Republique Algerienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider - BISKRA
Faculté des Sciences Exactes,Sciences de la Nature et de la Vie
Département d'informatique

Mémoire

Présenté pour obtenir le diplôme de Master académique en

Informatique

Option: **Génie Logiciel et Systèmes Distribués**

Un outil de transformation de BPNs vers CBPNs

Par:
AILANE Amira

Bennoui Hammadi MCA Superviseur

Année universitaire: 2019/2020

Remerciements

Tous mes remerciements et ma gratitude pour Allah le tout puissant qui m'a donné le courage et la volonté d'aller jusqu'au bout de ce niveau.

Mes vifs remerciements également à ***Mr Bennoui Hammadi*** *d'avoir assister et bien suivre mon travail ainsi que pour ses conseils précieux.*

Je voudrais également exprimer ma gratitude aux membres du jury pour avoir lu et évalué mon mémoire.

Je n'oublie jamais de remercier tous mes professeurs d'informatique qui m'ont appris les bases de l'informatique.

Je tiens à remercier l'etudiante ***ph.D Soumia Menacer*** *pour ses conseils, sa patience sans fin. Ainsi que monsieur* ***Ben Terki Aboubaker Seddiq*** *pour ses interventions importantes.*

Enfin, j'exprime ma gratitude à ma mère, à mon père et à toute ma famille.

Résumé

Dans ce rapport, nous présentons un outil qui permet d'éditer un réseau de Petri comportemental et de le transformer en un réseau de Petri comportemental coloré. L'objectif d'introduction des réseaux de Petri comportementaux colorés est d'attaquer le problème de complexité spatiale des modèles non colorés. La conversion est faite par un algorithme de transformation.

Abstract

In this report, we present a tool that allows to model a behavioral Petri net and transform it into a colored behavioral Petri net. The goal of introducing colored behavioral Petri nets is to reduce the spatial complexity of uncolored models. The conversion is performed by a translation algorithm.

Contents

Table de matières ii

Liste de tables iii

Liste de figures iv

Introduction générale viii

I État de l'art 1

1 Concepts de base 2

Introduction . . . 3

1.1 Réseaux de Petri (Définition) . . . 3

1.1.1 Réseaux de Petri (Formellement) . . . 4

1.1.2 Propriétés des RDPs . . . 5

1.1.3 Analyse des RDPs . . . 6

1.2 Réseaux de Petri colorés . . . 6

1.2.1 Multi-ensemble . . . 6

1.2.2 Réseaux de Petri colorés(Formellement) . . . 7

1.3 Diagnostic à base de modèle . . . 8

1.3.1 Diagnostic à base de cohérence . . . 9

1.3.2 Diagnostic à base d'abduction . . . 9

1.3.3 Approche unificative du DBM . . . 9

1.4 Behavioral Petri Nets(Réseaux de Petri comportementaux) . 10

1.4.1 Modèle BPN . . . 11

1.4.2 B-W analyse . . . 12

Conclusion . . . 13

2 Transformation de BPNs aux CBPNs 14

Introduction . . . 15

2.1 Colored Behavioral Petri Net(réseaux de Petri colorés comportementaux) . . . 15
2.1.1 Colored Behavioral Petri Net(Formellement) . . . 15
2.1.2 Réseaux de Petri colorés pour le diagnostic à base de modèle . . . 17
2.1.3 CBPN pour le diagnostic à base de modèle . . . 18
2.2 Processus de diagnostic à base de CBPNs . . . 19
2.2.1 Formalisation de problème de diagnostic utilisant CBPNs 19
2.2.2 Résolution de problème de diagnostic . . . 20
2.2.3 Procédure de transformation de BPN à CBPN . . . 23
Conclusion . . . 25

II Développement d'un outil de transformation d'un BPN vers un CBPN 26

3 Analyse et modélisation 27
Introduction . . . 28
3.1 Spécification et analyse de besoin . . . 28
3.2 Conception . . . 28
3.2.1 Conception globale . . . 29
3.2.2 Conception détaillée . . . 32
Conclusion . . . 38

4 Implémentation 39
Introduction . . . 40
4.1 Outils et langages de développement . . . 40
4.1.1 Langage de programmation Python . . . 40
4.1.2 Éditeur de programmation PyCharm . . . 40
4.1.3 Package Tkinter(Tool Kit Interface) . . . 41
4.1.4 Système de préparation de documents LaTeX . . . 41
4.1.5 Éditeur (TeX MAKER) . . . 41
4.1.6 XML(Extensible Markup Language) . . . 41
4.2 Implémentation . . . 42
4.2.1 L'application principale . . . 42
4.2.2 Fonctionnalités de l'application . . . 45
Conclusion . . . 54

Conclusion générale xii

Bibliography xiii

Liste de tables

4.1 Versions Logicielles/Matérielles 42

Liste de figures

1.1 Exemple illustrant le franchissement d'une transition, le marquage avant et après le franchissement[8] 5
1.2 Le diagnostic comme interaction de l'observation et de prédit[4] 9
1.3 Exemple de BPN[3] 11
1.4 Sémantique de transition Or 12

2.1 Exemple de CBPN 17
2.2 exemple de CBPN[7] 22
2.3 Graphe d'accessibilité en arrière [7] 23
2.4 Procédure de transformation [7] 24
2.5 Exemple de BPN 24
2.6 Modèle CBPN 25

3.1 Interaction entre les composants 29
3.2 Architecture global de l'application 30
3.3 Module de modélisation(Éditeur graphique) 31
3.4 Module de transformation 31
3.5 Diagramme de classe BPN 32
3.6 Diagramme de classe CBPN 32

4.1 Interface Main. 43
4.2 Menu Ouvrir BPN. 44
4.3 Menu Nouveau. 44
4.4 Menu Exit. 44
4.5 Menu Aide. 44
4.6 Barre d'outils. 45
4.7 Création d'une place. 46
4.8 Création d'une transition And. 46
4.9 Création d'une transition Or. 46
4.10 Modèle BPN. 47
4.11 Fichier XML pour BPN dans figure 4.10 47

4.12 Modèle CBPN. 48
4.13 Fichier XML pour CBPN 4.12 49
4.14 BPN 1 50
4.15 Matrice FW de t0 51
4.16 CBPN à partir de BPN 1 (Figure 4.14) 52
4.17 BPN 2 53
4.18 CBPN à partir de BPN 2(Figure 4.17) 54

Liste d'algorithmes

1 Dessiner une place 35
2 Dessiner une transition And 36
3 Dessiner une transition Or 36
4 Dessiner Arc 36
5 Transformation de BPN à CBPN 37
6 Dessiner CBPN 38

Introduction générale

Introduction générale

De nos jours, le génie logiciel s'appuie sur les langages de modélisation formel pour décrire, analyser et diagnostiquer des systèmes, dont on mentionne les réseaux de Petri, les automates, le pi-calcul, ... etc où le langage de modélisation est choisi en fonction de puissance d'expression et de disponibilité de techniques d'analyse.

En intelligence artificielle, le diagnostic basé-modèle est considéré comme l'un des rares succès où un modèle structurel et/ou comportemental du système à diagnostiquer est utilisé pour prédire son comportement normal ou fautif afin de le comparer avec celui observé.

Dans notre travail, nous présenterons les réseaux de Petri comme un outil formel pour représenter les modèles de comportement causal du système à diagnostiquer. Les réseaux de Petri sont largement utilisés dans le diagnostic basé sur un modèle à cause des techniques d'analyses qu'ils fournissent dont les plus importantes sont : les graphes d'accessibilités et les techniques algébriques. La version réseau de Petri qui à été utilisée pour représenter un modèle causal est le réseau de Petri comportemental (BPN) qu'a deux types de transitions: l'un pour modéliser la conjonction (type AND) et l'autre pour modéliser la disjonction (type OR), mais, les BPNs facent un problème de complexité spatiale car chaque instance d'une variable du système étudié est modélisée par une place correspondante. Cela est dû au fait qu'il n'existe qu'un seul type de jetons.

Dans le chapitre 2, on va proposé deux solutions alternatives permettant de faire face à un tel problème. La première est basée sur l'usage de réseaux de Petri colorés (RdPC) tel que définis par Jensen. Un tel usage ne permet pas de résoudre le problème de manière efficace car les RdPCs ne modélisent pas le OU logique sauf si nous les enrichissons par des arcs inhibiteurs. En plus, l'analyse d'accessibilité en arrière des RdPC requis par le processus de diagnostic nécessite une inversion du modèle. Ceci va conduire à avoir un problème de complexité mais cette fois-ci spatiale et temporelle.

L'autre alternative est l'utilisation d'une classe particulière de RdPCs

appelée CBPNs (Colored Behavioural Petri Nets) qu'à été introduite récemment par Mancer *et al.* dans [7]. L'idée des CBPNs est d'attacher à chaque transition une matrice décomposable décrivant ses manières de franchissement. Cette matrice facilite le processus d'inversion du modèle grâce à sa capacité à se diviser en deux matrices, car après avoir inverser le modèle (inverser les arcs), les matrices affectées aux places d'entrées deviendront dédiées aux places de sorties et vice versa.

L'objectif de notre projet, est d'implémenter une application qui permet de transformer un BPN à un CBPN.

Ce rapport commence par une introduction qui présente le problème et une proposition pour le résoludre, puis il se devise en deux parties, la première se concentre sur le niveau théorique (état de l'art) et la deuxième montre notre contribution (l'implémentation).

La première partie contient deux chapitres, le premier décrit les concepts de base concernant les réseaux de Petri et le problème de diagnostic, il montre comment on considère le BPN comme une version de réseaux de Petri qui décrit un modèle causal. Le deuxième chapitre montre les problèmes qui face les réseaux de Petri colorés dans le domaine de diagnostic, puis il présente le CBPN pour repésenetr un modèle causal afin de terminer le processus de diagnostic. A la fin du chapitre, on trouve une description complète de la procédure de transformation d'un BPN à un CBPN .
La deuxième partie, se devise en deux chapitres aussi, le premier chapitre donne la spécification et les conceptions nécessaire pour mettre en oeuvre l'application. Le dernier chapitre mentionne tous les outils de programmation et de rédaction qui nous aide pour accomplir ce travail avec une description concernant l'application et comment l'utiliser. Ce chapitre se termine par des exemples de transformation.

Part I

État de l'art

Chapter 1

Concepts de base

Introduction

Depuis des années, de nombreux travaux de recherche ont été consacrés au problème de diagnostic de pannes à cause de demandes de réparation des systèmes qui tombent en panne. La reprise rapide de fonctionnement normal demande un diagnostic minimal et rapide. Le diagnostic basé sur les modèles causaux et l'un des approches qu'a gagné beaucoup d'attentions ces dernières décennies.

Les réseaux de Petri ont été utilisés pour représenter les modèles causaux, et ainsi exploiter leurs techniques d'analyse pour mettre en œuvre efficacement les mécanismes de raisonnement du diagnostic.

Dans ce chapitre, on va commencer par un rappel sur les réseaux de Petri (classiques et colorés), ainsi qu'une vue simplifiée sur les approches de diagnostic à base de modèle. A la fin, on va définir les réseaux de Petri comportementaux BPNs.

1.1 Réseaux de Petri (Définition)

Les réseaux de Petri (RdP) sont l'un des formalismes mathématiques décrivant les systèmes à événements discrets qui peuvent être caractérisés par : concurrence, distribution, l'indéterminisme ... etc. Comme un langage de modélisation, il représente le comportement d'un système dynamique par un graphe biparti orienté composé de deux ensembles de noeuds, des places et des transitions qui sont reliées – entre deux nœuds différents (place-transition ou transition-place) – par des arcs valués et orientés. La place avec un arc orienté d'elle vers une transition s'appelle place d'entrée, par contre, la place avec un arc orienté d'une transition vers elle-même s'appelle place de sortie [7].

Les places peuvent contenir un ensemble de jetons (le jeton décrit une ressource présente dans la place ou une condition satisfaite), le mouvement des jetons dans le modèle représente le comportement du système dynamique, la distribution des jetons dans le modèle s'appelle le marquage.

Le franchissement d'une transition mène à retirer un nombre déterminé de jetons de ses places d'entrée et ajouter un nombre déterminé de jetons dans ses places de sortie, la transition ne peut pas franchir sauf si elle est sensibilisée.

1.1.1 Réseaux de Petri (Formellement)

Définition 1 *Un réseau de petri est un 5-quintuplet* $RP = (P, T, F, W, \mu_0)$*[8] ,tel que :*

- $P = \{p_1, p_2, ..., p_m\}$ *est un ensemble fini de places.*
- $T = \{t_1, t_2, ..., t_n\}$ *un ensemble fini de transitions.*
- $F \subseteq (P \times T) \cup (T \times P)$ *est l'ensemble fini d' arcs (relation de flux).*
- $W : F \to \{1, 2, ...\}$ *la fonction de poids.*
- $\mu_0 : P \to \{1, 2, ...\}$ *le marquage initial.*

$P \cup T \neq \emptyset$ et $P \cap T = \emptyset$.
Pour chaque $x \in P \cap T$ on utilise les notations classiques ${}^\bullet x = \{y | yFx\}$ et $x^\bullet = \{y | xAy\}$ pour déterminer les entrées et les sorties du noeud x. Si ${}^\bullet x = \emptyset$ le noeud est dit noeud source alors que si $x^\bullet = \emptyset$ le noeud est dit noeud puits. La fonction de marquage $\mu : P \to N$ associé à chaque place un nombre entier positif qui détermine le nombre de jetons, le marquage initial μ_0 représente le point de départ du comportement du réseau de Petri [7].

Définition 2 *un réseau de Petri marqué est le pair* (N, μ) *tel que N est un RdP et* μ *est un marquage.*

Le comportement d'un système peut être décris en terme des états et leurs changements, l'état ou le marquage de RdP est changé selon des règles de franchissement du transitions comme suit :

- Une transition t est dite sensibilisée si chacune de ses places d'entrée est marqué par au moins $W(p, t)$. $\mu(p) \geq W(p, t), \forall p \in^\bullet t$.
- Une transition sensibilisée peut être franchie.
- Le franchissement de t se traduit par:
  - Le retrait de $W(p, t)$ jetons de chaque $p \in^\bullet t$.
  - L'ajout de $W(t, p)$ jetons à chaque $p \in t^\bullet$.

