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

Université Ammar Telidji Laghouat

Faculté des Sciences et Sciences de l'Ingénierie
Département de Génie Informatique

Mémoire de Master

Filière: Informatique

Option: Réseaux, systèmes et applications répartis (**ReSar**)

Thème:

Ver un verificateur des modeles pour la validation des reseaux

Présenté par:

Benhammar Fatima

Soutenu devant le jury composé de:

Mr	X. Xxxxxx	Université de Laghouat	(Président)
Mr	X. Xxxxxx	Université de Laghouat	(Examinatrice)
Mr	X. Xxxxxx	Université de Laghouat	(Examinateur)
Mr.	T. Allaoui	Université de Laghouat	(Rapporteur)

Année 2011

Je dédie ce mémoire
à
mes parents
qui m'ont toujours soutenu et encouragé
au cours de la réalisation de ce mémoire de fin d'étude.
à
tous les professeurs et enseignants universitaires
qui m'ont permis, par leurs efforts, d'atteindre un tel niveau de formation.

Remerciements

Je remercie Dieu tout Puissant de m'avoir permis de mener à terme ce mémoire de fin d'étude.

En préambule à ce mémoire, je souhaite adresser ici tous mes remerciements aux personnes qui m'ont apporté leur aide et qui ont ainsi contribué à l'élaboration de ce mémoire.

Qu'il me soit permis de rendre un vibrant hommage à mon encadreur Mr Guelloma Younes,, qui était pour moi un immense honneur de travailler avec lui durant mon projet fin d'étude , et par coïncidence Dieu a voulu que je travail encore une fois avec lui dans ce mémoire. Donc un grand remerciement à lui pour avoir bien voulu superviser ce travail et donner de son temps et de son intelligence à la réussite de ce projet qui pour moi représente un modèle de réussite et une source de motivation permanente, je le remercie également pour sa disponibilité, son sens aigu de l'humanisme pédagogique, et enfin sa rigueur intellectuelle et morale qui est pour moi plus qu'un exemple à suivre, un modèle.

Je remercier également toute ma famille, surtout ma mère et Hadj Ahmed et que j'aime tant. Vous avez été mes modèles pendant toutes ces années et je fais tout pour devenir quelqu'un d'aussi bien que vous.

Enfin je remercie les membres du jury qui ont bien voulu accepter, et ce malgré, leur lourdes et exaltantes responsabilités pour procéder à l'évaluation de ce modeste travail.

Résumé

Vue l'importance des système informatique en particulier les réseaux dans notre quotidien(domestique, commercial).Malheureusement on rencontre des problèmes de la validation des réseaux ce qui nous donne la réflexion de s'assurer que les système se comportent et échange leurs information d'une manière correcte. donc on doit vérifier les système pour cela on a besoin de combiner 2 approches de vérification formelle(parmi ces approches le modèle checking, théoreme proving). Dans ce mémoire nous proposons une bibliothèque pour vérifier les systèmes à événements discrets. Nous vérifions les modèles avec deux idées distinctes :Dans la première idée, nous avons utilisé les méthodes de simplification présentées sont applicables sur les réseaux de pétri .La deuxième idée en utilise les expressions rationnelles pour modéliser tout phénomène régulier dans le temps ou dans l'espace finie ou infini.

Mots-clés : la vérification des réseaux, méthodes formelles,réseaux de pétri,expression rationnelle, Routage,modélisation, Simulation, NS-2.

Abstract

Thesignificance of the particular computer system networks in our daily lives (residential, commercial).Unfortunately the problems of validation networks are found which gives us reflection to ensure that the system behave and exchange their information in a correct manner. So we must check the system for that you need to combine two approaches to formal verification (these approaches the model checking, theorem proving). In this paper we propose a library to check the discrete event systems. We test the models with two distinct ideas : The first idea, we used the simplification methods presented are applicable to networks kneaded The second idea uses regular expressions to model any regular phenomenon in the time or in the. finite or infinite. space

Key-words : verification of networks, formal methods, Petri networks, regular expression, routing, modeling, simulation, NS-2.

Table des matières

Table des matières vi

Liste des figures viii

Introduction générale 1

1 Notions générales 3
1.1 Introduction . . . 4
1.2 Les Méthodes Formelles pour la vérification des systèmes : . . . 4
1.3 Réseaux de Pétri . . . 8
1.3.1 Notions de Base : . . . 9
1.3.1.1 les intérêts de réseaux de pétri : . . . 10
1.3.1.2 les limites de réseaux de pétri : . . . 11
1.4 l'expression rationnelle : . . . 12
1.4.1 Construction d'automates finis à partir des expressions rationnelles : . . . 12
1.5 Méthodes et outils d'analyse des UPPAAL et SPIN : . . . 14
1.5.1 UPPAAL . . . 14
1.5.1.1 Présentation : . . . 14
1.5.2 SPIN Model Checker . . . 14
Conclusion . . . 15

2 Généralités sur réseaux 17
2.1 Introduction . . . 18
2.2 Les réseaux . . . 18
2.2.1 Que signifie le réseau . . . 18
2.2.2 Les réseaux mobiles : . . . 19
2.2.2.1 Les réseaux mobiles avec infrastructures : . . . 19
2.2.2.2 Les réseaux mobiles sans infrastructures (AD HOC) : . . . 19
2.2.2.3 Définition d'un réseau AD HOC : . . . 20
2.2.2.4 Les caractéristiques des réseaux $ADHOC$: . . . 20
2.2.2.5 Les domaines d'applications des réseaux$ADHOC$: . . . 21
2.2.2.6 Définition d'un routage : . . . 21
2.2.2.7 Définition des Protocoles : . . . 22
2.2.2.8 Classification des protocoles de routage : . . . 22
2.2.2.8.a Les protocoles de routage $proactifs$: . . . 23
2.2.2.8.b Les protocoles de routage réactifs (à la demande) . . . 23
2.2.2.8.c Les protocoles de routage hybrides : . . . 23
2.2.3 La simulation : . . . 23
2.2.3.1 Méthodologie générale : . . . 24
2.2.3.2 Le simulateur avec NS-2 : . . . 25
2.2.3.3 Définition : . . . 25

2.2.3.4 Les étapes de simulation : . . . 25
2.2.3.5 La réalisation d'une simulation . . . 25
2.3 Conclusion . . . 25
Conclusion . . . 25

3 conception et réalisation de l'outil de validation des réseaux 27
3.1 Introduction . . . 28
3.2 La modélisation : . . . 28
3.2.1 L'étape de modélisation . . . 28
3.2.2 Définition d'un modèle : . . . 29
3.2.3 Création des modèles : . . . 30
3.3 Exemple de modélisation : . . . 30
3.3.1 Exemple 1 : . . . 30
3.3.2 Exemple 2 : . . . 31
3.3.3 Exemple 3 : . . . 31
3.3.4 Modèle statique de notre bibliothèque . . . 31
3.3.5 Spécification de l'outil réalisé : . . . 31
3.3.5.1 création de project : . . . 32
3.3.5.2 création de réseaux de pétri . . . 32
3.3.5.3 l'affichage de création de réseaux de pétri : . . . 34
3.3.5.4 création avec expression rationnelle : . . . 35
3.3.5.5 le tirage d'une transition de réseaux de petri : . . . 35
3.3.5.6 création avec expression rationnelle : . . . 36
3.3.5.7 Aide : . . . 37
3.4 conclusion . . . 37

Conclusion et Perspectives 39
RÃ©fÃ©rences . . . 40

Bibliographie 41

Liste des figures

1.1 les étapes de la vérification de modèle. 5
1.2 Les Techniques de vérification formelle. 6
1.3 Méthode générale de modélisation et d'analyse basée sur les réseaux de Pétri. 8
1.4 Exemple de réseau de Pétri 9
1.5 (A) RdP marqué avant franchissement de T1 (B) RdP après franchissement de T1. 11
1.6 Réseau de Pétri d'une pièce contenant une porte-fenêtre et une climatisation. 11
1.7 (A) RdP marqué avant franchissement de T1 (B) RdP après franchissement de T1. 13
1.8 construire les automates qui vont reconnaître des expressions plus complexes avec l'expression rationelle. 13

2.1 les étapes de la vérification de modèles.. 19
2.2 Le modèle des réseaux mobiles avec infrastructure. 20
2.3 Le modèle des réseaux mobiles sans infrastructure.. 20
2.4 Application de secours des réseaux AD HOC. 21
2.5 Applications commerciales des réseaux AD HOC. 21
2.6 Le chemin utilisé dans le routage entre la source et la destination.. 22
2.7 Classification des protocoles de routage. 23
2.8 Méthodologie d'une simulation. 24
2.9 Les étapes de simulation. 26
2.10 La réalisation d'une simulation. 26

3.1 La réalisation d'une simulation. 28
3.2 La modélisation d'un réseau ad hoc. 29
3.3 La modélisation d'un réseau ad hoc.. 30
3.4 La réalisation d'une simulation. 30
3.5 Le modèle statique de notre bibliothèque. 32
3.6 page d'aceuil. 32
3.7 création de rdp 33
3.8 création de réseaux de pétri linéaire 33
3.9 création de réseaux de pétri linéaire. 33
3.10 l'affichage de graphe. 34

Introduction générale

Depuis le milieu du vingtième siècle, les systèmes informatiques occupent une place croissante dans notre quotidien. les systèmes informatiques temps réel sont présents dans de multiples secteurs d'activités : contrôle des systèmes automatises de production, aide a la conduite des véhicules (en particulier dans le domaine de l'aviation) ou gestion des flux d'information sur des réseaux locaux et sur Internet. Au cours des dernières années, les systèmes temps réel se sont progressivement établis comme une discipline a part entière qui rassemble une forte communauté issue a la fois du monde académique et de l'industrie. L'évolution de ces systèmes se caractérise par une complexité croissante et un rôle toujours plus critique.

Leurs mise au point est un problème complexe pour lequel il est recommande d'utiliser des techniques de vérification pour la validation des systèmes.
Le but général de ce mémoire, est de proposer une solution pour vérifiée un système avec des méthodes formelles.

En est arrivé à un stade ou le réseau informatique a ressues à s'imposer dans presque tout les domaines d'activité (professionnelle, commercial, domestique). La gestion des réseaux est donc indispensable. Il faut souvent avoir recours à des techniques d'administration pour pouvoir contrôler son fonctionnement afin d'exploiter au mieux les ressources disponibles, il existe plusieurs aspect de problème de la validation des réseaux :

– Le fait qu'un scénario ne marche pas dans la simulation n'implique pas forcement sont échec dans la réalité.

– Difficulté d'obtention d'un réseau fiable (Nombre important des nœuds, les fonctionnalités ne sont pas toujours implémentées, l'hétérogénéité des nœuds).

– La plupart des simulateurs réseaux se focalisent seulement sur des protocoles simples et simulent des protocoles indépendamment.

– De nombreuses différences entre les résultats en simulation par rapport aux résultats en réel.

– En conclure de ces problèmes, comment construire un réseau fiables, et s'assurer que les systèmes se comportent et échangent leurs informations d'une manière correcte ? Donc on a besoin de valider le système.

Les objectifs de ce mémoire de master est d'étudier la possibilité de combiner ces deux approches de vérification formelle (*modlechecking*, *preuvedethormes*) pour valider un système :

– Modélisation et analyse des modèles en utilisant la théorie structurelle des *RdPs* .

– Vérification *modlechecking* des modèle en réseaux.

– Essayer d'élaborer un nouveau modèle de vérification, qui répond plus efficacement aux besoins, en particulier dans les situations issues du domaine d'application que nous considérons (les réseaux).par
– Essayer d'élaborer une nouvelle bibliothèque dédiée au domaine des réseaux afin de répondre aux exigences de standardisation.

– En effet, il existe des outils et logiciels de validation formelle de données telles que SPIN, UPPAAL,nous tenterons de concevoir un modèle.

Notre mémoire est structuré en trois chapitres :Dans le 1^{er} *chapitre*, nous allons aborder des notions générales sur les deux approches de vérification formelle (*preuvedethormes*, *modlechecking*), nous donnons une idée générale sur notre problématique, citons nos motivations et nos objectifs.
Le 2^{eme}*chapitre* , consacré à notre étude de cas :

– les réseaux.

– Le simulateur.

– Problèmes rencontrés.

le 3^{eme}*et dernier chapitre*, illustre la modélisation de réseau puis une méthode de vérificateur des modèles pour la validation de réseau.
la *conclusion de ce mémoire* résume les travaux faits durant toutes nos études ainsi que des possibles améliorations futures.

Notions générales

1

Sommaire

1.1 Introduction 4
1.2 Les Méthodes Formelles pour la vérification des systèmes : 4
1.3 Réseaux de Pétri 8
1.3.1 Notions de Base : 9
1.3.1.1 les intérêts de réseaux de pétri : 10
1.3.1.2 les limites de réseaux de pétri : 11
1.4 l'expression rationnelle : 12
1.4.1 Construction d'automates finis à partir des expressions rationnelles : 12
1.5 Méthodes et outils d'analyse des UPPAAL et SPIN : 14
1.5.1 UPPAAL 14
1.5.1.1 Présentation : 14
1.5.2 SPIN Model Checker 14
Conclusion 15

Dans ce chapitre ,on présente les deux approches de vérification formelles des modèles dans sa forme la plus générale, indépendamment de tout domaine d'application. On présentera dans la première partie la phase de vérification formelle en s'intéressant particulièrement à la vérification des systèmes (modèle). Dans la deuxième partie, on présente les principes de base de réseau de pétri. On termine le chapitre en présentant des outils usuels pour la vérification.

1.1 Introduction

Les tests et la simulation contribuent depuis longtemps à la validation de systèmes. Cependant, ces techniques ne permettent d'explorer qu'une partie des comportements possibles. Elles diffèrent en cela des techniques de vérification formelle, qui garantissent qu'une propriété est vérifiée pour la totalité des exécutions possibles du système.

