République Algérienne Démocratique et Populaie
Ministère de l'enseignement supérieur et de la recherche scientifique
Université Mohamed Khider - BISKRA
Faculté des Sciences Exactes et des Sciences de la Nature et de la vie
Départment d'informatique

GLSD/M2/2020

Mémoire fin d'etude

Présenté pour obtenir le diplôme de Master académique en

Option : **Génie Logiciel et Système Distribué**

Implémentation parallèle d'une technique d'analyse en arrière de marquages accessibles dans les RDPs

---

Par :
SAHRAOUI Rihab

Encadré par :
BENNOUI Hammadi

Année universitaire : 2019/2020

Remerciments

En premier lieu, je dois remercier Dieu le tout puissant qui m'a donné le courage et la force pour réaliser ce travail

Je remercie fortement mon encadreur : **M.Bennoui Hammadi** de m'avoir orienté par ses conseils dans le but de mener à bien ce travail.

Je tiens également à exprimer ma gratitude aux membres du jury pour la lecture et l'évaluation de mon travail.

Je n'oublie jamais de remercier tous mes professeurs au département d'informatique qui m'ont enseigné les principes de base de l'informatique.

Mon derniers remerciments, mais non des moindres, je voudrais remercier ma famille et mes amis pour tout leur soutien et leur compréhension durant cette année de mes études.

Et en fin, je remercie tous ceux qui ont contribué de loin ou de prés à réalisation de ce travail.

Résumé

L'analyse B-W est une technique particulière de résolution de problèmes de diagnostic. Elle est basée sur les BPNs, qui sont destinés à représenter le comportement causal du système à diagnostiquer. Le principe de cette technique utilise deux types de jetons ; jeton normal noir représente la satisfaction de la condition associée à une place marquée et jeton inhibé représente la non-satisfaction de la condition. Une parallélisation de l'analyse B-W à été est proposée. Une telle parallélisation est basée sur un ensemble de processus dérivés de la structure du modèle BPN.

Mots clés : *Analyse B-W, BPN, Processus, Parallèlisme.*

Résumé

B-W analysis is a diagnostic problem solving technique, defined for BPN, intended to represent the causal behavior of the system to be diagnosed. The principle of this technique uses two types of tokens ; black normal token represents satisfaction of the condition associated with a marked place and inhibitor token represents non-satisfaction of the condition. A distributed approach for parallel B-W analysis has been proposed. This approach is based on a set of processes derived from the structure of the BPN model.

Key words : *B-W analysis, BPN, process, Parallel*

Table des matières

Liste des tableaux

Table des figures

List of Algorithms

Introduction générale

Introduction générale

Le problème de diagnostic de fautes (ou des pannes) dans les systèmes technologiques a reçu une intention particulière depuis plus de trois décennies à raison de l'importance de sécurité et d'efficacité de fonctionnement.

Une variété d'approches ont été proposées dans la littérature d'I.A, chacune est basée sur un niveau de détail choisi pour le modèle du système et les genres des fautes qui nécessitent d'être diagnostiquer. Pour utiliser ces approches, nous devons représenter la structures et/ou le comportement du système à diagnostiquer. Lorsqu'un comportement anormal du système est observé, la tâche de diagnostic consiste à localiser les sources des pannes responsables du mal fonctionnement observé.

En fait, un diagnostic est défini somme un ensemble d'hypothèses (suppositions) sur la présence de certaines pannes qui peuvent expliquer le dysfonctionnement observé. Généralement, il y a deux notions logiques d'explication proposées dans la littérature :

- Approche basée consistance : ce type d'approche est utilisée dans le cas où le modèle représente le comportement normal du système à diagnostiquer. Elle désigne l'ensemble des suppositions qui gardent la consistance avec les observations.
- Approche basée abduction : cette approche est utilisée lorsque on a un modèle de comportement anormal du système à diagnostiquer. La conjonction des hypothèses avec le modèle doivent prédire l'observation.

Dans notre étude, nous nous focalisons aux approches utilisant un modèle décrivant le système dans le cas de mal fonctionnement. Avec ce genre d'approches, les différentes dépendances entres les pannes et les manifestations (ou observations) sont décrites dans le modèle comportemental. Ces dépendances décrivent les relations de causalités entre les différentes entités du modèle. De nombreux formalismes ont été proposés pour représenter les modèles causaux, tels que les formalismes logiques, les réseaux de Petri (RDP).

Dans notre travail, on s'intéresse à une classe particulière de RDP appelée BPN (Behavioral Petri net) qu'à été proposé dans [Por93] pour représenter le modèle. Le processus de résolution exploite une analyse en arrière de marquages accessibles (BW-Analysis : BackWard Analysis) du modèle BPN en question. Le principe de cette technique est d'utiliser deux types de jetons, jeton normal (noir, noté b) et un jeton inhibé (blanc, noté w) visant à modéliser la vérité ou la fausseté de la condition associée à une place.

L'une des particularités de l'analyse BW est son adéquation à une implémentation parallèle. Ce travail présente une approche parallèle pour l'analyse B-W dans laquelle un ensemble de processus est dérivé à partir de la structure du modèle BPN.

Ce mémoire est structuré en trois chapitres. Les deux premiers constituent la partie état de l'art et le dernier concentre sur l'implémentation parallèle de l'analyse B-W.

Première partie

ETAT DE L'ART

Chapitre 1

Diagnostic basé modèle

Chapitre 1

Diagnostic basé modèle

Introduction

La surveillance d'un système a but de traiter tous les comportements qui s'écartent du comportement attendu. Les principaux éléments d'un système de surveillance sont la détection, la localisation, le diagnostic et le traitement d'erreurs [VK94].

Le diagnostic est l'élément le plus important dans le système de surveillance, il consiste à localiser la cause de mal fonctionnement du système.

La littérature d'intelligence artificielle (I.A) fait état de deux familles d'approches permettant de résoudre un problème de diagnostic :

- Approches basées sur un modèle : ces approches sont utilisées dans le cas où il est possible de fournir un modèle qui décrit le fonctionnement normal et/ou anormal du système à diagnostiquer.[DMR92]
- Approches basées sur les heuristiques : ces approches sont basées sur les expériences et les intuitions des opérateurs considérés comme experts du domaine, a cause de l'impossibilité de traduire toutes les informations de fonctionnement du système sous forme d'un modèle.[DMR92]

Dans notre travail, nous nous focalisions sur les approches basées modèles, au début nous présentions des notions préliminaires concernant un problème de diagnostic basé-modèle. Dans la section trois nous donnerons une définition du modèle comportemental, et pour résoudre un problème de diagnostic, trois approches importantes seront montrés dans la section quatre. Dans les sections 5 et 6 nous présentons la définition du problème de diagnostic et sa résolution, finalement la section 7 conclut le chapitre.

1.1 Notions préliminaires

1.1.1 Description de système

La description d'un système est conçue pour formaliser d'une manière abstraite le concept d'un composant et le concept d'une collection de composants en interaction. Pour la représentation des connaissances, on utilise la logique du premier ordre. La définition suivante introduire par Reiter en [Rei87] présente cette notion.

Definition [Bla+06] : La description d'un système S est un pair : $<$ ***BM,COMPS*** $>$ où :

- ***BM*** est le modèle comportemental, il est formé de formules de premier ordre représentant les connaissances autour de S.
- ***COMPS*** un ensemble de constantes représentant les composants du système.[SLA04]

Le modèle comportemental ***BM*** utilise un prédicat ***AB(.)*** qui est interprété comme "anormal". L'argument d'un tel prédicat appartient nécessairement au ***COMPS***.

1.1.2 Observation

Le processus de diagnostic est basé sur l'utilisation d'un ensemble d'observations afin de détecter les pannes du système. L'absence de ces observations provoque la non possibilité de détecter aucune panne.

Definition [CT90] : Une observation ***OBS*** d'un système ***S*** est un ensemble fini de formules de premier ordre qui définit les entrées ***I*** (input) et les sorties ***O*** (output) de ***S*** [Rei87].

1.1.3 Diagnostic

Supposons qu' un système ***S*** $=<$ ***BM,***$\{ c_1, ..., c_n \} >$ est dans un état de mal fonctionnement, et soit ***OBS*** une observation sur ***S*** qui est en conflit avec le comportement normal de ***S***. La préparation de S passe nécessairement par une étape de diagnostic afin d'expliquer le dysfonctionnement observé.

Le rôle principale d'un processus de diagnostic est de spécifier l'ensemble de composants qui lorsqu'ils sont supposés en panne vont expliquer l'observation. La figure suivante montre que le diagnostic basé modèle est fondé sur l'interaction entre les observations si les prédictions obtenues à partir de la description du système [**16** ; Rei87].

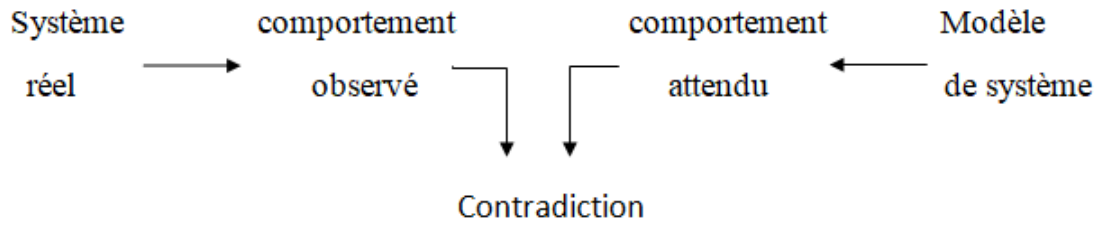

FIGURE 1.1 – Interaction entre l'observation et le comportement attendu.

1.2 Le modèle comportemental

L'approche de diagnostic basé modèle a besoin d'un modèle descriptif du système à diagnostiquer pour représenter le comportement normal ou/et anormal et des hypothèses permettant d'expliquer les observations. L'ensemble des comportements normaux et anormaux du système doit être représenté d'une manier uniforme[CT91].
Dans cette approche, chaque composant de ***COMPS***$=\{ c_1, ..., c_n \}$ est caractérisé par un ensemble de modes comportementaux $\{correct, fault_{i1}, \ldots, fault_{im}\}$ où

correct correspond au comportement correct du composant, et $fault_{ij}$ correspond aux divers comportements anomaux de tel composant.
Parmi les différents modèles permettant la modélisation de comportement du système, il y a les modèles causaux qui sont utilisés pour décrire les évolutions du système depuis un état vers un autre [CT90]. Les modèles causaux utilisent un langage logique basé sur les clauses de **HORN**.
Chaque clause a la forme suivante : $A_1 \wedge ..... \wedge A_n \rightarrow B$ où :

- $\wedge$ est le symbole de conjonction,
- $\rightarrow$ est le symbole d'implication,
- $A_1 \wedge ..... \wedge A_n$ est le corps de la clause (peut être vide),
- B est la tête de la clause.

