes. Si le jeton qui a été produit par

a est utilisé, b dépend causalement de a. Ainsi, le réseau A ci-dessus a deux exécutions maximales, qui peuvent être caractérisées par les ordres partiels $\begin{matrix} a \\ b \end{matrix}$ et a→b. D'autre part, selon la philosophie collective des jetons, tout ce qui est présent à la place du milieu après l'apparition de a est le nombre 2. Les conditions préalables au déclenchement de b ne changent pas et, par conséquent, b est toujours causalement indépendant de a [7].

Ce qui suit illustre que les deux philosophies produisent des notions incomparables d'équivalence.



Dans la philosophie du jeton collectif, la précondition de b exprimée par la place au milieu est redondante, et donc A doit être équivalent à B. Cependant A et B ne sont pas entièrement équivalents de bissimulation simultanée (une causalité respecte l'équivalence basée sur l'approche du jeton individuel), car B n'a pas l'exécution a→b. D'autre part, A est l'équivalent de la bisimulation entièrement concurrente de C ci-dessous [7].



En fait, C est le réseau d'occurrences obtenu à partir de A. Dans la philosophie des jetons individuels, A et C ont les exécutions $\begin{matrix} a \\ b \end{matrix}$ et a→b. Cependant, dans la philosophie du jeton collectif A n'a pas de course a → b et ne peut donc pas être équivalente à C d'une manière qui préserve la causalité [7].

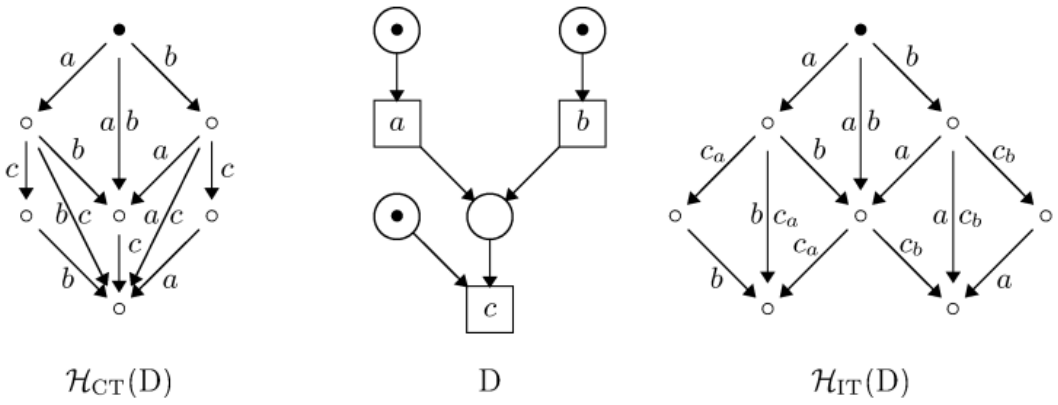


Figure 2.9 : interprétation individuelles et collective des jetons

Le réseau de Pétri D ci-dessus (ignorez HCT (D) et HIT (D) pour l'instant) illustre comment l'interprétation collective des jetons donne lieu à des relations causales qui ne peuvent pas être exprimées par des ordres partiels. Selon l'interprétation symbolique collective, ce réseau présente une causalité disjonctive : c est causalement dépendant de a ∨ b. En revanche, sous l'interprétation du jeton individuel, D admet deux exécutions, l'une dans laquelle c ne dépend que de a, et l'autre dans laquelle c ne dépend que de b [7].

Antoni Mazurkiewicz a plaidé une fois pour la lecture collective de ce réseau en laissant a et b être des contributions de 1 £ de deux écoliers pour acheter un cadeau pour leur professeur. L'acte d'acheter le cadeau, qui ne coûte que 1 £, est représenté par c. Or l'interprétation symbolique individuelle suggère que le cadeau est acheté à partir de la contribution soit de l'un soit de l'autre, alors que l'interprétation symbolique collective n'admet qu'une seule exécution complète, dans laquelle l'achat du cadeau est causé par la disjonction des deux contributions. Ce dernier serait une description plus juste de l'état des choses prévu [7].

5. Réseaux de Pétri Réversibles

La réversibilité causale signifie qu'un événement qui provoque d'autres événements ne peut être annulé qu'après que les événements provoqués ont été annulés en premier. Notre travail consiste à modéliser ce type de réversibilité au sein des réseaux de pétri. Nous allons définir les différents types d'exécution dans les réseaux de pétri réversible [1][2]

5.1.Exécution en avant (franchissement de la transition en avant)

Dans un réseau de pétri pour qu'une transition t soit franchissable ou active, il faut que le nombre de jetons dans les places pi entrant de t soit supérieur au poids des arcs liant les pi à t. L'exécution en avant des transitions peut créer de nouveau lien entre jetons des places entrant.

Si une liaison préexistante apparait dans un arc sortant d'une transition alors elle est également une condition préalable à la transition. Si une transition bifurque entre les places p1 et p2 alors les jetons transférer vers ses places ne sont pas connecter entre eux dans les places entrant de la transition. Les liaisons qui sont créées par la transition sont exactement celles qui apparaissent dans la postcondition d'une transition mais pas dans sa précondition. Ceci permettra par la suite d'énoncer l'inversion de la transition par la destruction des liaisons [1].

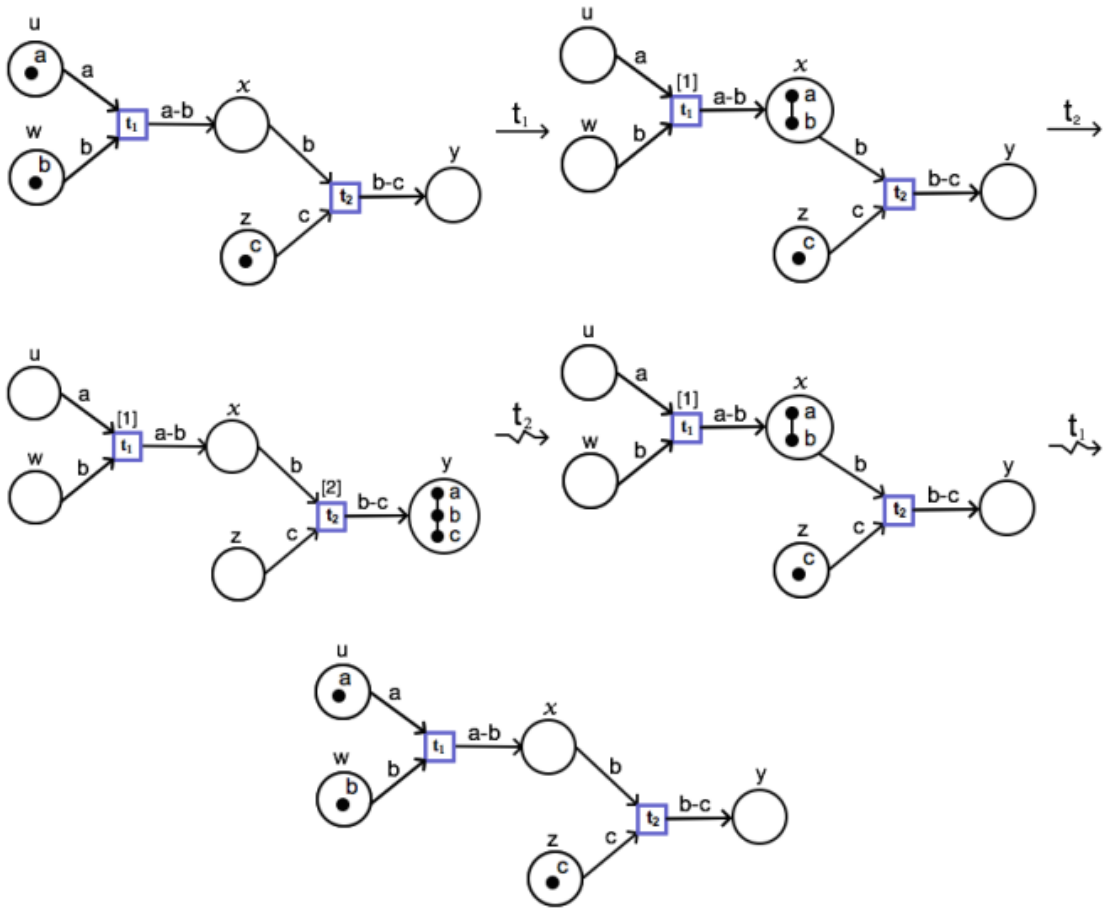


Figure 2.10 : Exécution en avant et en arrière d'un réseau de pétri

Un exemple de transition vers l'avant peut être vu dans les trois premières étapes de la figure ci-dessus.

Dans la figure ci-dessus les liaisons sont indiquées après l'exécution des transitions sur les arcs sortant par exemple après exécution de la transition t2 nous avons la liaison a-b-c. L'exécution en avant ne peut créer que de nouveau lien.

5.2.Bactracking (le retour en arrière)

Passons maintenant à la forme la plus simple de réversibilité, à savoir le retour en arrière. Ainsi une transition est bt-active (active pour le retour en arrière) si elle est la dernière transition à

avoir été exécutée. Ainsi, lorsqu'une transition est inversée en retour arrière, tous les jetons et toutes les liaisons de la postcondition de la transition seront transférés à la place entrante de la transition et toutes les liaisons nouvellement créées seront rompue. Dans les deux dernières étapes de la figure ci-dessus, nous observons que les transitions t1 et t2 sont inversées. Les jetons sont préservés tout au long de l'exécution d'un réseau de pétri réversible, les liens peuvent être créées pendant l'exécution en avant et détruit pendant l'exécution en arrière [1].

5.3.Causal reversing (inversion de cause)

Nous allons maintenant considérer la causalité entre les transitions dans un réseau de pétri et la réversibilité dans un ordre respectant la causalité. L'inversion d'une transition dans un ordre respectant la causalité est mise en œuvre exactement de la manière que le retour en arrière, c'est-à-dire que le jetons sont déplacer des places de sorti aux places d'entrée de la transition, tous les liens créés par la transition sont rompus. A la différence que l'inversion causale permet la permutation de transition inverse qui n'ont pas de relation causale entre elles [1].

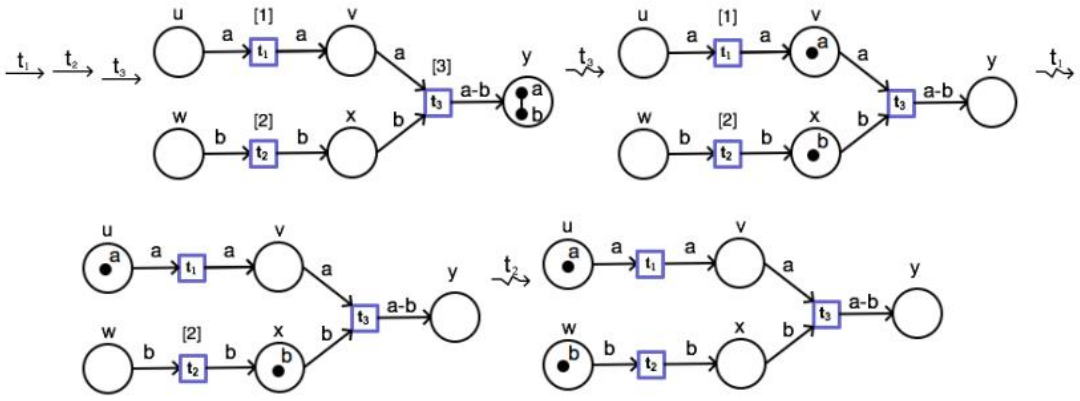


Figure 2.11 : Exemple d'ordre causale

Un exemple de réversibilité d'ordre causal est illustré à la figure ci-dessus. Ici, nous avons deux transitions indépendantes, t1 et t2, qui précèdent causalement la transition t3. En supposant que les transitions ont été exécutées dans l'ordre t1, t2, t3, l'exemple montre une inversion d'ordre causal ou t3 est inversée, suivi de l'inversion de ses deux causes t1 et t2. En général, elles peuvent être inversées dans n'importe quel ordre.