Les méthodes de vérification formelles sont des techniques permettant de raisonner rigoureusement, a l'aide de logiques mathématiques, sur des programmes informatiques ou des matériels électroniques, afin de démontrer leur validité par rapport a une certaine spécification. Ces méthodes permettent d'obtenir une très forte assurance de l'absence de bug dans les logiciels, c'est-a-dire d'acquérir des niveaux d'évaluation d'assurance élèves, elles sont basées sur les sémantiques des programmes, c'est-a-dire sur des descriptions mathématiques formelles du sens d'un programme donné par son code source (ou parfois, son code objet). Cependant, elles sont généralement couteuses en ressources (humaines et matérielles) et actuellement réservées aux logiciels les plus critiques. Leur amélioration et l'élargissement de leurs champs d'application pratique sont la motivation de nombreuses recherches scientifiques en informatique.
Dans ce qui suit, les deux principales techniques de vérification formelle sont « preuve de théorèmes » et « le modèle *checking* ». Dans la première un modèle du système à vérifier est vu comme un ensemble d'axiomes qui servent à prouver une propriété. Cette technique n'est pas automatique ; des outils d'aide à la preuve existent mais demandent une forte expertise. Le modèle *checking*, introduit il y a 20 ans, regroupe plusieurs techniques entièrement automatiques dans lesquelles la propriété à vérifier est testée de façon exhaustive sur l'ensemble des exécutions possibles du système.

1.2 Les Méthodes Formelles pour la vérification des systèmes :

Définition :

La vérification de modèles est basée sur trois étapes :La modélisation du système réel, la spécification des propriétés à vérifier et finalement la vérification proprement dite. La figure suivante montre l'exemple d'une mobylette modélisée par un dessin à la main faisant abstraction de plusieurs éléments du système réel. La propriété "je peux immobiliser la mobylette à tout moment est spécifiée dans un langage naturel et la vérification est effectuée par un être humain qui va vérifier si cette propriété est valide dans le modèle [BAR98].

Vérification

La vérification répond à la question « Construisons nous correctement le modèle ? (*isthesystembeingbuiltright*?), La vérification consiste à s'assurer que le modèle fonctionne comme le concepteur le désire (pas d'erreur de logique), ce qui nécessite l'isolation des erreurs (étape la plus difficile) afin de les corriger. La vérification est plus facile si on commence par un modèle simple qu'on améliore progressivement. Les techniques suivantes

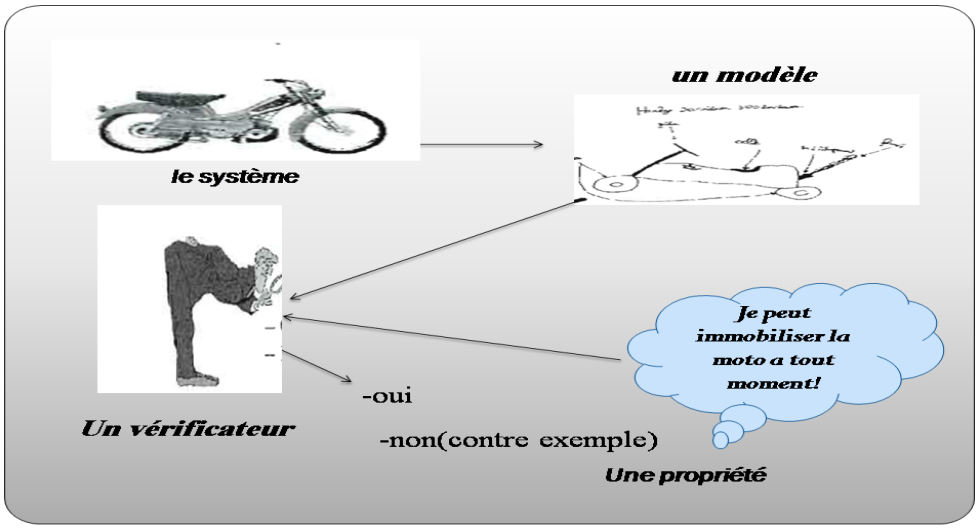


Figure 1.1 – *les étapes de la vérification de modèle.*

permettent l'isolation des erreurs (ou comportement à avoir) :

Considérer toujours que le modèle contient des erreurs.
Effectuer des tests :
Tester seulement une partie du modèle.
Tester le modèle dans des conditions limites. Pour cela : analyser la trace du modèle pour vérifier l'acheminement, et les changements d'état.

Validation :

La validation consiste à évaluer l'adéquation du système développé vis-à-vis des besoins exprimés par ses futurs utilisateurs. La validation cherche à répondre à la question "Construisons nous le bon modèle ?" (*istherightsystembeingbuilt*?).Trois questions doivent être posées :

– – – – – Le modèle représente-t-il correctement le système réel ?

– Les données sur le comportement généré par le modèle, sont-elles caractéristiques du système réel ?.

– L'utilisateur a-t-il confiance dans les résultats du modèle ?

Les Techniques de vérification formelle

Les méthodes de vérification formelle (voir figure 1.2) peuvent être classées en deux grandes approches : l'approche déductive (theorem proving) et l'approche comportemental (model checking). Ces deux techniques ont eu beaucoup de succès et ont été intégrées dans plusieurs outils de vérifications des systèmes [JR99].

Les modèles de vérification :

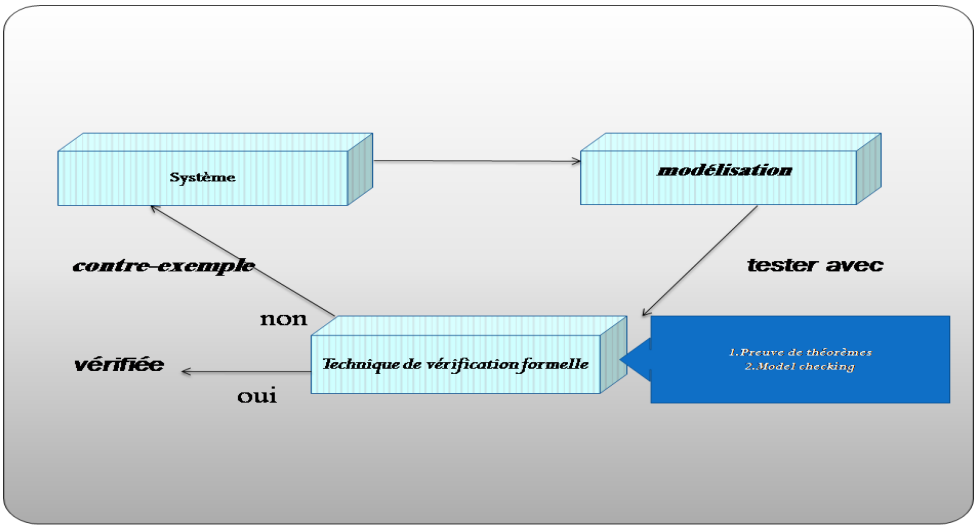


Figure 1.2 – *Les Techniques de vérification formelle.*

On peut distinguer deux principales approches permettant la vérification ou la preuve formelle de propriétés sur des modèles formels :

Preuve de théorèmes (*Theoremproving***) :**

– *Lapreuvedethormes* : est une technique où le système et les propriétés recherchées sont exprimés comme des formules dans une logique mathématique. Cette logique est décrite par un système formel qui défini un ensemble d'axiomes et de règles de déduction. Les étapes pendant la preuve font appel aux axiomes et aux règles, ainsi qu'aux définitions et lemmes qui ont été possiblement dérivés. Au contraire du *modlechecking*, le *theoremproving* peut s'utiliser avec des espaces d'états infinis à l'aide de techniques comme l'induction structurelle. Son principal inconvénient est que le processus de vérification est normalement lent, sujet à l'erreur, demande beaucoup de travail et des utilisateurs très spécialisés avec beaucoup d'expertise[HMo6].

– **Théorème** : proposition qui peut être mathématiquement démontrée, c'est-a-dire une assertion (proposition mathématique admise) qui peut être établie comme vraie au travers d'un raisonnement logique construit a partir d'axiomes (vérité indémontrable qui doit être admise) .Une fois le théorème démontre, il devient alors une hypothèse utilisable.

– **Preuve** :permet d'établir une proposition a partir de propositions initiales, ou précédemment démontrées a partir de propositions initiales, en s'appuyant sur un ensemble de règles de déduction. Une fois démontrée, la proposition peut ensuite être elle-même utilisée dans d'autres démonstrations. Dans ce cas, on la nomme généralement lemme. La preuve, bien que nécessaire a la Classification de la proposition comme "théorème", n'est pas considère comme faisant partie du théorème.

– **Assistants de preuve** :Un assistant de preuve est un logiciel permettant l'écriture et la vérification de preuves mathématiques, soit sur des théorèmes au sens usuel des ma-

thématiques, soit sur des assertions relatives a l'exécution de programmes informatiques. L'écriture de preuves entièrement formelles est une activité extrêmement fastidieuse ; de nombreuses étapes qui seraient sautées, car considérées comme évidentes pour le lecteur familier des mathématiques, doivent être décortiquées dans les plus grands détails. Cependant, l'assistant de preuve peut fournir plus ou moins d'automatisation pour limiter le travail de l'utilisateur humain. De nombreux assistants de preuves sont disponibles, certains sont inspires de l'isomorphisme de $Curry - Howard$: chaque preuve acceptée fournit un lambda-terme dont le type est exactement le théorème. Créer ce terme correspond à construire la preuve et vérifier la preuve correspond à typer ce terme, ce qui est une opération simple exécutable par un programme de petite taille. L'assistant de preuves a pour fonction de construire ce lambda-terme a partir de la preuve fournie par l'utilisateur, qui peut ensuite être type soit par l'assistant de preuves, soit par un autre programme .

Le modèle checking(MC)

Les étapes nécessaire de la création d'un*modlechecking* : Le *modlechecking* permet de vérifier la sureté de fonctionnement d'un système en en 3 étapes :

– $Lapremiretapedu ModleChecking$: consiste a exprimer le modèle considéré au moyen d'un graphe oriente, formé d'un ensemble de nuds et de transitions. Chaque nud représente un état du système, chaque transition représente une évolution possible du système d'un état donné vers un autre état. Parallèlement, le système est décrit par un ensemble de propositions logiques atomiques (ex. i=2, le processeur 3 est en attente, ...)[1a]. Chaque état du graphe orienté est étiqueté par l'ensemble des propositions atomiques vraies a ce point d'exécution. Un tel graphe est appelle structure de *Kripke* [MON03].

– **La deuxième étape du** *ModleChecking* : consiste à exprimer la négation de la formule de logique temporelle que nous souhaitons tester. La négation de cette formule est donc elle même transcrite sous forme d'une structure de *Kripke*, capable de reconnaitre exactement l'ensemble des exécutions satisfaisant la négation de la formule donnée. Par exemple, on pourra transcrire une formule logique LTL (logique temporelle linéaire) en un automate de Buchi [RDP99] non-déterministe.

– **La troisième et dernière étape** :consiste à réaliser le produit cartésien synchrone des deux structures de *Kripke* obtenues précédemment. Si le langage reconnu par le produit est vide, alors le système satisfait la formule de logique. Sinon, toute séquence appartenant au langage du produit constitue un contre exemple a la spéciation. Enumérer explicitement tous les états de l'automate peut être couteux, c'est pourquoi on Procède généralement par des méthodes symboliques, introduites par*KenMcMillan* et *EdClarke* [MOCH99].

– **La structure de** *Kripke* :représentant l'évolution du système via tous ces états accessibles est généralement obtenu a partir d'une description formelle du système (Algèbre de processus, automates, Réseaux de Pétri, etc..)[KP08]

1.3 Réseaux de Pétri

Historique :

Les réseaux de Pétri, introduits dans la thèse de Carl Adam Pétri en 1962,[RDP99] sont des outils graphiques et mathématiques permettant de modéliser le comportement dynamique des systèmes à événements discrets comme les systèmes manufacturiers, les systèmes de télécommunications, les réseaux de transport. D'un côté, leur représentation graphique permet de visualiser d'une manière naturelle le parallélisme, la synchronisation, le partage des ressources, les choix. De l'autre, leur représentation mathématique permet d'établir les équations d'état, à partir desquelles il est possible d'évaluer les propriétés du modèle et de les comparer avec le comportement du système modélisé [MD08]. Un critère important qui peut conduire au choix des réseaux de Pétri est le besoin de pouvoir établir des preuves formelles de propriétés (vivacité ou atteignabilité par exemple) devant absolument être satisfaites par l'application. La possibilité d'effectuer des simulations permet en outre de mesurer les performances que l'on peut espérer du système. Enfin, la représentation graphique d'un modèle et les règles de fonctionnement en font un outil essentiel pour la communication entre les différents acteurs du projet. Les paragraphes suivants proposent quelques propriétés pouvant être démontrées grâce à cet outil, ainsi que les méthodes d'analyse permettant de les vérifier.[Petri62]

Les réseaux de Pétri offrent un outil formel et une bonne représentation graphique qui permettent de modéliser et d'analyser les systèmes discrets particulièrement les systèmes concurrents et parallèles. La facette graphique des réseaux de Pétri, nous aide à comprendre aisément le système modélisé. Par ailleurs, leur puissance d'expression mathématique permet de simuler des activités dynamiques et concurrentes. L'intérêt majeur de ces réseaux réside dans leurs possibilité d'analyser les systèmes modélisés, grâce aux modèles de graphes et aux équations algébriques.

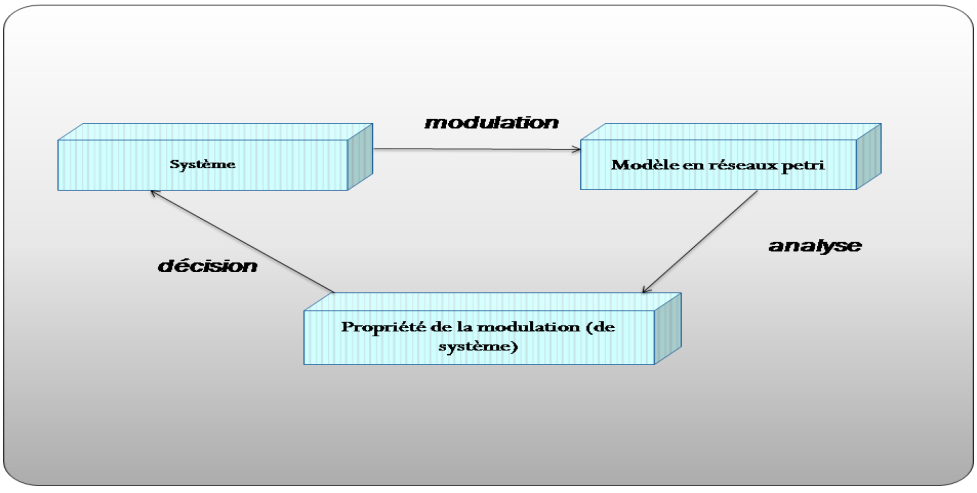


Figure 1.3 – *Méthode générale de modélisation et d'analyse basée sur les réseaux de Pétri.*

1.3.1 Notions de Base :

Les types de variables

En général lorsqu'on construit un modèle mathématique d'un système, on classe les variables en deux catégories :**Les variables continues** :sont des variables qui prennent leurs valeurs sur le domaine des réels.
Les variables discrètes :sont des variables qui prennent leurs valeurs sur un domaine dénombrable comme l'ensemble des entiers naturels, ou bien sur des ensembles dont le nombre l'élément est fini.