L'ensemble de ces clauses est appelé un programme logique. Chaque élément de la clause est représenté sous forme d'un prédicat, et l'ensemble de prédicats d'un modèle comportemental représenté par ces clauses est reparti en deux sous-ensembles [CT91] :

- *Symboles abducibles* Correspondent aux modes comportementaux des composants de S et ne peuvent jamais apparaitre dans la tête d'une clause dans **BM**.
- *Symboles non abducibles* Correspondent aux conditions contextuelles qui apparaissent seulement dans le corps d'une clause.

Un exemple simple considère le problème de modélisation du comportement d'un circuit digital contenant les portes logique **AND** [CT91 ; Ham91]. On peut distinguer trois modes comportementaux d'une porte **AND** {correct, stuck-at-0, stuck-at-1}.
La description du modèle comportemental de ce circuit est [Ham91] :

- $And - gate(X) \wedge correct(X) \wedge inp1(X, X1) \wedge inp2(X, X2) \rightarrow out(X, fand(X1, X2))$.
- $And - gate(X) \wedge stuckat0(X) \rightarrow out(X, 0)$
- $And - gate(X) \wedge stuckat1(X) \rightarrow out(X, 1)$

Où :
fand (X1, X2) est le *AND* logique de X1 et X2. On note que $\{correct, stuck-at-0, stuck-at-1\}$ est l'ensemble de symboles abducibles dans ce modèle ; les conditions qui apparaissent seulement dans le corps d'une clause dans **BM** (par exemple : *inp1, inp2*) sont des symboles non abducibles.

1.3 Approches basées-modèle

Le diagnostic est basé sur un ensemble d'hypothèses sur la présence de certaines pannes qui peuvent expliquer l'observation en utilisant un modèle.
En générale, il y a différentes formalisations de cette notion de raisonnement qu'ont été proposées dans la littérature [CT91 ; Ham12] :

— *Approche basée consistance* : désigne l'ensemble des supposisons qui gardent la consistance avec les observations (c.à.d ; si une valeur d'un paramètre est observée, alors le diagnostic ne doit pas prédire une valeur différente de la valeur observée sur le même parametère). Autrement dit, cette approche est utilisée si le modèle représente le comportement normal du système [SLA04].
— *Approche basée abduction* : Les hypothéses en conjonction avec le modèle doivent prédire l'observation. Autrement dit, elle est utilisée si le modèle représente le comportement anormal du système [SLA04].

1.3.1 Approche basée consistance

Elle est utilisée lorsque le modèle est une représentation du comportement correct du système à diagnostiquer. Dans cette approche, le modèle comportemental est construit selon la méthodologie suivante :

1. Pour chaque composant **c** de **COMPS**, qui à la possibilité d'être en panne, on a l'hypothèse ¬**AB(c)** .
2. La description du fonctionnement normal du système sous forme des implications en se basant sur l'hypothèse que tous les composants fonctionnent correctement.

On suppose que **S= (BM, COMPS)** est le système à diagnostiquer et **O(I)** est la sortie attendu.
Ceci formalisé ainsi par :

$$BM \cup I \cup \{\neg AB(c) \mid c \in COMPS\} \vdash O(I).$$

Où :

— **BM** : le modèle compotemental,
— **I** : les entrées,
— **O** : les sorties,
— $\vdash$: signifie la derivation.

Supposons qu'il y a une contradiction entre **O** la sortie observée, et **O(I)**, on peut conclure que le système a mal fonctionné. Ceci donne l'implication suivante :

$$BM \cup I \cup \{\neg AB(c) \mid c \in COMPS\} \vdash (O(I) \Rightarrow \neg \mathbf{O}).$$

Alors, intuitivement, on a :

$$BM \cup I \cup \{\neg AB(c) \mid c \in COMPS\} \text{ est inconsistante.}$$

L'objectif du diagnostic basé consistance est de rendre la consistance à la formule précédente via la supposition que le comportement de certains composants est anormal.

- **Critères de préférence**
  Dans ce paragraphe, on présente deux critères utilisés dans l'approche basée consistance pour raffiner l'ensemble des diagnostics [Ham12].

  1. **Maximisation du nombre de composants décrits**
     Parmi les diagnostics possibles (noté $\triangle$), on prend ceux qui comportent un nombre maximal de composants. Dans la plupart des cas, l'incomplétude des connaissances provoque la non possibilité d'obtenir des diagnostics complets, donc il sera nécessaire d'ajouter des hypothèses additionnelles pour rendre l'ensemble de diagnostics complet.
  2. **Minimisation de l'ensemble de composants anormaux**
     Parmi les diagnostics complets possibles, on préfère ceux qui incluent un nombre minimal de composants anormaux.

1.3.2 Approche basée abduction

Le diagnostic basé abduction est proposé initialement pour un modèle qui représente le comportement anormal du système sous forme de relation Cause-Effet. La méthodologie suivie dans cette approche est [SLA04] :

— Les hypothèses possibles sont les pannes initiales
— La description du système décrit les relations de causalité entre les manifestations et les causes initiales sous forme d'implications.

Puisque le comportement correct du système à diagnostiquer n'est pas modélisé, les sorties attendues ne peuvent pas prédites. Donc, il est impossible de détecter une contradiction entre les sorties observées et celles attendues.
Contrairement au diagnostic basé consistance, il n'y a aucune inconsistance à expliquer :

$$\mathbf{BM} \cup \mathbf{OBS} \cup \{\neg AB(c)\} \textit{ est toujours consistante.}$$

Soit *CO* une combinaison des sorties à expliquer. Le diagnostic abductif pour CO est l'ensemble $\triangle$ tel que :

— BM $\cup$ I $\cup\triangle \vdash$CO et
— BM $\cup$ I $\cup\triangle$ est consistante tel que CO $\subseteq$ O.

– **Critères de préférence**

1. **Maximisation des sorties**
   Parmi l'ensemble de diagnostics, sélectionner celui qui explique un nombre maximal des points observables. La confidentialité de cette approche est augmentée avec le nombre des points observables expliqués.
2. **Minimisation des composants anormaux**
   Parmi l'ensemble de diagnostics, on préfère ceux qui incluent un nombre minimal des composants anormaux.

1.3.3 Approche d'intégration

Cette approche est une combinaison des deux type de raisonnement basé consistance et basé abductive, cette combinaison peut être vue comme [CT91] :

— Extension des approches basées abduction pour supporter les modèles de comportement correct.
— Extension des approches basées consistance pour supporter les modèles de comportement anormal.

1.4 Problème de diagnostic

Dans [CT91], les auteurs analysent en détail la notion d'un problème de diagnostic, il se caractérise par différents types de données : les données contextuelles et les observations. En particulier, une majeure distinction existe entre les données contextuelles et les observations. Les données contextuelles sont un ensemble de paramètres qui offrent des informations sur le cas sous examen.
Par exemple, l'âge et le sexe d'un patient (diagnostic médical), les entrées dans d'autres applications. Ces données sont très importantes puisque elles offrent au diagnosticien la possibilité de faire des prédictions sur le comportement du système à diagnostiquer (en effet, un patient est un homme, permet au physicien d'exclure certaines pathologies et de se concentrer sur d'autres pathologies [**17** ; Rei87])
Les données correspondant aux observations jouent un rôle totalement différent dans le

processus de diagnostic.
Un problème de diagnostic est définit ainsi :

Definition [SLA04] : Un problème de diagnostic **DP** d'un modèle comportemental est un triple : DP = < (*BM*, *COMPS*), *CXT*, *OBS* > où :
- (BM, COMPS) : est la description du système
- CXT : sont les données contextuelles
- OBS : est l'ensemble d'observation à expliquer (manifestations)

1.5 Résolution d'un problème de diagnostic

Comme nous avons vu dans la section précédente, un diagnostic peut être caractérisé comme un générateur d'explications pour un ensemble d'observations (symptômes). Dans la littérature, il existe deux notions logiques pour le terme explication [CT91] :
- Explication par consistance : dans un tel cas, un diagnostic explique une observation m si elle ne contredit pas m
- Explication par abduction : dans un tel cas, un diagnostic explique une observation m si elle supporte directement m

La définition suivante proposée dans [CT91] est basée sur la reformulation d'un problème de diagnostic comme un problème d'abduction avec des contraintes de cohérence.

Definition [Por93] : Etant donné un problème de diagnostic DP = < (*BM*, *COMPS*), *CXT*, *OBS* >
Le problème d'abduction correspondant à DP est définit comme :
AP = < (*BM*, *COMPS*), *CXT*, (ψ^+, ψ^-) > où :

- $\psi^+ \subseteq$OBS
- ψ^- = {($\neg f(x)$ | $f(y) \in$OBS , pour chaque valeur admissible x de f différente de y}

ψ^+ est l'ensemble d'observations qui doivent être prédites par une solution. ψ^- est l'ensemble d'observations négatives (l'ensemble des valeurs qui sont en conflit avec les observations), on note que ψ^- peut être infini.

Definition [Rei87] : Soit (BM, COMPS) une description d'un système. L'assignation W pour COMPS est l'ensemble des symboles abducibles tel que pour chaque composant c $\in$ COMPS , W contient exactement une seule instanciation du prédicat $\alpha(c)$ où α est un symbole abducible.

Definition [CT91] : Etant donné un problème abductif AP = < (*BM*, *COMPS*), *CXT*, (ψ^+, ψ^-) >.
L'assignation W pour COMPS est une explication pour AP si :
- W couvre ψ^+ ; c-à-d, $\forall$ m $\in \psi^+$ BM $\cup$ CXT $\cup$ W $\vdash$ m
- W est consistante avec ψ^- ; c-à-d, BM $\cup$ CXT $\cup$ W ψ^- est consistante et $\forall$ n $\in \psi^-$ BM $\cup$ CXT $\cup$ W $\nvdash$ n.

1.6 Conclusion

Dans ce chapitre, on a présenté les approches basées-modèle, notre travail est destiné à la résolution d'un problème de diagnostic des pannes d'un système en se basant sur un modèle causal. Comme nous avons indiquer au début, le formalisme des RdPs est utilisé comme moyen à la fois de représentation du modèle et de raisonnement. Donc, le chapitre

suivant sera consacré à présenter les aspects théoriques à implémenter. En particulier, les BPNs et l'analyse BW ainsi que comment implémenter une telle analyse parallèlement.

Chapitre 2

Une technique d'analyse basée sur les BPNs pour la résolution de problème de diagnostic

Introduction

Durant ces dernières décennies, de nombreux travaux ont été proposé pour la résolution des problèmes de diagnostic avec la proposition de différentes approches. Nous avons choisi l'approche qui utilise un model décrivant le comportement du système dans de cas de mal fonctionnement.

Dans ce chapitre, on essayera de donner en détail comment l'analyse d'accessibilité en arrière des réseuax de Petri (PN ou RDP) serve de manière efficace pour résoudre des problèmes de diagnostic à base de modèle PN et plus particulièrement des modèles BPNs. Une telle analyse peut être implémentée de manière parallèle afin d'augmenter en plus l'efficacité.