5.4.Réversibilité hors de l'ordre causal

Nous allons maintenant parler de l'inversion hors de l'ordre causale. L'exécution d'une transition dans un réseau hors de l'ordre causale est la même que celle de l'ordre causale. A la différence que l'inversion dans un ordre non causal est faite de manière non ordonnée. Ainsi si

une transition t est inversée de manière non ordonnée, toutes les liaisons qui ont été créées par la transition t sont annulées. La destruction d'une liaison divise un composant en composant connectés plus petit, alors chacun de ces composant doit être replacé à l'endroit où le composant aurait existé si la transition t n'avait jamais eu lieu, c'est-à-dire exactement après la dernière transition qui implique des jetons du sous-composant. Si un jeton a et son composant connecté ont participé en dernier lieu à une transition avec une place de sortie x autre que y, alors le sous-composant est retiré de la place y et renvoyé à la place x, sinon il est renvoyé à la place où il se trouvait dans le marquage initial [1].

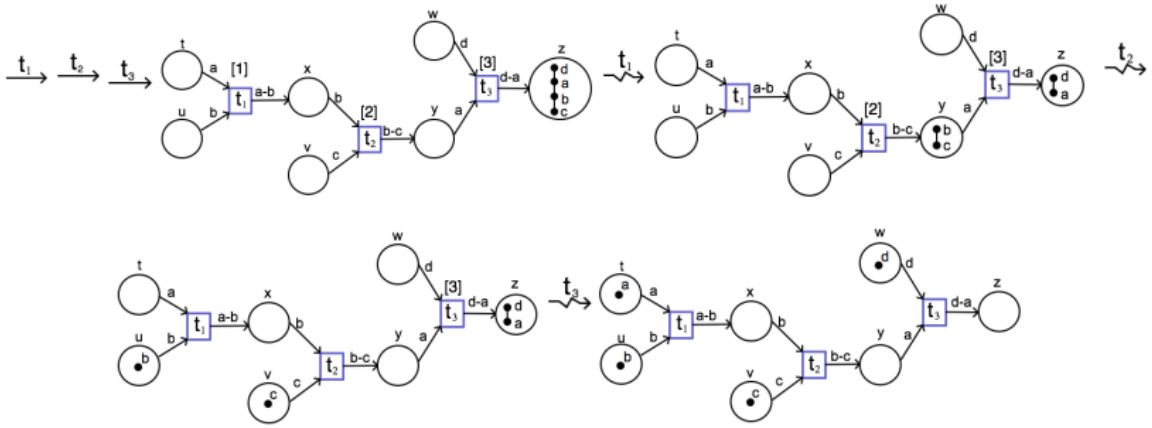


Figure 2.12 : Exemple d'ordre hors causalité

Dans le premier réseau représenté sur la figure ci-dessus, nous voyons que les transitions t1, t2 et t3 ont été exécutées dans cet ordre et que tous les jetons se trouvent maintenant à la place final z. Supposons que la transition t1 soit inversées hors de l'ordre. Comme nous l'avons déjà évoqué, l'effet de cette inversion devrait être la destruction du lien entre a et b. Cela signifie que le composant d-a-b-c est brisé en liaison d-a et b-c qui doit revenir en arrière dans le réseau de pétri pour capturer l'inversion de la transition. Néanmoins, les jetons de d-a doivent rester à la place z. Cela est dû au fait qu'il existe entre eux un lien qui n'a pas été inversé et qui était l'effet de la transition immédiatement précédente t3. Cependant, dans les cas de b et c le lien peut être inversée à la place y qui est la place ou les deux jetons sont connectés et d'où ils pourraient continuer à participer à tout autre calcul nécessitant leur coalition. Si la transition t2 est ensuite inversée, le lien entre b et c est détruit et les deux jetons peuvent donc retourner à leur place initiale, comme le montre le troisième filet de la figure ci-dessus. Enfin, si la transition t3 est inversée, le lien entre d et a se rompt et, étant donné qu'aucun des deux ne sont connecté à d'autres éléments, les jetons peuvent revenir à leur place initiale. A partir de cet exemple, nous observons que dans la réversibilité hors de l'ordre causal, une fois qu'une

transition est inversée, nous devons inverser toutes les liaisons qu'elle a créées et inverser les jetons de tous les éléments créés par la liaison rompue aussi loin en arrière que possible. Dans la réversibilité hors de l'ordre causale toutes transition exécutée peut être inverser à tout moment [1].

5.5.Réversibilité dans un réseau de pétri en utilisant l'interprétation individuelle de jeton

Soit N = (S, T, F, I, L) un réseau de pétri, soit u ∈ $T_\bullet$ tel que u = (X, t) le tir vers l'avant de t peut être représenté comme la destruction de tous les jetons de l'ensemble X et ensuite la production de jetons de l'ensemble $u^\bullet$. Donc, le retour en arrière de u consomme $u^\bullet$ et produit X. Dans cette partie, nous montrons comment cette vision intuitive de l'annulation peut assurer à la fois la cohérence d'un système et la réversibilité flexible.

La définition formelle du tir avant et arrière dans les réseaux de pétri est donnée par :
Pour un tir u ∈ $T_\bullet$ dans un réseau de pétri tel que u= (X, t), notons $^\bullet u$ = X l'ensemble des jetons d'entrée de u et $u^\bullet$ = {(u, k, s) / k < F (η(u), s)} l'ensemble des jetons de sortie de u.
Le tir en avant : la transition t est activé sous un marquage individuel M ∈ $S_\bullet$ si $^\bullet u \subseteq M$. Dans ce cas, t peut tirer sous M, ce qui donne M'= (M \ $^\bullet u$) ∪ $u^\bullet$ écrit comme M $\xrightarrow{u}_f$ M'ou M [u $>_f$ M'.
Le tir en arrière : la transition t peut être annuler sous un marquage individuelle M'∈ $S_\bullet$ si $u^\bullet \subseteq$ M' dans ce cas, l'annulation de t peut être déclencher M', ce qui donne M = (M'\ $u^\bullet$) ∪ $^\bullet u$, écrit comme M' $\xrightarrow{u}_f$ M ou M' [u $>_b$ M .

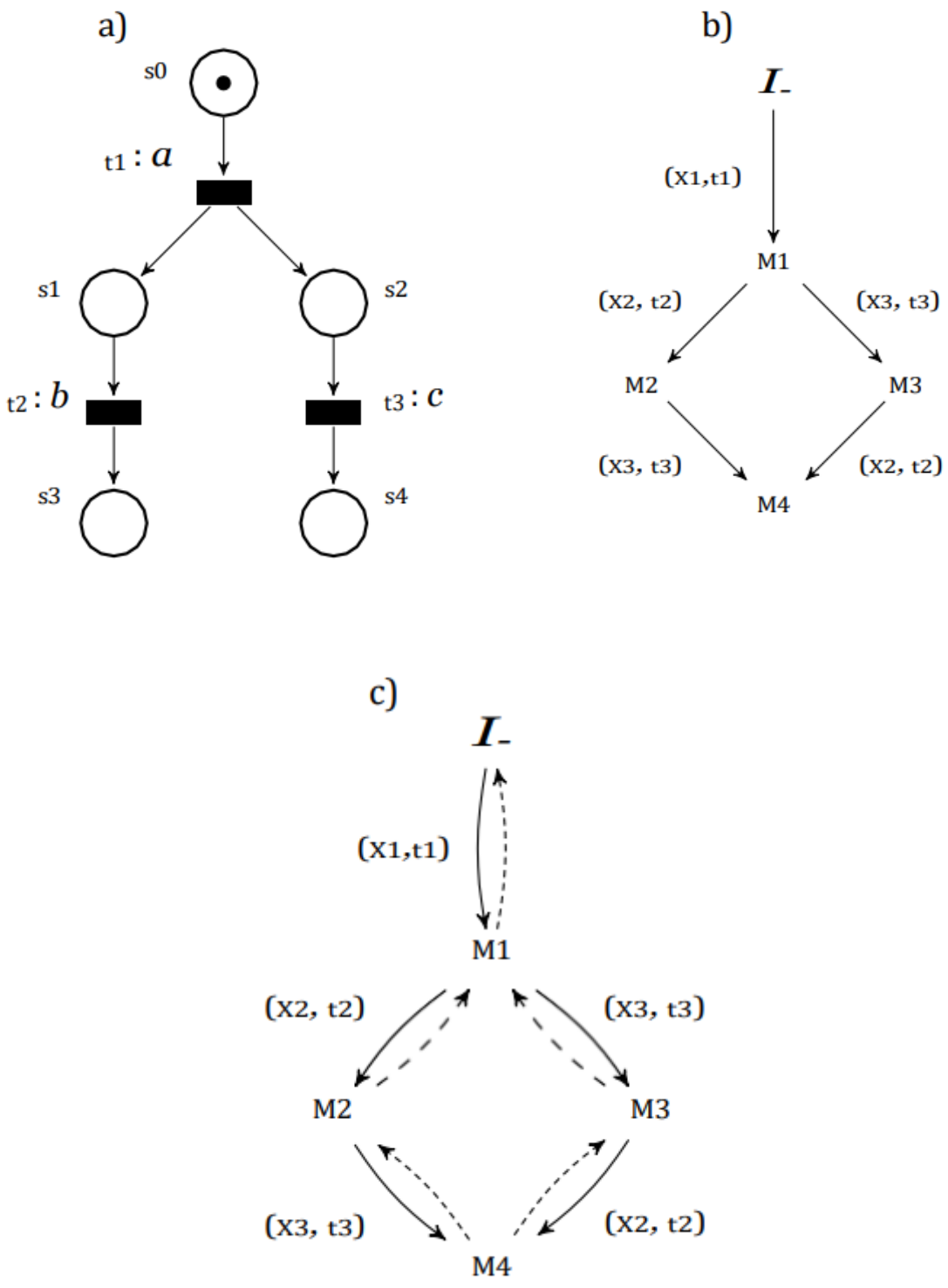


Figure 2.13 : Réversibilité dans un réseau de pétri d'interprétation de jeton individuel *a) réseau de pétri N ; b) graphe de marquage habituels ; c) graphe de marquage avec réversibilité causale*

Dans la figure ci-dessus a) N est le réseau de pétri ayant pour état de marquage initial $I_\bullet$ = {(*,0, s0)} nous permet. Marquer $I_\bullet$ nous permet de n'exécuter que (X1, t1) tel que X1 = {(*, 0, s0)}. Nous avons $I_\bullet$ $[(X1, t1)]_f$ M1 tel que M1 = {({(X1, t1)}, 0, s1), ({(X1, t1)}, 0, s2)}.

Dans l'état M1, nous avons trois possibles : exécuter (X2, t2), exécuter (X3, t3)

Ou annuler (X1, t1) :

M1 $[(X2, t2)]_f$ M2 tel que x2= {({(X1, t1)}, 0, s1)} et

M2 = {({(X1, t1)}, 0, s2), ({(X2, t2)}, 0, s3)} ;

M2 $[(X2, t3)]_f$ M3 tel que x3= {({(X1, t1)}, 0, s2)} et

M3 = {({(X1, t1)}, 0, s1), ({(X3, t3)}, 0, s4)} ;

M1 $[(X1, t1)]$ I_b .

Dans la figure ci-dessus, c) représente le graphe de marquage dans lequel un arc en pointillé signifie l'annulation d'une action, et le b) est le graphe de marquage habituel (sans réversibilité). Les deux graphes précédents ont exactement le même ensemble de marquage. Dans un réseau de pétri réversible, l'annulation d'un tir (X, t) génère un nouvel état de marquage. Il s'agit simplement de l'inverse de l'état passé de (X, t).