Concepts de base des réseaux de Pétri :

Définitions informelles :

Les réseaux de Pétri sont utilisés afin de modéliser le comportement dynamique de systèmes discrets. Un réseau de Pétri est un graphe orienté,pondéré et biparti, c'est à dire comprenant deux types de nœud : les places et les transitions. Les arcs peuvent relier une place à une transition ou une transition à une place.[RDP99]

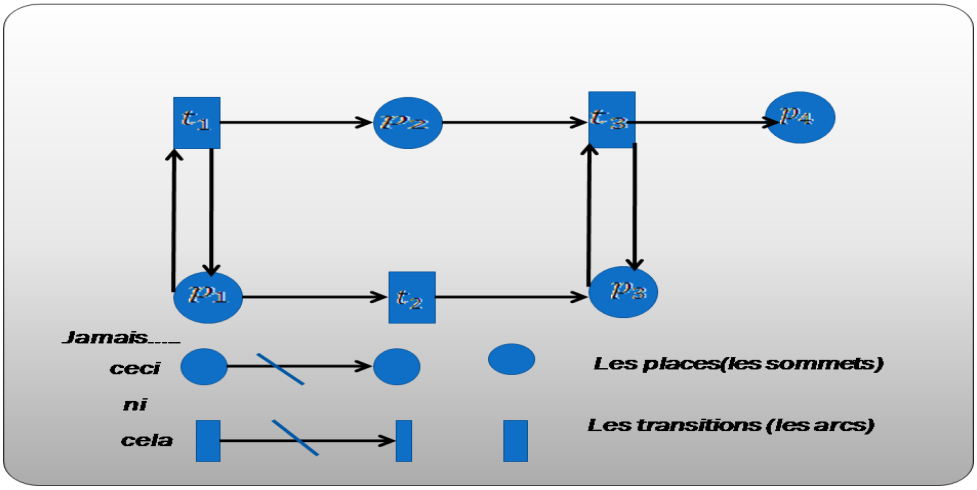


Figure 1.4 – *Exemple de réseau de Pétri .*

Définitions formelles :

- Façon plus formelle, un RdP peut-être défini par un 4-uplet <*P*, *T*, *Pr*, *Post*> tel que :
- *p*= P1, *P*2, ..., *Pn* :L est un ensemble fini et non vide de places ;
- *T*= T1,T2, ...,Tm : L est un ensemble fini et non vide de transitions ;
- *Pr* : $P \times T0,1$ est l'application d'incidence avant ;
- *Post* : P × T o, 1 est l'application d'incidence arrière.
- Pré(*Pi*, *Tj*) est le poids de l'arc (orienté) reliant la place Pi à la transition *Tj* ; ce poids vaut 1 si l'arc existe et o sinon.
- *Post* (*Pi* ,*Tj*) est le poids de l'arc (orienté) reliant la transition *Tj* à la place *Pi* .

Marquage des places :

Les places sont marquées par des jetons (points noirs) qui vont circuler dans les places selon certaines règles (définies ci-dessous). Cette circulation symbolise l'évolution dynamique du système. Le marquage initial (celui indiqué sur le dessin) donne la position initiale des jetons.

Règles de fonctionnement et circulation des jetons : Pour qu'une transition puisse être activée, la présence d'un jeton au moins est requise dans chaque place située en amont de la transition. L'activation (le tir) de la transition a pour effet de prélever ces jetons des places amont et de rajouter dans chaque place aval un nouveau jeton.
De façon plus formelle, le franchissement (tir) d'une transition Tj ne peut s'effectuer que si le marquage de chacune des places Pi directement en amont de cette transition est tel que :m Pi >= Pré (Pi ,Tj) (condition nécessaire).

Le franchissement (tir) de Tj consiste à retirer Pré(Pi ,Tj) jetons dans chacune des places directement en amont de Tj et à ajouter Post (Pk ,Tj)jetons dans chacune des places Pk directement en aval de Tj .
Modélisation de la concurrence (ou logique) et de la synchronisation (et logique)Concurrence à la fourniture de jetons dans une place :

En général lorsqu'on construit un modèle mathématique d'un système, on classe les variables en deux catégories :C'est la convergence d'arcs sur une place .
Concurrence à la consommation des jetons d'une place : C'est la divergence d'arcs à partir d'une place (voir figure b suivante).
Ce conflit structurel doit être arbitré par une règle de priorité quelconque lorsque le conflit est effectif (c'est-à-dire lorsque les transitions aval en compétition pourraient effectivement être activées).
Ne pas arbitrer un conflit effectif fait que le comportement du système n'est pas entièrement spécifié.
Synchronisation dans la consommation de jetons de plusieurs places :C'est la convergence de plusieurs arcs sur une transition (voir figure c suivante) .
Synchronisation dans la fourniture de jetons à plusieurs places : C'est la divergence d'arcs à partir d'une transition (voir figure d suivante).

exemple :

– Exemple :La figure suivante représente le graphe de Pétri d'une pièce comprenant une porte-fenêtre et une climatisation. L'objectif de ce système étant d'obtenir une température agréable dans la pièce.

1.3.1.1 les intérêts de réseaux de pétri :

Lors de son évolution, le réseau de Pétri parcourt séquentielle-ment les différents états du système modélisé. Cela peut être mis à profit pour :
Analyser en détail le comportement séquentiel du système.
Identifier les états en vue de la réalisation d'un graphe d'état (Markov).
Rechercher les blocages.
Rechercher les « conflits » (plusieurs transitions valides en même temps avec un devenir différent du réseau selon l'ordre du tir) .
L'utilisation des réseaux de Pétri pour l'identification des divers états du système en

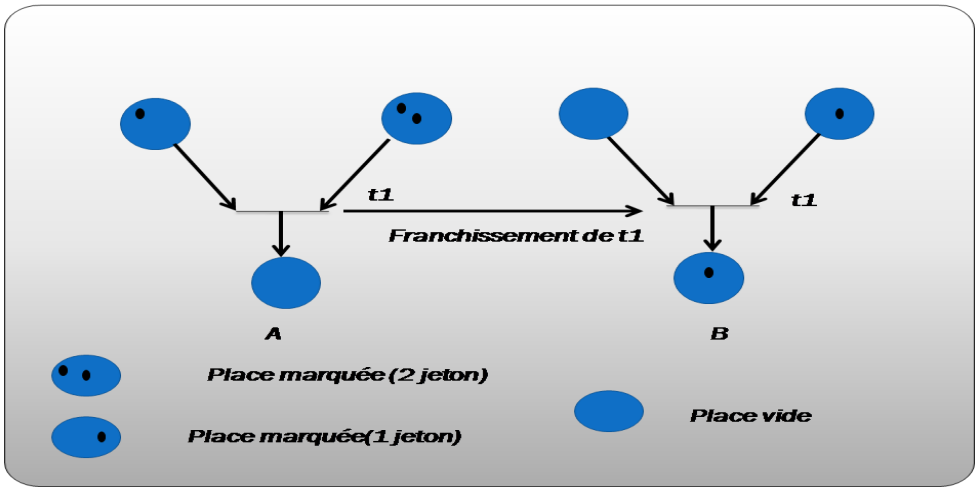


Figure 1.5 – *(A) RdP marqué avant franchissement de T1 (B) RdP après franchissement de T1. .*

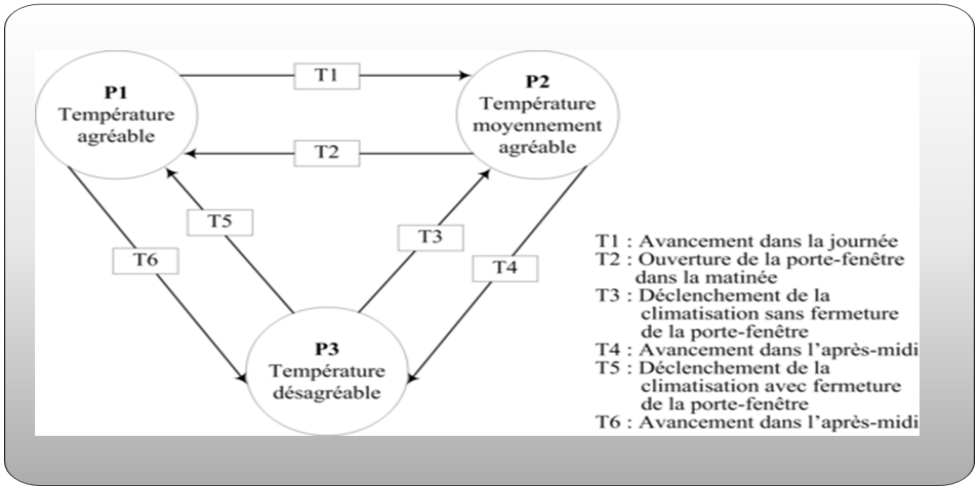


Figure 1.6 – *Réseau de Pétri d'une pièce contenant une porte-fenêtre et une climatisation.*

vue de générer le processus de Markov équivalent est l'une des applications les plus courantes dans le cadre de la sûreté de fonctionnement.

1.3.1.2 les limites de réseaux de pétri :

– – – – – – Rechercher les états non accessibles.

– Bien que le nombre d'états engendrés par un réseau de Pétri soit dénombrable, il n'est pas forcément fini. Dès que le système étudié est complexe, le nombre d'états engendrés est important et il n'est plus possible de tous les identifier. Cette méthode ne peut être utilisée pour générer un graphe de Markov que lorsque le nombre d'états n'est pas trop grand (jusqu'à quelques milliers). La représentation par réseaux de Pétri étant beaucoup plus condensée que celle des processus de Markov, elle est alors plus facile à maîtriser.

1.4 L'EXPRESSION RATIONNELLE :

– Une expression rationnelle peut modéliser tout phénomène régulier dans le tempe ou dans l'espace finie ou infini.

– Une expression rationnelle sont largement utilisées dans plusieurs domaines tel que recherche de motifs reconnaissance d'écriture (parole, bioinformatique, système exploitation) Force d'expression[EXP03].

– Un phénomène élémentaire constitue une entité indivisible est modélisée par une expression rationnelle.

–
– 0 : est une expression rationnelle (vide).

–
– 1 : est une expression rationnelle (comportement null).

–
– E : est une expression rationnelle.

–
– $EetF$: sont deux expression rationnelle alors E+F est expression rationnelle.

–
– $E.F$ est une expression rationnelle $E*$ est une expression rationnelle.

Comment vérifier ? Théorème : pour chaque expression il existe un automate équivalent automate de$trampson$.

1.4.1 Construction d'automates finis à partir des expressions rationnelles :

Nous allons voir comment il est possible de construire un automate fini non-déterministe qui reconnait une expression régulière. Cette méthode permet de produire des briques d'automates ayant un seul état initial mais aussi un seul état final.
Les expressions régulières sont définies de manière inductive (concaténation, union et étoile de $KLEENE$)[EXP04]. Commençons par voir comment on peut construireles automates des briques de base :
Nous pouvons maintenant construire les automates qui vont reconnaître des expressions plus complexes :

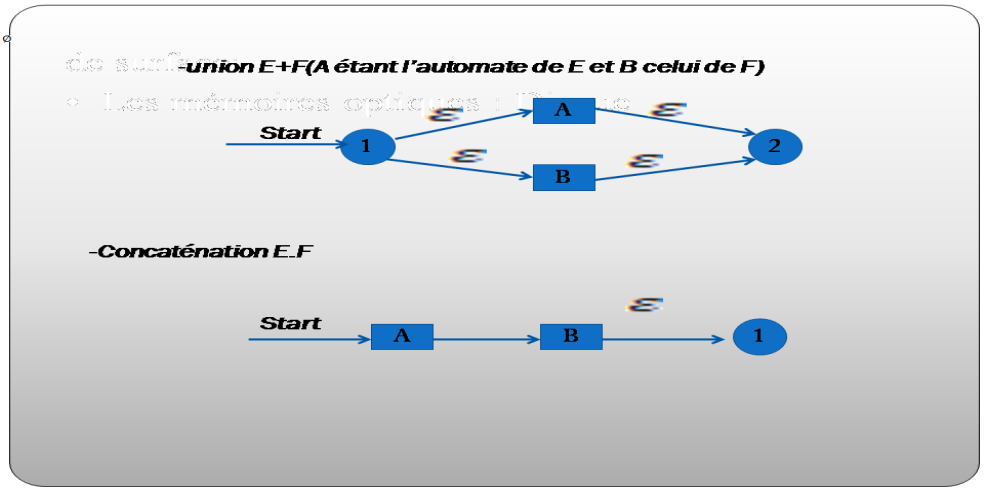


Figure 1.7 – *(A) RdP marqué avant franchissement de T1 (B) RdP après franchissement de T1. .*

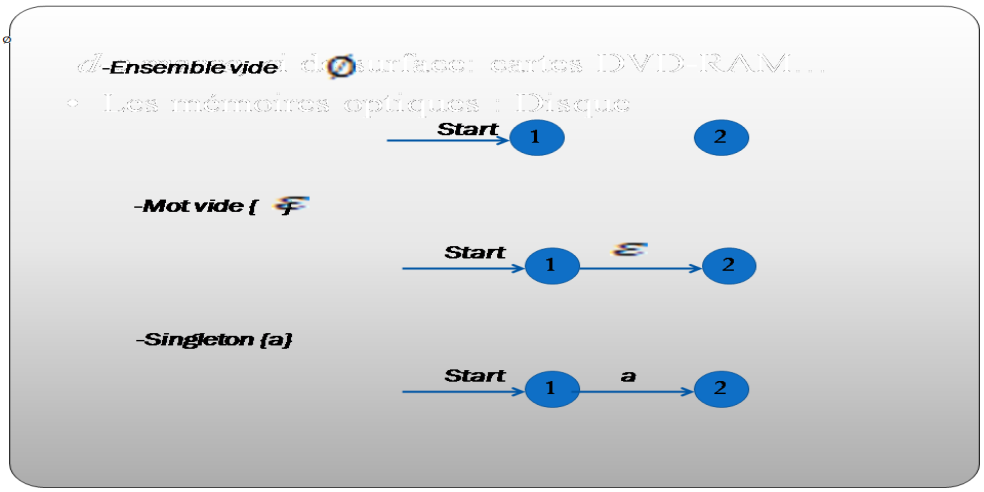
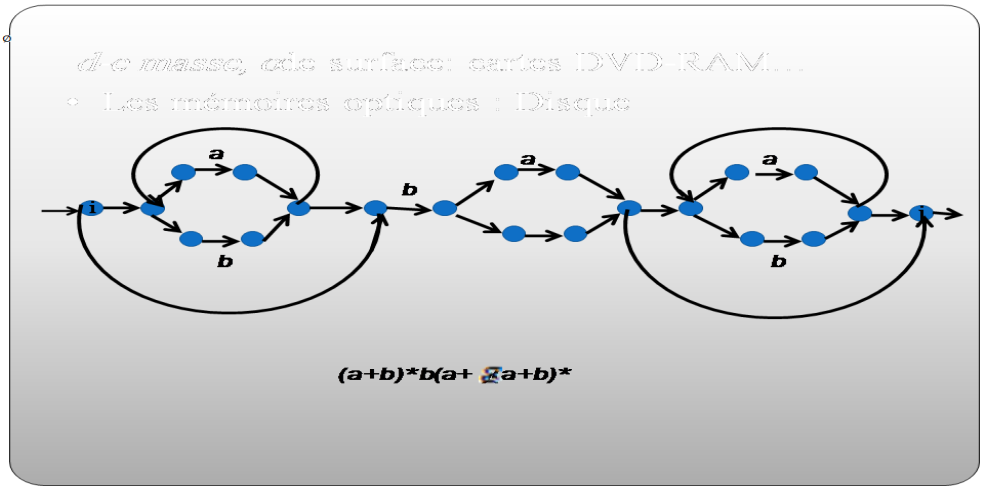


Figure 1.8 – *construire les automates qui vont reconnaître des expressions plus complexes avec l'expression rationelle.*



1

1.5 Méthodes et outils d'analyse des UPPAAL et SPIN :

1.5.1 UPPAAL

1.5.1.1 Présentation :

Principes *UPPAAl* est un outil de modélisation, simulation et vérification de systèmes temps réels. Il est basé sur la théorie des automates temporisés étendus par des variables et communication par canaux, et est composé de trois parties : un langage de modélisation, un simulateur, et un vérificateur de modèle.
. et le simulateur permet de visualiser le comportement du système.