2.1 Réseaux de Petri comportementaux (BPNs)

Comme nous avons indiqué, la méthode de diagnostic utilisée est basée sur un modèle décrivant le système dans le cas de mal fonctionnement. Au niveau de ce modèle, toutes les relations de causalité (ou dépendance) qui peuvent exister entre les pannes, ainsi qu'entre les observations seront exploitées dans le processus de diagnostic. Pour la représentation du modèle, le formalisme de réseaux de Petri et en particulier les BPNs à été utilisé dans la littérature, car il offre plusieurs techniques d'analyse pour accomplir la tâche de diagnostic en plus d'une description directe de toutes les relations de dépendance. Dans ce qui suit, nous illustrons ce formalisme à travers un exemple, pour cela on a besoin premièrement d'expliquer le concept de modèles causaux utilisés comme un outil de description du comportement anormal d'un système, c'est l'objectif de la première section. La deuxième section donne un rappel sur les réseaux de Petri. Ensuite, la présentation sera orientée vers la classe de BPNs et comment leurs analyse permet d'accomplir la tâche en question de manière plus efficace.

2.1.1 Modèles causaux

Dans un modèle causal, le comportement anormal d'un système est décrit par un ensemble d'entités et de relations entre ces entités. Chaque composant du système est

modélisé par une entité et les relations décrivent les causalités existantes entre elles.
En fait,les différents entitée du modèle peuvent être classées en trois types :

- Etats initiaux : sont les états qui n'ont pas de cause dans le modèle, Dans le cas d'un modèle de fonctionnement fautif il représente les perturbations initiales menant le système à tomber en panne.
- Etats internes : les conséquences des états initiaux qui ne sont pas observable ou mesurables, ils ne sont pas susceptibles de faire partie de toute solution car ils peuvent être expliqués en termes d'états initiaux.
- Manifestations : les entitées observables et représentent les conséquences des états internes (tous les points observables).

2.1.2 Réseaux de Petri : notions de base

Tout système peut être vu comme un ensemble de composants qui interagissent entre aux. Ces composants peuvent fonctionner indépendants les uns des autres, on dit qu'il y a parallélisme, c'est pourquoi on a besoin d'otils permettant de modéliser la concurrence. Les réseaux de Petri (PN) sont l'un des modèles formels les plus populaires et les plus avancés pour la description du comportement d'un system parallèle.
Cette section présente brièvement quelques définitions de base[**18**; Bla+06] concernant les réseaux de Petri.

Definition [Bla+06] : un RDP est un triple N= (P, T, F) dont :

- $P \cap T = \emptyset$
- $P \cup T \neq \emptyset$
- $F \subseteq (P \times T) \cup (T \times P)$
- $Dom(F) \cup Cod(F) = P \cup T$

Où

- P : est l'ensemble des places
- T : est l'ensemble des transitions
- F : est un ensemble d'arcs orientés

On utilise les notations classiques suivantes, pour chaque $x \in P \cup T$:

- $^\bullet x = \{ y / y\ F\ x\}$
- $x^\bullet = \{ y / x\ F\ y\}$

Un marquage est une fonction $\mu : P \rightarrow \natural$, pour chaque place p on associe un nombre entier de jetons.

Definition [CT90] : Étant donné un RP N = (P, T, F) et un marquage μ , un RP marqué est définit par :
Σ= (P, T, F, μ) (ou bien Σ= (N, μ)).

Definition [SLA04] : soit un RDP marqué Σ= (P, T, F, μ); une transition $t \in T$ est franchissable dans μ ssi : $\forall\ p \in {}^\bullet t\ \mu(p) \geq 1$
Si $t \in T$ est franchissable dans μ alors :
t peut produire un nouveau marquage μ'(ou écrit $\mu[t > \mu']$) tel que :
$\forall\ p \in P$, $\mu'(p) = \mu(p)$-pré(p,t)+ post(p,t) où :

- pré (p, t) = 1 si $(p, t) \in F$; 0 sinon.
- post (p, t) = 1 si $(t, p) \in F$; 0 sinon.

L'ensemble des marquages qu'il est possible d'atteindre à partir de $\mu 0$, noté par R (N,$\mu 0$), est l'ensemble minimale tel que :

— $\mu 0 \in$ R ($\mu 0$)
— si $\mu 1 \in$ R (N,$\mu 0$) et $\mu 1$ [t > $\mu 2$ pour chaque t ∈ T, alors $\mu 2 \in$ R (N,$\mu 0$).

Definition [Por93] : Etant donné un RDP marqué Σ= (P, T, F, $\mu 0$) :

— Une place p ∈ P est k-bornée ssi : ∀μ ∈R(N,$\mu 0$) μ(p)≤k ;
— Σ est k-bornée, ssi : ∀ pi ∈ P (p_i est k_i-bornée et k= max (k_i)) ;
  — Si Σ est 1-bornée alors il est dit *safe*
  — ∀ p ∈ P(μ(p) ≤1) → le marquage est *safe*

Definition [Rei87] : un RDP marqué Σ = (N, $\mu 0$) est non déterministe ssi : ∀μ ∈ R(N,$\mu 0$), ∀ $t_1, t_2 \in T$ μ[t_1 > et μ[t_2 > Dans ce cas, on dit que t_1, t_2 sont concurrentes.

2.1.3 Réseaux de Petri comportementaux (BPNs)

Dans [Bla+06], un modelé RDP comportemental (BPN : Behavioral Petri Net) est utilisé pour adresser le problème de diagnostic basé-modèle. Ce formalisme à modéliser le comportement du système à diagnostiquer.
Les BPNs comprennent deux types de transition, des transitions régulières AND (les transitions de RDP classique), et des transitions OR qui différent des transitions ordinaires, Ce dernier type de transition simplifie la tâche de modélisation.

Definition [CT91] : BPN est 4-tuples N= (P,T_n, T_or,F) où :

1. $T_n \cap T_{or} = \emptyset$
2. (P,($T_n \cup T_{or}$),F) est un réseau
3. F^+ (fermeture transitive de F) est non réflexive
4. ∀ p ∈ P (|• p|≤ 1 ∧ (| p• |≤ 1)
5. ∀p_1, p_2 ∈P ((•p_1= •p_2) ∧ ($p_1^\bullet = p_2^\bullet$) → $p_1 = p_2$
6. ∀$t \in T_n$(|• t |= 1∧ | $t^\bullet$ |> 0) ∨ (|• t |> 0∧ |• t |= 1)
7. ∀$t \in T_{or}$(|• t |= 2∧ | $t^\bullet$ |= 1)

Les transitions T_n sont des transitions d'un RDP classique et peuvent être classés selon :

— Transition linéaire : $t \in T$:|• t |=| $t^\bullet$ |= 1
— Transition fork : $t \in T$:| $t^\bullet$ |> 1
— Transition join : $t \in T$:|• t |> 1

La figure suivante montre la sémantique de la transition OR

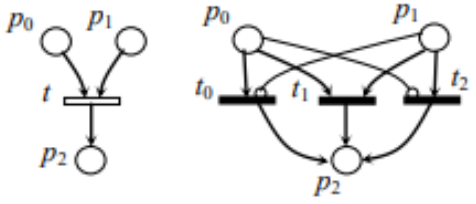

FIGURE 2.1 – la sémantique de la transition OR.

Les transitions T_{or} sont introduites pour représenter les connections logiques OR, Elles sont représentées graphiquement par des rectangles, le franchissement d'une transition de ce genre peut être effectué si et seulement si toutes ses places d'entrées sont marquées. Une conséquence évidente de non réflexive de F^+ (l' axiome 3 de la définition 6), est qu'elle définit un ordre partiel dénotée par « $\prec$ » sur les transitions d'un BPN
Soit t_1 et t_2 deux transitions : $t_1 \prec t_2 \Leftrightarrow t_1 F^+ t_2$.

Definition [Ham91] : Étant donné un BPN, un marquage μ , t est franchissable dans μ (on peut le exécutée) ssi : elle est sensibilisée dans μ et $\nexists t' \prec t$ tel que t' est sensibilisée dans μ.

Definition [Ham12] : un marquage initial μ_0 d'un BPN est un marquage safe si :
μ_0 (p)=1 alors p est une place source ($^\bullet p = \emptyset$)

Definition [VK94] : un BPN marqué est un paire (N, μ) où :
- N : est un BPN,
- μ Est un marquage initial ou est un marquage tel que $\exists \mu_0$ et $\mu \in$ R(N, μ_0)

On dit que μ est un marquage final si : $\nexists$ une transition franchissable dans μ .

Théorème [Bla+06] : Un BPN marqué est safe et déterministe.

Théorème [CT90] : Dans le marquage d'un BPN, il existe exactement un seul marquage final (la démonstration de ces théorèmes est donnée dans [Bla+06]).

2.1.3.1 Représentation d'un model causal par un BPN

Le formalisme le plus populaire pour la représenter le comportement causal du système à diagnostiquer est les RDP, L'objectif est traduire la description logique du problème de diagnostic en termes atteignabilité dans le modèle RDP, et d'appliquer ensemble d'algorithmes pour exploiter le parallélisme des évolutions concurrentes explicité par le modèle lui-même.
Chaque entité du modèle est représentée par une place dans le modèle BPN, et les relations de causalité sont représentées par des transitions.
Le modèle BPN obtenu est caractérisé par [Bla+06] :

- Une place source (p $\in$ P $|^\bullet p = \emptyset$) est une place représente un état initial du modèle causal.
- $\forall t \in T_n (|^\bullet t| > 1 \wedge | t^\bullet | = 1) \vee (|^\bullet t| = 1 \wedge (| t^\bullet | > 1)$, Dans le premier cas la panne correspondante à la place d'entrée peut conduire plusieurs éléments à tomber en panne dans un ordre indépendant. Dans le deuxième cas, la panne correspondante à la place de sortie est causée par la conjonction de toutes les pannes sont représentées par les places d'entrées de la transition.
- $\forall t \in T_{or} (|^\bullet t| = 2 \wedge | t^\bullet | = 1$, ce qui signifie que l'une des deux pannes est représentée par les deux places d'entrées provoque la panne correspondante à la place de sortie.
- Le modèle BPN doit être acyclique, c'est une hypothèse commune dans la modélisation du comportement causal d'un système.

2.1.3.2 Exemple d'illustration

L'exemple est un extrait depuis [Bla+06], il représente le comportement anormal d'un moteur de voitures. On peut distinguer les entités suivantes :

— Les états initiaux : ils modélisent les états suivants :{'*Pist* − *ring* − *state*'(*worn*),' *Pist* − *state*(*worn*),' *Road* − *cond*'(*poor*),' *Ground* − *clear*'(*low*)}.
— Les états internes : qui modélisent les panes des autres éléments.
— Les manifestations : Elles correspondent aux points d'observation {'*Ex*−*smoke*(*black*),' *Oil*− *ligh*'(*red*),' *Temps* − *ind*'(*red*)}.

Les transitions représentent les liaisons de dépendances entre ses entités.