5.6.Inverser les réseaux de pétri avec des jetons multiples

Dans les réseaux de pétri nous avons un formalisme, qui présente des jetons individuels qui peuvent être connectés ensemble via des liens. Une hypothèse des réseaux de pétri est que les jetons sont distincts par paires. Pour assouplir cette restriction, la multiplicité de jetons dans laquelle un modèle peut contenir plusieurs jetons du même type a été introduit par les chercheurs. Il a été observé que la possibilité de déclencher une transition plusieurs fois en utilisant différents ensembles de jetons peut introduire un non-déterminisme, également connu sous le nom de conflit en arrière, lors du retour en arrière. De plus, deux approches ont été identifiées pour définir la sémantique réversible en présence de tels conflits en arrière, inspirées par les interprétations du jeton individuel et du jeton collectif, définies pour raisonner sur la causalité dans les réseaux de Pétri. Dans l'approche des jetons individuels, plusieurs jetons du même type résidant au même endroit sont distingués en fonction de leur chemin causal, alors que dans l'interprétation des jetons collectifs, ils ne sont pas distingués. Le modèle des réseaux de pétri à jetons multiples a été étudié dans le cadre de l'approche des jetons collectifs, produisant une forme de réversibilité hors de l'ordre causal. Dans ce travail, nous appliquons plutôt l'interprétation des jetons individuels pour définir une sémantique causale, et nous

établissons qu'en fait l'ajout de plusieurs jetons n'ajoute pas à l'expressivité du modèle, en ce que pour tout réseaux de pétri avec plusieurs jetons il existe un réseau de pétri équivalent avec un seul jeton de chaque type [10].

Pour apprécier les défis induits par l'introduction de plusieurs jetons et la différence entre les interprétations individuelles et collectives des jetons, considérons l'exemple de la Figure ci-dessous (a). Dans cet exemple, nous pouvons voir un modèle de réseaux de pétri d'un assemblage/démontage d'un stylo. Le produit est constitué de l'encre, du gobelet et du bouton du stylo, modélisés par des jetons c et b, respectivement. On peut observer que les transitions, en plus de transférer des jetons entre les places, ont la capacité de créer des liens. Ainsi, le processus de fabrication du stylo nécessite la mise en place de l'encre à l'intérieur du gobelet, modélisée par la création du lien i-c par la transition t1, et par la suite, la mise en place du bouton sur le gobelet pour compléter [10].

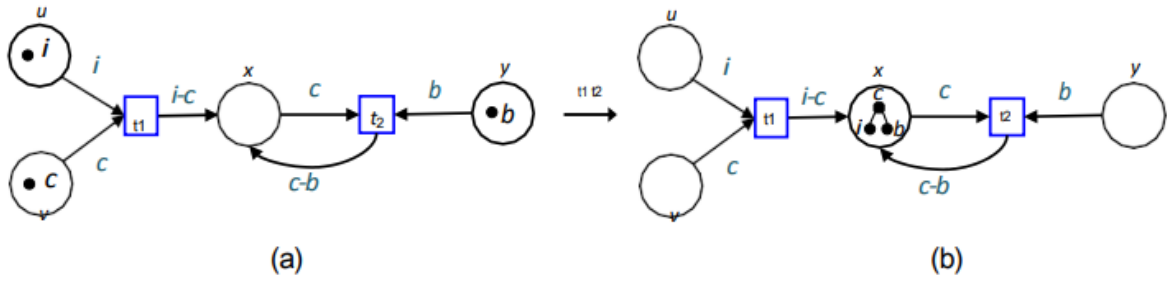


Figure 2.14 : Assemblage/démontage d'un stylo par réseau de pétri

L'assemblage, modélisé comme la création de la liaison c-b par la transition t2. L'inversion d'une transition dans les réseaux de pétri a pour effet de rompre les liens créés par la transition et de renvoyer les jetons/liens des places sortants vers les places entrants de la transition. Des mécanismes ont été développées afin de modéliser la réversibilité de retour en arrière, causale et hors causale pour le modèle. En particulier, dans l'exemple de la figure ci-dessus (b), l'inversion de la transition t2 entraînera la destruction de la liaison c-b et le retour du jeton b à la place y.

Supposons que nous souhaitons étendre le modèle de la figure ci-dessus (a) pour l'assemblage de deux stylos. Étant donné que dans les réseaux de pétri, les jetons sont uniques, il serait nécessaire d'introduire trois nouveaux jetons distincts et de cloner les transitions tout en renommant leurs zones pour tenir compte des noms des nouveaux jetons à utiliser, ce qui entraînerait une expansion considérable du modèle pour chaque nouveau stylo à produire. Ainsi, une extension naturelle du formalisme consiste à assouplir cette restriction et à permettre

à plusieurs jetons du même type d'exister au sein d'un modèle. A cet effet, considérons le scénario de la figure ci-dessous (a) présentant un système avec un stylo déjà assemblé/échantillon en place x et deux éléments de chacun des composants encre, gobelet et bouton.

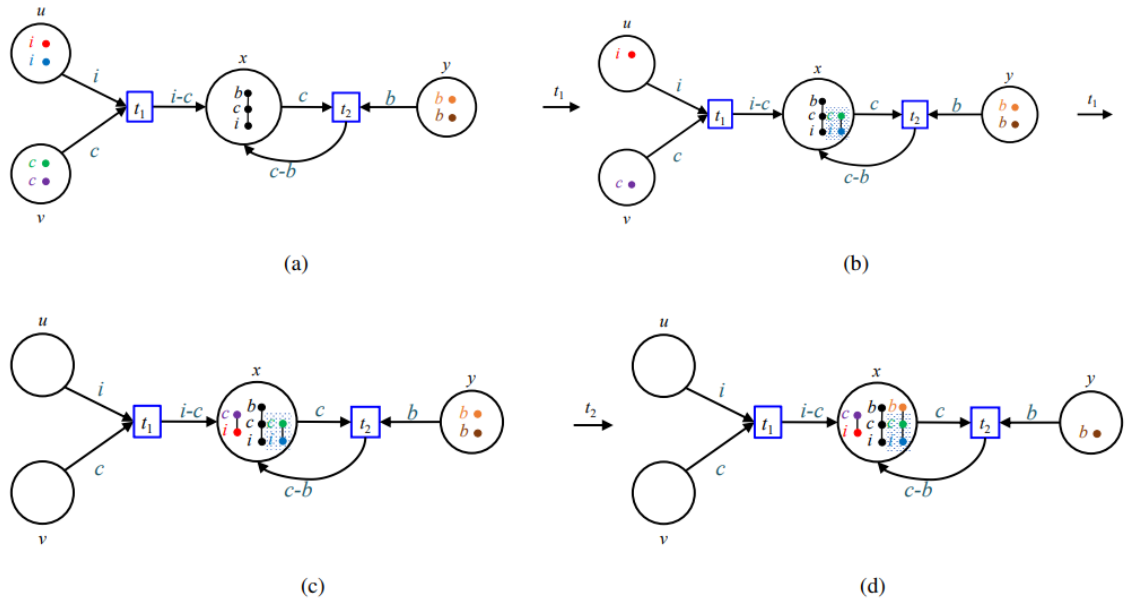


Figure 2.15 : Assemblage/démontage des stylo par réseau de pétri

Un problème qui se pose dans ce nouveau paramètre est celui dû à la présence de plusieurs jetons du même type, le phénomène de non-déterminisme vers l'arrière se produit lorsque les transitions sont inversées. Par exemple, après l'exécution de la transition t1 deux fois et t2, deux stylos assemblés existeront à la place x et ainsi qu'un composant i-c, comme on le voit sur la figure ci-dessus (d). Supposons que dans cet état la transition t1 soit inversée. Dans l'interprétation collectif des jetons, toutes les instances de la liaison i-c sont considérées comme identiques. En conséquence, n'importe laquelle de ces liaisons pourrait être détruite lors de l'inversion de la transition t1. Cependant, dans l'interprétation des jetons individuels, les divers jetons d'encre et de gobelet sont distingués en fonction de leur chemin causal. Par conséquent, la première exécution de la transition t1 donne le réseau de la figure ci-dessus (b). Compte tenu de ces relations causales entre transitions, sous une sémantique de réversibilité causale, la composante i-c spécifique ne doit pas être décomposée tant que la transition t2 n'est pas inversée. De même, le stylo préexistant ne doit pas être décomposé en ses parties car il n'a été créé par aucune des transitions. Au lieu de cela, l'inversion de la transition dans le réseau de

pétri de la figure ci-dessus (d) devrait rompre la liaison dans le composant constitué de la liaison simple i-c. A noter que ceci est compatible avec la compréhension que le démontage du produit ne permettrait pas la séparation de l'encre de l'intérieur du gobelet avant le retrait du bouton, puisque cela est inclus dans le couple du gobelet et du bouton [10].

En conséquence, nous observons que suite à l'interprétation individuelle des jetons, l'inversion d'un calcul nécessite de garder une trace du comportement passé dans le contexte de l'exemple, en distinguant les jetons impliquant le stylo préexistant et les jetons utilisés pour déclencher chaque transition [10].

Conclusion

Ce chapitre propose une approche réversible des réseaux de pétri en se basant sur les trois formes de réversibilité qui sont le retour en arrière, l'inversion causale et l'inversion hors de l'ordre causale. Dans le chapitre suivant nous allons faire la conception et l'implémentation d'un éditeur graphique de réseau de pétri.

Chapitre 03 : Conception et implémentation

Dans cette partie, nous introduisons la phase de conception de notre projet visant à développer une application permettant de dessiner et de simuler un réseau de Pétri réversible avec la notion de jeton individuel. Cette étape est cruciale pour concevoir une architecture logicielle solide et modulaire, en mettant l'accent sur la convivialité, la réversibilité du réseau de Pétri et la manipulation des jetons individuels. Ensuite nous allons faire l'implémentation.

1. Conception

1.1.Objectifs de la partie conception :

L'objectif principal de cette partie est de définir une conception détaillée de l'application qui répond aux besoins fonctionnels identifiés. Nous visons à concevoir une architecture logicielle bien structurée qui permettra une gestion efficace des réseaux de Pétri réversibles en se basant sur les jetons individuels. Nous cherchons également à fournir une interface utilisateur intuitive et conviviale pour faciliter le dessin et la simulation des réseaux de Pétri.

L'architecture de notre application se compose de plusieurs composants clés.

Tout d'abord, nous avons le module de dessin qui permettra aux utilisateurs de créer et de manipuler graphiquement les éléments du réseau de Pétri.

Ensuite, nous avons le module de simulation réversible qui exécutera les comportements du réseau en respectant les principes de réversibilité.

Enfin, nous avons le module de jetons individuels, qui sera responsable de la création, du déplacement, de l'analyse et de la suppression des jetons individuels dans le réseau.