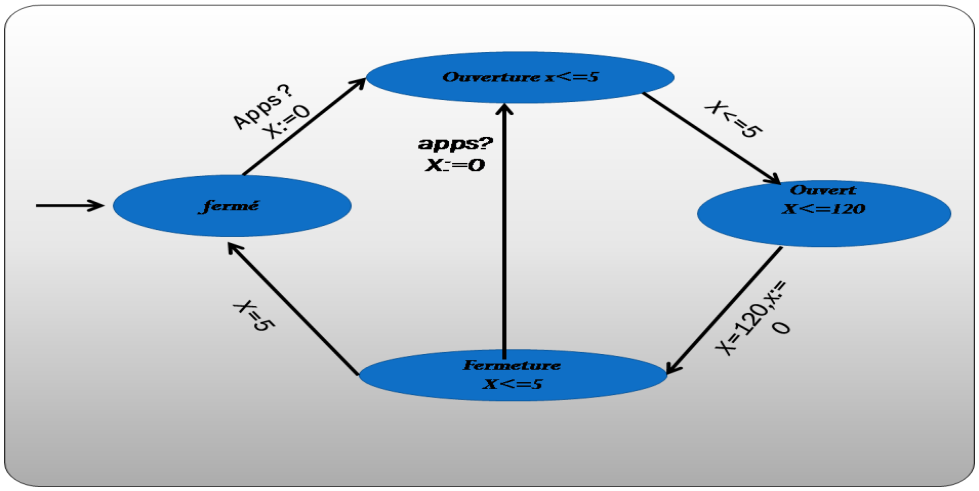
Vérification *UPPAAL* est conçu pour vérifier les propriétés d'invariance et d'atteignabilité des états, en explorant systématiquement l'espace d'états des automates du modèle. Une trace de diagnostic est générée, qui indique la cause du succès ou de l'échec de la vérification effectué.[UPP13]

Les principales caractéristiques *d'Uppaal* **:**

Nous donnons rapidement les principales caractéristiques *d'Uppaal* pour une présentation des plus récents développements, qui seront illustrées dans les paragraphes suivants : Pour *Uppaal*, un modèle consiste en un ensemble d'automates temporises, qui communiquent par une synchronisation binaire, utilisant des canaux et une syntaxe du type émission/réception. Ainsi sur le canal c, un émetteur envoie le signal c ! Et un récepteur se synchronise avec lui par le signal complémentaire c ? [UPP03]. Les automates temporises *d'uppaal* sont une variante des modèles originaux. Ce sont évoluent de manière synchrone avec le temps, et des variables entières discrètes invariant, qui doit être satisfaite pendant toute la durée passée dans cet état.une transition de l'automate est étiquetée par :
Exemple : L'automate temporisé de la figure (a) décrit une barrière qui s'ouvre(en moins de 5 secondes) a l'approche d'une voiture et reste ouverte pendant 120 secondes avant de se reformer. L'automate part de l'état initial fermé. lorsqu'une voiture s'approche de la barrière (signal *app* ?) ,il passe dans l'état ouverture pour arriver a l'état ouvert (en remettant l'horloge x a zero) après un délai inférieurs ou égal a 5 secondes. Dans cet état le temps s'écoule et l'approche d'autre voitures (*app* ?) remet a chaque fois l'horloge x a zero. Lorsque x vaut 120, la barrière commence a se refermer (l'automate va dans l'état fermeture).elle se rabat après 5 secondes (l'automate revient a l'état fermé) , a moins qu'une autre voiture ne s'approche.

1.5.2 SPIN Model Checker

Un modèle checker, l'outil logiciel qui effectue la vérification de modèle, examine tous les scénarios possibles du système de manière systématique. De cette façon, il peut être démontré qu'un modèle de système donné répond véritablement à une certaine propriété[SPIN03].



SPIN est un modèle checker basé sur l'approche automate développé par*Gerard J.Holzmann* pour vérifier les protocoles de communication. Il a été largement répandu dans les industries qui construisent des systèmes critiques. Les modèles devant être vérifiés par SPIN sont écrits en *Promela*, un langage qui décrit le comportement du système. Les propriétés à vérifier doivent être formulées en Logique Temporelle Linéaire (LTL)[SPIN97]. Etant données la spécification formelle du comportement du système et la spécification d'une propriété en logique temporelle, SPIN modèle checker vérifie si cette propriété est satisfaite. En effet, le mode de vérification de SPIN détermine si le modèle proposé en *Promela* satisfait ou non une propriétéLTL. ceci est effectué en menant, si nécessaire, une vérification exhaustive i.e. en parcourant l'ensemble des exécutions possibles du système. Le principe de vérification de SPIN est qu'il transforme, à partir des propriétés spécifiées, leur négation en automates de*Bchi* . SPIN calcule ensuite le produit synchronisé de cet automate avec celui du système. Le résultat est encore un automate de *Bchi* dont on vérifie si le langage est vide ou non. S'il est vide, alors il n'existe aucune exécution violant la formule de départ, sinon il accepte au moins une exécution (ou mot du langage de l'automate, chaque lettre du mot étant représentée par une action du système ou de la propriété)[SPIN96]. Un automate de *Bchi* accepte un mot (et donc son langage est non-vide) si et seulement si l'exécution du système passe infiniment souvent par un ou plusieurs des états acceptants de l'automate. I.e. qu'il existe un cycle accessible depuis l'état initial qui comporte un état acceptant. Pour prouver que toutes les exécutions du système sont valides vis-à-vis de la propriété à vérifier, il suffit de prouver qu'il n'existe pas de cycle acceptant accessible depuis l'état initial. Si ce mot existe, il va permettre de trouver un ensemble de transitions des automates du système qui viole la propriété. Un mot qui viole la propriété s'appelle un contre-exemple. L'ensemble des transitions pour lesquelles la propriété n'est pas vérifiée forment les contre-exemples qui peuvent être générés par SPIN. Ces contre-exemples sont censés permettre à l'utilisateur de modifier par la suite son modèle (ou la propriété si elle a été mal exprimée), afin de corriger les erreurs éventuelles[SPIN66].

Conclusion

Malgré la grande importance attribuée par la communauté scientifique aux modèles de vérification des programmes, cette technologie reste encore très jeune et difficilement applicable à des programmes réels.. Les outils disponibles sont pour la plupart de nature académique et leur utilisation nécessite un haut niveau d'expertise et une connaissance profonde des formalismes utilisés, p.ex., la logique temporelle. A notre connaissance, ces

outils n'ont jamais été utilisés par des non spécialistes dans des projets de grande envergure, alors que les outils de test et de preuve automatique se sont largement répandus et sont maintenant couramment utilisés.
Cette approche présente deux avantages. D'une part, elle permet de trouver et de corriger les bogues plus tôt et limite ainsi leur coût. d'autre part, il se peut qu'à une certaine étape le modèle de vérification « modèle checker » ne soit plus en mesure de vérifier le programme en raison du problème d'explosion combinatoire. Il est alors très appréciable pour l'utilisateur de savoir que son programme est valide jusqu'à un certain point et que certaines modifications sont potentiellement source d'erreurs.

Généralités sur réseaux 2

Sommaire

2.1 Introduction 18
2.2 Les réseaux 18
2.2.1 Que signifie le réseau 18
2.2.2 Les réseaux mobiles : 19
2.2.2.1 Les réseaux mobiles avec infrastructures : 19
2.2.2.2 Les réseaux mobiles sans infrastructures (AD HOC) : 19
2.2.2.3 Définition d'un réseau AD HOC : 20
2.2.2.4 Les caractéristiques des réseaux *ADHOC* : 20
2.2.2.5 Les domaines d'applications des réseaux*ADHOC* : 21
2.2.2.6 Définition d'un routage : 21
2.2.2.7 Définition des Protocoles : 22
2.2.2.8 Classification des protocoles de routage : 22
2.2.3 La simulation : 23
2.2.3.1 Méthodologie générale : 24
2.2.3.2 Le simulateur avec NS-2 : 25
2.2.3.3 Définition : 25
2.2.3.4 Les étapes de simulation : 25
2.2.3.5 La réalisation d'une simulation 25
2.3 Conclusion 25
Conclusion 25

Dans ce chapitre, nous allons présenter la validation des protocoles réseaux et les caractéristiques inhérentes ainsi que quelques domaines d'application de ces réseaux .

2.1 Introduction

Le développement des réseaux sans fil et des réseaux mobiles ouvre une nouvelle ère dans le domaine des télécommunications, aujourd'hui avec l'évènement du WIF et de la téléphonie mobile, et demain avec les technologies des réseaux. A l'heure actuelle, l'essentiel de l'effort de recherche porte sur les réseaux mobiles et sur les problématiques liées aux réseaux, car ces technologies couvrent tout le domaine lié au monde des réseaux (routage, gestion de la mobilité, auto-configurabilité, qualité de service, redondance des architectures).
La validation de protocoles de communication utilise depuis longtemps plusieurs méthodes complémentaires : simulation, test et expérimentation sur prototypes, et méthodes formelles. Ces dernières offrent un complément précieux en assurant une plus grande confiance dans les résultats et en offrant des possibilités de validation impossible en simulation ou sur prototype.
La stratégie (ou le protocole) de routage est utilisée dans le but de découvrir les chemins qui existent entre les nœud. Le but principal d'une telle stratégie est rétablissement de routes qui soient correctes et efficaces entre une paire quelconque d'unités, ce qui assure l'échange des messages d'une manière continue. A cause des limitations des réseaux , la construction des routes doit être faite avec un minimum de contrôle et de consommation de la bande passante et de l'énergie. Suivant la manière de création et de maintenance de routes lors de l'acheminement des données, les protocoles de routage peuvent être séparés en deux catégories, les protocoles pro-actifs et les protocoles réactifs.

En effet, les expérimentations sur prototypes sont limitées par le matériel nécessaire et les conditions pouvant être reproduites. La simulation permet l'exécution de gros systèmes et l'introduction de conditions simulées (surcharge d'un lien réseau, etc.), mais elle ne permet pas l'étude exhaustive de toutes les exécutions du système : il est alors impossible d'être certain qu'une propriété sera toujours vérifiée.
Or dans certains contextes, comme les systèmes critiques, cette notion de confiance dans la validation est primordiale. Les méthodes formelles sont donc indispensables pour compléter la validation de nombreux protocoles. En particulier, *lemodlechecking* qui permet la vérification de propriétés exprimées en logique temporelle sur l'ensemble des états accessibles d'un modèle du système est très utilisé dans le domaine de la validation des systèmes temps réel communicants Cependant, il n'existe à ce jour que peu de travaux traitant de la validation formelle dans le domaine des réseaux.
.

2.2 Les réseaux

Pour résoudre les problèmes liés à un système donné, on fait appel à des définitions de base.

2.2.1 Que signifie le réseau

Un réseau en général est le résultat de la connexion de plusieurs machines entre elles, afin que les utilisateurs et les applications qui fonctionnent sur ces dernières puissent échanger des informations (ressources, applications ou matériels, connexion à internet, etc.)

Le terme réseau en fonction de son contexte peut désigner plusieurs choses.il peut désigner l'ensemble des machines, ou l'infrastructure informatique d'une organisation avec les protocoles qui sont utilisés, ce qui est le cas lorsque l'on parle de internet. Le terme réseau peut également être utilisé pour décrire la façon dont les machines d'un site sur un réseau local (réseau Ethernet, Token Ring, étoile, bus,etc).

Aujourd'hui, les réseaux se retrouvent a l'échelle planétaire. Le besoin d'échange de l'information est en pleine évolution. Pour se rendre compte de ce problème il suffit de voir comment sécuriser ou vérifier l'échange des informations entre les machines (retours, Switch....).