Après le franchissement on obtient un nouveau marquage μ' :

$$\mu'(p) = \mu_0(p) - W(p, t) + W(t, p)$$

Exemple 1 *L'exemple (Figure 1.1) suivant montre le changement de marquage après le franchissement d'une transition utilisant ces règles :*

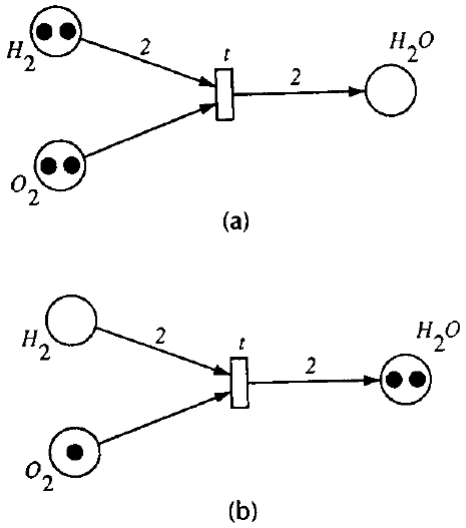


Figure 1.1: Exemple illustrant le franchissement d'une transition, le marquage avant et après le franchissement[8]

1.1.2 Propriétés des RDPs

Atteignabilité (Accessibilité)

On dit qu'un marquage μ est atteignable à partir d'un marquage μ_0 s'il $\exists$ une séquence de franchissements menant de μ_0 à μ . On note par $R(\mu_0)$ l'ensemble de marquages atteignables à partir de μ_0.

Le problème d'atteignabilité est de trouver si $\mu \in R(\mu_0)$ et la séquence $s =< t_1, t_2, .., t_n >$ de transitions à franchir pour atteindre μ à partir de μ_0. L'ensemble de marquages accessibles à partir d'un marquage μ_0 est :

- $\mu_0 \in R(u_0)$.
- si $\mu_1 \in R(\mu_0)$ et $\mu_2 \in R(\mu_1)$ alors $\mu_2 \in R(\mu_0)$.

Bornitude

On dit qu'un réseau de Petri est K-borné si le nombre de jetons dans chaque place ne dépasse pas un nombre fini K pour tout marquage accessible à partir de μ_0 .
$\mu(p) \leq k$, $\forall \mu \in R(\mu_0)$ et $\forall p$.

RDP Safe

Si un réseau de Petri est 1-borné (K=1), alors le réseau de Petri est dit safe.

1.1.3 Analyse des RDPs

Dans la littérature des RdPs, plusieurs méthodes d'analyse des modèles RdPs ont été définies dont la plus connue et utilisée est celle basée sur le concept de graphe d'accessibilité. Dans la suite, on essayera de décrire brièvement le principe d'une telle méthode qui nécessite une énumération de tous les états que le système peut l'atteindre.

Graphe d'accessibilité

Le graphe d'accessibilité montre tous les états possibles de l'RDP partant de l'état initial μ_0 :

- Soit $T(\mu_0)$ l'ensemble de transitions sensibilisées dans μ_0 : alors on obtient Card(T(μ_0)) nouveaux marquages (nœuds) par tirage de ces transitions.
- En partant de l'un de ces nouveaux marquages, par exemple μ, Card(T(μ)) nouveaux marquages peuvent être atteints à partir de μ ces marquages peuvent être représentés par un arbre où :
  - La racine représente μ_0.
  - Une nouvelle couche de nœuds représentant l'ensemble de marquages obtenus de ceux représentés par la couche précédente de nœuds en tirant une parmi les transitions sensibilisées.

1.2 Réseaux de Petri colorés

Un modèle de réseau de Petri appelé réseau de Petri coloré (RdPC) a été introduit afin de résoudre le problème de complexité des réseaux de Petri (un modèle RdP d'un système donné a tendance à devenir trop complexe, que ce soit la taille du modèle ou l'analyse, même pour des systèmes de taille modeste). Le principe d'un modèle RdPC est d'étendre la notion de jeton avec une couleur, par commodité nous utilisons la notation de « la couleur du jeton ». Ça permet de décrire des processus similaires de manière uniforme et succincte sans perdre la capacité de les distinguer.

1.2.1 Multi-ensemble

Un multi-ensemble est un ensemble où des éléments individuels peuvent apparaître plusieurs fois.

Définition 3 *Un multi-ensemble m sur un ensemble S est une fonction m* $\in [S \to N]$ *dénoté* $\Sigma_{s \in S} m(s)'s$ *tel que :*

- $\forall s \in S$ *ssi* $m(s) \neq 0$.
- S_{MS} *est l'ensemble de tous les multi-ensemble sur S.*

$\forall m, m_1, m_2 \in S_{MS}$ et $\forall n \in N$:

$|\mathrm{m}| = \Sigma_{s \in S} m(s)$.

$m_1 + m_2 = \Sigma_{s \in S}(m_1(s) + m_2(s))'s$.

$m1 \neq m2 = \exists s \in S : m_1(s) \neq m_2(s)$.

$m_1 \leq m_2 = \forall s \in S : m_1(s) \leq m_2(s)$.

1.2.2 Réseaux de Petri colorés(Formellement)

Définition 4 *Un réseau de Petri coloré RdPC est le quintuplet* $N = (\Sigma, P, T, A, C)$ *où [7]:*

$P \cap T = \emptyset$ *et* $P \cup t \neq \emptyset$.

$A \subseteq (T \times P) \cup (T \times P)$.

$C \in [p \to 2^{\Sigma}]$.

P, T et A sont définis de la même manière que dans les réseaux de Petri (A et F réfère à la même chose). Σ est un ensemble non vide de types dit ensemble de couleurs (ensemble d'ensembles de couleurs). C est la fonction de couleurs qui associée à chaque place p un ensemble de couleurs (on écrit $C(p) \subseteq \Sigma$); c'est à dire, que chaque place p peut contenir un ou plusieurs jetons dont chacun porte une couleur appartenant à l'ensemble de couleurs de p. La fonction de marquage μ définit à chaque place un multi-ensemble de couleurs tel que : $\forall p \in P : \mu(p) \in C(p)_{MS}$ Le franchissement d'une transition entraîne le déplacement de jetons dans le modèle. Ces jetons sont déterminés par les expressions d'arc (qui se composent de variables typées, constantes, de fonctions ou même d'opérateurs).
L'évaluation d'une expression d'arc est un multi-ensemble de couleurs, il peut être attaché à chaque transition une expression booléenne (avec des variables) appelée guard qui spécifie les liaisons pour lesquelles elle est évaluée à vrai, un Binding (liaison) est une affectation des valeurs de données aux variables libres apparaissant dans l'expression d'arc entrant ou la guard d'une transition.

Définition 5 *Un RdPC marqué est le pair (N,μ) où N est un RDPC et μ est une fonction définie sur P tel que :*
$\mu(p) \in C(p)_{MS}, \forall p \in P$

Une transition t est sensibilisée s'il existe une liaison tel que:

- Le résultat de l'évaluation de chacune des expressions d'arc entrant est présente dans la place d'entrée correspondante.
- Le guard (s'il existe) est satisfait.

Lorsqu'une transition t est sensibilisée dans un marquage μ où $\mu[t > \mu^{'}$,le nouveau marquage $\mu^{'}$ est calculé comme suivant :
$\forall p \in P : \mu^{'}(p) = \mu(p) - E(p,t) < b > + E(t,p) < b >$, où E(x,y) est l'expression associée à l'arc (x,y) et E(x,y)<b> est l'évaluation de l'expression d'arc avec le binding b.

1.3 Diagnostic à base de modèle

L'avancement de la conception moderne et de la technologie de fabrication nous ont permis de construire des systèmes de haute complexité. Lorsque ces systèmes ne fonctionnent pas correctement, ils doivent être réparés. Pour les réparer if faut passer par l'étape de diagnostic pour savoir les causes de ce mal fonctionnement.
Dans le domaine de l'intelligence artificielle, plusieurs tentatives ont été proposées pour définir des approches de diagnostic qui ont abouti à la proposition de deux approches déffirentes: l'une est basée sur les systèmes expert "baséheuristique" et l'autre est basée sur les modèles. Le diagnostic expérimental "basé heuristique" représente l'expérience comme d'un ensemble de règles de la forme "symptôme $\rightarrow$ cause" ce qui rend la tâche de raisonnement facile. Cette approche souffre de la difficulté d'acquisition des connaissances et les limites de leur domaines d'application

L'idée clé de l'approche de diagnostic à base de modèle (DBM) [4] est de remplacer la codification de l'expertise par le modèle de la structure et/ou le comportement du système à diagnostiquer. Le processus de diagnostic prend en entré le modèle du système en cours d'examen et l'observation (le comportement actuel observé du système) pour atteindre la solution qui représente le diagnostic (la cause de panne).

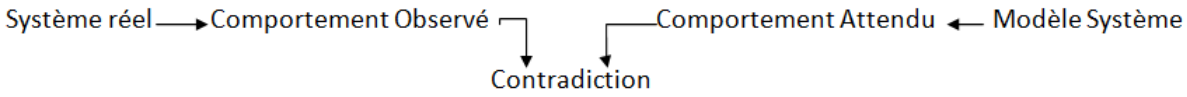


Figure 1.2: Le diagnostic comme interaction de l'observation et de prédit[4]

1.3.1 Diagnostic à base de cohérence

Cette approche exige la présence du comportement correct dans le modèle pour savoir comment le système se comporte normalement, puis le processus de diagnostic va comparer le comportement observé du système réel par rapport au comportement prédit à partir du modèle.

1.3.2 Diagnostic à base d'abduction

Cette approche est dédiée pour traiter les modèles qui décrivent le comportement fautif. Chaque composant est caractérisé par un ensemble de modes de comportement. Donc, on trouve dans le modèle la relation causal entre le symptôme et la faute. On peut exploiter le modèle causal du système à diagnostiquer.

Modèle causal

un modèle causal est un couple (V,E) où V est un ensemble d'entités, noté états, et E est un ensemble de relations de cause-effet entre les états. Pour la raison du diagnostic, les états sont classés en causes initiaux , états internes et manifestations. Les états d'un modèle causal sont utilisés pour représenter les états (états partiels) du système modélisé. Les causes initiaux représentent les états initiaux à partir desquels commence toute évolution dans le système. Les états internes décrivent la partie inobservable du système comme des conséquences des états initiaux. Les manifestations représentent les états observables du système comme des conséquences des états internes. Chaque état peut être instancié en lui affectant une valeur à partir d'un ensemble fini et prédéfini noté $valeurs_admissibles$. Il est important de garder à l'esprit que chaque état doit prendre au plus une valeur à un instant donné et qu'il peut être présent sur le modèle ou absent [6].

1.3.3 Approche unificative du DBM

Cette approche est défini comme un problème d'abduction avec contraints de cohérence [5].

Donc, une observation doit être prédite par un diagnostic avec la satisfaction de certaines contraintes de cohérence. Formellement,un problème de diagnostic est donné par le triple $DP = (BM, INIT, < \Psi^+, \Psi^- >)$,où BM est le modèle comportemental du système à diagnostiquer, INIT est l'ensemble des instances de causes initiaux en fonction desquelles les observations doivent être expliquées,Ψ^+ est un sous-ensemble d'observations que doit impliquer une solution de DP ,Ψ^- est l'ensemble de toutes les valeurs possibles qui sont en conflit avec l'observation faite (qui sont connues pour être absentes dans le cas examiné).

Soit OBS l'ensemble d'observations obtenues, ainsi,$\Psi^+ \subseteq OBS$ et $\Psi^- = \{m(x)|m(y) \in OBS, x \neq y\}$ où m est une manifestation et
$x, y \in valeurs\_admissible(m)$.
Une solution d'un DP est l'ensemble $\Delta \subseteq INIT$ où Δ prédit chaque paramètre dans Ψ^+ et ne prédit aucun paramètre dans Ψ^-:

$$\forall x \in \Psi^+, BM \cup \Delta \vdash x$$
$$\forall y \in \Psi^-, BM \cup \Delta \nvdash y$$

1.4 Behavioral Petri Nets(Réseaux de Petri comportementaux)

Un modèle BPN (Behavioral Petri Net) a été introduit afin de décrire le comportement causal du système. Par cela, il peut être considéré comme la version réseau de Petri d'un modèle causal. En fait, l'une des principales caractéristiques pour lesquelles un tel modèle de réseau est défini est de trouver une alternative aux formalismes logiques de telle sorte que la sémantique précise d'un modèle causal puisse être donnée en termes de structure et comportement de réseau de Petri. Pour plus de détails sur les BPN, une description complète peut être trouvée dans [9].

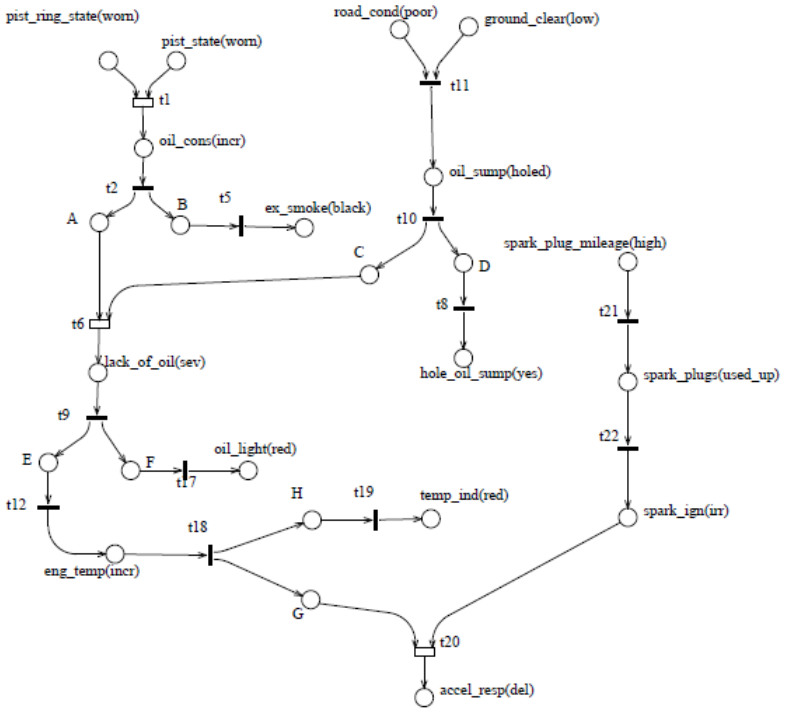


Figure 1.3: Exemple de BPN[3]

1.4.1 Modèle BPN

Définition 6 *un BPN est un quadruple N=(P,T_N,T_{OR},F)où :*

1. $T_N \cap T_{OR} = \emptyset$
2. *N=(P,*$T_N \cup T_{OR}$*,F) est un RdP ordinaire .*
3. F^+ *est irréflexif .*
4. $\forall p \in P(|^\bullet p| \leq 1) et (|p^\bullet| \leq 1)$
5. $\forall p_1, p_2 \in P((^\bullet p_1 =^\bullet p_2) et (p_1^\bullet = p_2^\bullet) \rightarrow p_1 = p_2)$
6. $\forall t \in T_N : (|^\bullet t| = 1 et |t^\bullet| \geq 0) ou (|^\bullet t| \geq 0 et |t^\bullet| = 1)$
7. $\forall t \in T_{OR} : (|^\bullet t| = 2 et |t^\bullet| = 1)$

L'ensemble des transitions d'un BPN est partitionné en deux sous-ensembles, où les transitions T_N représentent les transitions habituelles (conjonction des causes), les transitions T_{OR} sont destinées à représenter la disjonction logique 'ou'. Une transition (représentée graphiquement comme un rectangle vide) est dite une transition OU et elle représente une macro transition dont la sémantique peut être donnée en terme de réseau de Petri avec des arcs inhibiteurs.