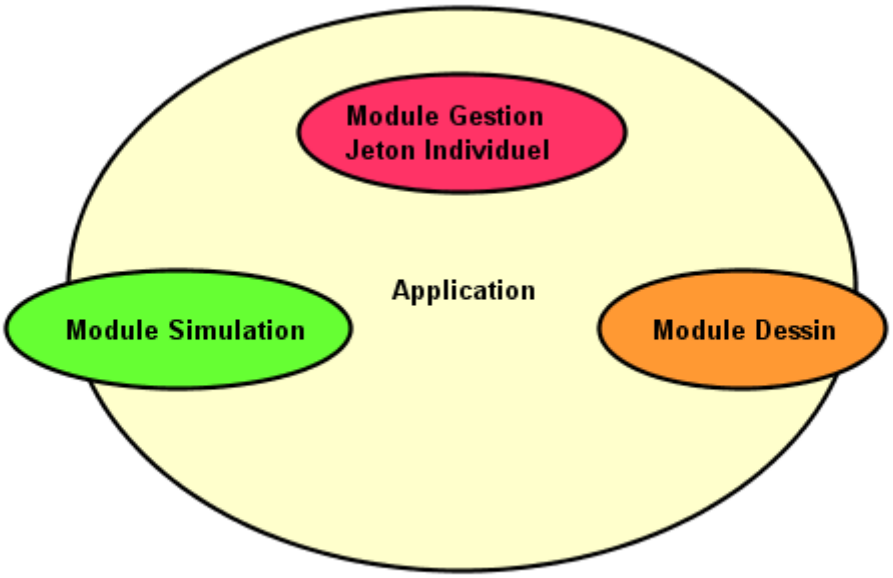


Figure 3.1 : Les modules de notre application

1.2.Méthodologie de conception utilisée :

Pour concevoir notre application, nous adoptons une approche orientée objet basée sur le langage de modélisation UML (Unified Modeling Language). L'utilisation d'UML nous permettra de visualiser et de spécifier les différentes entités, leurs relations et leurs interactions dans l'application. Cette méthodologie de conception nous offre une représentation claire et structurée de notre architecture logicielle, favorisant ainsi une compréhension approfondie et une mise en œuvre cohérente de notre application.

1.3.Exigence fonctionnelle de l'application :

Les exigences fonctionnelles du système définissent les fonctionnalités et les actions spécifiques que l'application doit offrir pour répondre aux besoins des utilisateurs. Dans notre cas, nous avons défini :

- Fonctionnalité de dessin des réseaux de Pétri :
  - Permettre à l'utilisateur de créer et de modifier graphiquement des places, des transitions et des arcs pour construire le réseau de Pétri.
  - Prise en charge de différentes formes et styles visuels pour les éléments du réseau.
- Fonctionnalité de manipulation des éléments du réseau :
  - Permettre à l'utilisateur de déplacer, et supprimer les éléments du réseau (places, transitions, arcs).
  - Prise en charge de la connexion et de la déconnexion des arcs entre les places et les transitions.
  - Possibilité de définir des propriétés spécifiques pour chaque élément du réseau (marquage initial).
- Fonctionnalité de simulation réversible :
  - Exécuter la simulation du réseau de Pétri en respectant les règles de réversibilité.
  - Gérer l'évolution des jetons individuels dans le réseau lors de la simulation.
  - Permettre à l'utilisateur d'exécuter, de mettre en pause, de reprendre et de réinitialiser la simulation.
  - Afficher visuellement l'état du réseau pendant la simulation, y compris les changements de marquage et les mouvements de jetons.
- Fonctionnalité de gestion de jeton individuel :
  - Mise en place d'un mécanisme de distinction de chaque jeton (identificateur).
  - Identifier pour chaque jeton sa place mère, sa position.
  - Retracer le parcours de chaque jeton à chaque étape, les transitions franchis.
- Fonctionnalités supplémentaires :
  - Possibilité de sauvegarder et de charger les réseaux de Pétri dans différents formats de fichier.
  - Analyse des propriétés du réseau, telles que la détection de blocages ou la vérification de propriétés spécifiques.

1.4.Flux d’utilisation de l’application

Notre application propose un flux d'utilisation organisé en plusieurs étapes distinctes.

Voici une description du déroulement de ce flux :

1. L'utilisateur a la possibilité de créer un nouveau fichier rdp ou de le charger depuis un fichier sauvegardé
2. Lors de la création d’un nouveau fichier rdp, l’utilisateur peut dessiner le rdp à l’aide des differents outils proposés dans l’application
3. Lors du chargement d’un fichier d’un rdp, l’utilisateur a la possibilité de modifier le rdp
4. L’étape suivante dans le cas d’une création d’un nouveau rdp ou d’un chargement, on peut simuler l’exécution d’un rdp
5. Après la simulation, il est possible d’analyser le réseau avec le graphe de marquage, parcours jeton, …
6. Il est aussi possible de visualiser la présence d’un blocage ou pas dans un rdp
7. Lors de l’analyse en cliquant sur un jeton, on peut visualiser son identificateur, la transition franchis, sa place mère.
8. Après toutes les opérations effectuées sur le rdp, il est possible de sauvegarder dans un fichier afin de pouvoir le réutiliser plus-tard.

Toutes les étapes décrites ci-dessus sont résumés dans la figure suivante.

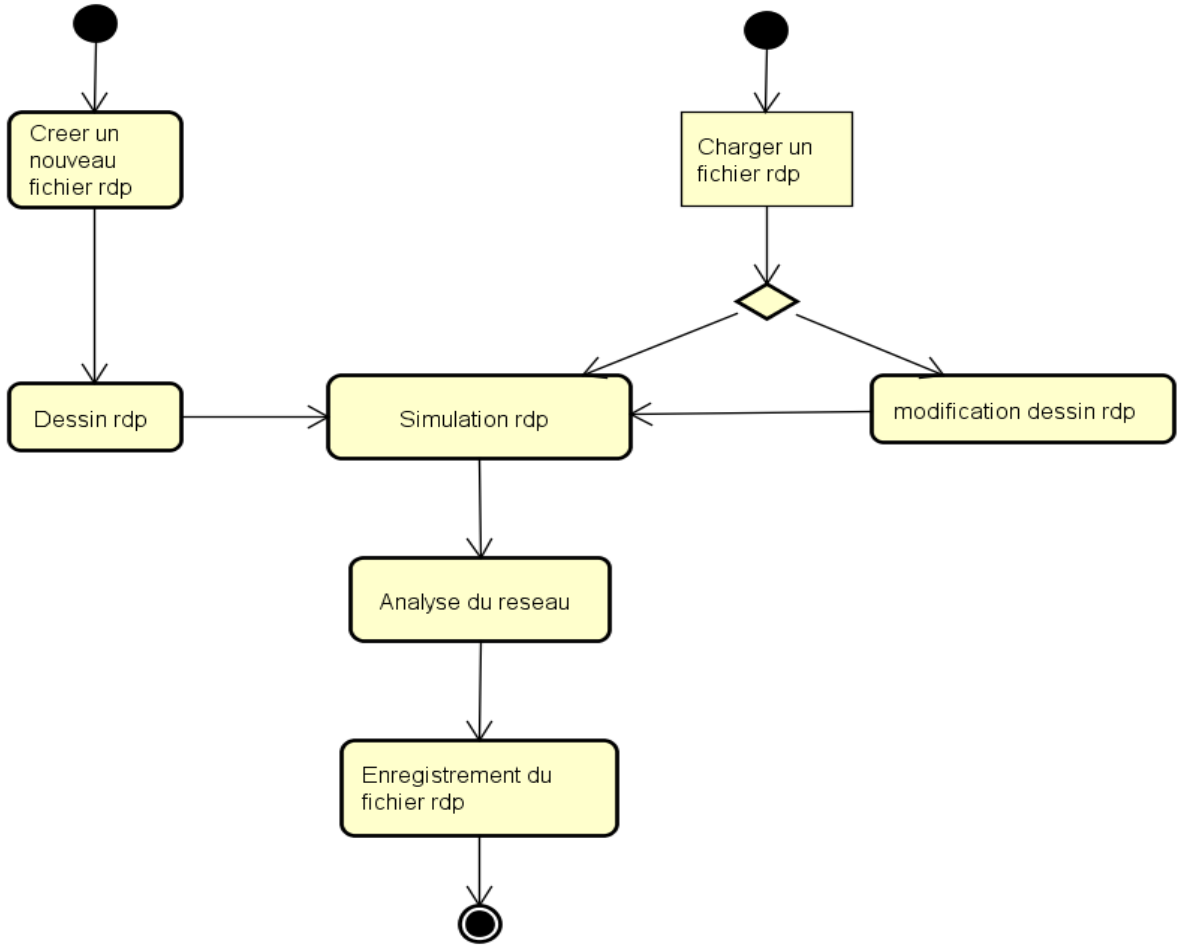


Figure 3.2 : Flux d'utilisation de l'application

1.5.Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation permet de décrire les besoins des utilisateurs et de capturer les exigences fonctionnelles d'un système.

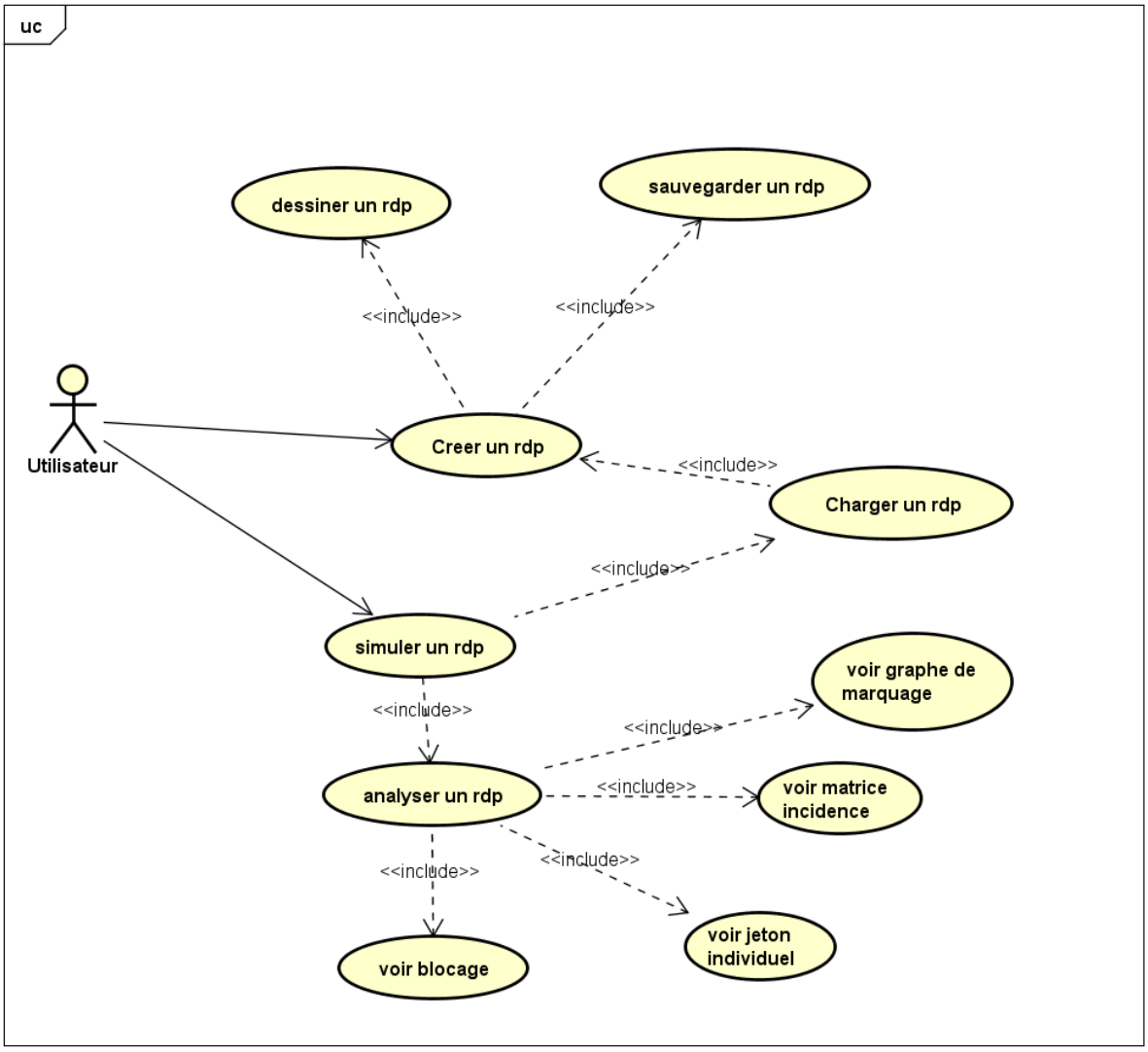


Figure 3.3 : Diagramme de cas d'utilisation

1.6.Diagramme de classe

Le diagramme de classe permet de décrire la structure de classe d'un système ainsi que les interactions entre les différentes entités.