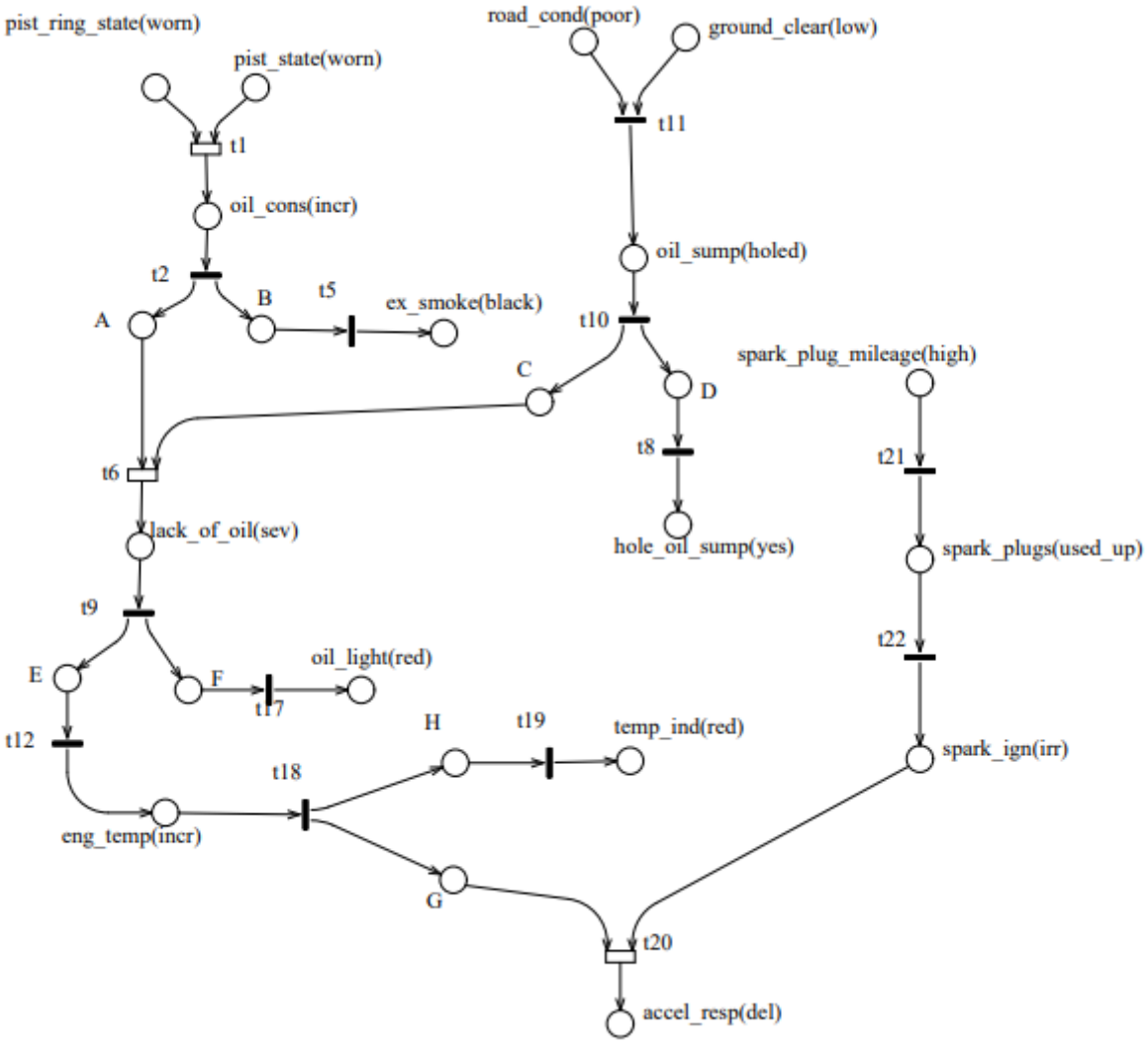

FIGURE 2.2 – Exemple d'un BPN représentant le comportement anormal d'un moteur d'une voiture

2.2 Analyse B-W

Dans notre travail, la technique de diagnostic utilisée est L'analyse B-W, elle est basée sur l'analyse de marquages atteignable d'un modèle RDP.

2.2.1 Discussion informelle

Étant donnée un $\Sigma = (N, \mu_0)$ on peut obtenir plusieurs marquages par le franchissement des transitions sensibilisées, on peut aussi atteindre d'autres marquage a partir de nouveau marquages.
La taille d'arbre de couverture s'augmente infinitivement si le RDP n'est pas borné. Pour

rendre limité, il faut introduire un symbole spécial ω.

Cette méthode d'analyse permet de détecter si le système modélisé possède de bonnes propriétés (l'absence d'interblocage. . .).
On peut utiliser le résultat d'algorithme arbre de couverture pour résoudre des problèmes de diagnostics, mais son implémentation est complexe. Pour cela la technique d'analyse d'atteignabilité en arrière ont été proposées.
Le principe d'analyse en arrière (B-W analysis) proposée dans [Bla+06] utilise deux types de jetons, jeton normal (noir, noté b) et un jeton inhibé (blanc, noté w).
Lorsque nous modélisons les conditions par des places, la présence de jeton noir signifie que la condition correspondante est satisfaite, A l'inverse, le marquage d'une place par un jeton blanc signifie que la condition n'est pas satisfaite, si la place n'est pas marqué aucune information n'est donnée concernant la condition qui lui est associée.
Logiquement donc, ces conditions correspondent à considérer trois valeurs logiques $\{vrai, faux, inconnu\}$

2.2.2 Principe et cadre d'utilisation

On a besoin de présenter certains concepts de base relatifs aux RDP [Por93 ; Bla+06] pour montrer comment l'analyse en arrière exploitée dans le diagnostic.

Definition [Poo89] : soit un BPN N= (P, T_n, T_{or}, F), un marquage b-w est une fonction $\mu : p \rightarrow \{b, w, 0\}$
- Si μ(p)=b alors la place p est marquée par un jeton normal (noir)
- Si μ(p)=w alors la place p est marquée par un jeton inhibé (blanc)
- Si μ(p)=0 alors la place p est vide.

La notation de règles de franchissement en arrière a été introduite dans [Bla+06 ; Por93] pour l'analyse b-w . Elle nécessite de considérer le concept d'ordre partial entre les transitions dans un modèle RDP.

$$\text{t1} \prec \text{t2} \Leftrightarrow \text{t2} \succ \text{t1}\ (\forall t1, t2 \in (T_n \cup T_{or})).$$

Definition [] : Etant donnée N= (P, T_n, T_{or}, F), μ un marquage b-w et t $\in (T_n \cup T_{or})$:
- t est b-franchissable dans μ si : $\forall p \in t^\bullet(\mu(p) = b \wedge p^\bullet = \emptyset)$;
- $\neg$ t est b-franchissable dans μ si : $\forall p \in t^\bullet(\mu(p) = w \wedge p^\bullet = \emptyset)$;
- t est b-franchissabledans μ si : $\forall p \in t^\bullet(\mu(p) = b \wedge p^\bullet \neq \emptyset)$ et $\nexists t' \succ t(t'$ou$\neg t'$ est franchissable dans $\mu))$;
- $\neg$ t est b-franchissabledans μ si : $\forall p \in t^\bullet(\mu(p) = w \wedge p^\bullet \neq \emptyset)$ et $\nexists t' \succ t(t' ou \neg t'$ est franchissable dans $\mu))$;

Definition [] : Étant donné un BPN N= (P, T_n, T_{o_r}, F), un marquage b-w μ est inconsistant ssi : $\exists$ t $\in T_n$ et $\exists p_i, p_j \in t^\bullet$ tel que : $\mu(p_i) = b$ et $\mu(p_j) = w$.
En autre termes, si on a une transition fork , il n'est pas possible d'avoir des jetons de type différents marquant ses places de sortie.
En l'analyse b-w, il est possible de forcer le franchissement en arrière de transition de type"fork" par la définition suivante.

Definition [] : soit un BPN N= (P, T_n, T_{o_r}, F), une transition$t \in T_n$, tel que $t^\bullet = \{p_1, \ldots, p_r\}$ est forcée dans le marquage b-w μ ssi :

1. t est non b-franchissable dans μ ;
2. μ est non inconsistant ;
3. $\exists\, p_i\ (1 \leq i \leq \mu(p_i) \neq 0$;
4. $\nexists\, t' \prec t (t' ou \neg t'$ est b-franchissable ou forcée dans μ).

L'étape d'abstraction peut être formalisée comme la suite :
si t est forcée alors :

$$\forall p_i \in t^\bullet \mu(p_i) = 0 \Rightarrow (\mu(p_i) = b \vee \mu(p_i) = w), \text{ tel que } \mu \text{ n'est pas inconsistant.}$$

Ceci implique qu'il faut marquer les places vides dans la sortie de t avec le même type de jetons contenus dans les places marquées.

Definition [Moz92] : Étant donnée un BPN, μ un marquage b-w , b-étap dans μ est l'ensemble maximal s $=\{t_{i_1} \ldots t_{i_r} \neg t_{j_1} \ldots \neg t_{j_s}\}$ qui a la propriété que chaque : $t_{i_k} (1 \leq k \leq r)$ et chaque $\neg t_{j_h} (1 \leq h \leq s)$ est b-franchissable dans μ
Chaque s' $\subseteq$ s est dit b-sous-étape.
Pour monter comment un nouveau marquage b-w est produit à partir d'un ancien marquage dans l'analyse b-w, les règles de franchissement en arrière suivantes ont été introduites par [Bla+06 ; Por93].