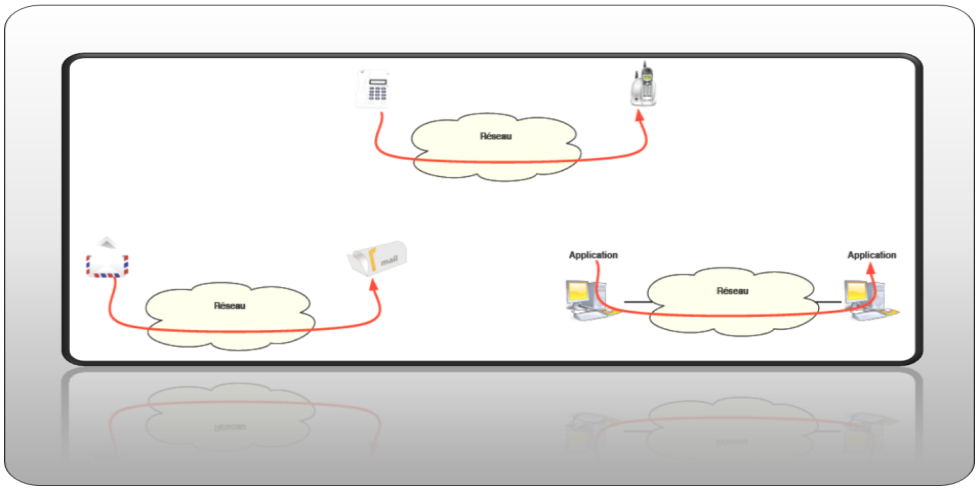


Figure 2.1 – *les étapes de la vérification de modèles..*

2.2.2 Les réseaux mobiles :

Le réseau mobile est un ensemble de nuds mobiles qui permet aux usagers d'accéder à l'information indépendamment de leurs positions géographiques, et sans utiliser une infrastructure câblée.
Les réseaux mobiles (ou sans fil), peuvent être divisés en deux classes :

- Des réseaux avec infrastructure.
- Des réseaux sans infrastructure..

2.2.2.1 Les réseaux mobiles avec infrastructures :

2.2.2.2 Les réseaux mobiles sans infrastructures (AD HOC) :

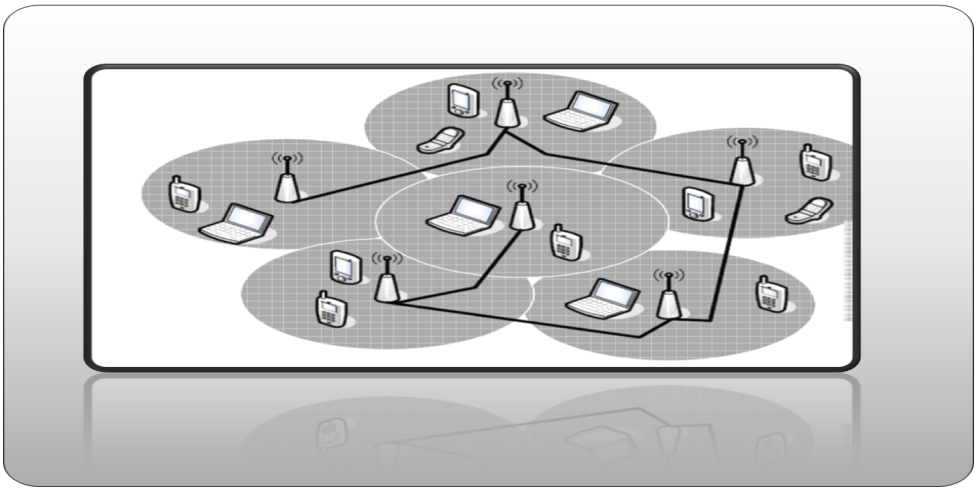


Figure 2.2 – *Le modèle des réseaux mobiles avec infrastructure.*

2.2.2.3 Définition d'un réseau AD HOC :

Un réseau mobile AD HOC, appelé généralement MANET (Mobile AD HOC NETwork), est un ensemble d'unités mobiles qui se déplacent dans un territoire quelconque. Le seul moyen de communication est l'utilisation des ondes radio qui se propagent entre Les différents nuds mobiles, sans l'aide d'une infrastructure préexistante ou d'une administration centralisée.[30]

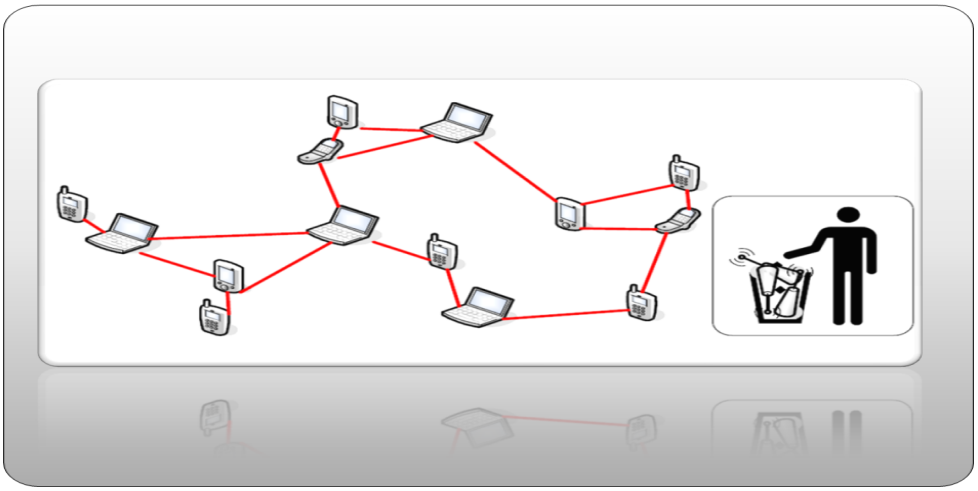


Figure 2.3 – *Le modèle des réseaux mobiles sans infrastructure..*

2.2.2.4 Les caractéristiques des réseaux *ADHOC* :

Les réseaux *ADHOC* présentent de nombreuses caractéristiques dont certaines leur sont bien spécifiques et les différencient des réseaux mobiles classiques.

– **Une topologie dynamique :**les unités mobiles du réseau, se déplacent d'une façon libre et arbitraire. Par conséquent, la topologie du réseau peut changer à des instants

imprévisibles d'une manière rapide et aléatoire.

2.2.2.5 Les domaines d'applications des réseaux*ADHOC* :

On utilise La technologie *ADHOC* dans plusieurs applications civiles est également apparue.Parmi les applications des réseaux *ADHOC* , on peut distinguer les domaines suivants :

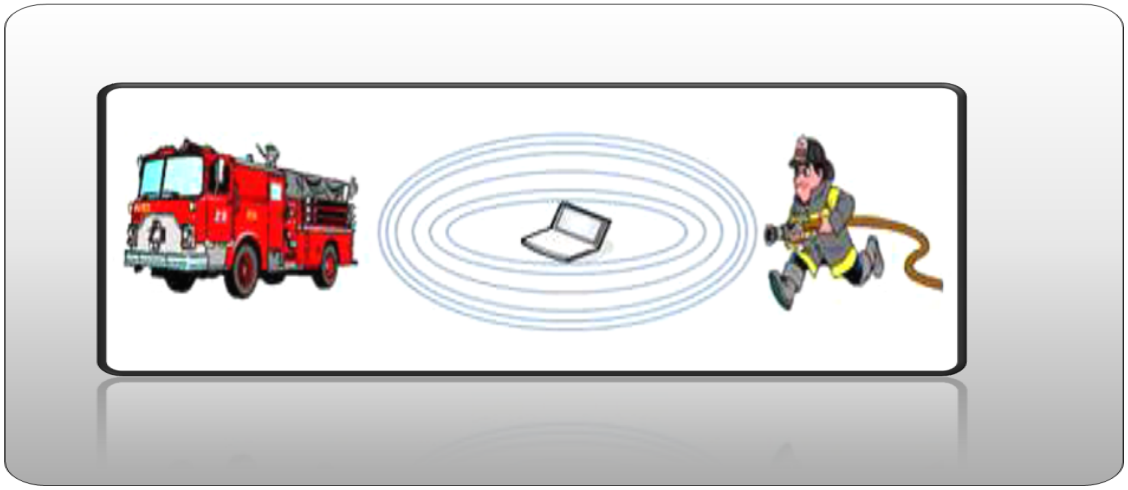


Figure 2.4 – *Application de secours des réseaux AD HOC.*

– Services d'urgence : opération de recherche et de secours des personnes, tremblement de terre, feux, dans le but de remplacer l'infrastructure filaire(figure).

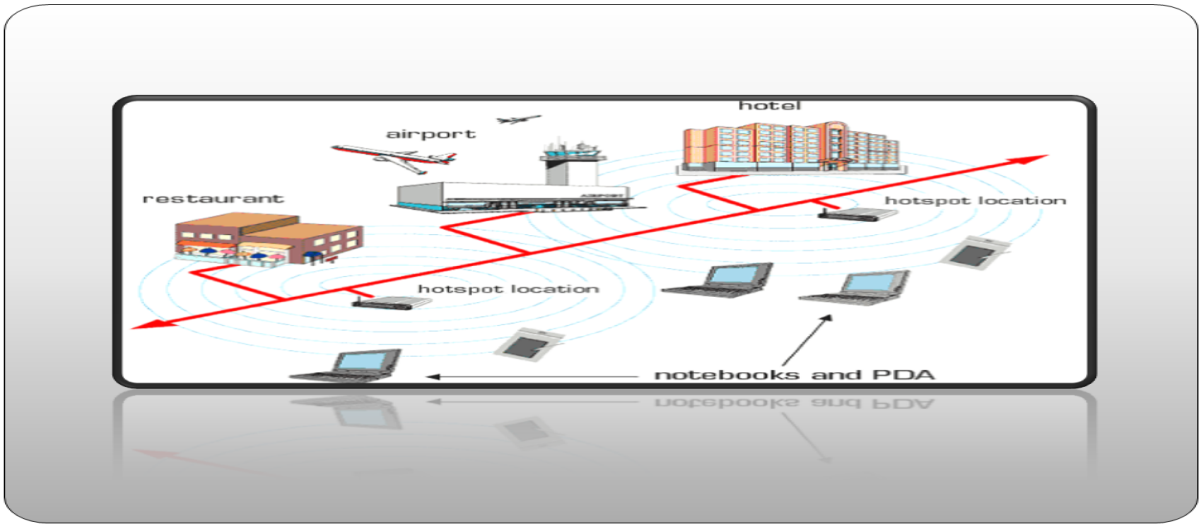


Figure 2.5 – *Applications commerciales des réseaux AD HOC.*

2.2.2.6 Définition d'un routage :

Méthode et protocoles qui permettent de trouver un chemin optimal pour échanger des informations entre machines sur un réseau (au sens d'un certain critère de performance (bande passante, délai et la qualité de donnée, etc...)).

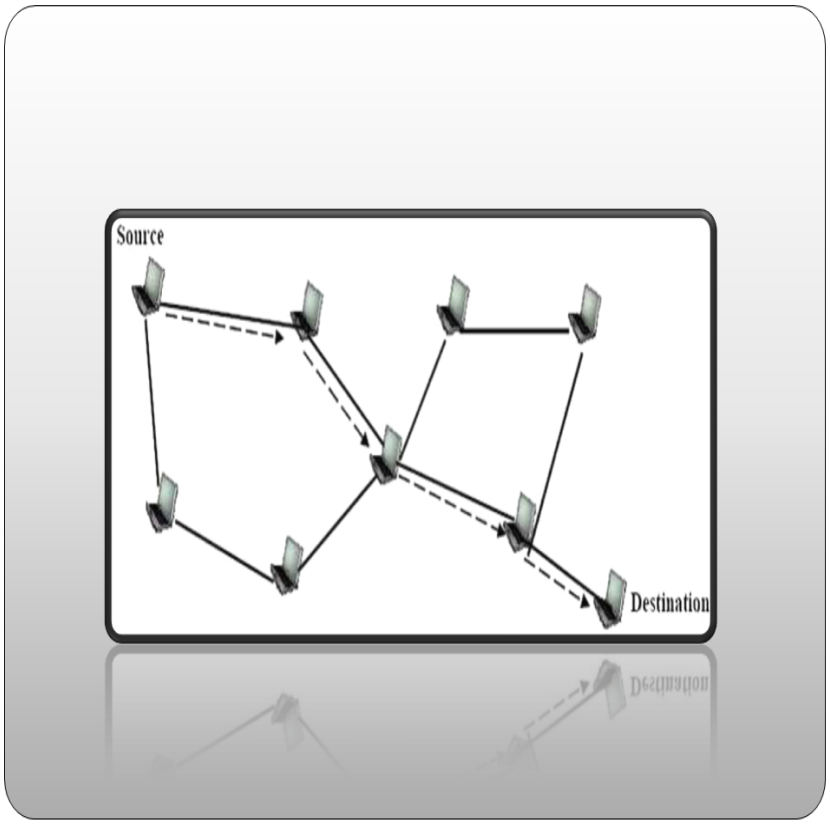


Figure 2.6 – *Le chemin utilisé dans le routage entre la source et la destination..*

2.2.2.7 Définition des Protocoles :

Un protocole est une méthode standard qui permet la communication entre les machines :

– Ensemble de règles et de procédures à respecter pour émettre et recevoir des données sur le réseau.

– TCP/IP : Transmission$ControlProtocol$ / $InternetProtocol$.

– Défini la norme de communication, (en fait un ensemble de protocoles) des ordinateurs reliés à Internet.Va contenir les protocoles($HTTP$,FTP,$SMTP$,etc)

2.2.2.8 Classification des protocoles de routage :

Les protocoles de routage peuvent être partagés en trois classes, les protocoles *proactifs*, les protocoles réactifs et les protocoles hybrides. Comme il est illustré dans la figure sui-

vants.

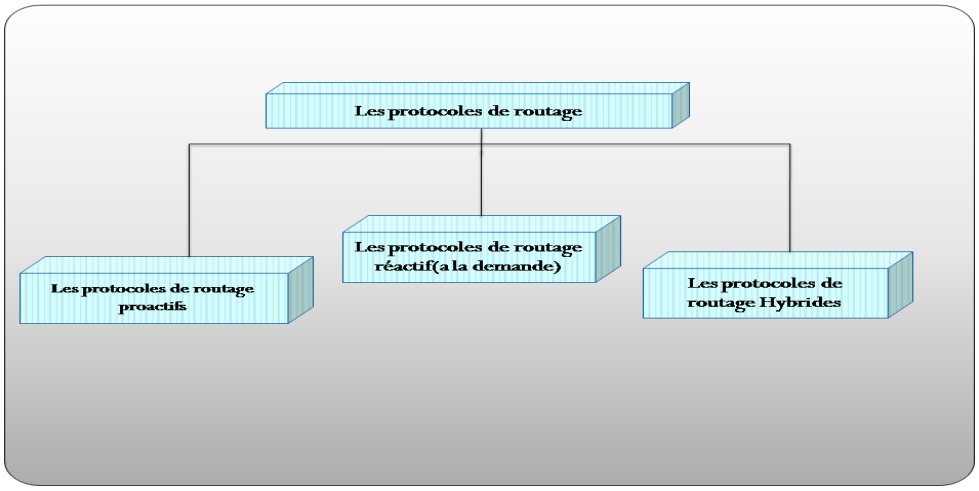


Figure 2.7 – *Classification des protocoles de routage.*

2.2.2.8.a Les protocoles de routage $proactifs$:

Les protocoles de routage$proactifs$ exigent une mise à jour périodique des données de routage qui doit être diffusée par les différents nuds de routage du réseau. Le protocole le plus connus dans cette classe est $OLSR$ ($Optimizedlinkstateroutingprotocol$).