Dans cette section, nous allons définir les différentes classes qui composent le système, ainsi que leurs attributs et leurs méthodes.

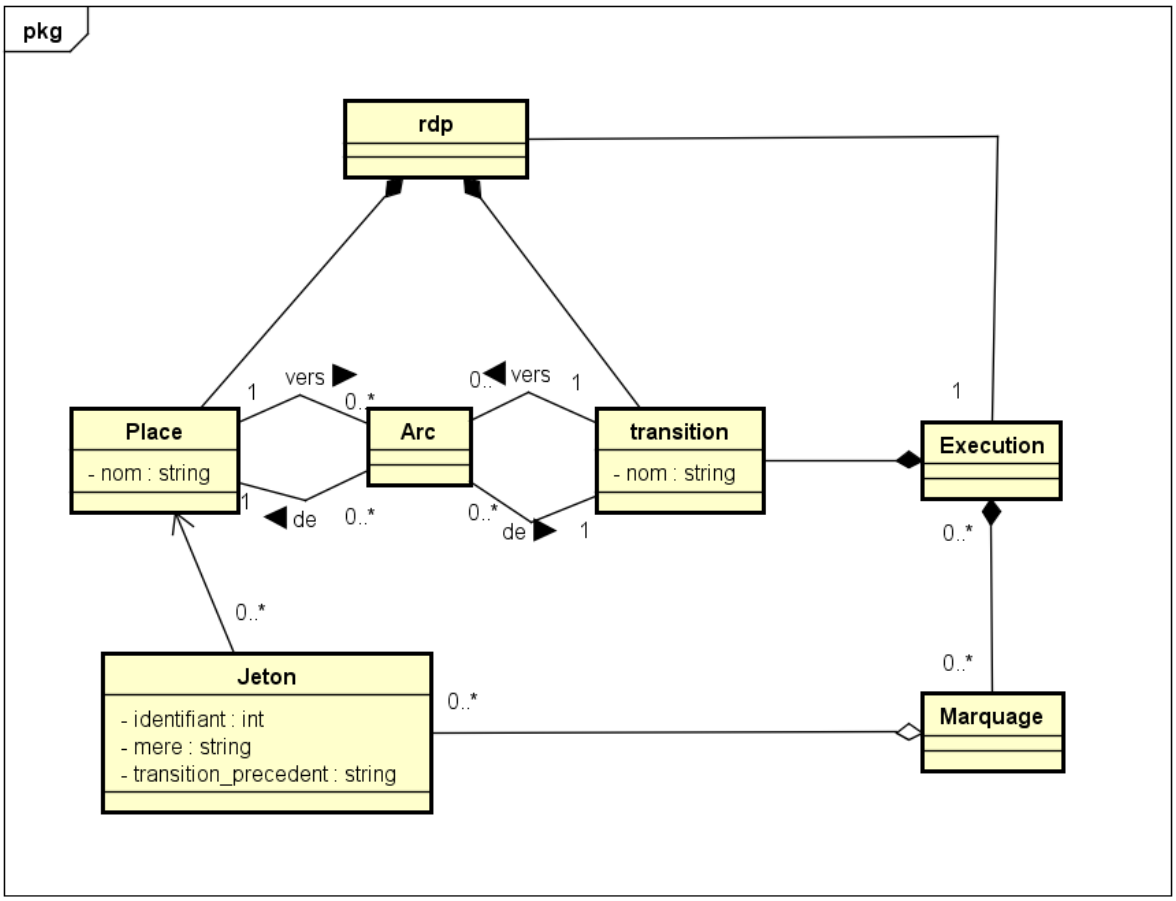


Figure 3.4 : Diagramme de classe

2. Implémentation

2.1.Outils de développement

2.1.1. IntelliJ IDEA

IntelliJ IDEA est un IDE intelligent et tenant compte du contexte qui permet de travailler sur toutes sortes d'applications en Java et dans d'autres langages de la JVM tels que Kotlin, Scala et Groovy. De plus, grâce à ses puissants outils intégrés, IntelliJ IDEA Ultimate vous aide à créer des applications web full stack avec JavaScript et les technologies connexes, ainsi que la prise en charge avancée de frameworks populaires tels que Spring, Spring Boot, Jakarta EE, Micronaut, Quarkus et Helidon. Les plugins gratuits développés par JetBrains peuvent compléter IntelliJ IDEA et permettre de travailler avec d'autres langages de programmation tels que Go, Python, SQL, Ruby et PHP [11]

Figure 3.5 : IntelliJ IDEA

2.1.2. Java

Java est un langage de programmation orienté objet. Java permet de créer des applications complète en utilisant les structure de données classiques (tableaux, fichier) et l'allocation dynamique de la mémoire pour créer des objets en mémoire. La classe, au sens de la programmation objet, remplace la notion de structure, qui est un ensemble de données décrivant une entité (un objet en Java). Le langage Java facilite également la création d'applications interactives et permet à l'utilisateur de piloter son programme dans un ordre non imposé par le logiciel en utilisant les interfaces graphique (GUI : Graphical User Interface) [12]

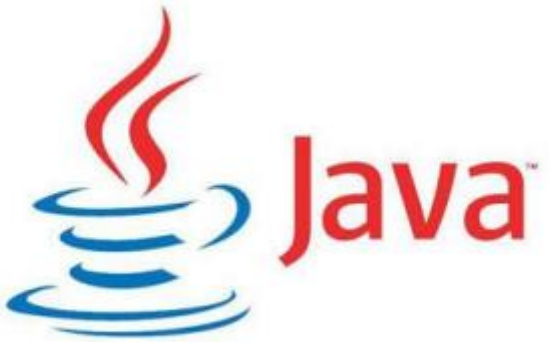


Figure 3.6 : JAVA

2.1.3. JDK

Le Java Development Kit (JDK) comprend un ensemble de bibliothèques logicielles de base du langage de programmation Java, ainsi que des outils pour compiler et convertir le code Java en bytecode destiné à la machine virtuelle Java. Selon la plate-forme Java1 adaptée (et évidemment la version Java visée), il existe diverses versions du JDK [12] :

JSE pour la Java 2 Standard Edition également désignée J2SE ;

JEE, sigle de Java Enterprise Edition également désignée J2EE ;

JME 'Micro Edition', destinée au marché mobile ;

À chacune de ces plateformes correspond une base commune de Development Kits, plus des bibliothèques additionnelles spécifiques selon la plate-forme Java que le JDK cible, mais le terme de JDK est appliqué indistinctement à n'importe laquelle de ces plates-formes.

Figure 3.7 : JDK

2.2.Présentation de quelques interfaces

Dans cette section, nous présentons quelques interfaces que nous avons réalisé.

2.3.Fenêtre principale

La figure ci-dessous présente la fenêtre principale de notre éditeur.

Figure 3.8 : Fenêtre principale

Fenêtre principale

Barre de menu

Figure 3.9 : Barre de menu

Barre de menu

La barre des menus comporte des menus classiques comme menu Fichier, Editer, Outils, Aide.

Fichier

Ce menu possède les actions habituelles comme Nouveau, Ouvrir, Enregistrer sous.

Outils

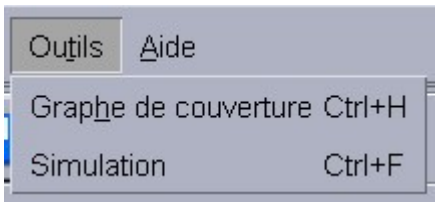


Figure 3.10 : Menu Outils

Menu Outils

Ce menu permet d'ouvrir la fenêtre de simulation pour lancer la simulation. Il permet aussi d'ouvrir la fenêtre graphe de couverture pour voir le graphe de couverture.

Barre d'outils

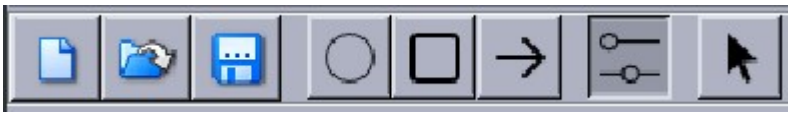


Figure 3.11 : Barre d'outils

Barre d'outils

New : Permet d'ouvrir une nouvelle fenêtre avec un réseau vide.

Open : Permet d'ouvrir un réseau de Pétri précédemment enregistré.

Save : Enregistre le réseau courant dans un fichier en lui donnant un nom.

Place : permet de dessiner une place dans l'afficheur.

Transition : permet de dessiner une transition dans l'afficheur.

Arc : permet de dessiner un arc reliant transition et place dans l'afficheur.

Attribut : permet de donner le nombre de jeton spécifique a une place, le poids d'un arc et la matrice d'incidence.

Select : permet de sélectionner les dessins dans l'afficheur.

Afficheur

Cette fenêtre permet, comme son nom l'indique, d'afficher un réseau via une Page.

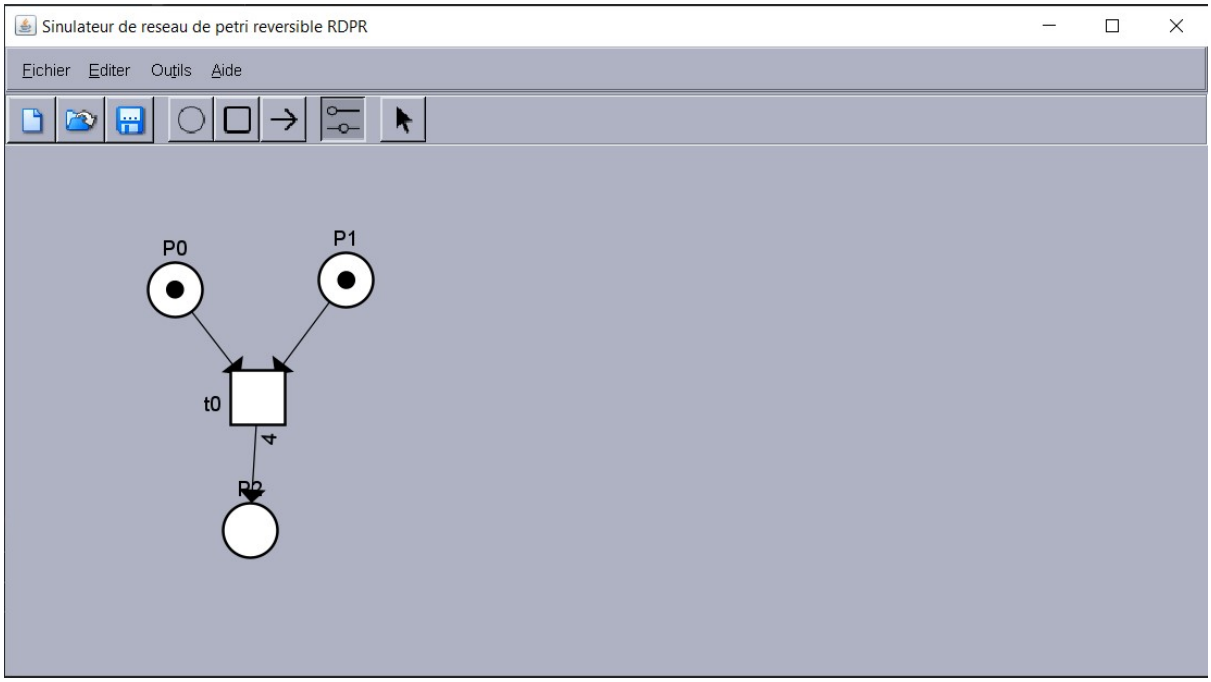


Figure 3.12 : Afficheur

C'est cette fenêtre qui gère les évènements claviers et souris ainsi que les menus contextuels.