En comparaison avec les modèles causaux, chaque place représente une instance particulière d'une entité (ou état) du modèle causal, les instances d'états initiaux sont représentées par des places sources, tandis-que les instances de manifestations sont représentées par des places puits. Les transitions sont utilisées pour représenter les relations de cause-effet entre les instances d'états dont les dépendances sont supposées fonctionnelles [9].

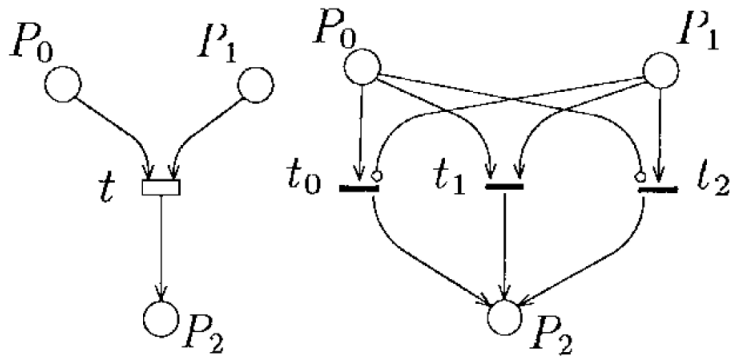


Figure 1.4: Sémantique de transition Or

1.4.2 B-W analyse

L'analyse B-W, une technique d'analyse particulière, consiste en une analyse d'accessibilité vers l'arrière effectuée avec deux types différents de jetons appelés respectivement jetons inhibiteurs (blanc) et jetons normaux (noir). Le sens du jeton noir est que nous pouvons associer une condition à une place et un jeton noir (normal) dans une place signifie que la condition est satisfaite. Par contre, un jeton blanc (anormal) représente une condition non satisfaite dans la place. Si aucune contrainte est imposée sur la place, elle ne sera pas marquée (vide), ça nous conduit à considérer trois valeurs logiques $\{vrai, faux, inconnu\}$ [3].

Définition 7 *Soit* $N=(P, T_N, T_{OR}, F)$ *un BPN, un B-W marquage est une fonction* $\mu(p) \rightarrow \{b, w, 0\}$. *Si* $\mu(p) = b$ *alors la place p est marquée par un jeton normal (black), si* $\mu(p) = w$ *alors la place p est marquée par un jeton anormal-inhibiteur-(white), si* $\mu(p) = 0$ *alors la place p est vide.*

Conclusion

Dans ce chapitre, nous avons présenté les réseaux de Petri et leurs techniques d'analyse comme un moyen de représenter des modèles causaux, puis de les remplacer par des versions colorés pour réduire la taille des modèles. Nous avons également introduit les réseaux de Petri comportementaux (BPN) en supposant qu'il s'agit d'une autre version de réseaux de Petri qui représente un modèle causal, et qui vise à résoudre un problème de diagnostic basé sur un modèle.

Dans le chapitre suivant, nous présenterons deux modèles (les réseaux de Petri colorés et les réseaux de Petri de comportement colorés) et les utiliserons pour représenter un modèle causal afin de compléter le processus de diagnostic basé sur un modèle.

Chapter 2

Transformation de BPNs aux CBPNs

Introduction

Le processus de diagnostic fait partie importante de la réparation des systèmes, en recherchant les causes profondes qui conduisent à un dysfonctionnement (pannes). Récemment, le diagnostic à base de modèle (MBD) devient le plus pratique et le plus utilisable, au lieu des systèmes experts de première génération.

Il y a beaucoup d'outils de modélisation formelle qui nous permet de modéliser les systèmes, l'un de ces modèles est les BPNs qui sont caractérisés par une complexité spatiale même pour des systèmes de taille modérée [9]. L'objectif de notre projet est l'implémentation d'un outil software qui produit un nouveau modèle à partir d'un BPN. Ce modèle sera utilisé pour accomplir la tâche de diagnostic au lieu de BPN mais l'analyse à base de ce modèle sera moins compliquée. Ce modèle est appelé CBPN(Colored Behavioral Petri Network).

Dans ce chapitre, nous allons présenter ce modèle CBPN, voir les avantages du CBPN dans le diagnostic basé sur un modèle en comparant par les difficultés auxquels le RdPC est confronté dans ce domaine, puis on va introduire une procédure qui permet de passer de BPN à CBPN.

2.1 Colored Behavioral Petri Net(réseaux de Petri colorés comportementaux)

L'article [7] présente une nouvelle approche basée sur les RDPC pour le diagnostic basé sur un modèle causal. Il introduit une classe particulière de RdPC appelés réseaux de Petri colorés comportementaux (CBPN) pour décrire le comportement causal du système en cours d'examen. La principale caractéristique des CBPN est que l'étiquetage des transitions avec des matrices comme alternative pour déterminer les différentes manières dont une transition peut se déclencher (franchir). Chaque ligne de la matrice détermine les couleurs des jetons impliquées lorsque la transition se déclenche par rapport à une manière de franchissement spécifique. Il convient de noter que dans les CBPN, l'expression d'arc consiste en une variable typée pour un arc d'entrée d'une transition et une fonction sélective pour sa sortie.

2.1.1 Colored Behavioral Petri Net(Formellement)

Définition 8 *Un CBPN est un 6-tuple* $N = (\Sigma, P, T, A, C, FW)$ *où :*

- (Σ, P, T, A, C) *est un RdPC.*
- $FW : T \rightarrow MAT_{n,m}(\Sigma \cup \{\varepsilon\})$.
- A^+ *est irréflexif.*

Un CBPN est un RdPC particulié qu'est introduit principalement pour décrire le comportement causal du système en cours d'étude. L'ensemble des places peut être partitionné en trois sous-ensembles: Causes initiaux (Ic), états internes (Is) et manifestations(Mn).
$P = (Ic \biguplus Is \biguplus Mn)$,où $Ic = \{p | p \in P, {}^\bullet p = \emptyset\}$, $Mn \subseteq \{p | p \in p, p^\bullet = \emptyset\}$ et $Is = P \backslash (Is \bigcup Mn)$. Les transitions peuvent être classées comme transitions *fork*, $t \in T : |t^\bullet| > 1$, transitions *Join*, $t \in T : |{}^\bullet t| \geq 1$. De plus, elles sont étiquetées avec leurs matrices FW (Firing Way) de façons de franchissement. Soit t une transition avec n manière de franchissement et m places avec lesquels elle est connectée ($m = |{}^\bullet t| + |t^\bullet|$). La matrice de façon de franchissement $FW = [c_{i,j}]$ de t est une $n \times m$ de couleurs $c_{ij} \in \bigcup_{\omega \in \Sigma^\omega}$ il y compris la couleur vide ε.

La matrice FW contient une ligne pour chaque façon de franchissement et une colonne pour chaque place connectée. Ainsi, le c_{ij} est constitué de la couleur retirée de (ajoutée à) la place j lorsque t est franchisée par rapport à la façon de franchissement i. Notez que si t est une transition fork, elle est destinée à diviser la couleur du jeton d'entrée (lorsque plusieurs conséquences sont présentes). C'est pourquoi les couleurs d'une même rangée sont similaires. Alors que si elle est *join*, nous rencontrons trois cas d'utilisation possibles. La première est comme d'habitude lorsque t est une conjonction de causes. La deuxième est constituée de Ou logique où t devient franchissable si au moins une de ses places d'entrée est marqué. Par conséquent, nous pouvons dériver la dernière, qui est Ou exclusif où t devient activée si et seulement si une seule de ses places d'entrée est marquée. La matrice FW peut être décomposée en $FW_{in_{n \times |{}^\bullet|t}}$ et $FW_{out_{n \times |t^\bullet|}}$ comme sous-matrices d'entrée et de sortie respectivement.

Définition 9 *Un marquage initial d'un CBPN est un marquage safe μ_0 tel que :$\forall p \in P : \mu_0(p) \neq \emptyset \rightarrow p \in Ic$.*
Un CBPN marqué est le pair (N,μ) où N est un CBPN et μ est un marquage avec la propriété suivante:$\forall p \in P : |\mu(p)| \leq 1$.

Définition 10 *Un CBPN est safe si:*

$$\forall p \in P, \forall \mu \in R(N, \mu_0) : |\mu(p)| \leq 1.$$

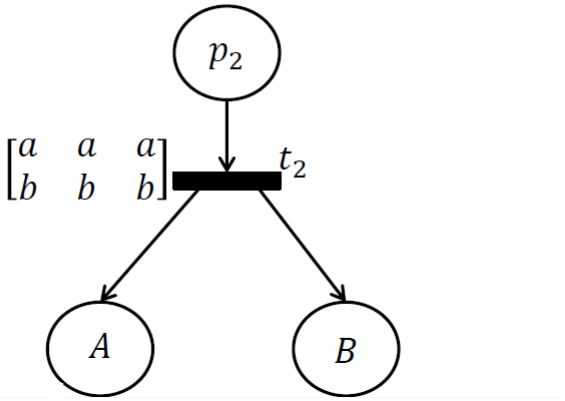


Figure 2.1: Exemple de CBPN

2.1.2 Réseaux de Petri colorés pour le diagnostic à base de modèle

Il a été remarqué que les BPNs offraient une description claire et précise du système à diagnostiquer, en plus de capturation de tous les aspects concernant la validation des modèles causaux. En fait,une représentation BPN devient assez complexe même pour les systèmes de taille modeste. La raison principale est que nous avons un seul type de jeton, qui nous permis de modéliser chaque valeur (instance) d'un état par une place. De plus, pour chacune il y a un chemin d'exécution comme un sous-réseau. Notez que le problème mentionné vient avec les places et aussi avec les transitions. D'une part, un sous-ensemble de places appartient au même état modélisé. D'un autre côté, un sous-ensemble de transitions partage le même domaine de places (entrées et sorties) et effectue la même action avec eux. Ici, nous utilisons le terme domaine de place pour désigner l'ensemble des places correspondant aux valeurs d'un état.
Pour cela, nous référons aux RdPCs mentionnés dans la section [1.2]. Nous pouvons définir une correspondance (mapping)entre un BPN et un RdPC comme suivant:

- Remplacer l'ensemble de places $\{p_1, ..., p_n\}$ qui ont le même domaine par une seule place p et on attache à p un ensemble de couleur $\{c_1, ..., c_n\}$ de telle façon que chaque couleur réfère à une place.

- Remplacer l'ensemble de transitions $\{t_1, ..., t_m\}$ qui ont le même do-

maine pour les entrées et les sorties par une seule transition t, qui peut franchir avec m différentes façons, chacune correspondante à une transition t_i.

- le marquage d'un RdPC est safe, chaque place peut contenir au maximum un seul jeton à la fois. Dans un marquage initial sauf les places sources peuvent être marquées.

Les RdPCs n'ont pas une représentation complète des transitions Or,c'est pourquoi il est proposé d'utiliser nouvelles transitions et des arc inhibiteur,nous avons à nouveau le problème de la complexité. Pour accomplir de processus de diagnostic utilisant un RdPC on doit faire une analyse en arrière sur un graphe d'accessibilité qui peut être vu comme une analyse en avant sur un graphe d'accessibilité d'un modèle RdPC inversé.pour inverser un modèle on doit:

- inverser la direction des arcs.
- inverser les fonctions de l' RdPC.

Nous avons mentionné qu'une transition peut se franchir de plusieurs manières, en plus de l'utilisation d'expressions d'arc pour déterminer les couleurs de jeton impliquées lorsqu'elle franchie. En effet, le processus d'investissement des arcs avec les expressions pour certains cas et difficile et impossible pour autres cas,l'article [6] traite le probléme de diagnostic utilisant les RdPCs.
L'article[7] présente une astuce pour faciliter la manipulation des expressions d'arcs.

2.1.3 CBPN pour le diagnostic à base de modèle

Afin de simplifier la tâche d'analyse, des propositions dans l'article[7] d'utiliser des matrices comme étiquettes pour les transitions où les différentes manières de franchissement sont déterminées . Chaque ligne de la matrice correspond à une manière de franchissement et détermine les couleurs des jetons impliqués.En addition, chaque colonne correspond à une place et affiche les couleurs des jetons voulues(qu'ils soient consommés ou produits) par cette transition. En fait, cette proposition est la caractéristique clé d'une classe particulière de RdPCs appelés CBPNs .Il convient de noter que nous n'avons pas l'intention de négliger les expressions d'arc, mais elles seront utilisées comme variables typées et expressions sélectives. Dans la section 2.1, une description complète des CBPN est donnée.

2.2 Processus de diagnostic à base de CBPNs

Il est clarifié dans la section précédente l'utilité d'utilisation de CBPN dans le diagnostic grâce à des matrices associées aux transitions qui rendent la tache d'analyse en arrière moins complexe,dans cette section on va formaliser le problème de diagnostic en terme de CBPN.