2.2.2.8.b Les protocoles de routage réactifs (à la demande)

Les protocoles de routage appartenant à cette catégorie, créent et maintiennent les routes selon les besoins. Lorsque le réseau a besoin d'une route, une procédure de découverte globale de routes est lancée, et cela dans le but d'obtenir une information spécifiée, inconnue au préalable.$AODV(ADHOCOn-demandDistanceVector)$, DSR ($DynamicSourceRouting$), sont les plus connus dans cette classe.

2.2.2.8.c Les protocoles de routage hybrides :

Dans ce type de protocole, on peut garder la connaissance locale de la topologie jusqu'à un nombre prédéfini (à priori petit) de sauts par un échange périodique de trame de contrôle, autrement dit par une technique $proactife$. Les routes vers des nuds plus lointains sont obtenues par schéma réactif, c'est-à-dire par l'utilisation de paquets, et de requêtes en diffusion. Le protocole ZRP ($ZoneRoutinierProtocol$) est le plus connu dans cette classe.

2.2.3 La simulation :

La simulation est un processus qui consiste à :
– Concevoir un modèle du système (réel) étudié.

– Mener des expérimentations sur ce modèle (et non pas des calculs).

– Interpréter les observations fournies par le déroulement du modèle et formuler des décisions relatives au système.

2.2.3.1 Méthodologie générale :

On distingue classiquement quatre phases distinctes : La modélisation (représenter le comportement du système), la programmation, l'expérimentation et l'interprétation des résultats (accompagnée d'actions).

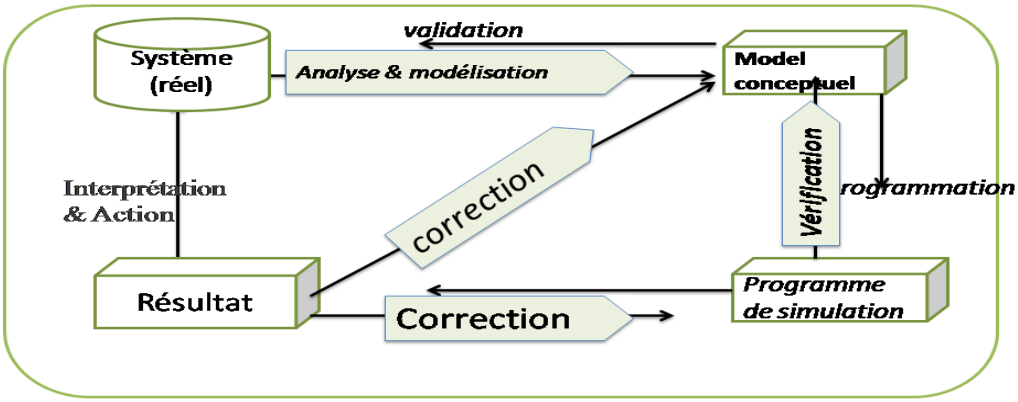


Figure 2.8 – *Méthodologie d'une simulation.*

– **Modèle conceptuel :** Le modèle n'est qu'une approximation du système, il est conditionné par l'objectif de l'étude.
– **Expérimentation :** Il s'agit de construire des théories, ou hypothèses, qui prennent en compte le comportement observé.
– passage du système au modèle conceptuel est une étape essentielle pour la simulation.

– utilise une modélisation conceptuelle par réseaux de Pétri.

– passage du modèle conceptuel au modèle/programme de simulation se fait en Utilisant le langage NS-2 ; ce langage de simulation permet également d'extraire des résultats issus de différentes expérimentations.

2.2.3.2 Le simulateur avec NS-2 :

2.2.3.3 Définition :

Simulateur NS2 (Network Simulator) est un simulateur des événements discretsà pour but de faire de la recherche de réseau. NS2 fournit un soutien substantiel pour simuler le TCP, UDP, IP et le routage, le multicast et les protocoles dans les réseaux filaires et les réseaux sans fils (local et satellite) avec un support de mobilité des nuds. NS2 est écrit en C++, et il y a une version orientée-objet de TCL, qui sappelle OTcl .
NS2 est un outil très puissant. Cependant quaujourdhui, il a développé les versions NS2 2.33, NS2 2.34 et NS2 2.35, NS2 ne soutient pas complètement les réseaux sans fils surtout les protocoles de routage multicanaux, multi-interface. NS2 a soutenu les types de protocole de routage de réseau fils : le routage unicast, le routage multicast, le routage hiérarchique. Pour un type de protocole de routage, NS2 fournit les fonctions API pour soutenir les utilisateurs qui simulent facilement le réseau. De nos jours, NS2 a soutenu 4 protocoles de routage dans le réseau mobile : DSDV, AODV, TORA et DSR, mais le soutien est élémentaire.

2.2.3.4 Les étapes de simulation :

La simulation avec NS-2 passe en général par trois phases pour crée des topologies réseaux :

– **Définition de la topologie du réseau :** on crée les nuds, avec les caractéristiques de chacun, ainsi que les liens entre les nuds. On peut définir sur chaque lien, le délai de transmission, la bande passante, le fait qu'il soit simplexe ou duplexe et le type de file d'attente se trouvant à son extrémité.

– **Spécification du scénario de la simulation :** l'utilisateur spécifie les différents agents de la communication qui vont agir pendant la simulation. Il spécifie aussi la succession des différentes opérations (à l'instant $t1$, envoi des données, à l'instant $t2$ arrêt d'émission).

– **Exploitation des résultats :** cette dernière phase consiste en un recueil des statistiques de la simulation. Ces dernières peuvent être exploitées directement par$NS-2$ en faisant appel aux outils qui l'accompagnent, ou bien elles seront archivées pour une utilisation ultérieure par d'autres outils de traitements statistiques.

2.2.3.5 La réalisation d'une simulation

On peut résumer les étapes de réalisation d'une simulation dans la figure suivants .

2.3 Conclusion

On a détaillé les définitions des différentes étapes d'étude de cas (réseau, simulateur, les protocoles...), ainsi que problèmes rencontré. On a pu constater que le rôle des protocoles de la couche liaison de données est primordial. En effet, ils sont charges de garantir la fiabilité de transfert de données dans un réseau de communication. Leur mise au point

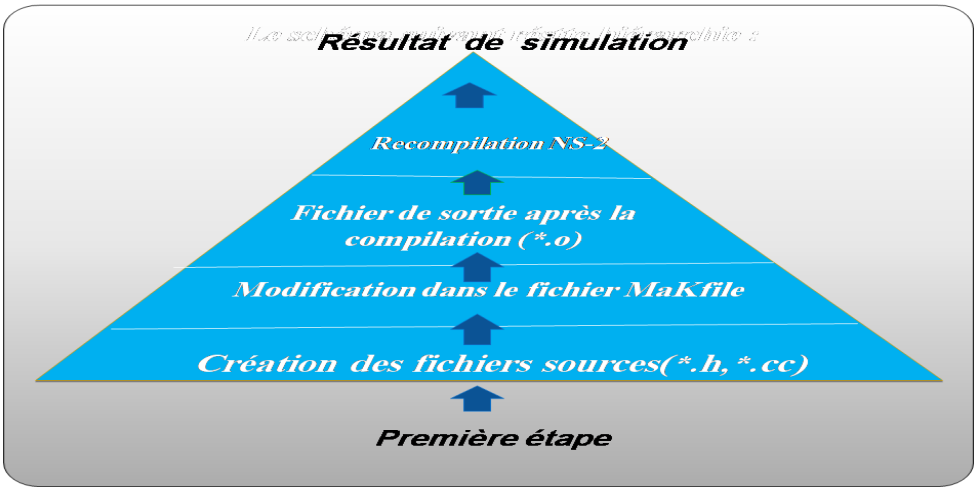


Figure 2.9 – *Les étapes de simulation.*

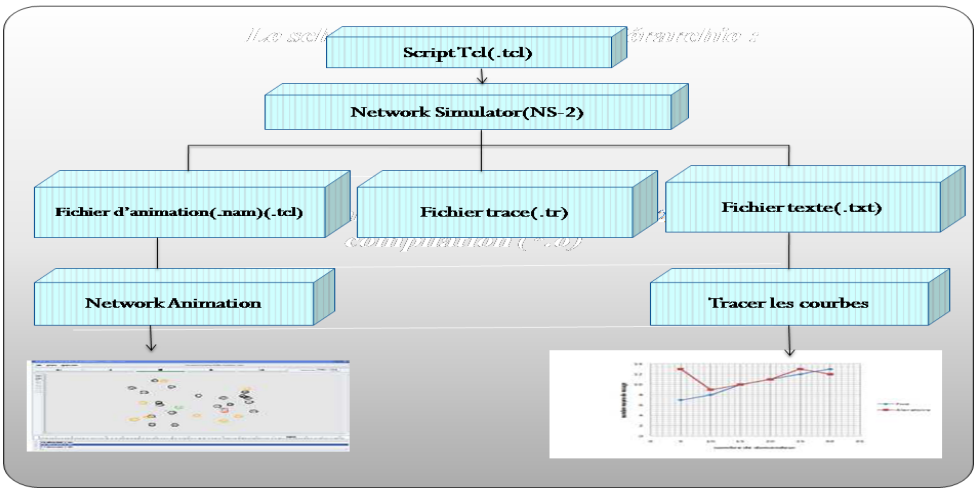


Figure 2.10 – *La réalisation d'une simulation.*

est un problème complexe pour lequel il est recommande d'utiliser des techniques de spécification et de vérification formelles. L'objectif de la suite du travail est de fournir une description formelle au protocole réseaux pour pouvoir ensuite vérifier formellement des propriétés sur le modèle obtenu, et cela en combinant les deux approches : model checking et preuve de théorèmes.

3 CONCEPTION ET RÉALISATION DE L'OUTIL DE VALIDATION DES RÉSEAUX

Sommaire

3.1 Introduction . . . 28
3.2 La modélisation : . . . 28
3.2.1 L'étape de modélisation . . . 28
3.2.2 Définition d'un modèle : . . . 29
3.2.3 Création des modèles : . . . 30
3.3 Exemple de modélisation : . . . 30
3.3.1 Exemple 1 : . . . 30
3.3.2 Exemple 2 : . . . 31
3.3.3 Exemple 3 : . . . 31
3.3.4 Modèle statique de notre bibliothèque . . . 31
3.3.5 Spécification de l'outil réalisé : . . . 31
3.3.5.1 création de project : . . . 32
3.3.5.2 création de réseaux de pétri . . . 32
3.3.5.3 l'affichage de création de réseaux de pétri : . . . 34
3.3.5.4 création avec expression rationnelle : . . . 35
3.3.5.5 le tirage d'une transition de réseaux de petri : . . . 35
3.3.5.6 création avec expression rationnelle : . . . 36
3.3.5.7 Aide : . . . 37
3.4 conclusion . . . 37

Pour faire le lien avec les chapitres précédent, on montre dans la partie suivante comment la modélisation s'applique dans les réseaux. Ensuite, on présente dans la deuxième partie l'architecture générale du programme proposé dans le projet.

3.1 Introduction

l'objectif principal de ce travail est la réalisation d'un outil de vérification de modèles adapté a la validation des différents scénarios ,modèles.des réseaux informatiques.pour cette raison,nous allons décrire la force d'expression des modèles que proposons pour implémentation dons ça première version de notre outil,puis en décrit les démarches de réalisation de l'outil qui se vent en future un logiciel complet de vérification et de validation des réseaux.

3.2 La modélisation :

On a vu dans le chapitre 1 que pour effectuer des opérations de *model – checking*il faut commencer par modéliser le système réel à l'aide d'un langage abstrait, ensuite on spécifie les propriétés qu'on veut vérifier à l'aide d'un langage déclaratif, et finalement on fait appel à un vérificateur pour vérifier si ces propriétés sont valides dans le modèle défini. Le système qu'on voudrait modéliser est constitué par toute entité dynamique dans le système informatique : les fichiers de journalisation de tout genre, les flux réseaux, les systèmes de gestion de bases de données, etc. (voir figure). Il est clair que la modélisation d'un tel système serait une tâche complexe et très lente.
En Recherche opérationnelle, un réseau est un ensemble des nœud qui sont reliés pour

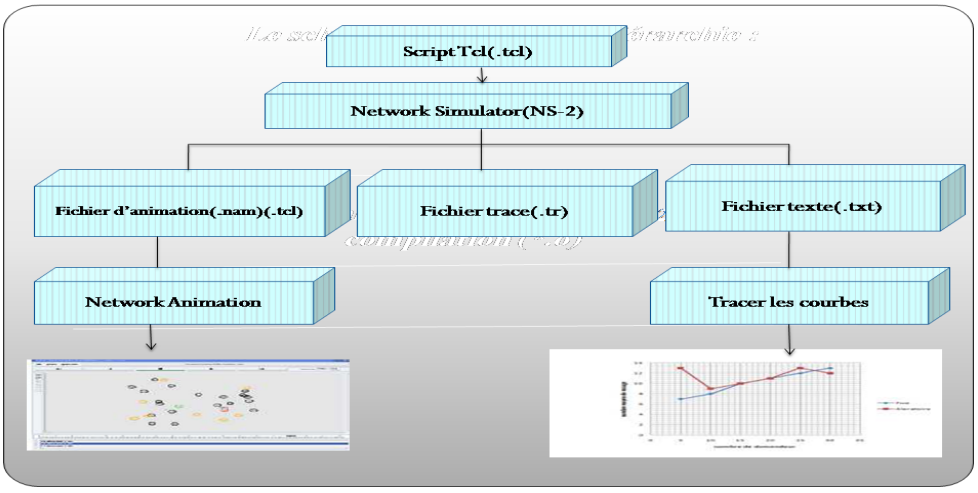


Figure 3.1 – *La réalisation d'une simulation.*

permettre un échange d'information (flot).

3.2.1 L'étape de modélisation

La modélisation au sens large est l'utilisation efficace d'une représentation simplifiée d'un aspect de la réalité pour un objectif donné. Loin de se réduire à l'expression d'une solution à un niveau d'abstraction plus élevé que le code [25], la modélisation en informatique peut-être vue comme la séparation des différents besoins fonctionnels et préoccupations extra-fonctionnelles (telles que : la sécurité, la fiabilité, l'efficacité, la performance, la flexibilité, etc.).