L'afficheur contient le réseau. En réalité, à l'exception des propriétés des éléments, elle gère presque tout ce qui concerne le réseau lui-même. En somme, elle est responsable de la création des composants graphiques du réseau et de la représentation du réseau.

Dans la figure ci-dessous nous montre un réseau de pétri et sa matrice d'incidence

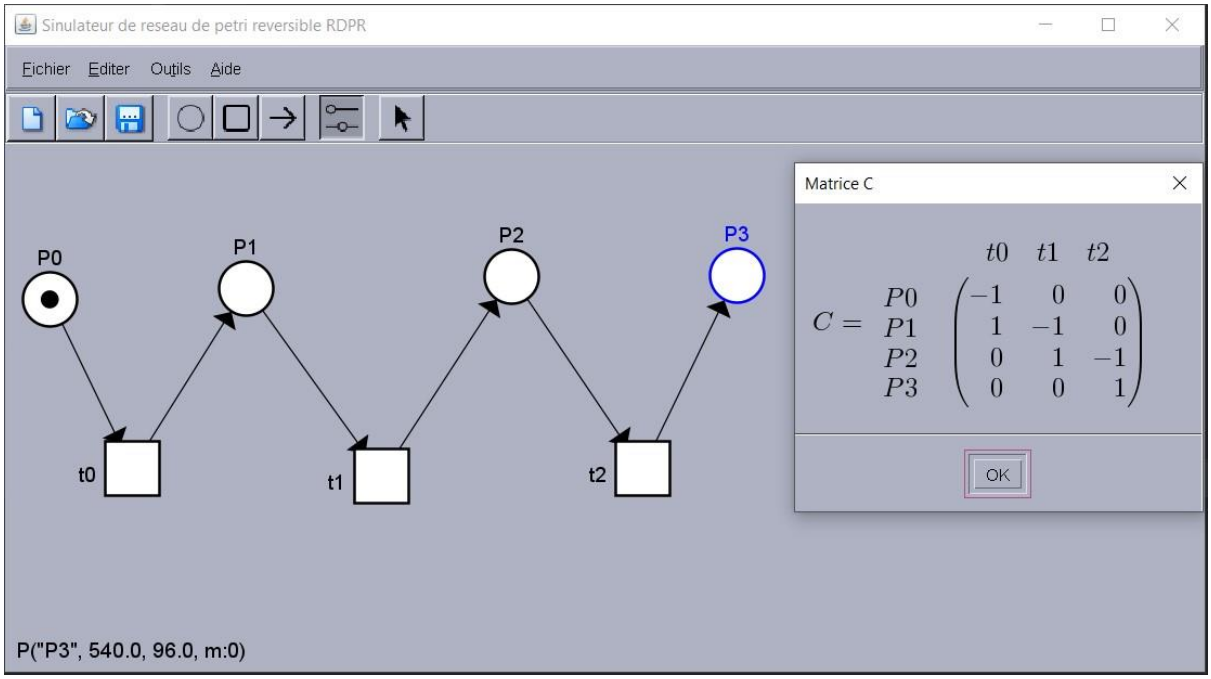


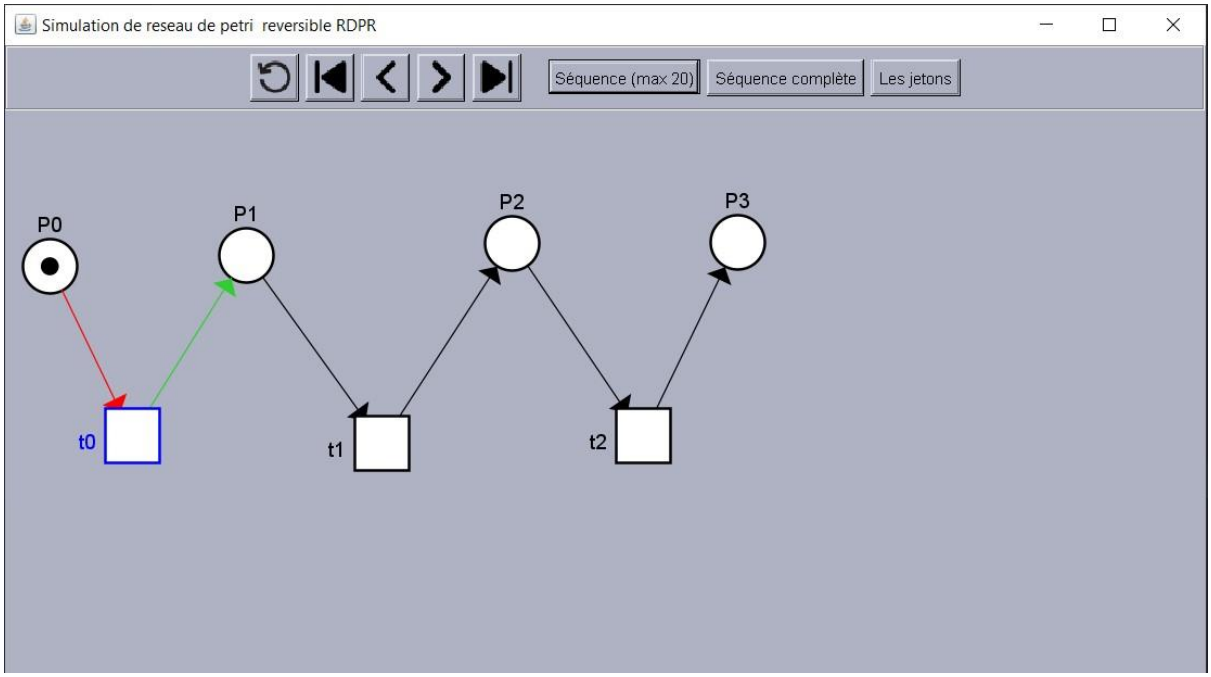
Figure 3.13 : Réseau de pétri avec sa matrice d'incidence

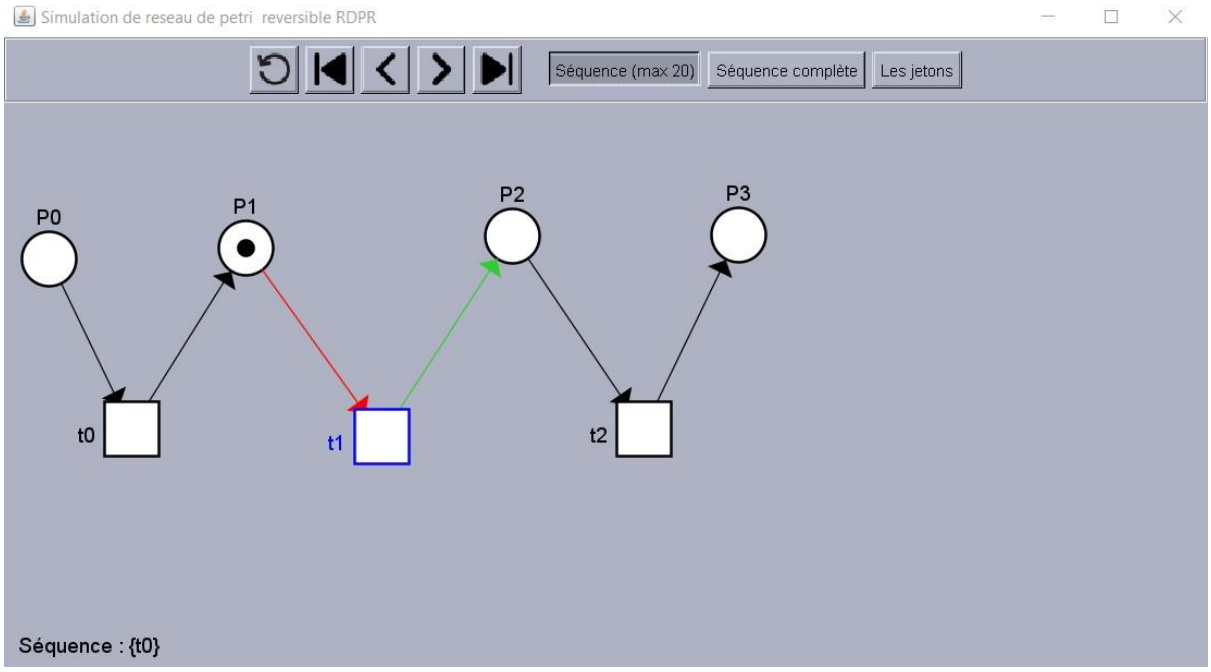
Dans la figure ci-dessus nous présentons un réseau le graphe de couverture du réseau de la figure ci-dessus.