2.2.1 Formalisation de problème de diagnostic utilisant CBPNs

Définition 11 *Un problème de diagnostic CBPNDP de CBPN est définie comme*
CBPNDP=(N,M^{init},$< M^+$,$M^- >$),où:

- $M^{init} = \{(p,c)|p \in Ic, c \in C(p)\}$
- $M^+ = \{(p,c)|p \in Mn, c \in C(p), \mu^{OBS}(p) = c\}$
- $M^- = \{(p,c)|p \in Mn, c \in C(p), \mu^{OBS}(p) \neq c\}$

N est un CBPN correspondant à un modèle causale ,M^{init} est un ensemble de couples (p,c) où en termes desquels le diagnostic sera donné. $< M^+, M^- >$ représentent l'observation obtenue, sont deux ensembles de couples (p,c) correspondant respectivement à Ψ^+, Ψ^-.

Définition 12 *Soit (N, μ) un CBPN marqué et le pair(p,c) ,$p \in P, c \in C(p)$:*

$$(N,\mu) \vdash (p,c) \longleftrightarrow \exists \mu^{'} \in R(N,\mu) : \mu^{'}(p) = c$$

Définition 13 *Soit (N,μ) un CBPN marqué et le pair(p,c) ,$p \in P, c \in C(p)$:*

$$(N,\mu) \nvdash (p,c) \longleftrightarrow \exists \mu^{'} \in R(N,\mu) : \mu^{'}(p) \neq c$$

Définition 14 *Soit un problème de diagnostic CBPNDP=(N,M^{init},$< M^+$,$M^- >$),un marquage initial μ^{init} est une solution à CBPNDP si :*

$$(N,\mu^{init}) \vdash M^+ \text{ et } (N,\mu^{init}) \nvdash M^-$$

Un marquage initial qui nous permet d'arriver à une observation obtenue n'existe pas dans M^- est un marquage solution pour CBPNDP.

2.2.2 Résolution de problème de diagnostic

Maintenant que nous savons comment définir le problème de diagnostic avec les CBPNs, on va accomplir la tâche de raisonnement, évidemment en exploitant les techniques d'analyse des RdPs classiques. Puisqu'un problème de diagnostic peut être considéré comme un problème abductif, un tel raisonnement devrait être effectué par une analyse en arrière sur un graphe d'accessibilité. Cela permet, à partir d'un marquage, de déterminer les marquages à partir desquels le marquage donné est accessible. En fait, cela correspond tout à fait à l'analyse du réseau inversé.

Une caractéristique importante de CBPN est que l'inversion de modèle peut être accomplir par une simple manipulation de matrices(FW) et l'inversion de la direction d'arc. Ici, nous avons éliminé le problème de l'inversion des expressions d'arc.

Inversion du modèle CBPN

Pour inverser un modèle CBPN il suffit d'inverser les directions d'arcs et inverser les matrices associées aux transitions.

Notons $t^{'}$ la transition correspondant à la transition t après inversion de la direction des arcs environnants de t. Donc, $t^{'\bullet} =^{\bullet} t$ et ${}^{\bullet}t^{'} = t^{\bullet}$. Rappelant que la matrice FW peut être décomposée en deux sous-matrices FW_{in} et FW_{out} où $FW_{in}(t) = FW_{out}(t^{'})$ et $FW_{out}(t) = FW_{in}(t^{'})$. On utilise la signification$(-^{B})$pour dénoter le processus d'analyse en arrière. La matrice $FW^{B}(t)$ correspondant à la matrice $FW_{in}(t)$ où :$FW^{B}_{in}(t) = FW_{out}(t)$ et $FW^{B}_{out}(t) = FW_{in}(t)$. Par conséquent,$E^{B}(x, y)$ se réfère à l'expression de l'arc(x,y), avec $x, y \in P \cup T$, il doit être clair que $E^{B}(x, y)$ est une variable typée, v_y, si E(x,y) est une fonction sélective et vice versa.

Définition 15 *Soit (N, μ) un CBPN marqué, une transition t est franchissable en arrière dans un marquage μ,(dénoté $\mu[t >^{B}$) si:*

$$\forall p \in t^{\bullet} : E^{B}(t, p) < b > \leq \mu(p) \wedge \nexists\ t^{'} \succ t : \mu[t^{'} >^{B}.$$

Le franchissement de t en arrière rend le réseau progressé une étape en arrière, conduire à produisant un nouveau marquage $\mu^{'}$, donné comme suivant:

$$\mu^{'}(p) = \mu(p) - E^{B}(t, p) < b > + E^{B}(p, t) < b >$$

Définition 16 *Soit (N, μ) un CBPN marqué, μ est dit incohérent si:*

$$\exists t \in T, \exists p, p^{'} \in t^{\bullet} : \mu(p) \neq \mu(p)$$

L'incohérence vient lorsque on trouve des places de sortie d'une transition fork marquées par des couleurs différentes. Il y a des cas où on trouve des places vides (non marquées), nous introduisons un autre concept appelé forced transition (transition forcée) qui est introduit dans [4] pour les BPNs. Ici, nous supposons que toutes les places non marquées nécessaires pour le franchissement en arrière de la transition souhaitée sont marqués par la même couleur que celles marquées.

Définition 17 *Soit (N, μ) un CBPN marqué et t une transition fork tel que $^{\bullet}t = \{p\}$ et $t^{\bullet} = \{p_1, .., p_m\}$, on dit que t est forcée au marquage μ si:*

- *t n'est pas franchissable en arrière dans μ .*
- *$\exists p_i (1 \leq i \leq m) \mid \mu(p_i) \neq \emptyset$.*
- *μ n'est pas incohérent .*
- *$\nexists t^{'} \succ t$ où $t^{'}$ est franchissable en arrière ou forcée en μ.*

Si t est forcée dans μ alors $\forall p, p^{'} \in t^{\bullet}$ où $\mu(p) = \emptyset \mu(p^{'}) \neq \emptyset$, considérons $\mu(p) = \mu(p^{'})$.

Graphe d'accessibilité en arrière

En partant d'un marquage $\mu \sqsubseteq \mu^{OBS}$ tel que $\forall (p, c) \in M^{+} : \mu(p) = c$, les nœuds terminaux du graphe d'accessibilité obtenu (par franchissement en arrière) sont soit des marquages initiaux soit des marquages incohérents. $\mu^{ini} \subseteq 2^{M^{init}}$ est l'ensemble de marquages initiaux μ_i où $\exists \mu^{'} \in R(N, \mu_i)$: $\mu \sqsubseteq \mu^{'}.\mu_i$ est dit une solution au CBPNDP s'il traite les deux contraintes de Définition 14. Il est nécessaire d'appliquer une analyse d'accessibilité en avant pour assurer que μ_i est cohérent avec M^{-} ou non(μ_i ne doit atteindre aucun élément de M^{-}).

Afin d'exécuter le processus de raisonnement en une seule phase, nous utilisons un nouveau concept qui est la couleur inhibiteur (on le dénote avec le symbole $c^{w}, c \in \Sigma$). Nous cherchons par là, à travers une analyse en arrière, bloquer la production de certaines couleurs [6]. Si $C(p) = \{c_i, c_j\}$ et $\mu(p) = c_i$, le marquage de p peut être donné par $\mu(p) = c_j^{w}$.
Afin de montrer comment les diagnostics sont calculés à l'aide de l'analyse CW, on utilise le CBPN suivant :

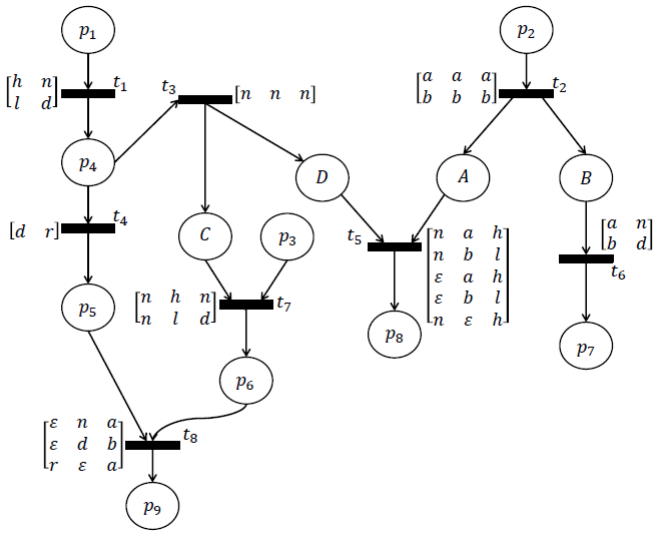


Figure 2.2: exemple de CBPN[7]

On considère le problème de diagnostic $DP = (N, M^{init}, < M^+, M^- >$) avec l'observation $\mu^{OBS} = p_8(h), p_9(a)$ ($p_9(a)$ référant à $p_9(b^w)$, $M^+ = \{(p_8, h)\}$ et $M^- = \{(p_9, b)\}$. On commence par le marquage $\{p_8(h), p_9(a)\}$ afin d'obtenir ce graphe d'accessibilité en arrière :

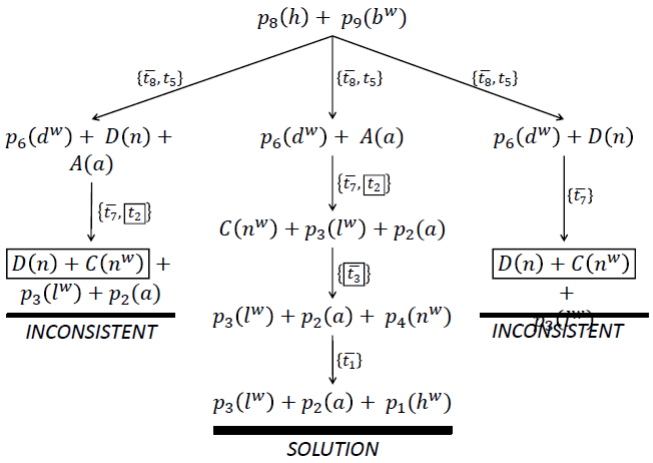


Figure 2.3: Graphe d'accessibilité en arrière [7]

2.2.3 Procédure de transformation de BPN à CBPN

Dans la suite, on utilise une procédure [7] qui nous aide à développer un outil qui transforme un BPN $N^{'} = \{P^{'}, T^{'}_{N}, T^{'}_{OR}, F^{'}\}$ vers un CBPN $N = (\Sigma, P, T, A, C, FW)$ équivalent.
La procédure contient six fonctions :

1. dt.
2. dp .
3. *In* .
4. *Out* .
5. *clr* .
6. *update* .

On utilise les fonctions d_p et d_t pour déterminer le domaine des places et le domaine des transitions respectivement pour indiquer $x \in (P^{'} \cup T^{'}_{N} \cup T^{'}_{OR})$ tel que ,$d_p : P^{'} \rightarrow P$ et $d_t : (T^{'}_{N} \cup T^{'}_{OR}) \rightarrow T$.
Les fonctions $In, Out : (T^{'}_{N} \cup T^{'}_{OR}) \rightarrow 2^{P}$, associant à chaque transition un ensemble de places, ont les mêmes interprétations que $^{\bullet}t$ et $t^{\bullet}$ respectivement, mais en termes de domaines. La fonction $clr : p^{'} \rightarrow \Sigma$ renvoie la couleur correspondante à une place$p \in P^{'}$. La fonction *update* qui fait la mise à jour de la matrice de transition en ajoutant une nouvelle manière de franchissement comme une ligne de couleurs.

Procedure 1 Translation
Input BPN $N' = (P', T'_N, T'_{OR}, F')$
Output CBPN $N = (\Sigma, P, T, A, C, FW)$

1: $\Sigma \leftarrow P \leftarrow T \leftarrow \emptyset$
2: **for** $\{p_1, \ldots, p_n\} \subset P'$ *such that* $d_p(p_1) = \ldots = d_p(p_n) = p$ **do**
3: $P \leftarrow P \cup \{p\}$
4: $C(p) \leftarrow \{clr(p_1), \ldots, clr(p_n)\}$
5: $\Sigma \leftarrow \Sigma \cup C(p)$
6: **end for**
7: **for** $\{t_1, \ldots, t_m\} \subset (T'_N \cup T'_{OR})$ *such that* $d_t(t_i) = \ldots = d_t(t_m) = t$ **do**
8: $T \leftarrow T \cup \{t\}$
9: **for** $t' \in \{t_1, \ldots, t_m\}$ **do**
10: **if** $t' \in T_N$ **then**
11: ▷ Let ${}^\bullet t' = \{p_1, \ldots, p_n\}$, $t'^\bullet = \{p'_1, \ldots, p'_k\}$
12: $update(FW(t), [clr(p_1) \ldots clr(p_n)\ clr(p'_1) \ldots clr(p'_k)])$
13: **else**
14: ▷ Let ${}^\bullet t' = \{p_1, p_2\}$, $t'^\bullet = \{p'\}$
15: $update(FW(t), [clr(p_1)\ clr(p_2)\ clr(p')])$
16: $update(FW(t), [clr(p_1)\ \varepsilon\ clr(p')])$
17: $update(FW(t), [\varepsilon\ clr(p_2)\ clr(p')])$
18: **end if**
19: **for** $p_i \in In(t_1)$ **do**
20: $A \leftarrow A \cup \{(p_i, t)\}$
21: $E((p_i, t)) \leftarrow v_{p_i}$
22: **end for**
23: **for** $p_i \in Out(t_1)$ **do**
24: $A \leftarrow A \cup \{(t, p_i)\}$
25: $E((t, p_i)) \leftarrow f(v_{p_1}, \ldots, v_{p_n}, p_i)$
26: **end for**
27: **end for**
28: **end for**

Figure 2.4: Procédure de transformation [7]

La procédure est décomposée en deux boucles, l'une (ligne 2-6) fait la transformation des places et l'autre (ligne 7-27) fait la transformation des transitions.

Exemple 2 *On exploite le BPN suivant pour le transformer à l'aide de la procédure précédente:*

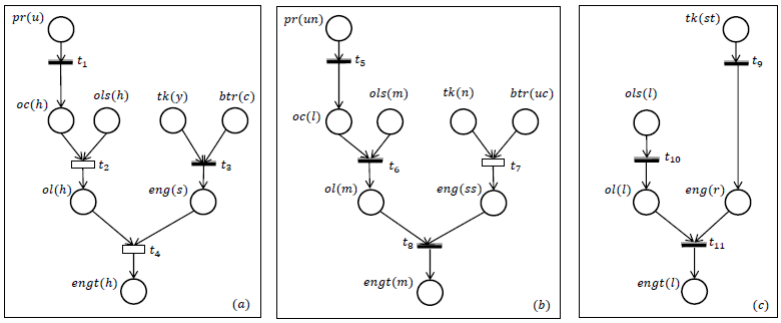


Figure 2.5: Exemple de BPN

La transformation de modèle BPN Figure 2.5 en utilisant la procédure précédente nous donnera le CBPN schématisé dans Figure 2.6.