1. $^\bullet p = \{t\} \wedge t \in T_n (^\bullet t = \{p_1, \ldots, p_r\}, r \leq 1)$;
2. $^\bullet p = \{t\} \wedge t \in T_{or} (^\bullet t = \{p_1, p_2\}\ )$

Pour chaque cas on peut distingue deux alternative. Soit μ' le marquage obtenu a partir d'une b-étape s dans μ :
1.a μ(p)=b $\rightarrow$ $\{t\}$ est une dit b-sous-étape de s, $\mu'(p_1) = \ldots = \mu'(p_r)$=b et μ' (p)=0 ;
1.b) μ(p)=w $\rightarrow$ $\{\neg t\}$ est une b-sous-étape de s et on a des choix non déterministes tel que :$< \mu'(p_1), \ldots, \mu'(p_r) >$ est une combinaison de valeurs de {b, w}, différent du cas : $\forall p_i$ $(1 \geq$ i $\geq$ r) $\mu'(p_i)$=b ;
2.a) $\mu(p) = w \rightarrow \{\neg t\}$est une b-sous-étape de s, $\mu'(p_0) = \mu'(p_1) = w$ et $\mu'(p) = 0$;
2.b) $\mu(p) = b \Rightarrow$on a des choix non déterministes :

— $\{t\}$ est une b-sous-étape de s, $\mu'(p_0)$=b,$\mu'(p_1)$=w et$\mu'(p) = 0$;
— $\{t\}$ est une b-sous-étape de s, $\mu'(p_0)$=b,$\mu'(p_1)$=b et$\mu'(p) = 0$;
— $\{t\}$ est une b-sous-étape de s, $\mu'(p_0)$=w,$\mu'(p_1)$=b et$\mu'(p) = 0$;

La figure suivante va présenter graphiquement les règles de franchissement en arrière

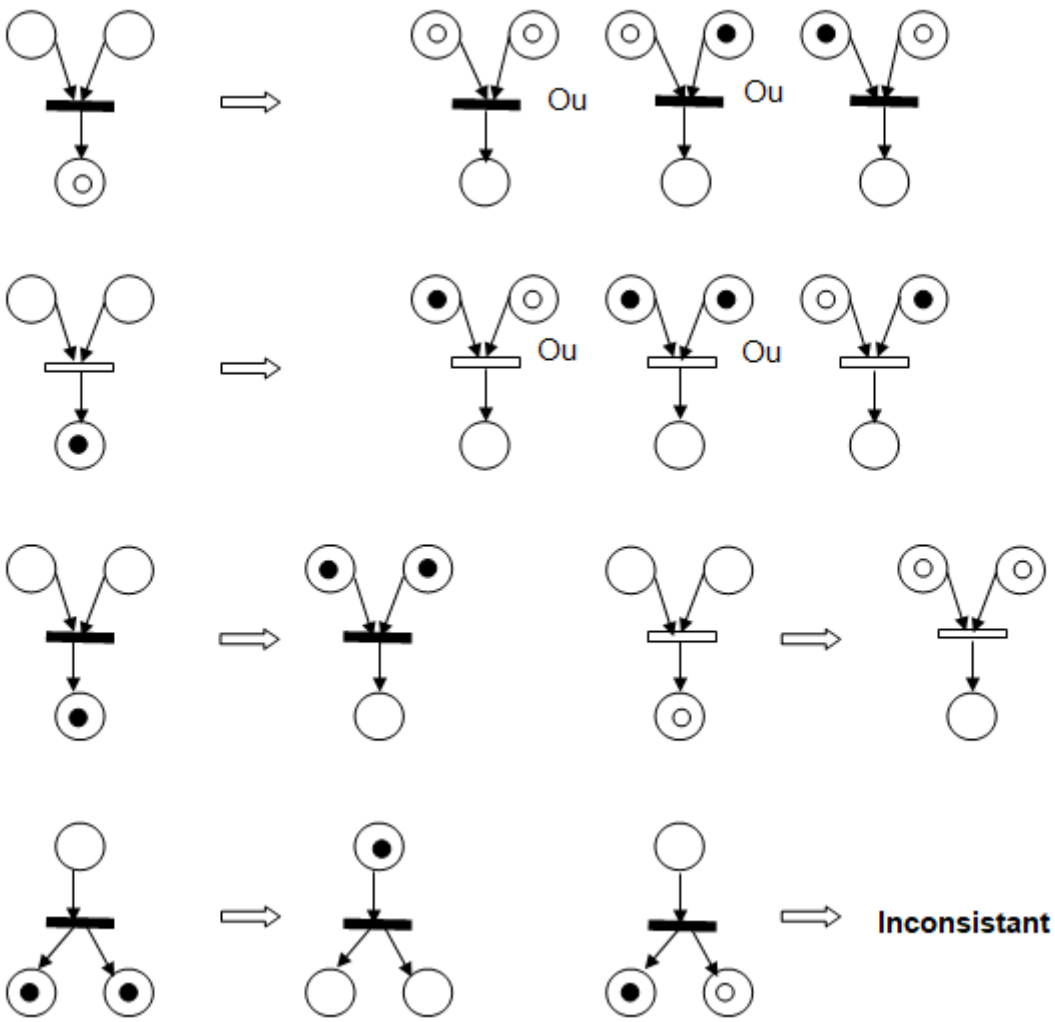

FIGURE 2.3 – les règles de franchissement en arrière d'un BPN

La condition d'arrêt du processus d'application de b-w partant d'un marquage b-w μ tel que : $\mu(p) \neq 0 \rightarrow p^{\bullet} = \emptyset$(p est une place puits) est :

1. Soit un marquage initial b-w.
2. Soit un marquage b-w inconsistant.

En le premier cas, les conditions initiales qu'elles n'ont pas de signification dans le cas sous examen sont représentées par des places sources vides.Par contre, les places sources qui sont marquées avec des jetons normales et inhibés représentent les conditions initiales qu'elles ont été démontrées vraie et fausse respectivement.
La figure 2.4 représente le résultat de l'analyse B-W de BPN figure 1, dont le marquage de démarrage est : $\{"Ex - smoke"(black)[w], "Oil - light(red)[b], "Temp - ind(red)[b]\}$
Dans le graphe de raisonnement en arrière pour l'analyse B-W, on utilise les notations :

- p [b] signifie que la place p est marqué avec un jeton noir ;
- p [w] que la place p est marqué avec un jeton blanc ;
- les arcs sont étiquetés avec b-étape ;
- les transitions soulignées représentent les transitions forcées

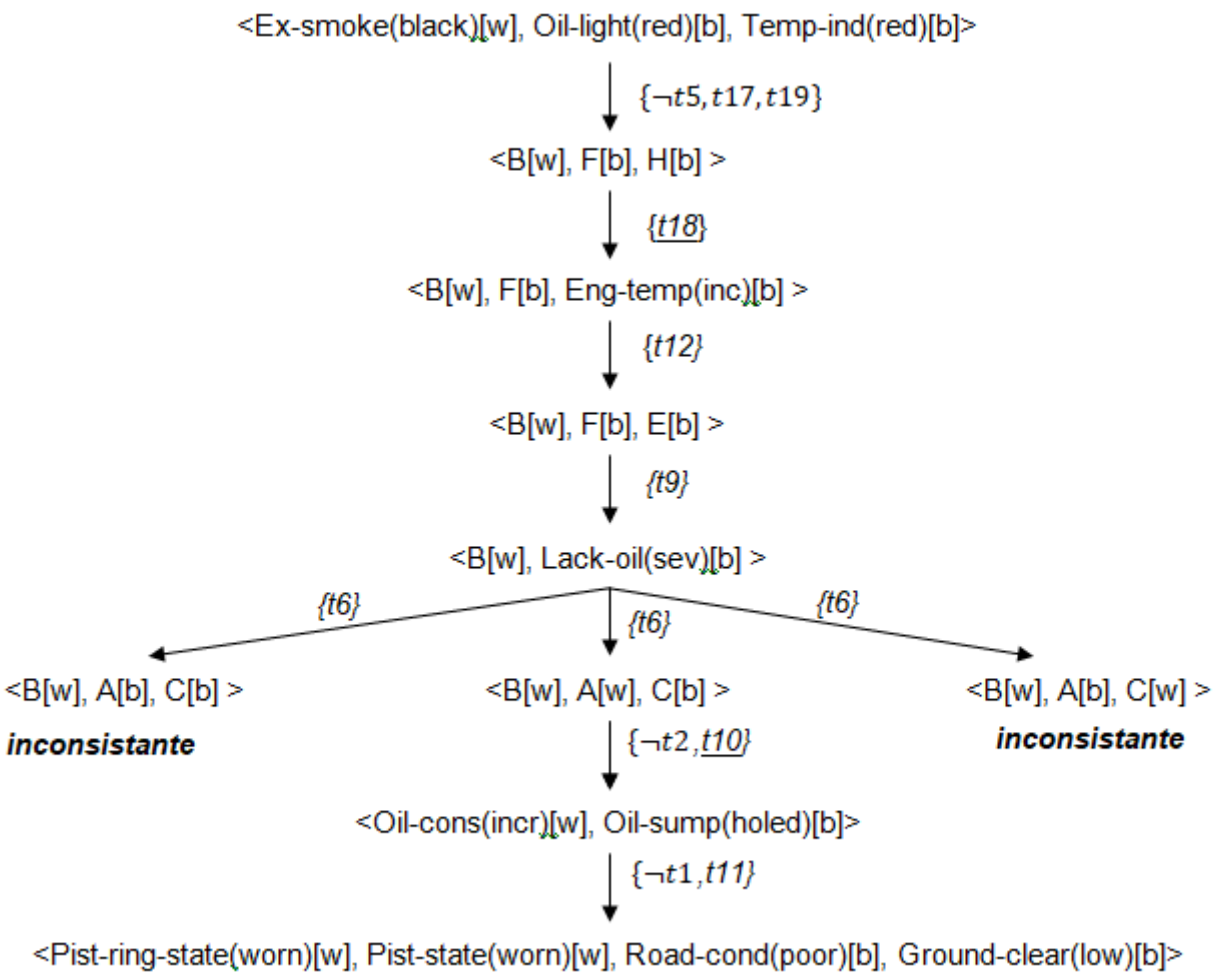

FIGURE 2.4 – graphe de raisonnement en arrière pour l'analyse B-W

2.3 Application de la technique B-W pour la résolution de problèmes de diagnostic

Comme nous avons vu dans le chapitre précédant, un problème de diagnostic est défini par DP= (<BM, COMP>, Exp, < ψ^+, ψ^- >) où :
BM : est le modèle comportemental, il est formé d'un ensemble de formules du premier ordre représentant les connaissances autour du système S.
COMP : est un ensemble de composants du système.
Exp : l'ensemble des causes initiaux menant le système a suivre un comportement anormale.
ψ^+ : est l'ensemble d'observations qui doivent être prédites par une solution.
ψ^- : est l'ensemble d'observations négatives (les valeurs qui sont en conflit avec les observations).

Definition [DMR92] : Étant donné un problème de diagnostic DP= (<BM, COMP>, Exp, < ψ^+, ψ^- >), E ⊆ Exp est une solution (diagnostic) pour DP ssi :

$$\forall m \in \psi^+ \; S \cup E \vdash m$$
$$\forall n \in \psi^- \; S \cup E \nvdash n.$$

Definition [16] : Un BPN de diagnostic est défini comme DP-BPN = $(N_s, P_E, < P^+, P^- >)$ où :

— $P_E = \{p \in P \mid \Phi(p) \in Exp\}$;
— $P^+ = \{p \in P \mid \Phi(p) \in \psi^+\}$;
— $P^- = \{p \in P \mid \Phi(p) \in \psi^-\}$.

Tel que :
N_s : est le BPN correspondant au système ;
P_E est l'ensemble des places représentant l'ensemble Exp ;
P^+ est l'ensemble des places représentant les manifestations observées, et qui doivent être couvertes par la solution ;
P^- : est l'ensemble des places représentant les manifestations absentes, et qui ne doivent pas être couvertes par la solution.
ϕ(p) est une fonction d'interprétation. Elle associe place du modèle une interprétation.

Definition [17] : un marquage μ de BPN covers un ensemble de places Q ssi :

$$\forall p \in Q \rightarrow \mu(p) = 1 .$$

Definition [18] : marquage μ de BPN zéro-covers un ensemble de places Q ssi :

$$\forall p \in Q \rightarrow \mu(p) = 0.$$

Théorème [SLA04] : soit DP-BPN = $(N_s P_E, < P^+, P^- >)$ un BPN de diagnostic, un marquage b-w initial μ_E est une solution pour DP-BPN ssi :
Le marquage b-w final μ de N_s) covers P^+ et zéro-covers P^-.