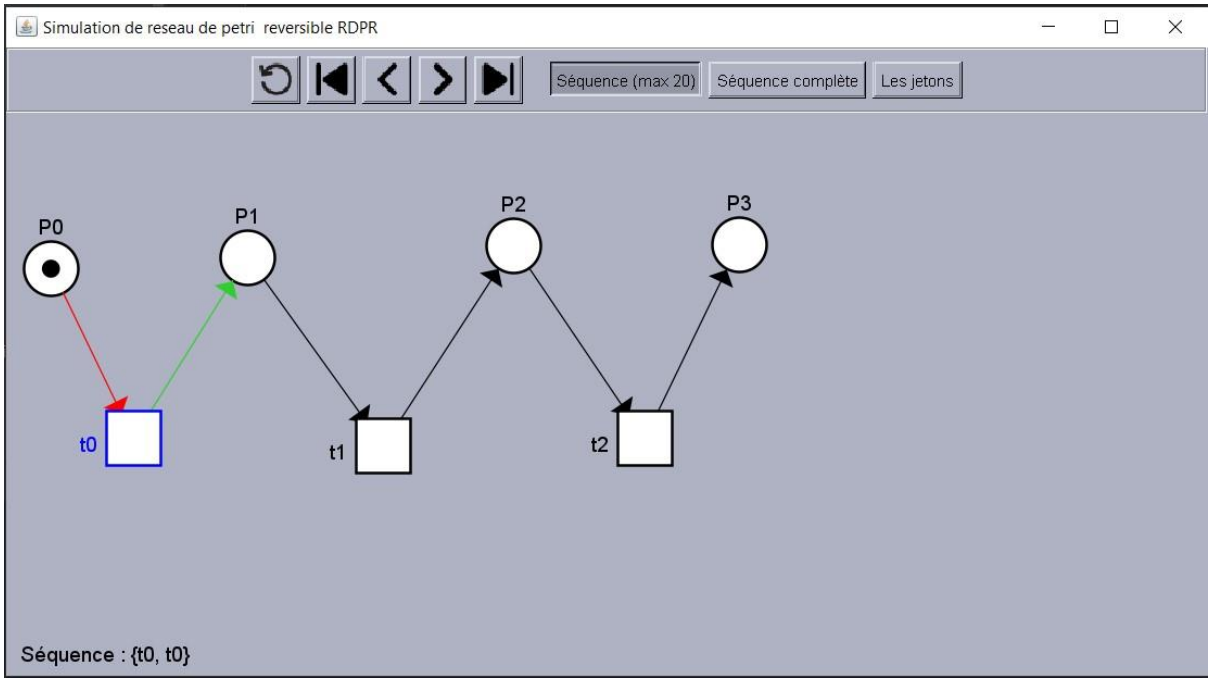
Figure 3.14 : Graphe de couverture

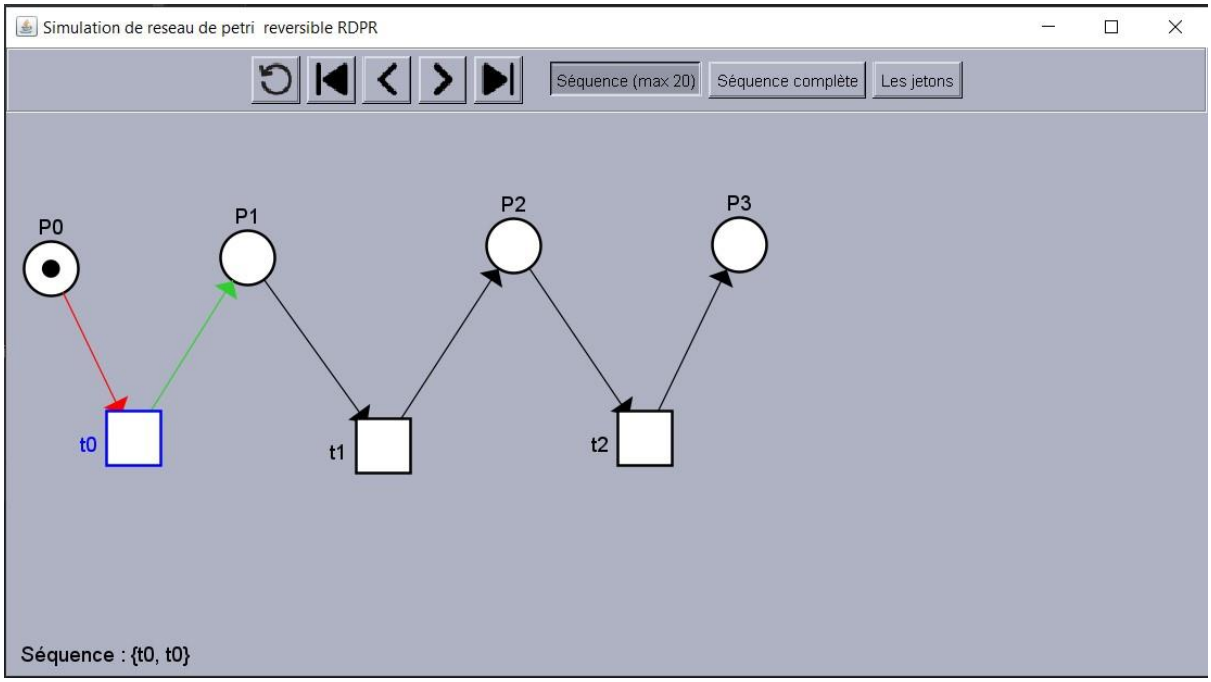
2.4.Exemple de simulation

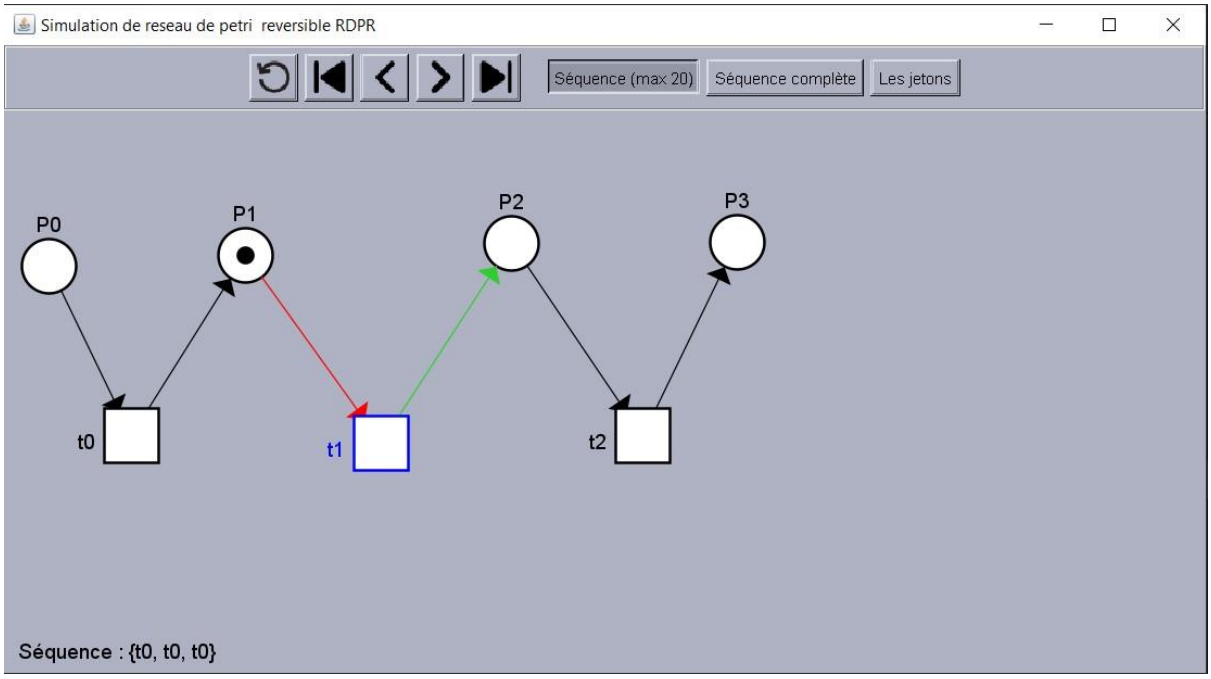
Les figures ci-dessous représente la séquence de simulation du réseau de pétri. La première la figure représente l'état initiale.

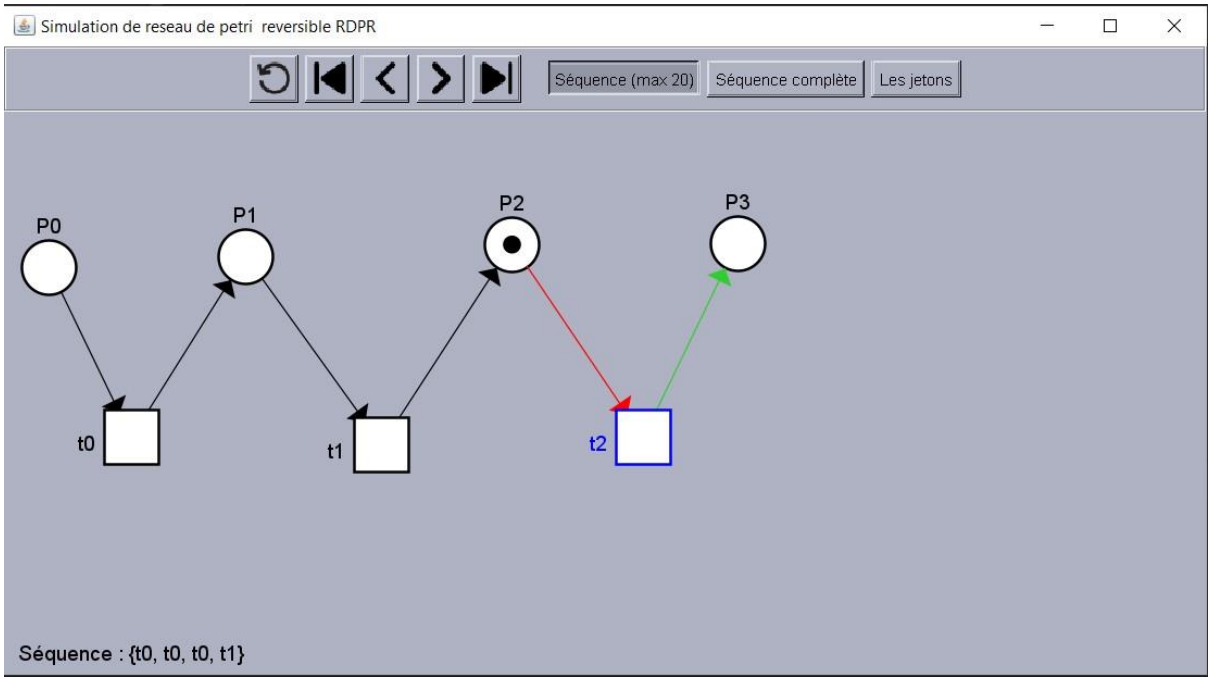

Simulation de reseau de petri reversible RDPR
Séquence (max 20)
Séquence complète
Les jetons
P0
P1
P2
P3
t0
t1
t2


Simulation de reseau de petri reversible RDPR
Séquence (max 20)
Séquence complète
Les jetons
P0
P1
P2
P3
t0
t1
t2
Séquence : {t0}


Simulation de reseau de petri reversible RDPR
Séquence (max 20)
Séquence complète
Les jetons
P0
P1
P2
P3
t0
t1
t2
Séquence : {t0, t0}


Simulation de reseau de petri reversible RDPR
Séquence (max 20)
Séquence complète
Les jetons
P0
P1
P2
P3
t0
t1
t2
Séquence : {t0, t0}


Simulation de reseau de petri reversible RDPR
Séquence (max 20)
Séquence complète
Les jetons
P0
P1
P2
P3
t0
t1
t2
Séquence : {t0, t0, t0}


Simulation de reseau de petri reversible RDPR
Séquence (max 20)
Séquence complète
Les jetons
P0
P1
P2
P3
t0
t1
t2
Séquence : {t0, t0, t0, t1}

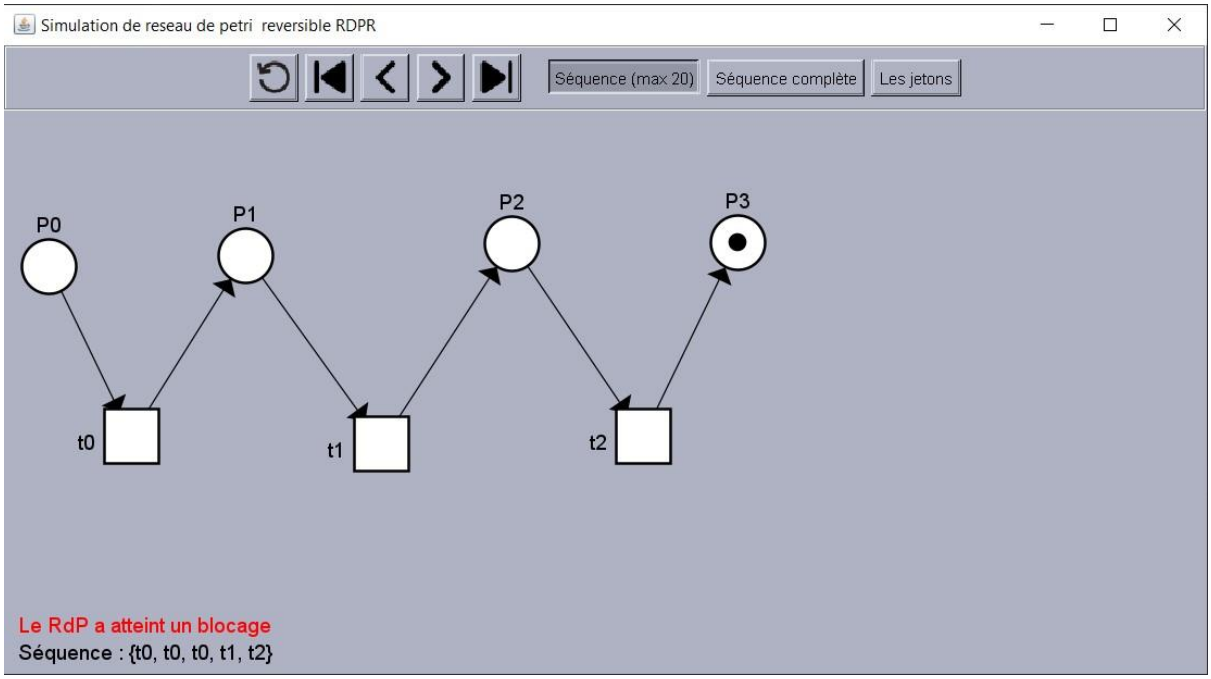


Figure 3.15 : Exemple de simulation

La figure ci-dessous montre la séquence de transition tiré par ordre.