– L'étape de modélisation est une phase essentielle à la simulation. Des différents points doivent être abordés :

– Définir l'objectif de la modélisation (lié au cahier des charges) : Pourquoi modélise-t-on ?

– Qu'étudie-t-on ? Que veut-on améliorer, ou faire ?

– Définir les éléments du système (via la réalisation d'une fonction, ou d'un processus) et les limites du système (les entrées, les sorties).

– Définir les interactions entre ces éléments (hiérarchie).

– Définir la dynamique du système (entités qui circulent entre les éléments, comportement du système au cours du temps).

– Abstraction (choisir les éléments du système pertinents pour l'étude).

– Formalisation, conceptualisation : Modèle mathématique (algèbre (max, +), chaînes de Markov), modèle logiciel (), modèle graphique (réseaux de Petri,bond graphs).

3.2.2 Définition d'un modèle :

Un modèle est un ensemble de règles, de formules, ou d'équations pouvant être utilisées pour prédire un résultat en fonction dun ensemble des nuds ou de variables d'entrée. Un réseau(exp : ad hoc) peut être modéliser par un graphe $Gt = (Vt, Et)$. Où : Vt représente l'ensemble des nœud (i.e. les unités ou les hôtes mobiles) du réseau et Et modélise l'ensemble les connections qui existent entre ces nœud. Si$e = (u, v)$? Et, cela veut dire que les nœud u et v sont en mesure de communiquer directement à l'instant t. La figure suivante représente un réseau*adhoc* de 10 unités mobiles sous forme d'un graphe :

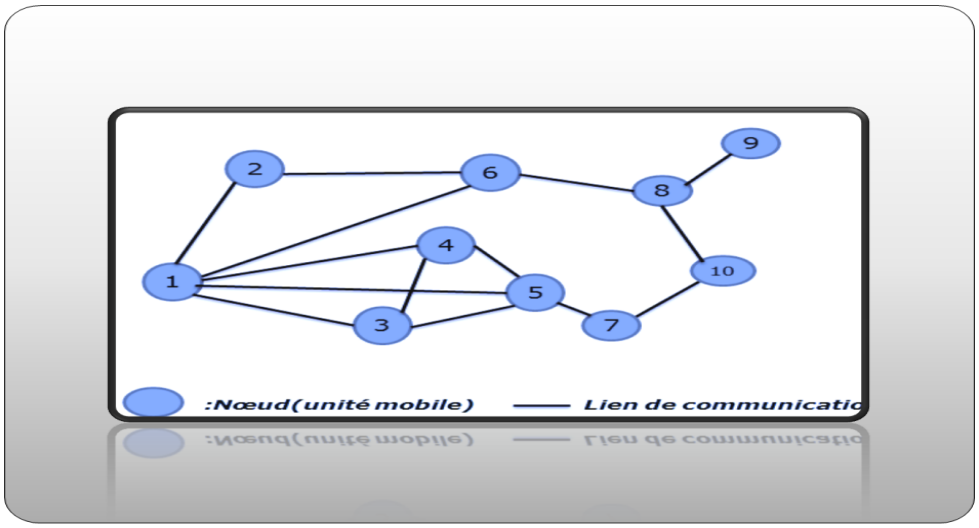


Figure 3.2 – *La modélisation d'un réseau ad hoc.*

La topologie du réseau peut changer à tout moment, elle est donc dynamique et imprévisible ce qui fait que la déconnexion des unités soit très fréquente. exemple

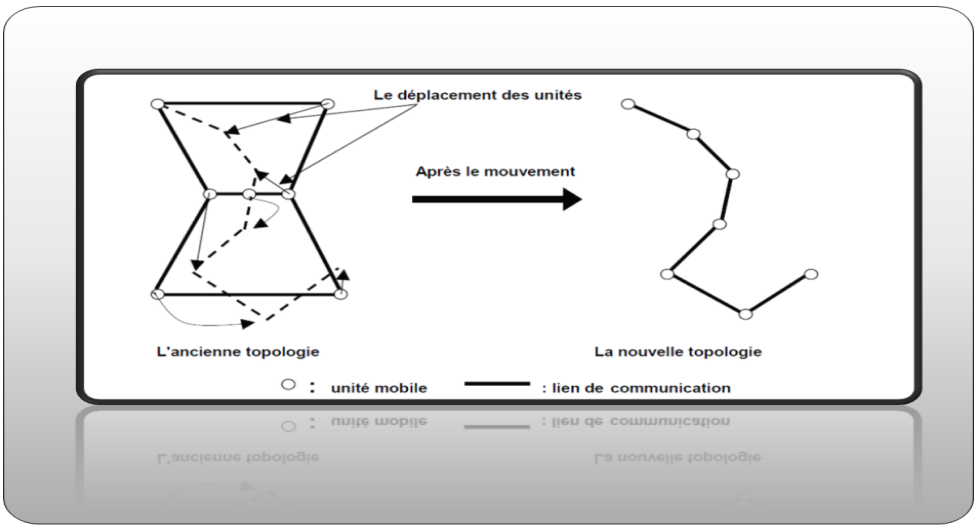


Figure 3.3 – *La modélisation d'un réseau ad hoc..*

3.2.3 Création des modèles :

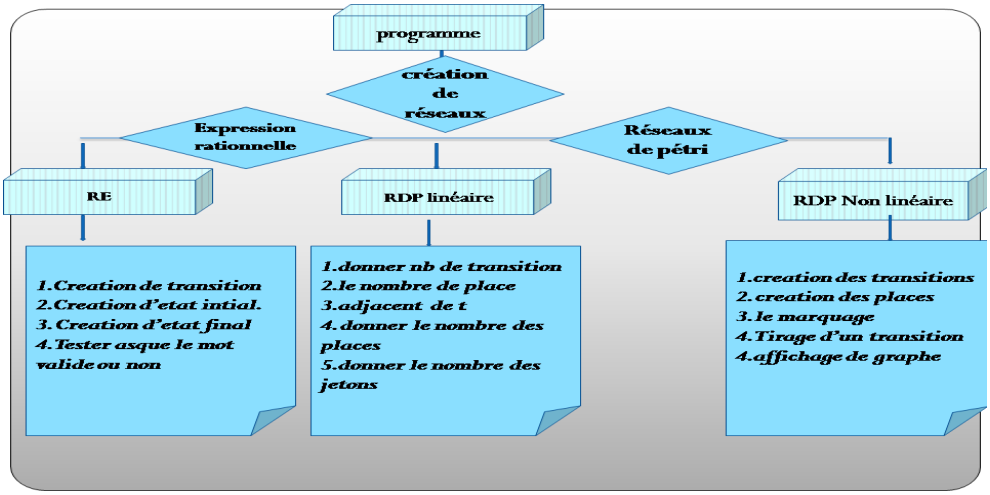


Figure 3.4 – *La réalisation d'une simulation.*

3.3 Exemple de modélisation :

3.3.1 Exemple 1 :

On imagine d'un réseaux statique ou *ADHOC* avec faible mobilité,et on veut étudier l'échange de message entre les nuds du réseaux . ce réseaux est contrent par des propriétés définie au préalable :chaque nud possède une fenêtre qui contient les message qu'il peut contenir dans sa mémoire.la technique utilisée est le *Broadcast* mais chaque nud ne peut

envoyer des messages qu'a un certain nombre de "voisin" .on peut modéliser ce réseaux utilisant un RDP :

- Chaque nud est modélisée par une place .
- La taille de fenêtre est modéliser par le nombre de jeton.
- Chaque transition lie les nuds qui ont une possibilité d'échanger les messages .
- Les messages perdues peuvent être calculés .
- Un fonction de la saturation de transition.
- La validation se fera par des tirages aléatoire jus qu'a saturation .

3.3.2 Exemple 2 :

On peut modélise le *multicast* comme des transition liens les nuds d'un même . en peut s'étendre aux réseaux $ADHOC$ a forte mobilité.

3.3.3 Exemple 3 :

Pour valides des on modélise un réseaux quelconque en utilisant une expression rationnelle ,exemple un réseaux se comporte de la maniéré suivants :

- Phénomène $''a''$ toujours suivant de $''b''$ suivant de $''d''$.
- Toujours commencer par un phénomène "c" suivit de n'importe quel autre phénomène.
- Système se repentant a l'infini peut être modélisé par $(c.(a.(b+d)*))$.
- soit le scénario suivant :b puis c ou d,a modélisé par$b.(c+d).a$.
- on peut vérifier la validation de ce scénario en se referont a $c.(a.(b+d))*$.
- $b.(c+d).a$ non valide.
- c.ad valide.

3.3.4 Modèle statique de notre bibliothèque

3.3.5 Spécification de l'outil réalisé :

on veut développer l'outil pour qu'il soit utilisable alors mointenable par la communauté pour usage professionnel on a séparer l'interface de la bibliothèque de fonctions. On a crée un langage facile de communication avec la bibliothèque.

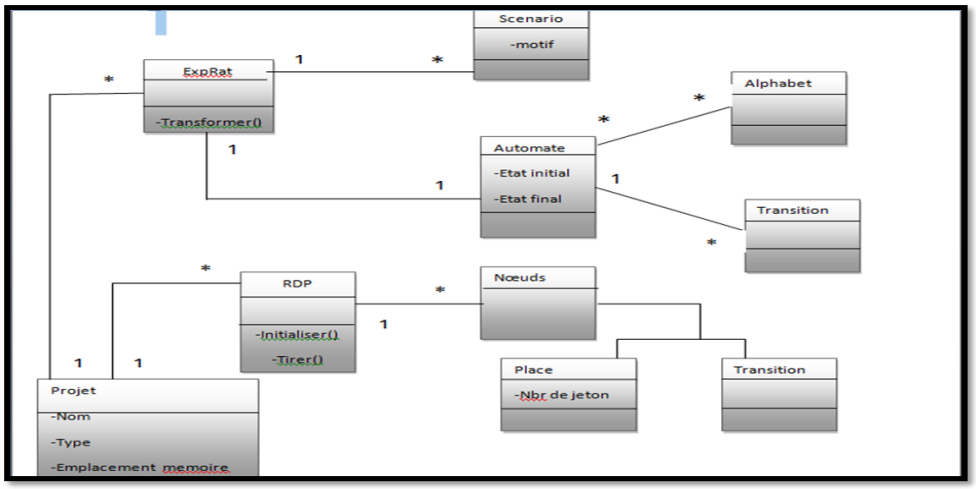


Figure 3.5 – *Le modèle statique de notre bibliothèque.*

3.3.5.1 création de project :

Linterface basique est fait dune manière a favoriser lutilisation des commandes développées dans des scripts systèmes ou dans dautres programmes, cest pour cette raison que linterface utilisée et plus technique que facile.

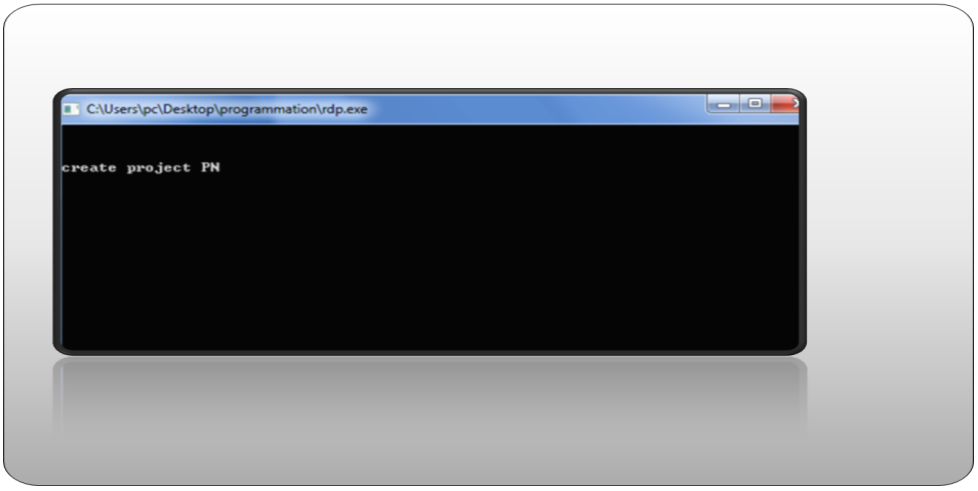


Figure 3.6 – *page d'aceuil.*

3.3.5.2 création de réseaux de pétri

La création d'un réseau de petri se fait en invoquant la commande create project RDP. le système de commandes permet de l'utiliser directement dans d'autres programmes. On peut utiliser d'autres commande pour créer les sommets du réseau et de le faire dérouler.