2.3.1 Exemple

Supposons dans notre exemple (la figure 2.4) l'observation est $\{''Ex{-}smoke''(black)[w], ''Oil{-}light''(red)[b], ''Temp - ind''(red)[b]\}$.
Les ensembles P^+ et P^- sont identifiées via une comparaison entre les valeurs des paramètres de l'observation OBS et celles attendues depuis le modèle.
$P^+ = \{''Oil - light(red)''[b], ''Temp - ind''(red)[b]\}$;
$P^- = \{''Ex - smoke''(black)[w]\}$
$P_E = \{''Pist{-}ring{-}state''(worn), ''Pist{-}state''(worn), Road{-}cond''(poor), Ground{-}clear''(low), ''Spark{-}plug - mileage''(high)\}$
La résolution de ce problème sera calculée par l'analyse b-w tous en considérant les différents cas du marquage b-w μ :

$$\mu(p) = \begin{cases} b & si\, p \in P^+ ; \\ w & si\, p \in P^- ; \\ 0 & sinon \end{cases}$$

Alors on peut obtenu toutes les solutions possibles par l'exploitation des différentes alternatives non déterministes. Dans le cas d'un marquage inconsistant, l'alternative sera ignorée et il faut examiner d'autres possibilités ; en prenant en considération le marquage

initial obtenu à la fin de l'analyse b-w.
La solution obtenue doit être constituée par des conditions correspondantes aux places marquées avec des jetons normales et par la négation des conditions correspondantes aux places marquées aves des jetons inhibés.
Le graphe de la figure 2.4 représente les calcules de la solution du problème DP-BPN par l'analyse b-w. Donc la seule solution possible est :
E={$''\neg Pist-ring-state''(worn), ''\neg Pist-state''(worn), Road-cond''(poor), Ground-clear''(low)$}

2.4 Exploitation de parallélisme dans l'analyse en arrière

Dans cette section, nous décrivons comment l'analyse b-w peut être effectuée à l'aide d'un ensemble de processus asynchrones, exécution concurrent et interaction par échange de messages.
Deux types d'approches différentes ont été proposés dans la littérature pour l'implémentation concurrent du jeu de jetons, généralement classé comme centralisé ou décentralisé. D'une part en l'approche centralisé, le jeu de jetons est effectué par un ensemble de processus dont l'exécution est contrôlée par un processus coordinateur, d'autre part les approches décentralisées reposent sur un ensemble de processus séquentiels fonctionnant de manière asynchrone et interagissant par un mécanisme de communication / synchronisation
L'approche que nous choisissons dans notre travail est **décentralisé**.
Notre approche fonctionne comme suit : la première étape est la dérivation d'une partition des transitions d'un modèle BPN donné puis un processus est associé à chaque sous-ensemble de transitions[Bla+06].

2.4.1 Partitionnement de réseau (BPN)

Pour partitionner l'ensemble des transitions, nous utilisons une relation binaire $\asymp$ définie sur $(T_n \cup T_o r) \times (T_n \cup T_{o_r})$.

Definition [19] : Soit t_i, t_j deux transition d'un BPN N, la relation $\asymp \subseteq$ T$\times$ T est défini par : $t_i \asymp t_j \Leftrightarrow |^\bullet t_i| = 1$ et $|t_j^\bullet| = 1 \wedge {}^\bullet t_i = t_j^\bullet$

Propriété [Bla+06] : Soit $t_i, t_j, t \in T$ être transition d'un BPN N, si $t_i \asymp t$ et $t_j \asymp t$ alors $t_i = t_j$.

Definition [20] : Étant donnée un BPN N, une transition t $\in$ T est dit transition de départ ssi :

$$|t^\bullet| > 1 \text{ ou } (|t^\bullet| = 1 \wedge t^\bullet \cap^\bullet t_j = \emptyset), \forall t_j \in T.$$

La définition ci-dessus correspond à dire qu'une transition de départ a soit plusieurs places de sortie, soit exactement une place de sortie.
Étant donné une transition de départ t, il est possible de définir un sous-ensemble de transition appelé sous-ensemble de partitionnement généré par t de la manière suivante :

Definition [21] : soit t $\in$T est une transition de départ d'un BPN N. sous-ensemble de transition $T_i \subseteq T$généré par la transition t est défini par :

1. $t \in T_i$
2. si $t_i \in T_i \wedge \exists t_j \in T : t_i \asymp t_j$ alors $t_j \in T_i$

Une transition de départ t, son sous-ensemble de partitionnement correspondant T_i est construit en calculant la fermeture transitive $\asymp^+$ de la relation binaire $\asymp$.
Le processus s'arrête lorsque la première transition t_j, telle que $\forall t_k \in T\ t_j \not\asymp t_k$ est ajouté à T_i une telle transition t_j serait la transition de fin de T_i. Par la définition de la relation binaire, une transition de fin t peut être caractérisé de la manière suivante : soit (a) $|^\bullet\ t\ | > 1$ ou (b) $^\bullet t = \{p\}$ et p est une place source ou (c) $^\bullet t = \{p\}$ et $^\bullet t \subseteq t_j^\bullet$ et t est une transition de départ.
Étant donné une partition de l'ensemble des transitions, il est possible de générer des sous-ensemble indépendants appelés **sous-ensemble générés par transition**.

Definition [22] : soit N= (P, T_n, T_{o_r}, F) être un BPN et soit $T = T_n \cup T_{or}$. Étant donné une partition $\{T_1, \ldots, T_n\}$ de T, on définit les sous-ensemble générés par transition (TGS) de N. la partition à être des réseaux $N_i = (P_i, T_i, F_i)(i = 1, \ldots, n)$où :

— $P_i = P \cap (^\bullet T_i \cup T_i^\bullet)$
— $F_i = F \cap ((P_i \times T_i) \cup (T_i \times P_i))$

Étant donné un TGS N_i, on dénote :
S (N_i) est une transition de départ
E (N_i) est une transition de fin
La figure montre les sous-ensembles de partitionnement T_i, obtenus à partir du réseau de la figure 2.2

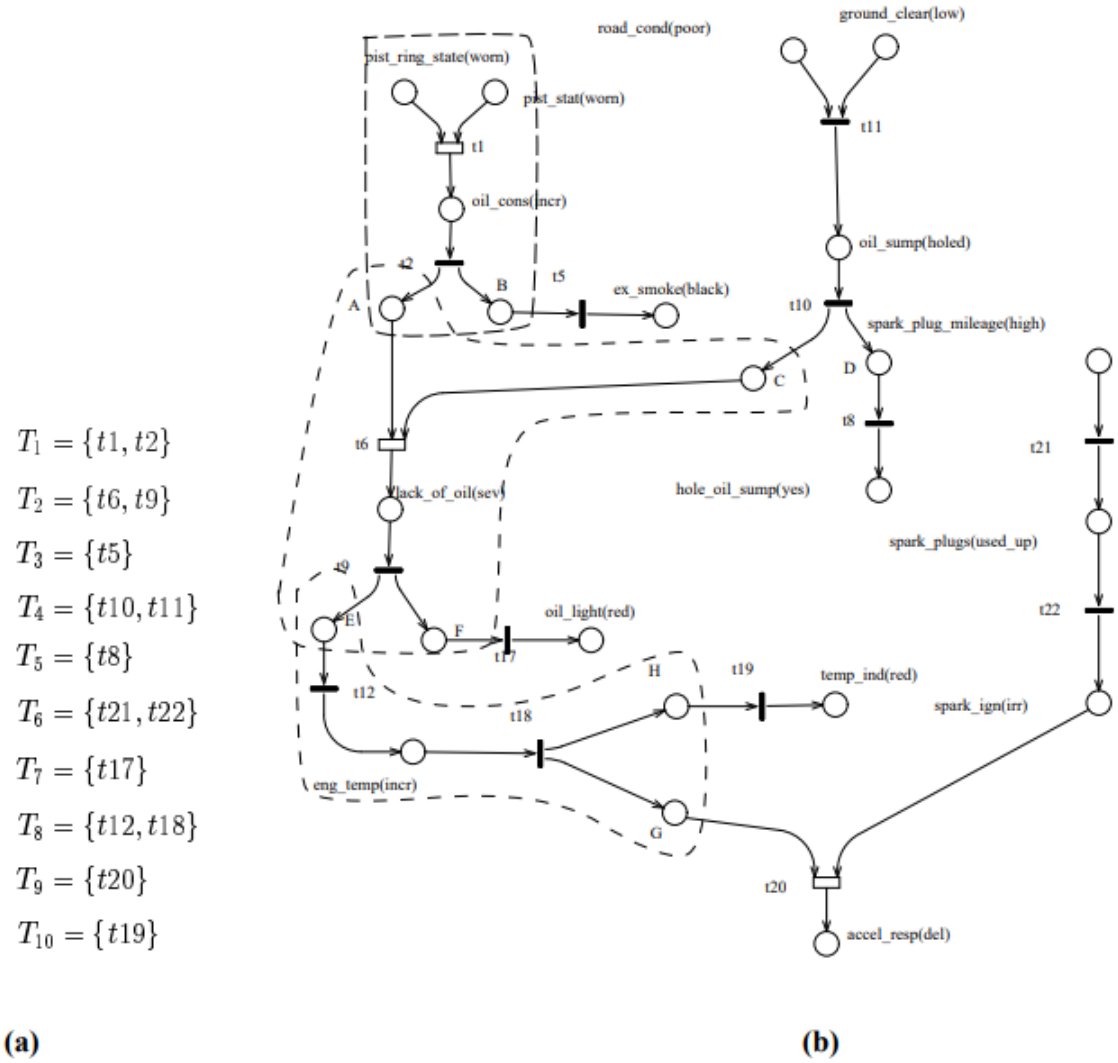

FIGURE 2.5 – un exemple d'application de la technique de partitionnement

2.4.2 Dérivation de processus

Etant donné un modèle BPN, un processus est associé à chaque TGS généré, chaque processus est construit en fournissant, pour chaque transition dans un TGS donné, une activité mettant en œuvre sa règle de franchissement en arrière, en composant séquentiellement les activités ci-dessus à partir de la transition de départ et ajouter consécutivement celles liées aux transitions dans le même TGS. On utilise Un processus supplémentaire appelé processus d'assemblage (AP) pour collecter les résultats partiels des processus contenant les places sources et il est responsable de l'assemblage de la solution finale.