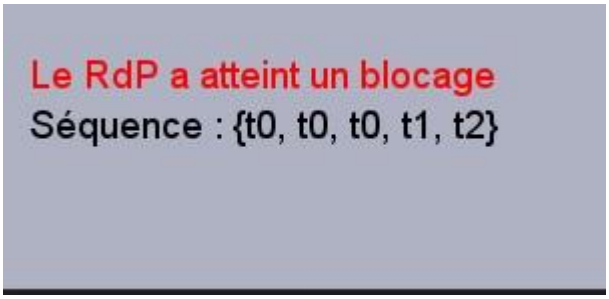


Figure 3.16 : Séquence de transition tirée par ordre

Lorsque l'utilisateur veut sauvegarder un fichier il clic sur save et choisi l'emplacement et le nom du fichier.

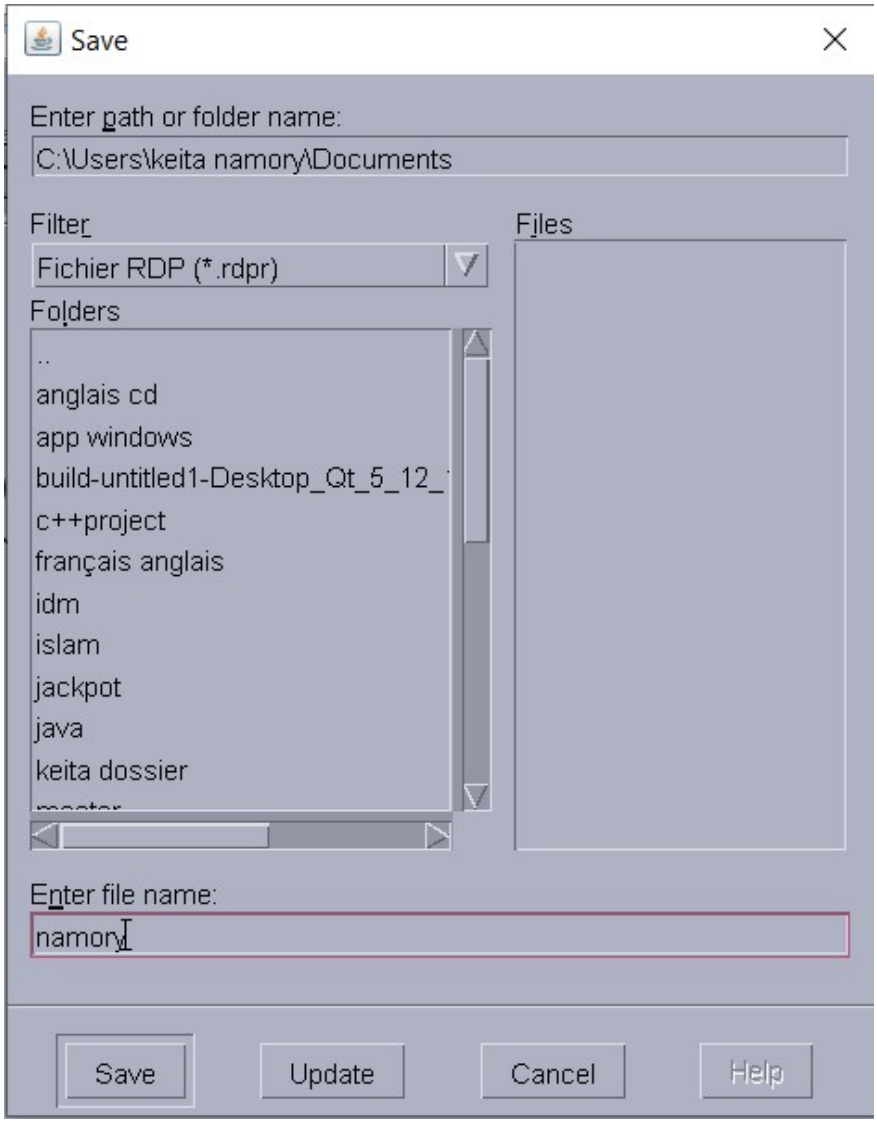


Figure 3.17 : Sauvegarde d'un fichier

Lorsque l'utilisateur veut ouvrir un fichier il clic sur open et choisi le fichier qu'il veut ouvrir au format (.rdpr)

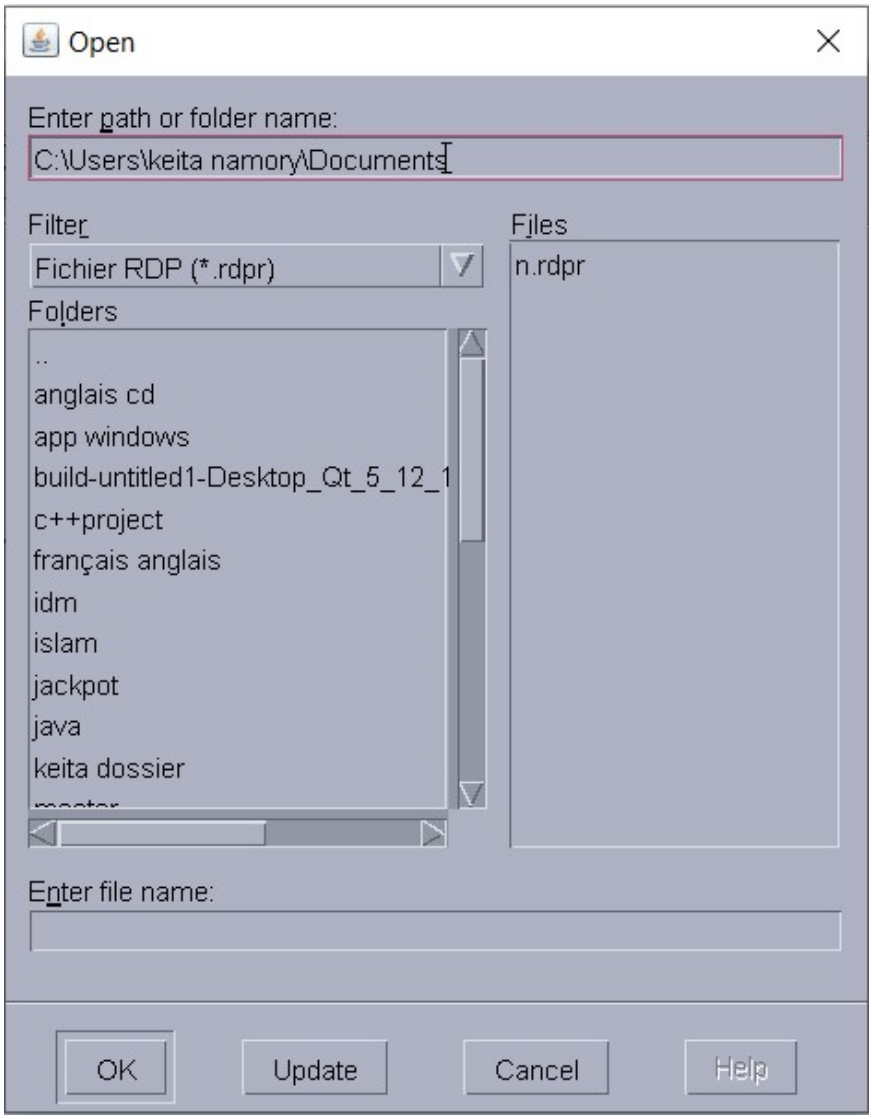


Figure 3.18 : Ouvrir un fichier .rdpr

Conclusion

Dans cette partie, nous avons présenté la conception de notre application tels que les modules, les exigences fonctionnelles, et quelques diagrammes UML. Ainsi, nous avons fourni une vue d'ensemble de l'architecture logicielle que nous allons concevoir. De plus avec UML, nous nous assurons d'avoir une base solide pour la création d'une application fonctionnelle et conviviale pour les utilisateurs.

Conclusion Générale

Notre projet de fin d'étude consiste à créer et mettre en œuvre une application graphique pour éditer ensuite simuler les réseaux de pétri réversible. Nous avons fait ce travail en passant par plusieurs phases. En premier nous avons mené une recherche sur le domaine ensuite une analyse du projet pour faire la conception et l'implémentation de notre éditeur.

La réalisation d'un éditeur de réseau de pétri réversible constitue une avancée significative dans le domaine de la modélisation avec les réseaux de pétri. Cet éditeur offres aux utilisateurs un plateforme pour représenter et analyser des comportement complexe grâce à la réversibilité des transitions. L'éditeur offre aux utilisateur la possibilité de créer, visualiser et analyser des réseaux de pétri avancés, en prenant en compte la réversibilité des transitions. La réversibilité des transitions permet aux utilisateurs d'explorer différents scenarios et de revenir en arrière dans l'exécution, offrant ainsi une plus grande souplesse.

Nous n'avons pas pu implémenter notre éditeur en utilisant l'interprétation individuelle des jetons et la relation de causalité. Ainsi l'interprétation individuelle des jetons et la relation de causalité feront l'objet de nos future perspectif pour que notre éditeur soit complet.

Bibliographie

[1] A. Philippou and K. Psara, ‘Reversible Computation in Petri Nets’, Apr. 2018, [Online]. Available: http://arxiv.org/abs/1804.04607

[2] A. Benamira, ‘CAUSAL REVERSIBILITY IN INDIVIDUAL TOKEN INTERPRETATION OF PETRI NETS’, *Computer Science*, vol. 21, no. 4, pp. 489–511, 2020, doi: 10.7494/csci.2020.21.4.3728.

[3] D. Boughareb, ‘Méthodes Formelles pour le Parallélisme’, Université 08 mai 1945, Guelma, 2017.

[4] H. Diab, ‘ÉVALUATION DE MÉTHODES FORMELLES DE SPÉCIFICATION’, Université de Sherbrooke, Québec, 1999.

[5] R. Lilia, ‘Proposition d’un modèle base sur les réseaux de pétri pour modéliser les microservices’, Universite 08 Mai 1945, Guelma, 2022

[6] G. Selma and C. Imene, ‘Un environnement graphique pour l’analyse des réseaux de pétri temporels’, Université de Larbi Ben M’hidi d’Oum el Bouaghi, 2021.

[7] R. J. Van Glabbeek, ‘The individual and collective token interpretations of Petri nets’, in *Lecture Notes in Computer Science*, Springer Verlag, 2005, pp. 323–337. doi: 10.1007/11539452_26.

[8] M. Riyadh ABDMEZIEM, ‘An Automatic Petri-net Generator for Modeling Multi-agent Systems (Master thesis)’, 2012, doi: 10.13140/RG.2.2.18312.01280.

[9] K. Barylska, M. Koutny, Ł. Mikulski, and M. Piątkowski, ‘Reversible computation vs. reversibility in Petri nets’, *Sci Comput Program*, vol. 151, pp. 48–60, Jan. 2018, doi: 10.1016/j.scico.2017.10.008.

[10] A. Philippou and K. Psara, ‘Token Multiplicity in Reversing Petri Nets Under the Individual Token Interpretation’, in *Electronic Proceedings in Theoretical Computer Science, EPTCS*, Open Publishing Association, Sep. 2022, pp. 131–150. doi: 10.4204/EPTCS.368.8.

[11] L. Rayene and B. Omram, ‘Proposition d’une méthodes alpha-réduit pour les réseaux de pétri récursifs sous une sémantique de vrai parallélisme’, Université de Larbi Ben M’hidi d’Oum el Bouaghi, 2022.

[12] B. Khaled and A. Chemseddine, ‘Conception et réalisation d’un éditeur graphique de réseau de pétri’, Université d’Ibn Khaldoum, Tiaret, 2027.