Figure 3.7 – *création de rdp*

Figure 3.8 – *création de réseaux de pétri linéaire*

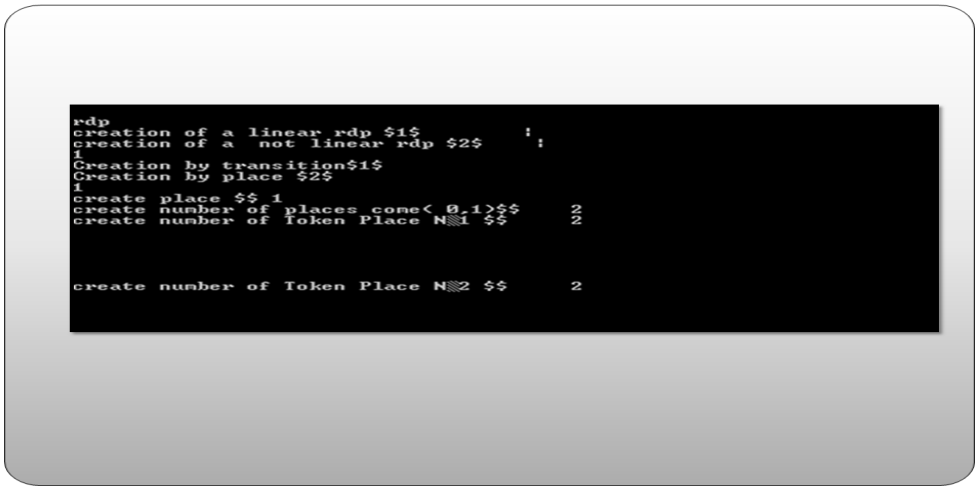
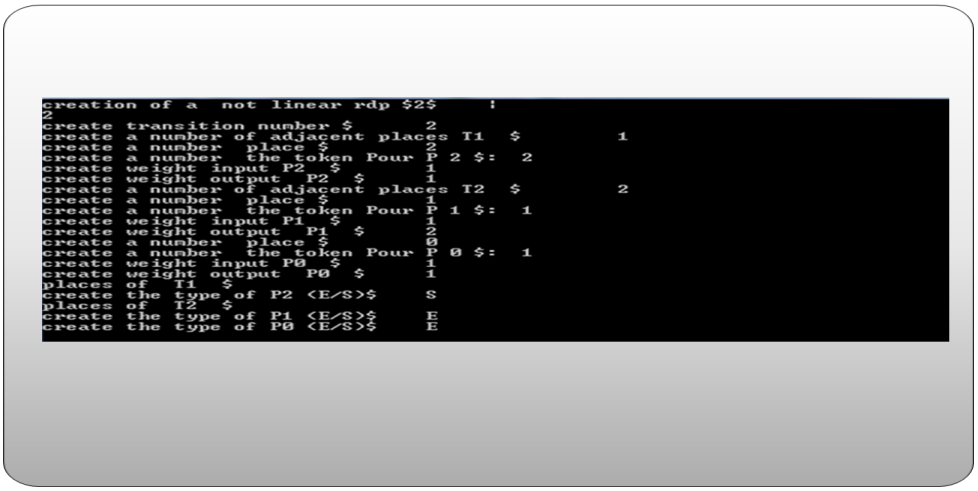


Figure 3.9 – *création de réseaux de pétri linéaire.*



3.3.5.3 l'affichage de création de réseaux de pétri :

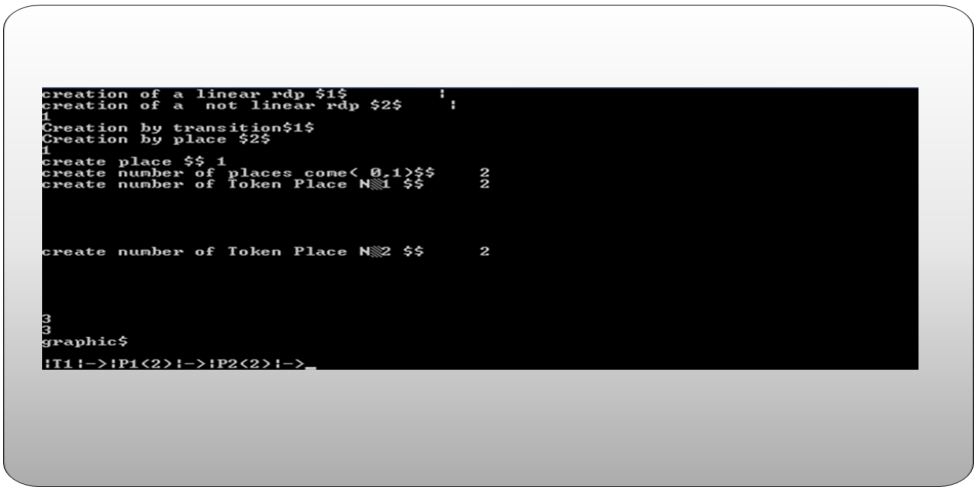
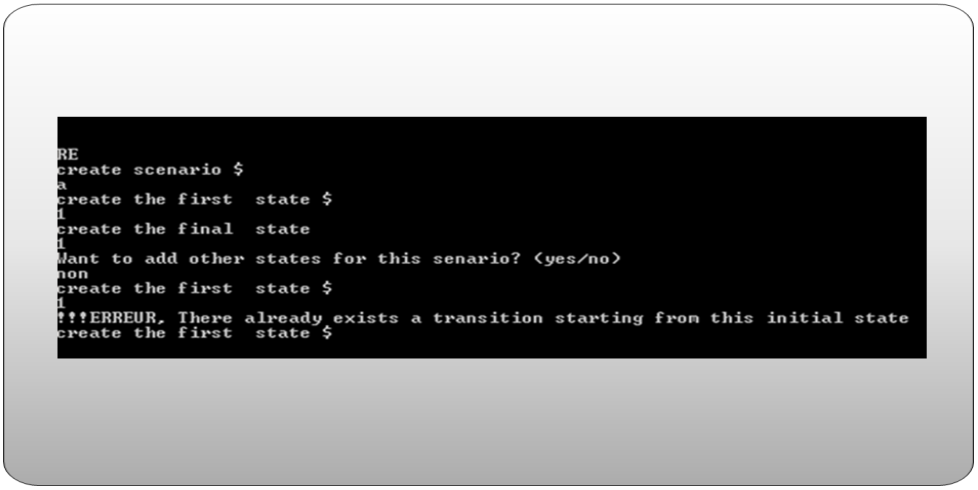
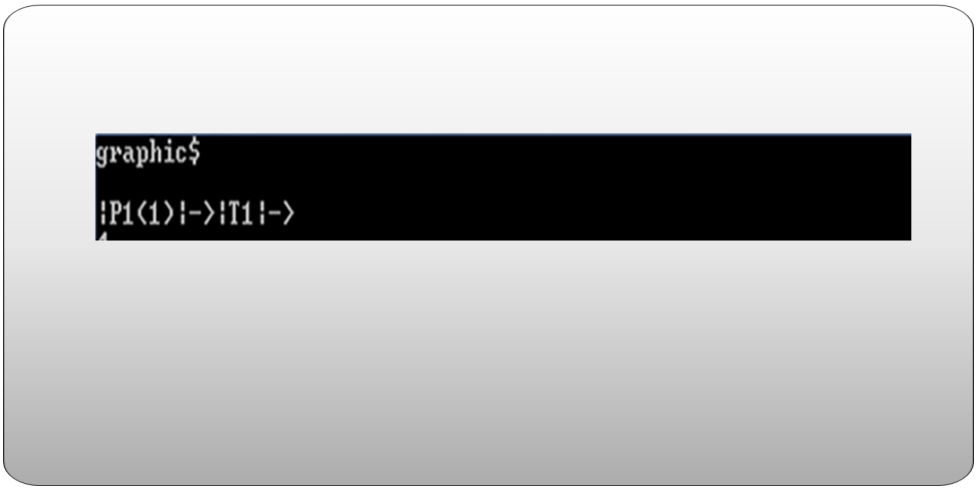


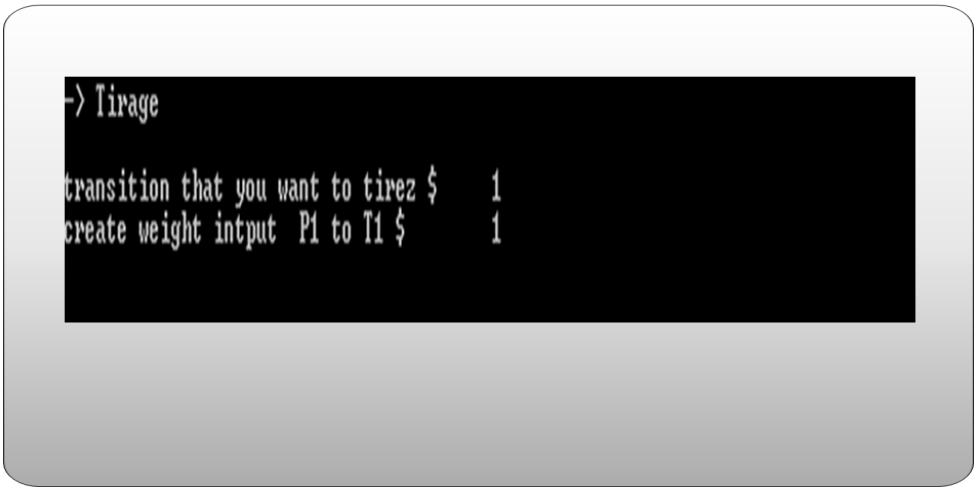
Figure 3.10 – *l'affichage de graphe.*

3.3.5.4 création avec expression rationnelle :

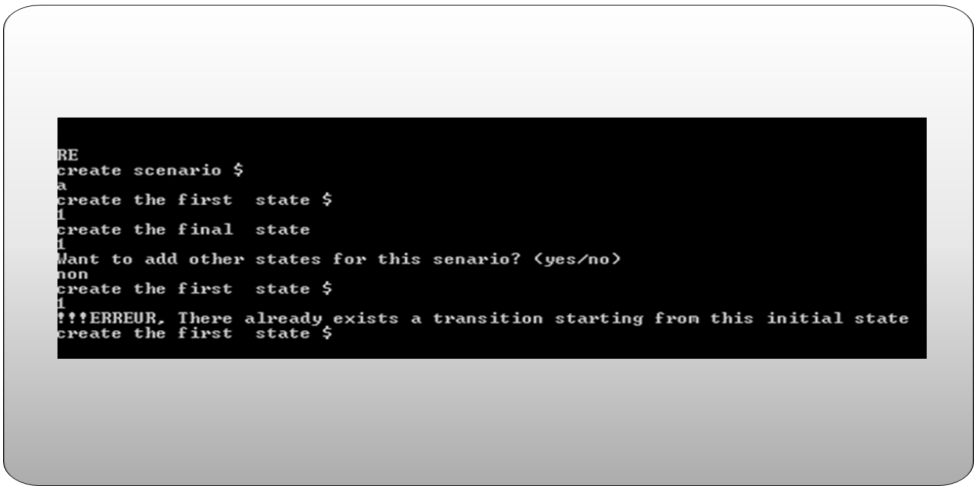


3.3.5.5 le tirage d'une transition de réseaux de petri :



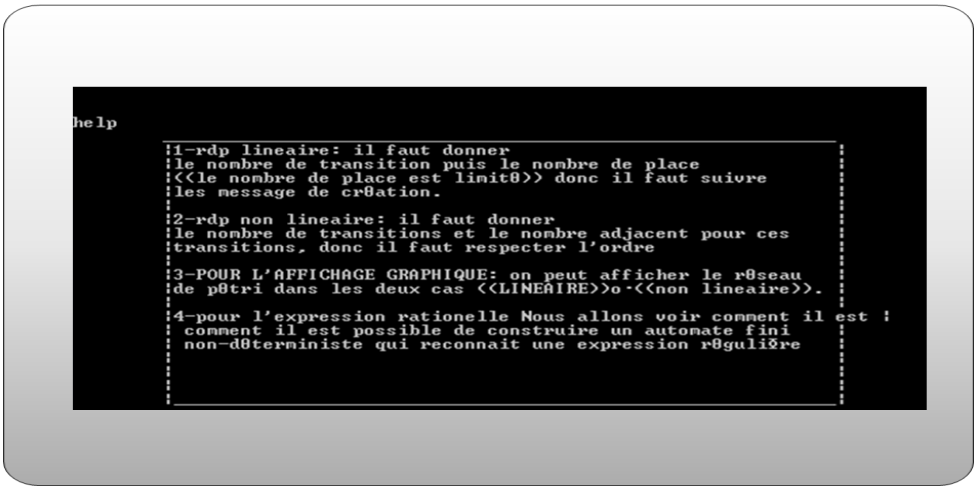


3.3.5.6 création avec expression rationnelle :



3.3.5.7 Aide :

On peut soliciter une page daide :



Nous avons présenté à titre d'exemple les possibilités de créer un modèle se basant sur un réseau de petri avec notre bibliothèque. Il y a d'autres commandes pour les expressions rationnelles et les automates d'états finis.

3.4 CONCLUSION

Nous avons présenté dans ce chapitre les outils modélisé dans notre bibliothèque équipée d'un langage de script facile à intégrer. Nous avons donné quelques exemples de modélisation de réseaux et enfin nous avons montré quelques captures decran de lutilisation de notre outil.

Conclusion et Perspectives

Lesmé©thodes de vé©rification formelle se ré©pandent trÃ¨s vite dans tous les domaines, et traitent de plus en plus de donnÃ©es sensibles. Ce succÃ¨s est du au fait que les mÃ©thodes formelles sont des mÃ©thodes rigoureuses et basÃ©es sur la thÃ©orie et donc garantissent un niveau de sÃ©curitÃ© trÃ¨s Ã©levÃ© par rapport aux mÃ©thodes usuelles (test et simulation). Il existe deux approches principales de mÃ©thodes formelles : le model-checking et la preuve. La complÃ©mentaritÃ© entre les deux approches nous a motive a les combiner pour vÃ©rifier des propriÃ©tÃ©s sur une Ã©tude de cas. Durant notre mÃ©moire, nous avons explorÃ© la possibilitÃ© de combinÃ© ces deux approches Ã travers le protocole de communication . Nous avons d'abord propose une modÃ©lisation du protocole au moyen dâun rÃ©seau de PÃ©tri sous la plate forme en c++.

Bibliographie

[MOD98] M. Barjaktarovic. , *state-of-the-art. In formal methods. WetStone Technologie,Inc.*, 1998.

[MODCH99] Edmund M. Clarke, Jr., *Orna Grumberg and Doron A. Peled, Model Checking. MIT Press*, 1999.

[MD08] Kais Klai and Laure Petrucci.*Modular construction of the symbolic observation graph. In 8th International Conference on Application of concurrency to System Design (ACSD), pages 88-97*, 2008.

nite objects, Handbook of Theoretical Computer Science : Formal Models and Semantics, tome B (Jan Van Leeuwen,ed.), MIT Press, 1999.

[Petri62] etri (Carl Adam). *mit Automaten.â These de PhD, Bonn : Institut fur Ins-trumentelle Mathematik, Schriften des IIM Nr.* 2,1962.

[SPIN03] G. J. Holzmann. *SPIN Mo del Checker, Primer and Reference Manual. Addison-Wesley*, 2003.

[SPIN97] G. J. Holzmann. *The Model Checker SPIN. IEEE Transactions on software enginee-ring, Vol. 23(5)*, 1997.

[SPIN96] G. J. Holzmann, *Peled et Mihalis Yannakakis. On Nested Depth First Search.In The Spin Veri cation System, pp 23-32. American Mathematical So ciety*, 1996

[SPIN66] Y. Vardi. *An automata-theoretic approach to linear temp oral logic. Dans Faron Moller and Graham M. Birtwistle, editors, Pro ceedings of the 8th Ban Higher Order Workshop, Lecture Notes in Computer Science, Vol. 1043, pp. 238.266, Springer-Verlag*, 1996

[EXP03] D. Ziadi, J.-L. Ponty et J.-M. Champarnaud, *Passage dâune expression rationnelle Ã un automate fini non-dÃ©terministe , Bulletin of the Belgian Mathematical Society Simon Stevin* , 4-2(1997), pp.177â203

[EXP04] M. Champarnaud and G. Duchamp, *Derivatives of rational expressions and related theorems Theoret. Comp. Sc.*, 313-1(2004), pp. 31â44