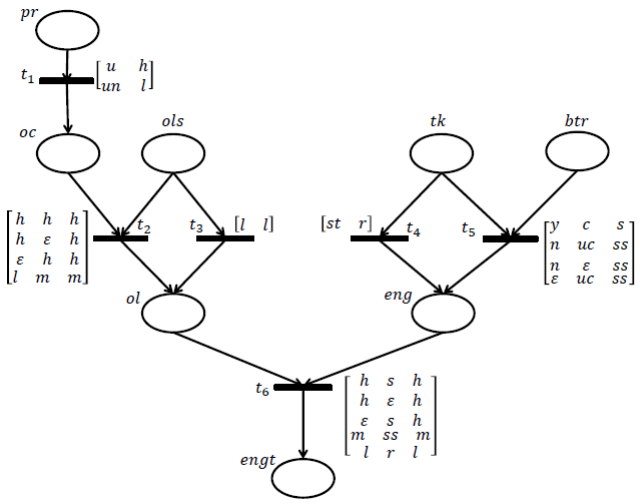


Figure 2.6: Modèle CBPN

Conclusion

Dans ce chapitre, nous avons introduit le modèle CBPN et nous avons vu les différences entre l'utilisation des RdPCs et des CBPNs pour le diagnostic basé sur un modèle. Nous avons terminé le chapitre avec la procédure de passage des BPNs vers des CBPNs.

Dans le chapitre suivant on va donner la procédure de réalisation de l'outil proposé pour convertir un BPN à un CBPN.

Part II

Développement d'un outil de transformation d'un BPN vers un CBPN

Chapter 3

Analyse et modélisation

Introduction

Après avoir appris les points théoriques nécessaires, nous passons au développement de l'application. Ce chapitre est composé de trois sections, nous commençons par présenter l'objectif de notre projet. Ensuite, nous donnons des modèles abstrait de l'application et les diagrammes de classe adéquats de l'application. Enfin, on va décrire les structures de données et les algorithmes qu'on a utilisé lors du développement.

3.1 Spécification et analyse de besoin

L'analyse des besoins est la première étape dans le cycle de vie d'un logiciel, elle sert à définir précisément les besoins de notre logiciel, les services qui seront rendus à l'utilisateur et les contraintes sur lesquelles ce logiciel devra fonctionner. Comme il est clarifié dans le chapitre précédent, notre algorithme prend comme entrée un modèle BPN qui a quelques contraintes et propriétés pour générer un modèle CBPN.

Comme une première étape pour accomplir cet outil, les tâches principaux qui doivent mises en points sont les suivantes :

- l'outil doit présenter un moyen simple et interactif pour l'acquisition du modèle. Ce modèle est décrit à l'aide d'un BPN.
- Si une place est produite, il est nécessaire d'attribuer et marquer cette place par un domaine et une valeur qui peut être modélisé dans cette place.
- Il est nécessaire de fournir une technique de sauvegarde utilisant cet outil.
- Enfin,l'outil doit accomplir la transformation de modèle d'entrée pour générer le modèle de sortie, ce qui doit être enregistré et visualisé dans une interface.

3.2 Conception

La conception est un processus qui a plusieurs niveaux, commençant par une conception globale plus abstraite qui sera raffiné jusqu'à atteindre un niveau moins abstrait à partir duquel il est simple de générer le code source.

3.2.1 Conception globale

L'objectif de la conception est de montrer l'ensemble des composants et sous composants constituant l'outil et les relations existantes entre eux.

Notre outil peut être vu en premier niveau d'abstraction comme un ensemble de trois composants. Le premier composant reçoit des entrées de l'utilisateur, le deuxième composant exploite les sorties du premier composant. Le troisième composant affiche les résultats de deuxième composant. On peut schématiser cette interaction entre les composants par Figure 3.1 :

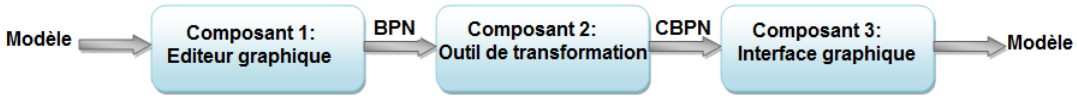


Figure 3.1: Interaction entre les composants

Voici une brève description de chaque composant:

Éditeur graphique: Ce composant représente l'interface qui interagit avec l'utilisateur d'une part et avec l'autre composant d'autre part. Il permet à l'utilisateur de modéliser le comportement d'un système par un BPN qui sera utiliser comme une entrée pour le composant 2.

Outil de transformation: Le rôle de ce composant est le principal et le plus important dans notre projet, car il fonctionne pour convertir le modèle (BPN) qui est prend entré en fonction de l'algorithme de transformation vers un autre modèle (CBPN)équivalent.

Interface graphique: Lorsque la conversion est terminée, l'interface nous permet d'afficher le modèle finale (CBPN).

Après avoir fixé l'objectif de notre application et les services nécessaires qu'il doit fournir,on peut donner la Figure 3.2 pour expliquer l'architecture générale de notre application:

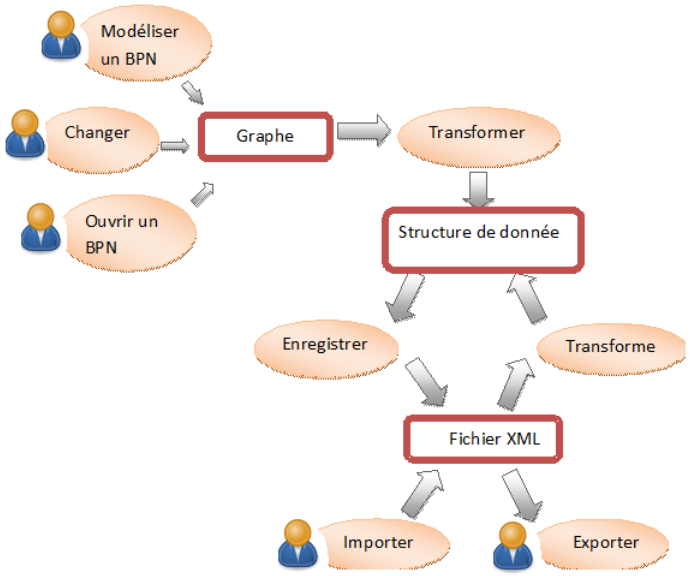


Figure 3.2: Architecture global de l'application

Selon l'architecture globale de notre application, l'utilisateur peut créer un modèle de type BPN qui sera stocké dans un fichier avec extension XML. Ce fichier sera généré après la désignation du modèle par l'utilisateur ou il existe déjà (anciens). Une fois le fichier est prêt, nous pouvons appliquer le processus de transformation partant de fichier XML pour avoir la structure de donnée du modèle CBPN qui sera stocké dans un fichier avec extension XML aussi.

Pour plus de détails, on peut voir la structure de chaque composant et sa fonctionnalité comme suivant:

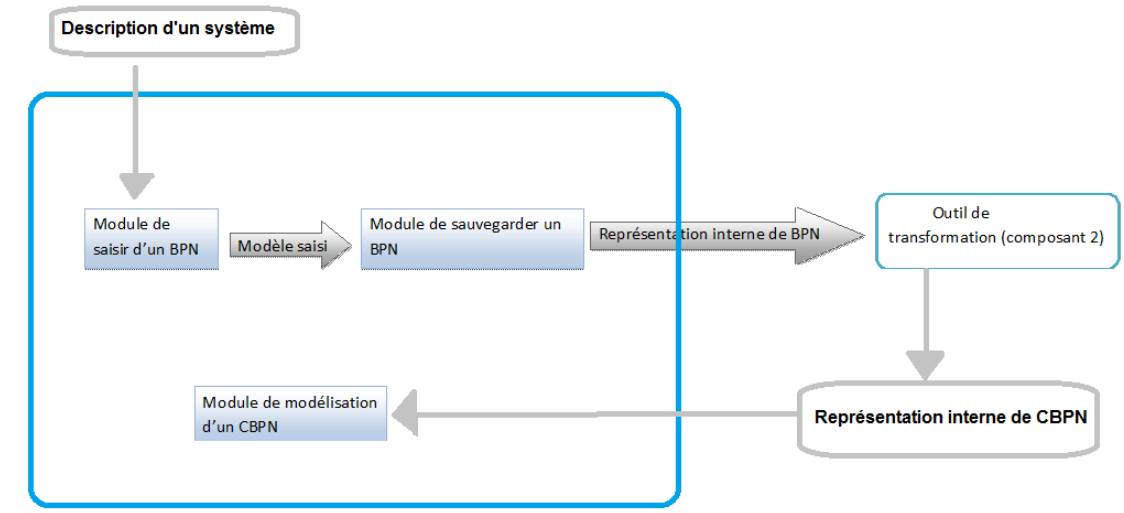


Figure 3.3: Module de modélisation(Éditeur graphique)

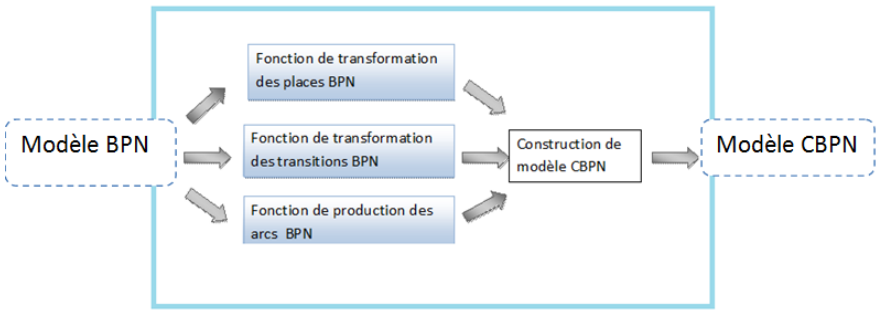


Figure 3.4: Module de transformation

Diagrammes de classe

On va présenter les classes qu'on a utilisé et les différentes relations entre celles-ci par deux diagrammes de classes .

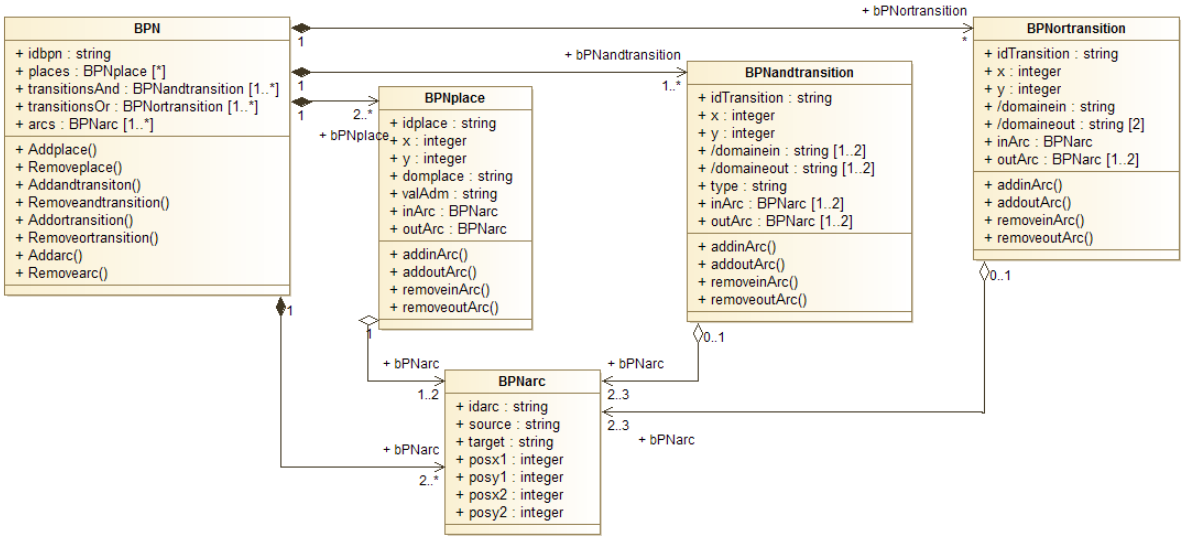


Figure 3.5: Diagramme de classe BPN

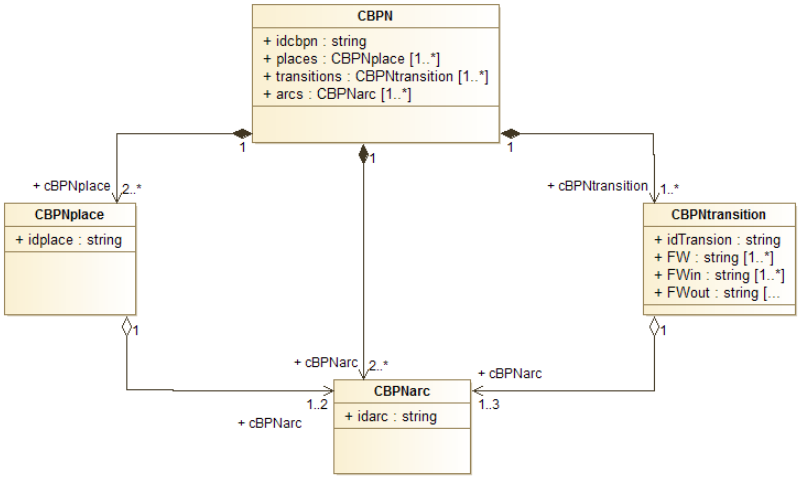


Figure 3.6: Diagramme de classe CBPN

3.2.2 Conception détaillée

La conception détaillée consiste à fournir pour chaque composant identifié dans le niveau précédent les algorithmes et les structures de données permettant d'implémenter les tâches correspondantes.

Dans ce qui suit, nous allons présenter les structures de données et les algorithmes nécessaires pour implémenter les fonctionnalités de notre outil qui sera détailler dans le chapitre 4 .

Les structures des données

Les structures de données qui correspondent au modèle d'entrée BPN:

Classe BPNplace :

Cette classe est caractérisée par les attributs suivants:

idplace: c'est l'identificateur de place de type entier, qui sera envoyé par la fonction qui désigne la place (cercle).

x,y: de type entier, ces deux attributs représentent les coordonnés de la place(cercle) dans l'interface graphique.

domplace: une chaine de caractères indiquant le domaine modélisé par la place.

valAdm: une chaine de caractères indiquant la valeur admissible qui peut être modélisée dans la place.

inArc: une liste contenant des objets de type BPNarc où la place représente la cible de l'arc(BPNarc). La taille de cette liste doit être 1 ou 0 .

outArc: une liste contenant des objets de type BPNarc où la place représente la source de l'arc(BPNarc). La taille de cette liste doit être 1 ou 0.