Selon la définition 6, chaque transition t d'un BPN N= (P, $T_n, T_o r, F$) peut être classée comme transition Join, fork ou transition linéaire. À chaque classe de transition, une activité simple, exécutant la règle de franchissement en arrière pour cette classe, comme illustré dans la figure 2.6, la notation suivante a été utilisée : les listes de paramètres d'entrée et de sortie ont été spécifiées en utilisant les mots-clés in et out, tandis que le symbole '□' est utilisé pour désigner différentes alternatives mutuellement exclusives. Dans l'activité correspondant aux jointures de transitions, nous désignons comme A_i une parmi les 2n-1 instanciations possibles des valeurs aux variables de sortie (sauf que en affectant b à toutes) l'activité correspondant aux transitions linéaires a été omise car elle

correspond à une copie du contenu de sa variable d'entrée unique à la variable de sortie unique.

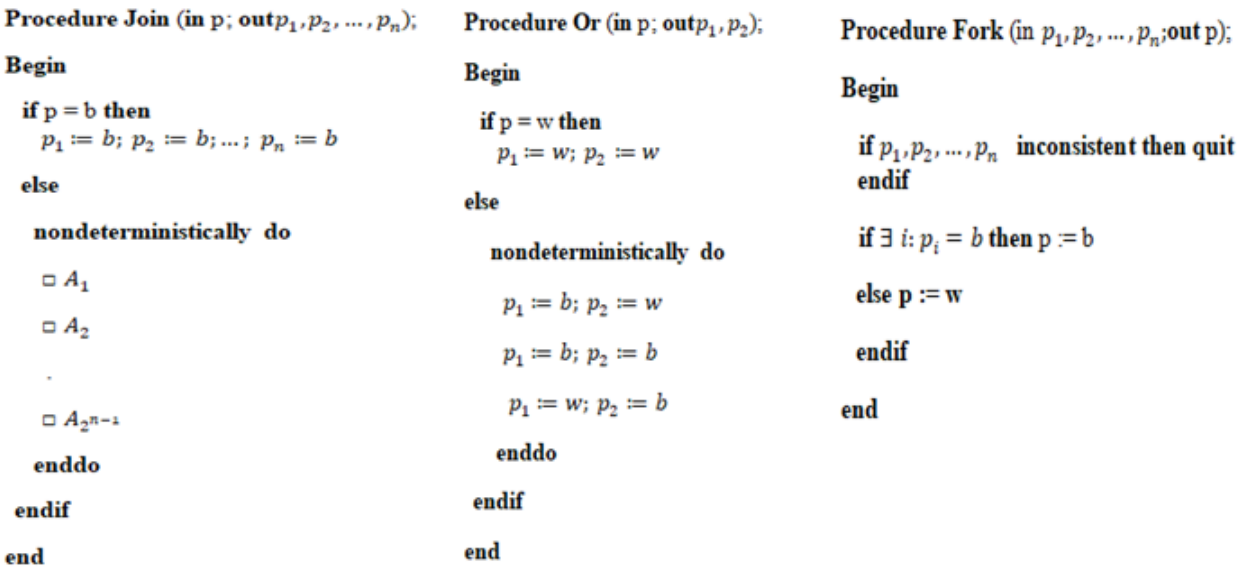

FIGURE 2.6 – activités associées aux classes de transition

Le comportement d'une activité est dérivé de la règle de franchissement en arrière, définie pour la classe de transitions correspondante, de la manière suivante :
Étant donné une transition t, nous introduisons dans l'activité associée un variable de sortie pour chaque place $^\bullet$ t et une variable d'entrée pour chaque place t $^\bullet$. Le variable contient des jetons noirs (b) ou blancs (w) ; si aucun jeton n'est contenu dans une place, le contenu de la variable correspondante n'est pas défini, pour qu'une activité démarre, le contenu de toutes ses variables d'entrée doit être défini et à la fin, le contenu de ses variables de sortie est mis à jour. Une activité correspondant à une transition ayant plusieurs places de sortie est appelée point de synchronisation, car toute sa variable d'entrée doit être définie pour l'activité démarre.L'ensemble des variables de sortie d'une activité t1 peut ne pas être dissocié de l'ensemble des variables de sortie d'une autre activité t2, la fin de t1 peut impliquer le début de t2 ; t2 reçoit en entrée les données produites par t1.
Dans ce cas, nous disons que t1 et t2 sont des activités qui interagissent. Le contenu du variable correspondant aux places puits est considéré comme un élément de données fourni par l'environnement externe, tandis qu'à d'autres places, les éléments de données sont fournis par d'autres activités.
Il est noter que l'ensemble de TGS ne forme pas de partition sur l'ensemble des places d'un BPN donné. Cela implique que certains processus ont des variables correspondant à la même place dans le BPN. Dans ce cas, nous considérons les variables comme des tampons d'entrée et de sortie d'un canal de communication entre les processus. Plus précisément si N_i, N_j sont deux TGS tels que $^\bullet E(N_i) \cap S(N_j)^\bullet \neq 0$, alors il existe un canal de communication entre les processus Π_i et Π_j, correspondant respectivement à N_i et N_j, et le flux de message se produit de manière asynchrone de Π_i à Π_j.
L'utilisation de TGS pour dériver l'ensemble de processus effectuant la technique d'analyse B-W peut entraîner des problèmes de blocage.En fait, il peut arriver que certains processus contenant des places puits ne soient jamais actifs en raison du marquage initial particulier. . Par conséquent, certains autres processus en attente de traitement des

messages inactifs sont bloqués. Cet effet se propage à travers tous les processus, produisant ainsi un blocage. Dans le modèle d'exécution séquentielle, les transitions qui ne sont jamais actives sont forcées de se déclencher (ce sont des transitions forcées). Des transitions forcées pourraient également être utilisées dans le modèle d'exécution parallèle, mais une solution plus efficace peut être conçue. Ce problème peut en fait être résolu en observant que si la transition de départ d'un TGS ne peut pas recevoir de jetons à ses places de sortie, alors ces places peuvent être supprimées. Si toutes les places de sortie de la transition de départ d'un TGS sont supprimées, le TGS global peut être supprimé. Considérant le programme concurrent, cette étape correspond à l'élimination des canaux et processus d'entrée. Pour obtenir un programme concurrent sans blocage, une étape de simplification sur tous les TGS est effectuée.

L'étape de simplification est définie comme suit :

1. étant donné un TGS N_i, supprimer tous les places p $\in$ S $(N_i)^\bullet$ qui sont des places puits non marqués.
2. si S $(N_i)^\bullet$ devient vide, retirez N_i de l'ensemble des TGS.
3. si N_i a été supprimé, supprimez tous les places dans$^\bullet E(N_i)$.

L'étape de simplification est effectuée de manière itérative en considérant chaque TGS tour à tour. Après l'étape de simplification, l'ensemble des processus correspondant au TGS restant est construit. De cette façon, aucun processus n'attendra un message qui ne peut pas être envoyé, le programme résultant est donc sans blocage.
L'ensemble de TGS correspondant à un BPN donné peut être considéré comme un modèle, pour l'ensemble de processus effectuant une analyse B-W, qui peut être instancié selon le marquage initial pour l'analyse B-W. Lorsqu'un tel marquage est spécifié, une étape de simplification est réalisée et seul le processus correspondant au TGS restant est construit et programmé pour exécution. La construction de l'ensemble de TGS ne doit être effectuée qu'une seule fois pour un modèle BPN donné, l'étape de simplification doit être effectuée chaque fois qu'un nouveau marquage initial pour l'analyse B-W est spécifié .[Bla+06].

2.4.3 Plusieurs solutions et vérification des incohérences

L'objectif est de concevoir est d'analyser toutes les évolutions possibles du système. Afin d'accomplir cette tâche, le comportement des activités élémentaires doit être modifié comme expliqué ci-dessous. Lorsqu'un choix non déterministe est présent, chaque alternative possible est générée puis propagée entre les activités, permettant le calcul de l'ensemble global des solutions au sein d'une même exécution. Il convient de noter que la génération exhaustive d'alternatives implique que chaque lieu peut contenir simultanément plus de jetons, chacun étant lié à une alternative particulière. Pour gérer correctement cette situation, il est nécessaire de suivre les différentes évolutions générées. Cela peut être accompli en modifiant l'interprétation des variables, en les considérant comme des tampons de mémoire qui peuvent contenir plusieurs éléments du type (c, v) où c $\in \{b, w\}$ représente soit un jeton noir (c = b) ou un jeton blanc (c = w) et v appelé **identificateur d'évolution (eid)**, est un carectère désigne un chemin dans le graphe de raisonement en arrière, d'une manière similaire le contenu de chaque message échangé entre les processus peut être défini.
En utilisant cette méthode, toutes les alternatives possibles, générées lors de l'analyse B-W, peuvent être considérées dans la même exécution ; cela signifie que toutes les solutions possibles au problème de diagnostic peuvent être collectées à la fin de l'analyse. Il

convient de noter que de cette manière, à la fin de l'analyse, le graphe d'accessibilité en arrière n'est pas disponible pour le processus d'assemblage. Néanmoins, il peut être facilement construit par AP si les éléments de données sont modifiés pour inclure également des informations sur le franchissement de la transition [Bla+06].
Le choix d'un ensemble de processus fonctionnant de manière asynchrone implique qu'il n'y a pas de représentation explicite du marquage, qui est plutôt divisé entre différents processus. Ce cas implique un protocole pour la vérification des incohérences. En fait, un processus découvrant l'incohérence d'un marquage donné devrait signaler l'incohérence à tous les autres processus gérant différentes parties d'un même marquage. Quelle que soit l'absence d'un état global, il se peut que les processus découvrant l'incohérence ne connaissent pas les identités des autres processus détenant une partie du même marquage. Nous proposons une solution appelée i-propagation, basée sur la propagation des points de synchronisation des incohérences. Le processus découvrant le marquage incohérent produit un élément de données du type (i, v) où i signifie incohérent et v est l'eid actuel, un tel élément de données est ensuite propagé au moyen de messages à d'autres processus, l'incohérence est décourageant au moyen d'une opération de fusion (merge), définie sur l'élément de données susmentionné, effectuée à chaque point de synchronisation. L'opération de fusion (merge) est une opération commutative définie comme suit[Bla+06] :

Definition [23] : Soit v'un préfixe de v et c, c' ∈ (b, w) tel que c ≠ c'

- merge($< c, v >, < c, v' >$) $=< c, v >$
- merge($< c, v >, < c', v' >$) $=< i, v >$
- merge($< c, v >, < i, v' >$) = $merge(< i, v >, < c, v' >) =< i, v >$

Etant donné que la première opération après un point de synchronisation correspond à une transition de classe fork, le résultat de la fusion est mis au place d'entrée d'une telle transition. L'opération merge est effectuée sur des éléments de données avec le même eid, contenus dans des tampons de mémoire différents correspondant aux places de sortie de la transition.A la fin de l'analyse, le processus d'assemblage vérifie tous les éléments de données contenus dans les places source. Si l'un de ces places contient un élément de données du type (i, v), alors à partir de chaque place source, les éléments de données (c, v), qui sont liés à la même évolution v, sont supprimés et ne sont pas considérés comme une solution finale. [Bla+06].