Classe BPNandtransition :

Les attributs les plus importants de cette classe sont:

idTransition: c'est l'identificateur de transition (de type AND) de type entier, qui sera envoyé par la fonction qui désigne la transition (rectangle noir).

x,y: de type entier, ces deux attributs représentent les coordonnés de la transition (rectangle noir) dans l'interface graphique.

inArc: une liste qui contient des objets de type BPNarc où la transition est la cible de cet arc (BPNarc), l'ajout d'un objet dans cette liste est limité par des conditions (une transition de type AND peut contenir deux arcs de sorties au cas où elle contient un seul d'entrée, comme elle peut contenir un arc de sortie en cas où elle contient deux arcs d'entrée).

outArc: une liste qui contient des objets de type BPNarc où la transition est la source de cet arc (BPNarc).

domainein: une liste de chaines de caractères, contenant les domaines des places d'entrée de la transition.

domaineout: une liste de chaines de caractères, contenant les domaines des places de sortie de la transition.

Classe BPNortransition:

L'objet de type BPNortransition a les mêmes attributs de l'objet BPNandtransition. Ce type de transition se diffère au l'autre type dans la représentation graphique (rectangle vide) et les arcs d'entrée et sortie où la transition de type OR contient un seul arc de sortie et deux arcs d'entrées.

Classe BPNarc :

Les attributs de cette classe sont:

idArc: c'est une chaine de caractères qui identifie l'arc (ligne), qui sera généré par la fonction uuid qui génère des identificateurs uniques .

source: objet de type BPNplace ou BPNandtransition ou BPNortransition.

target : objet de type BPNplace ou BPNandtransition ou BPNortransition.

posx1,posy1: de type entier, les coordonnées de début de ligne.

posx2,posy2: de type entier, les coordonnées de fin de ligne.

Classe BPN :

Cette classe est composée par des objets qui sont identifiés précédents:

idbpn: une chaine de caractères désignant le nom de modèle BPN.

places: liste des objets de type BPNplace.

transitionsAnd: liste des objets de type BPNandtransition.

transitionsOr: liste des objets de type BPNortransition.

arcs: liste des objets de type BPNarc.

Les structures de données qui correspondantes aux modèle de sortie CBPN:

Classe CBPNplace :

Cette classe contient les attributs idTransition, inArc, outArc et :

fwmat: (Firing way matrix), c'est une matrice qui décrit les manières de franchissement de transition (matrice de couleurs).

FWin,FWout: pour simplifier la tâche d'implémentation, on peut deviser la matrice fwmat en deux matrices, FWin matrice pour les couleurs de places d'entrées et FWout matrice pour les couleurs de places de sorties.

Classe CBPNarc :

Les attributs de cette classe sont similaires aux attributs de la classe BPNarc .

Classe CBPN :

Cette classe est composée par des objets qui sont identifiés précédents:

idcbpn: une chaine de caractères désignant le nom de modèle CBPN.

places: liste des objets de type CBPNplace.

transitions: liste des objets de type CBPNtransition.

arcs: liste des objets de type CBPNarc.

Les algorithmes

Pour modéliser un BPN on a besoin des fonctions suivantes:

Algorithm 1 Dessiner une place

Dessiner cercle(x,y)
Entrer le domaine et la valeur de cette place
Créer objet BPNplace p avec les attributs x,y,domaine et valeur
Ajouter p au liste de places BPN

Algorithm 2 Dessiner une transition And

Dessiner rectangle noir(x,y)
Créer objet BPNandtransition t avec les attributs x et y
Ajouter t au liste de transitions And de BPN

Algorithm 3 Dessiner une transition Or

Dessiner rectangle vide(x,y)
Créer objet BPNortransition t avec les attributs x et y
Ajouter t au liste de transitions Or de BPN

Algorithm 4 Dessiner Arc

Entrées: x,y
Sorties: A:BPNarc

s=ChercherSource(x,y) ,renvoyer l'objet qui représente la source d'arc
if (s!=null) **then**
  créer BPNarc a
  $a.x \leftarrow x$
  $a.y \leftarrow y$
  t=ChercherTarget($x^{'}, y^{'}$) ,renvoyer l'objet qui représente la cible d'arc
  if (t!=null) **then**
    if (s et t ne sont pas de même type) **then**
      $a.x2 \leftarrow x^{'}$
      $a.y2 \leftarrow y^{'}$
      Ajouter a au liste des arcs de sorties de s
      Ajouter a au liste des arcs de entrées de t
      Ajouter a au listes des arcs de BPN
      Dessiner un ligne qui relié entre (x,y) et ($x^{'}, y^{'}$)
    end if
  end if
end if

Une fois le BPN est prêt, on utiliser deux algorithmes ,l'un pour sauvegarder le BPN et l'autre pour le transformer.
Les algorithmes de sauvegarde de modèle(BPN ou CPBN)convertissent chaque classe en une balise parent et définissent ses attributs comme sous-balises.

Algorithm 5 Transformation de BPN à CBPN

Entrées: BPN $N' = (P', T'_N, T'_{OR}, F')$
Sorties: CBPN $N = (\Sigma, P, T, A, C, FW)$

for $\{t_1, \ldots, t_m\} \subset (T'_N \cup T'_{OR})$ *such that* $d_t(t_i) = \ldots = d_t(t_m) = t$ **do**
  $T \leftarrow T \cup \{t\}$
  for $t' \in \{t_1, \ldots, t_m\}$ **do**
    if $t' \in T_N$ **then**
      ▷ Let ${}^\bullet t' = \{p_1, \ldots, p_n\}$, $t'^\bullet = \{p'_1, \ldots, p'_k\}$
      $update(FW(t), [clr(p_1) \ldots clr(p_n)\ clr(p'_1) \ldots clr(p'_k)])$
    else
      ▷ Let ${}^\bullet t' = \{p_1, p_2\}$, $t'^\bullet = \{p'\}$
      $update(FW(t), [clr(p_1)\ clr(p_2)\ clr(p')])$
      $update(FW(t), [clr(p_1)\ \varepsilon\ clr(p')])$
      $update(FW(t), [\varepsilon\ clr(p_2)\ clr(p')])$
    end if
    for $p_i \in In(t_1)$ **do**
      $A \leftarrow A \cup \{(p_i, t)\}$
      $E((p_i, t)) \leftarrow v_{p_i}$
    end for
    for $p_i \in Out(t_1)$ **do**
      $A \leftarrow A \cup \{(t, p_i)\}$
      $E((t, p_i)) \leftarrow f(v_{p_1}, \ldots, v_{p_n}, p_i)$
    end for
  end for
end for

Algorithm 6 Dessiner CBPN

Entrées: C:CBPN

for p in C.places : **do**
  Dessiner une cercle sur (p.x,p.y)
end for
for t in C.transitions : **do**
  Dessiner un triangle sur (t.x,t.y)
end for
for a in C.arcs : **do**
  Dessiner un ligne de(a.posx1,a.posy1) à (a.posx2,a.posy2)
end for

Conclusion

Dans ce chapitre, nous avons montré l'analyse des besoins et les conceptions nécessaires comme une première étape qui nous aide pour la réalisation de notre outil.

Dans le chapitre suivant, on va mettre en œuvre les conceptions de notre projet.

Chapter 4

Implémentation

Introduction

Après les étape d'analyse et de conception mentionnées dans le chapitre précédent (chapitre 3), nous doivent passer aux prochaines étapes du projet, qui sont le codage et le test.

Ce chapitre comprend deux parties. La première présente brièvement les outils de développement et les langages de programmation que nous avons appris et utilisés dans la réalisation de notre projet. La deuxième partie présente les principaux résultats de l'implémentation de notre application finale.

4.1 Outils et langages de développement

Dans cette section, nous présentons les différents outils et langages, qui nous aident lors de la réalisation de notre projet au niveau de la programmation et l'édition du mémoire.

4.1.1 Langage de programmation Python

Python est un langage de programmation puissant et facile à apprendre qu'on a utilisé lors de réalisation de notre projet. Il dispose de structures de données de haut niveau et permet une approche simple mais efficace de la programmation orientée objet. En plus qu'un programme Python est court que les programmes d'autres langages, Python est un langage de programmation open source et non typé. Il est disponible pour tous ces systèmes d'exploitation (Windows, LINUX et Mac OS).

4.1.2 Éditeur de programmation PyCharm

PyCharm est un open source environnement de développement intégré utilisé pour programmer en Python. C'est un puissant assistant de codage, qui peut mettre en évidence les erreurs et offrir des correctifs rapides basés sur le débogueur Python intégré. C'est un éditeur approprié pour écrire et tester de nombreuses lignes de code, car il fournit une vue structurelle du projet et une navigation rapide dans les fichiers.

4.1.3 Package Tkinter(Tool Kit Interface)

Tkinter(*Tool Kit Interface*) [1]'est un package open source d'interface utilisateur graphique *GUI*, il est destiné au langage de programmation Python, nous avons préféré *Tkinter* pour développer les interfaces graphiques de notre application, car il est puissant et simple à apprendre. Il est disponible pour tous ces systèmes d'exploitation (Windows, LINUX et Mac OS).

4.1.4 Système de préparation de documents LaTeX

LaTeX est un système de composition puissant et flexible pour la production d'articles techniques et scientifiques de haute qualité. Il est basé sur le langage des balises. Il suit la philosophie de conception de séparer la présentation du contenu, ainsi les auteurs se concentrent sur ce qu'ils écrivent, pas sur ce qui est affiché, car l'apparence est gérée par LaTeX. L'apparence comprend de nombreux aspects, la structure du document (partie, chapitre, section, ..etc), des figures, des références croisées et bibliographies. Il est plus familier à un programmeur informatique, car il suit le cycle de compilation-exécution de code. Une fonctionnalité distinctive de LaTeX est son mode mathématique, qui permet de composer des formules complexes.

4.1.5 Éditeur (TeX MAKER)

TeX MAKER [2]est un éditeur gratuit et open source pour la rédaction des documents, basé sur le système LaTeX.Il prend en charge un correcteur orthographique puissant, la saisie semi-automatique du code et un afficheur *pdf*. Nous avons utilisé TeXMAKER pour rédiger notre rapport et faire notre présentation, car il produit des articles et des exposés de haute qualité.

4.1.6 XML(Extensible Markup Language)

XML est langage de balisage extensible, il représente un outil indépendant du logiciel et du matériel pour stocker et transporter des données. XML a été conçu pour être auto-descriptif. On a utilisé XML pour stocker les modèles d'entrées et de sorties.

4.2 Implémentation

En plus des outils mentionnés dans la section 4.1, notre outil de programmation est implémenté en utilisant la programmation orientée objet (POO), et un ensemble de logiciels et de matériels qui sont résumés dans le tableau suivant:

Software/Hardware	**Version**
OS	Microsoft Windows 7 , 64bits
CPU	Intel(R) Core(TM) i3-4500U CPU 1.80GHz 2.40GHz
RAM	4.00Go
L'interpréteur Python	3.7.0
PyCharm	2019.3.3
Tkinter	8.6

Table 4.1: Versions Logicielles/Matérielles

4.2.1 L'application principale

Figure 4.1 représente toute l'interface de notre application.

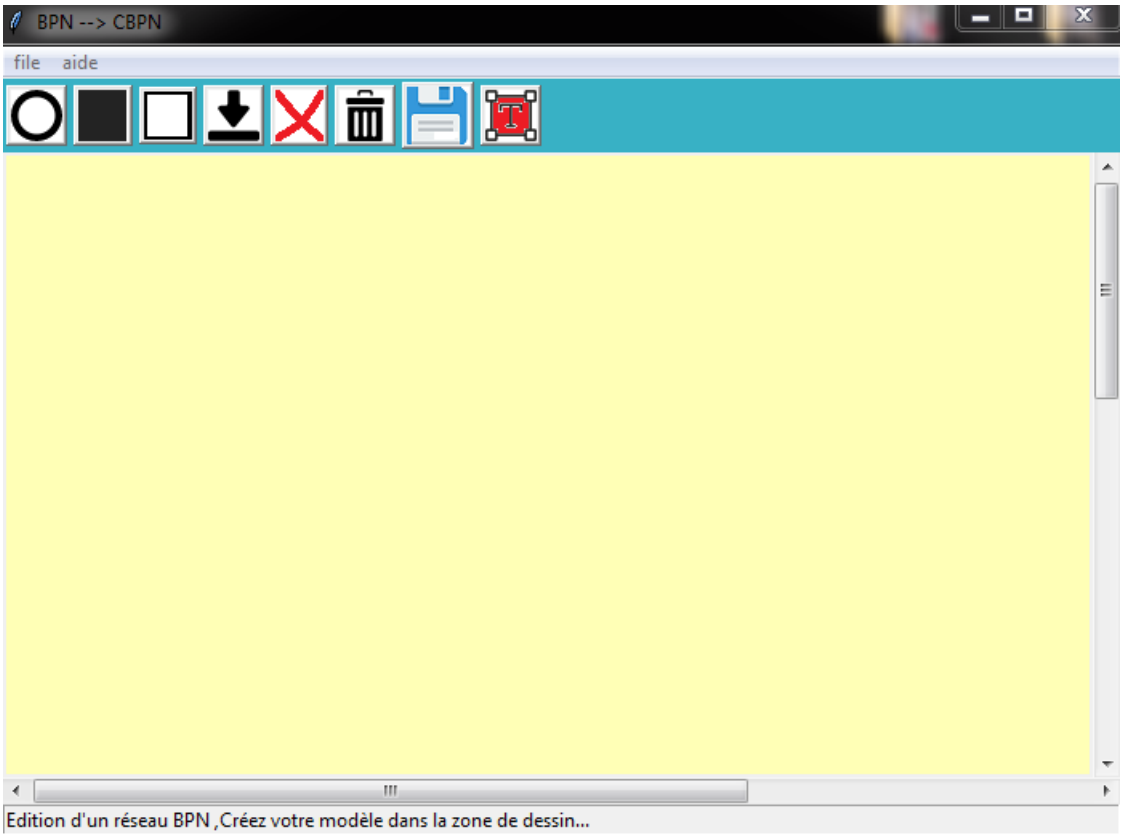


Figure 4.1: Interface Main.

On a implémenté une interface graphique simple et ordinaire pour nous permettre d'éditer le modèle d'entrée BPN et le sauvegarder dans un fichier XML. Cet outil d'édition se compose d'une barre de menus, d'une barre d'outils et d'une zone de dessin avec des barres de défilement et une barre d'état.

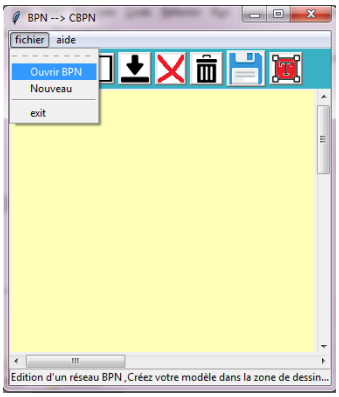


Figure 4.2: Menu Ouvrir BPN.

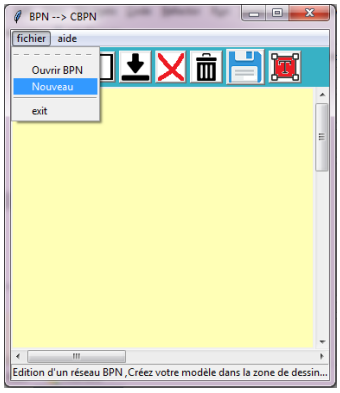


Figure 4.3: Menu Nouveau.

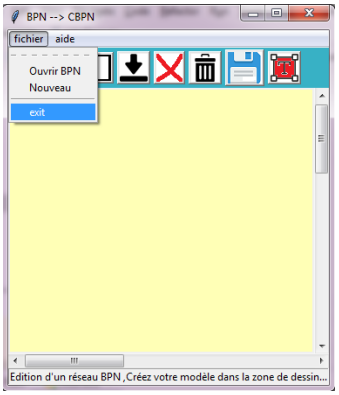


Figure 4.4: Menu Exit.

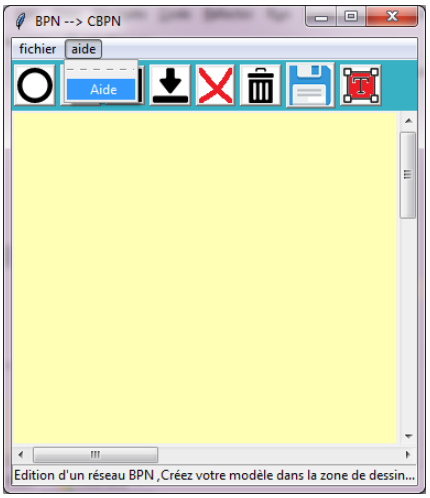


Figure 4.5: Menu Aide.

La barre de menus

L'application contient une barre de menus avec deux menus (Fichier et Aide) où le menu principale est le menu "Fichier" qui offre trois fonctionnalités à l'utilisateur *Figures 4.2, 4.3, 4.4*.

Ouvrir BPN: (*Figures 4.2*) Cette fonction permet à l'utilisateur de choisir un fichier XML qui contient la structure d'un BPN et de le dessiner dans la zone de dessin. Donc, elle permet d'ouvrir un BPN qui existe déjà.

Nouveau: *Figures 4.3* Cette fonction permet à l'utilisateur de créer une nouvelle zone de dessin pour éditer un nouveau BPN.

Exit: *Figures 4.4* permettez-nous de quitter l'application.

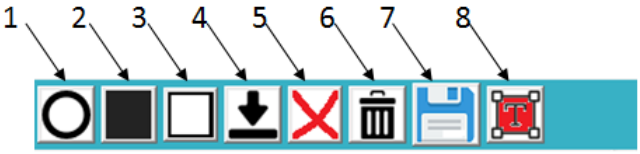


Figure 4.6: Barre d'outils.

La barre d'outils

La barre d'outils se compose de huit boutons qui nous aident à éditer, enregistrer et enfin convertir un BPN en un CBPN.

Dans la barre d'outils *Figure 4.6*, le bouton (1) est utilisé pour dessiner les places, le bouton (2) est utilisé pour dessiner les transitions de type AND, le bouton (3) est utilisé pour dessiner les transitions de type OR, le bouton (4) est utilisé pour dessiner les arcs, le bouton (5)est utilisé pour supprimer les objets individuels selon le choix de l'utilisateur, le bouton (6) pour la suppression de tout le modèle et pour vider la zone de dessin, le bouton (7) est utilisé pour enregistrer le modèle BPN qui est dans la zone de dessin dans un fichier XML, le bouton (8) est utilisé pour convertir le modèle BPN qu'est dans la zone de dessin ou qui est choisi par l'utilisateur.

4.2.2 Fonctionnalités de l'application

Notre outil consiste de trois modules. Un module pour éditer un BPN et le sauvegarder, un module pour convertir le BPN en un CBPN et un module d'édition et de sauvegarde du CBPN obtenu.

Édition d'un BPN

Nous avons commencé l'implémentation avec l'éditeur graphique *(Figure 4.1)* qui nous permet de créer un modèle BPN à l'aide de la barre d'outils *(Figure 4.6)*. L'utilisateur peut changer le nom de la place et remplir ses caractéristiques (le domaine et la modélisation de sa valeur) ou la supprimer

après la création de place immédiatement *(Figure 4.7)*, comme il peut supprimer ou modifier l'étiquette de transition aussi *(Figure 4.8)*. Une fois Le modèle BPN est prêt, l'utilisateur peut l'enregistrer au format XML *(Figure 4.9)*.

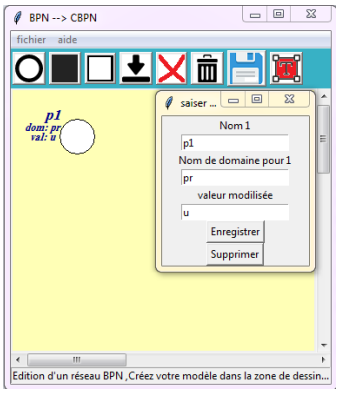


Figure 4.7: Création d'une place.

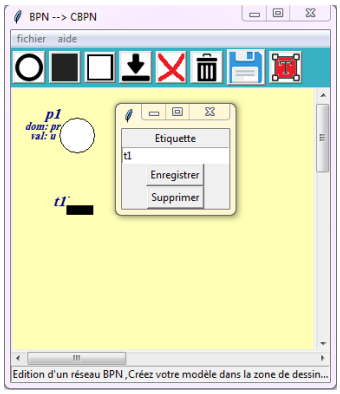


Figure 4.8: Création d'une transition And.

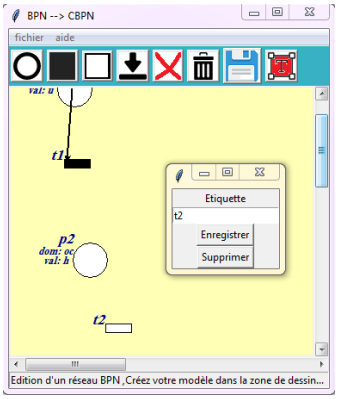


Figure 4.9: Création d'une transition Or.

La *Figure 4.10* montre le modèle BPN qui sera enregistré en tant que fichier XML *Figure 4.11*, et l'objectif de son enregistrement est d'actualiser (ré-ouvrir) le même modèle sans répéter la tâche d'édition.

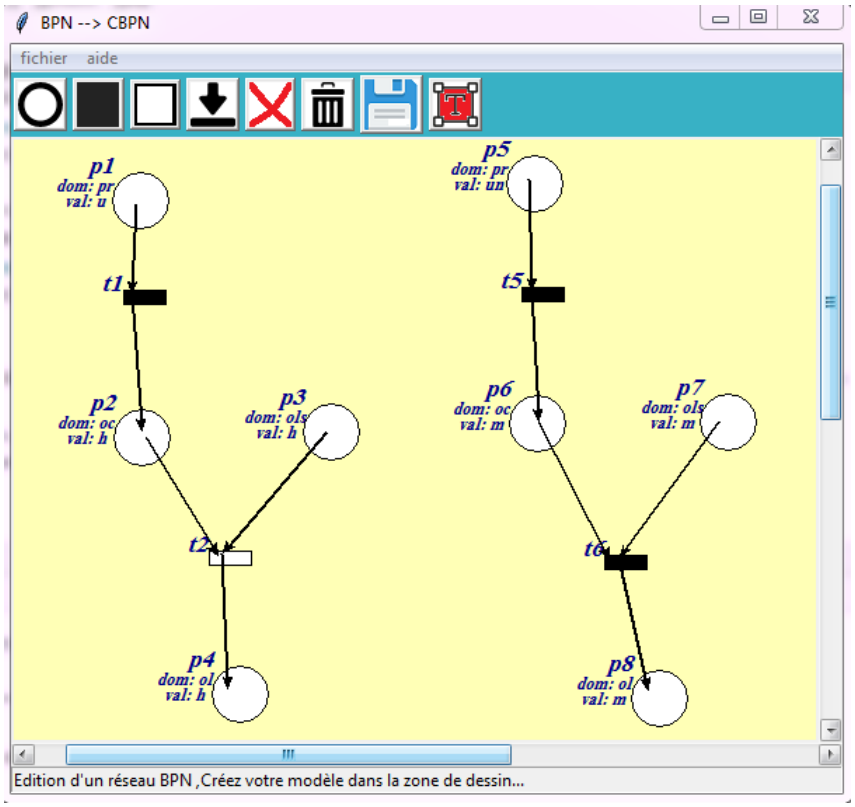
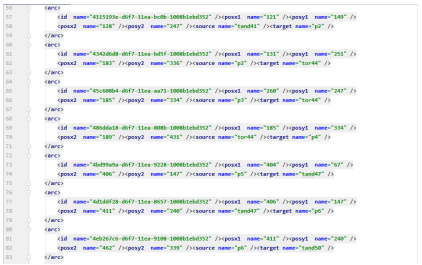


Figure 4.10: Modèle BPN.

(a) partie 1

(b) partie 2



(c) partie 3

(d) partie 4

Figure 4.11: Fichier XML pour BPN dans figure 4.10

Conversion d'un BPN à CBPN

Une fois le modèle BPN dessiné et enregistré correctement, nous pouvons le convertir en modèle (CBPN) en utilisant le bouton 8 (*Figure 4.6*) qui fournit une liste de modèles BPN. L'utilisateur peut en choisir un pour le convertir .

Édition d'un CBPN

Après la conversion du modèle BPN, nous obtenons la structure de données du modèle CBPN que nous essayons d'afficher sous forme de modèle graphique et le sauvegarder sous forme d'un fichier XML.
La figure 4.12 montre le nouveau modèle (CBPN) que nous avons obtenu par conversion du modèle BPN 4.10

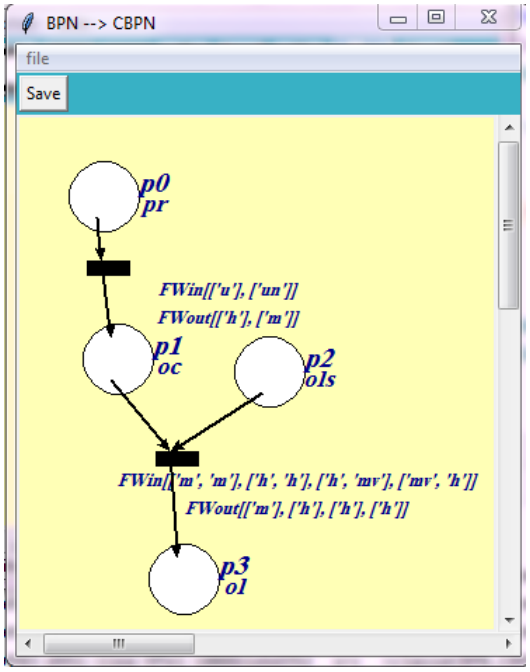


Figure 4.12: Modèle CBPN.

Il est clair que le système peut être décrit comme un modèle plus petit que le BPN qui sera équivalent à un CBPN. L'importance du modèle CBPN est montrée en comparant la taille du modèle BPN à celle du CBPN, utilisant les graphes ou les fichiers XML.
Le modèle CBPN est enregistré dans un fichier XML (*Figure 4.13*) .

```
<CBPN>
    <place><id name="p0" /><x name="37" /><y name="33" /><domaine name="pr" /><valeursAdmissible name="['u', 'un']" />
        <arcout name="df4ac2ec-d950-11ea-b7ba-1008b1ebd352" /></place>
    <place><id name="p1" /><x name="47" /><y name="149" /><domaine name="oc" /><valeursAdmissible name="['h', 'm']" />
        <arcin name="df4ac2ed-d950-11ea-ab90-1008b1ebd352" /><arcout name="df4ac2ee-d950-11ea-b023-1008b1ebd352" />
    </place>
    <place><id name="p2" /><x name="155" /><y name="158" /><domaine name="ols" /><valeursAdmissible name="['h', 'm']" /
        <arcout name="df4ae9fe-d950-11ea-a153-1008b1ebd352" /></place>
    <place><id name="p3" /><x name="94" /><y name="306" /><domaine name="ol" /><valeursAdmissible name="['h', 'm']" />
        <arcin name="df4ae9ff-d950-11ea-a437-1008b1ebd352" /></place>
    <transition>
        <id name="t 0" /><x name="50" /><y name="104" /><domainein name="['pr', 'pr']" />
        <domaineout name="['oc', 'oc']" /><FWin name="[['u'], ['un']]" /><FWout name="[['h'], ['m']]" />
        <arcin name="df4ac2ec-d950-11ea-b7ba-1008b1ebd352" /><arcout name="df4ac2ed-d950-11ea-ab90-1008b1ebd352" />
    </transition>
    <transition><id name="t 2" /><x name="99" /><y name="240" /><domainein name="['oc', 'ols', 'oc', 'ols']" />
        <domaineout name="['ol', 'ol']" />
        <FWin name="[['m', 'm'], ['h', 'h'], ['h', 'mv'], ['mv', 'h']]" /><FWout name="[['m'], ['h'], ['h'], ['h']]" />
        <arcin name="df4ac2ee-d950-11ea-b023-1008b1ebd352" />
        <arcin name="df4ae9fe-d950-11ea-a153-1008b1ebd352" /><arcout name="df4ae9ff-d950-11ea-a437-1008b1ebd352" />
    </transition>
    <arc><id  name="df4ac2ec-d950-11ea-b7ba-1008b1ebd352" /><source name="p0" /><target name="t 0" /></arc>
    <arc><id  name="df4ac2ed-d950-11ea-ab90-1008b1ebd352" /><source name="t 0" /><target name="p1" /></arc>
    <arc><id  name="df4ac2ee-d950-11ea-b023-1008b1ebd352" /><source name="p1" /><target name="t 2" /></arc>
    <arc><id  name="df4ae9fe-d950-11ea-a153-1008b1ebd352" /><source name="p2" /><target name="t 2" /></arc>
    <arc><id  name="df4ae9ff-d950-11ea-a437-1008b1ebd352" /><source name="t 2" /><target name="p3" /></arc>
</CBPN>
```

Figure 4.13: Fichier XML pour CBPN 4.12

Exemples de transformation

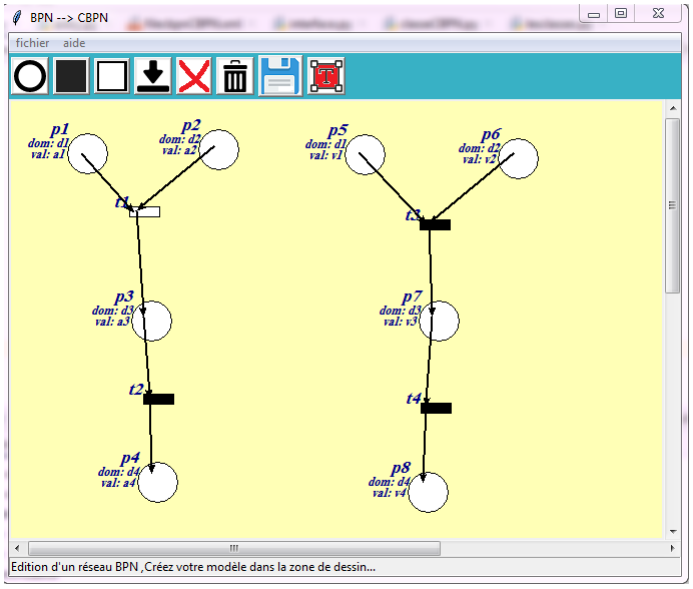


Figure 4.14: BPN 1

$P' = \{p1, ..., p8\}$,$T'_N = \{t2, t3, t4\}$,$T_{OR} = \{t1\}$

Place	Domaine	Valeur modélisé
p1	d1	a1
p2	d2	a2
p3	d3	a3
p4	d4	a4
p5	d1	v1
p6	d2	v2
p7	d3	v3
p8	d4	v4

On a domaine(p1)= domaine(p5)=d1. Donc, on va remplacer p1 et p2 par une seule place CBPN p0 avec des $valeursadmissibles\{a1, v1\}$.$P \leftarrow p0$.

$domaine(^\bullet t1) = domaine(^\bullet t3) = \{d1, d2\}$ et $domaine(t1^\bullet) = domaine(t3^\bullet) = \{d3\}$.

Donc,on va remplacer t1 et t3 par une seule transition CBPN t1 $T \leftarrow t1$. La matrice FW de t1 sera créer comme suivant:

Dans le BPN on a t1 de type Or qui a deux places d'entrées avec $\{v1, v2\}$ et une place de sortie avec la valeur $\{v3\}$. Selon le OU logique, on doit penser à trois suppositions, ça conduit à créer trois lignes. La première ligne contient trois colonnes v1 ,v2,v3 , la deuxième ligne contient mv ,v2,v3 et la troisième ligne contient v1,mv,v3.(dans l'application on a utilisé "mv" au lieu de ε)

Dans le BPN on a t3 de type AND qui a deux places d'entrées avec les valeurs $\{a1, a2\}$ et une place de sortie avec la valeur $\{a3\}$.Donc, on ajoute une ligne à la matrice avec trois colonnes contenant a1 ,a2,a3 .

$v1$	$v2$	$v3$
$a1$	$a2$	$a3$
mv	$a2$	$a3$
$a1$	mv	$a3$

Figure 4.15: Matrice FW de t0

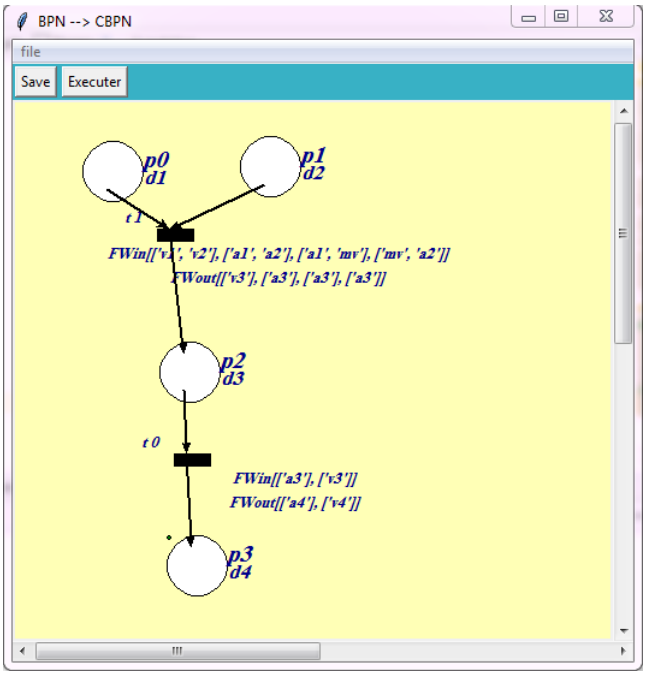


Figure 4.16: CBPN à partir de BPN 1 (Figure 4.14)

Place	Domaine	Valeur admissible
p0	d1	a1 v1
p1	d2	a2 v2
p2	d3	a3 v3
p3	d4	a4 v4

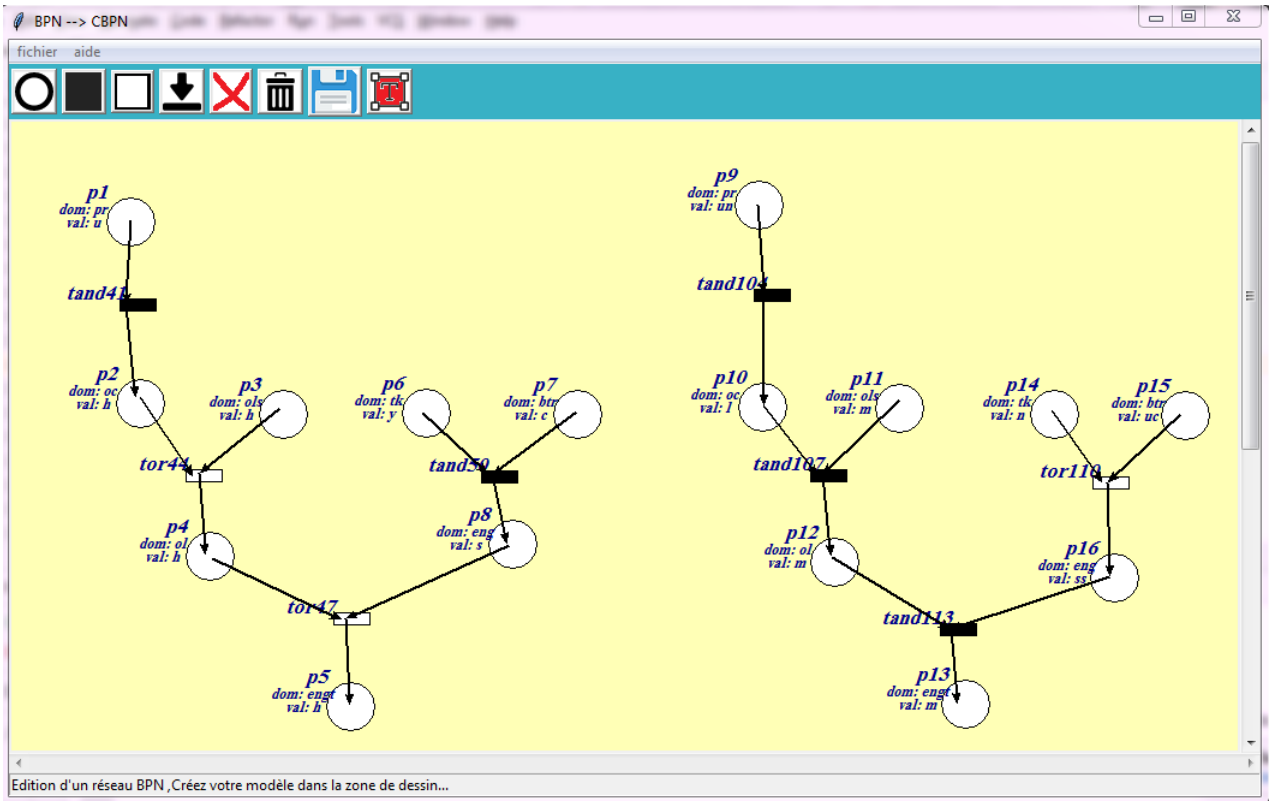


Figure 4.17: BPN 2

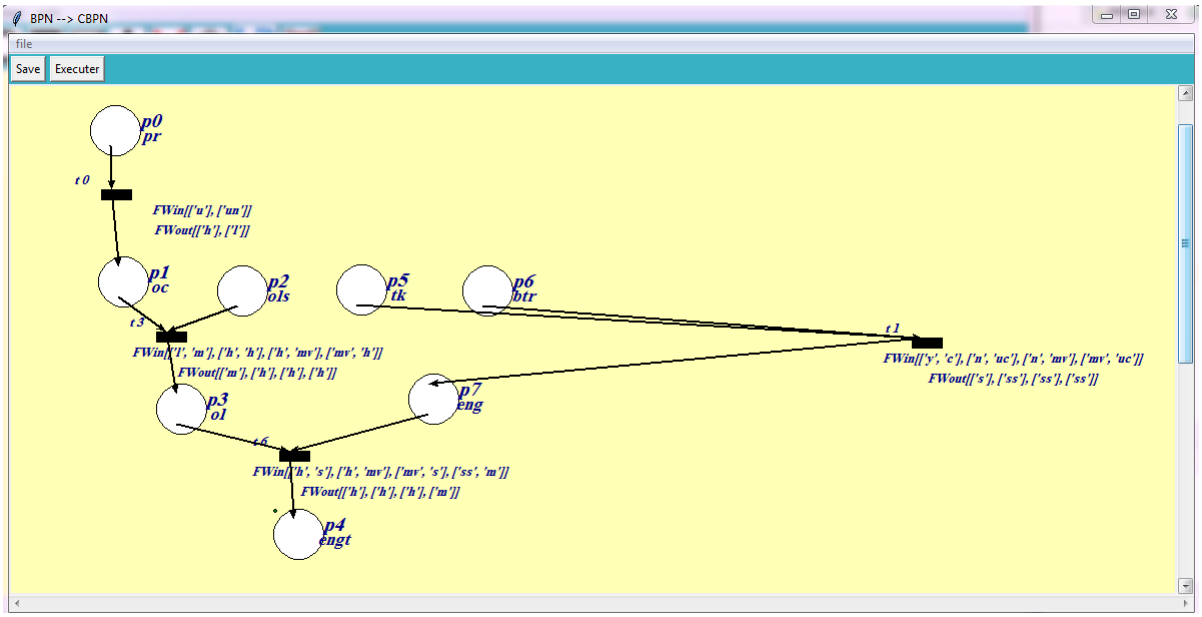


Figure 4.18: CBPN à partir de BPN 2(Figure 4.17)

Conclusion

Dans ce chapitre, nous avons présenté les outils que nous avons utilisés pour implémenter notre application, en commençant par un éditeur graphique qui nous permet d'éditer le modèle d'entrée BPN et puis le convertir simplement en un modèle qui est équivalent CBPN (moins compliqué en terme de taille), et à la fin nous avons affiché ce modèle dans une interface graphique.

Conclusion générale

Conclusion générale

Depuis presque trois décennies, les modèles causaux de comportement ont été montrés très utiles dans le raisonnement à base de modèle. En termes de notation, plusieurs outils ont été utilisés pour représenter de tels modèles causaux.

Dans ce mémoire, nous avons présenté les réseaux de Petri comportementaux colorés (CBPN) comme un formalisme mathématique pour représenter les modèles causaux. Les CBPNs sont caractérisés par des matrices associées aux transitions, qui décrivent les manières de franchissement des transitions. Ces matrices nous conduisent loin du problème d'inverser les expressions d'arcs qui face les RDPCs par la décomposition de chaque matrice en deux sous-matrices, une sous-matrice pour les places d'entrées et une autre pour les places de sorties. Lorsque on inverse les directions d'arcs, il suffit d'inverser les matrices d'entrées et sorties (la matrice d'entrée devient une matrice de sortie et vice versa).

Dans notre projet, nous avons essayé d'implémenter un algorithme convertissant un modèle d'entrée BPN en modèle CBPN. Le cycle de développement que nous avons suivis est montré au niveau de la deuxième partie de ce rapport.

Bibliography

[1] An Introduction to Tkinter. `http://effbot.org/tkinterbook/`, 2005. accessed on May 18, 2017.

[2] L'éditeur LaTeX universel. `http://www.xm1math.net/texmaker/index_fr.html`, 2017. accessed on May 18, 2017.

[3] Cosimo Anglano and Luigi Portinale. Bw analysis: a backward reachability analysis for diagnostic problem solving suitable to parallel implementation. In *International Conference on Application and Theory of Petri Nets*, pages 39–58. Springer, 1994.

[4] Hammadi Bennoui. *Distributed Causal Model-based Diagnosis: An Approach by Interacting Petri Nets*. PhD thesis, UNIVERSITE DE MOHAMED KHIDER BISKRA, 2012.

[5] Luca Console and Pietro Torasso. A spectrum of logical definitions of model-based diagnosis 1. *Computational intelligence*, 7(3):133–141, 1991.

[6] Soumia Mancer and Hammadi Bennoui. Coloured petri nets based diagnosis on causal models. In *PNSE@ Petri Nets*, pages 123–136, 2017.

[7] Soumia Mancer and Hammadi Bennoui. Distributed diagnostic problem solving with colored behavioral petri nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.

[8] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[9] Luigi Portinale. Behavioral petri nets: a model for diagnostic knowledge representation and reasoning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 27(2):184–195, 1997.