2.4.4 Exemple

Dans cette section, nous décrivons un exemple de dérivation du programme parallèle mettant en œuvre l'analyse B-W pour le BPN représenté sur la figure 2.5 et son exécution sur un marquage de départ particulier. Dans ce qui suit, nous désignons comme N_i le sous-réseau généré par la transition correspondant au sous-réseau de partitionnementT_i, et avec Π_i le processus correspondant à N_i.
Supposons que le marquage de départ pour l'analyse B-W est ex-smoke (black) [w], oil-light (red) [b], temp-ind (red) [b]. L'étape de simplification sur l'ensemble des TGS est réalisée comme suit :

— N9 est supprimé de l'ensemble de TGS puisque S $(N_9)^\bullet$ ne contient qu'une seule place non marqué.
— les places G et spark-ign (irr) sont supprimés à la suite de la suppression deN_9.
— les places G et spark-ign (irr) sont supprimés à la suite de la suppression de N_9.
— N_6 est supprimé car la place unique dans S $(N_6)^\bullet$ (spark-ign (irr)) a été supprimée à la suite de la suppression de N_9.
— N_5 est supprimé car S $(N_5)^\bullet$ ne contient qu'une seule place non marqué.

Après l'étape de simplification décrite, l'ensemble des processus correspondant aux TGS restants est construit. Sur la figure 2.7, l'interaction entre les processus est représentée par un graphique dans lequel les nœuds correspondent aux processus et les arcs dirigés correspondent aux communications sont représentés par des directions d'arc.

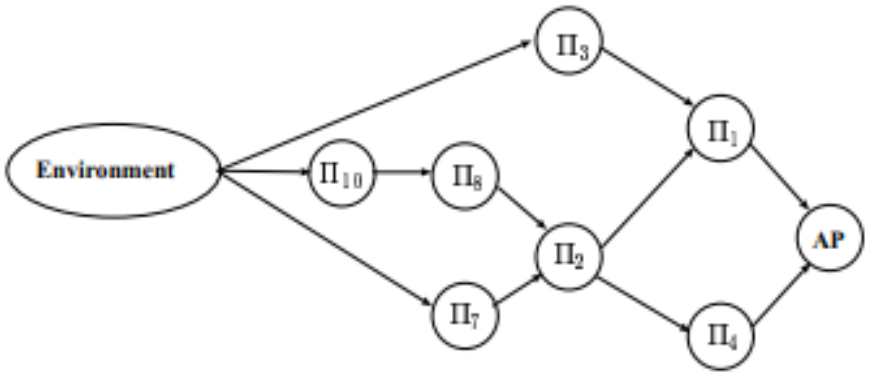

FIGURE 2.7 – structure d'interaction des processus correspondant au TGS dans a figure 2.5

On présente maintenant un exemple d'exécution du programme concurrent sur le marquage de départ défini ci-dessus. Par souci de simplicité dans la description, nous supposons que si plus d'alternatives sont générées par un processus, alors tous les messages correspondants sont envoyés dans un groupe de signaux à chaque processus de destination. De plus, nous ne considérons pas le fait que différents processus peuvent avoir besoin de temps différents pour effectuer un calcul interne avant d'envoyer un message. Cela implique qu'après réception d'un message, la prochaine action effectuée par un processus est l'envoi du message correspondant aux résultats de son calcul interne. Cet expédient nous permet de décrire l'exécution du programme exemple de façon étape par étape. Cette hypothèse n'est faite qu'à des fins de description ; en fait, dans les exécutions réelles, toutes les actions décrites se produisent de manière asynchrone.
Dans la description, nous utilisons la notation $\Pi_i \overset{Q}{\longrightarrow} \Pi_j : \{m_1, ..., m_k\}$ pour indiquer que le processus Π_i envoie au processus Π_j le groupe de messages $m_1, ..., m_k$ via le canal Q.
La notation $\Pi_j \overset{Q}{\rightsquigarrow} \Pi_i$ est utilisée pour indiquer que le processus Π_i attend un message du processus T_j via le canal Q.
La spécification du canal sur lequel les messages sont envoyés est nécessaire si un processus contient plus d'une place source (dans notre exemple Π_1 et Π_4). De cette façon, le processus d'assemblage peut construire la solution finale. Il convient de noter qu'un processus peut avoir plus de deux canaux d'entrée.
L'opération de fusion doit donc être appliquée de cette manière :
exemple si $\{m_1, ..., m_n\}$ est un ensemble de messages tels que m_1 est reçu du canal P_i, que la séquence résultante d'applications de l'opération de fusion est

$$\text{merge } (m_n, merge(m_{n-1}, merge(m_{n-2}, merge(\ldots merge(m_2, m_1)\ldots))))$$

la fonction tmerge * ($\{m_1, ..., m_n\}$) est utilisée pour désigner l'opération de fusion appliquée à un ensemble de messages. Cette fonction est récursivement définie comme

- merge * ($\{m_1, \tau\}$) = m_1
- merge * ($\{m_1, m_2, \ldots, m_n\}$)=merge(merge *($\{m_1, \ldots, m_(n-1)\}$), m_n)

Ou $m_1, .., m_n$ sont des messages et τ désigne le message vide.
dans ce qui suit, nous désignons respectivement C_1, C_2, C_3, C_4 le canal de communication pist-ring-state (worn), pist-state (worn), road-cond (poor) et ground-clear (low).
La génération d'un nouvel eid s'effectue par concaténation de l'eid précédent avec un

entier représentant une nouvelle évolution générée, séparée par le caractère '#'. Chaque élément de données fourni par l'environnement externe est supposé avoir le eid défini la chaîne vide ε.

Étape 0 : initialement tous les processus attendent une entrée. L'environnement externe effectue l'action suivante : Env $\rightarrow \Pi_3 < w, \epsilon >, Env \rightarrow \Pi_7 < b, \epsilon >, Env \rightarrow \Pi_1 0 :< b, \epsilon >$
. **Étape 1** : $\Pi_3 \rightarrow \Pi_1 :< w, >, \Pi_1 0 \rightarrow \Pi_8 :< b, >, \Pi_7 \rightarrow \Pi_(2) :< b, \epsilon >$.
$\Pi_2 \rightsquigarrow \Pi_1, \Pi_8 \rightsquigarrow \Pi_2$,$\Pi_2 \rightsquigarrow \Pi$, $\Pi_1 \overset{C_1}{\rightsquigarrow} AP$, $\Pi_1 \overset{C_2}{\rightsquigarrow} AP$, $\Pi_4 \overset{C_3}{\rightsquigarrow} AP$, $\Pi_4 \overset{C_4}{\rightsquigarrow} AP$
Étape 2 : $\Pi_8 \rightarrow \Pi_2 :< b, \epsilon >$.Π_2 effectue merge*($\{< b, \epsilon >, < b, \epsilon >\}$) , produire le message $< b, \epsilon >$;
$\Pi_2 \rightsquigarrow \Pi_1, \Pi_2 \rightsquigarrow \Pi_4$,$\Pi_1 \overset{C_1}{\rightsquigarrow} AP$, $\Pi_1 \overset{C_2}{\rightsquigarrow} AP$, $\Pi_4 \overset{C_3}{\rightsquigarrow} AP$, $\Pi_4 \overset{C_4}{\rightsquigarrow} AP$
Étape 3 : Π_2 produire trois messages correspondant à différentes affectations puisque la place de sortie de sa transition de fin (t6) contient $< b, \epsilon >$.
$\Pi_2 \rightarrow \Pi_1 : \{< b, \#1 >, < b, \#2 >, < w, \#3 >\}$
$\Pi_2 \rightarrow \Pi_4 : \{< b, 1 >, < w, 2 >, < b, 3 >\}$
$\Pi_2 \rightsquigarrow \Pi_1, \Pi_2 \rightsquigarrow \Pi_4$,$\Pi_1 \overset{C_1}{\rightsquigarrow} AP$, $\Pi_1 \overset{C_2}{\rightsquigarrow} AP$, $\Pi_4 \overset{C_3}{\rightsquigarrow} AP$, $\Pi_4 \overset{C_4}{\rightsquigarrow} AP$
Étape 4 : Π_1 effectue l'opération de fusion suivante :
merge*($\{< w, \epsilon >, < b, \#1 >\}$) $=< i, \#1 >$
merge*($\{< b, \epsilon >, < w, \#2 >\}$) $=< i, \#2 >$
merge*($\{< w, \epsilon >, < w, \#3 >\}$) $=< w, \#3 >$
$\Pi_1 \overset{C_1}{\rightsquigarrow} AP\{< i, \#1 >, < i, \#2 >, < w, \#3 >\}$
$\Pi_1 \overset{C_1}{\rightsquigarrow}$ AP$\{< i, \#1 >, < i, \#2 >, < w, \#3 >\}$
$\Pi_4 \overset{C_3}{\rightsquigarrow} AP\{< w, \#1 >, < b, \#2 >, < b, \#3 >\}$
$\Pi_4 \overset{C_4}{\rightsquigarrow} AP\{< w, \#1 >, < b, \#2 >, < b, \#3 >\}$
Étape 5 : AP rejette de toutes les sources les éléments de données liés aux évolutions #1 et #2 car il existe des éléments de données du type $< i, \#1 >$ and $< i, \#2 >$.
Les resultants finial sont :
$< pist{-}state{-}ring(worn)[w], pist{-}state(worn)[w], road{-}cond(poor)[b], ground{-}clear(low)[b] >$.

2.5 Conclusion

Dans ce chapitre, nous avons présenté les concepts de base des BPNs, ensuite l'approche adoptée pour la résolution d'un problème de diagnostic en parallèle. Le dernier chapitre sera consacré à décrire les étapes de développement d'un outil de dérivation de processus parallèle.

Deuxième partie

IMPLEMENTATION

Chapitre 3

Développement d’un outil de dérivation de processus parallèl

Chapitre 3

Développement d'un outil de dérivation de processus parallèle pour l'exploitation d'analyse B-W

Introduction

On a présenté dans le chapitre précédent une technique d'analyse basée sur les BPNs pour la résolution de problème de diagnostic. Pour exploiter cette technique on développe un outil basé sur les principes de la technique.
Le développement de l'outil suit le cycle de vie d'un logiciel.

3.1 Analyse de besoins

L'analyse de besoins est la première phase dans le cycle de vie d'un logiciel. Elle consiste à définir les besoins d'utilisateur et les contraintes. On s'intéresse au partitionnement du modèle BPN qui représente le comportement anormal du système et à l'application de l'analyse B-W par la dérivation de processus parallèles. Les fonctionnalités qui doivent être fornies par cette application peuvent être résumées comme suit :

- La représentation graphique du comportement anormal du système par un BPN.
- Le partitionnement des transitions.
- La dérivation d'ensemble de processus qui effectué l'analyse B-W pour obtenir les diagnostics (marquages initiaux) correspondant aux causes initiales du problème.

les contraintes
- le comportement du système doit être fini (sans cycle, sans objet isolé, il n'y a pas de transitions sources ou puits)

3.2 Conception globale

La conception globale montre d'une façon abstraite l'ensemble des modules et les relations entre eux.

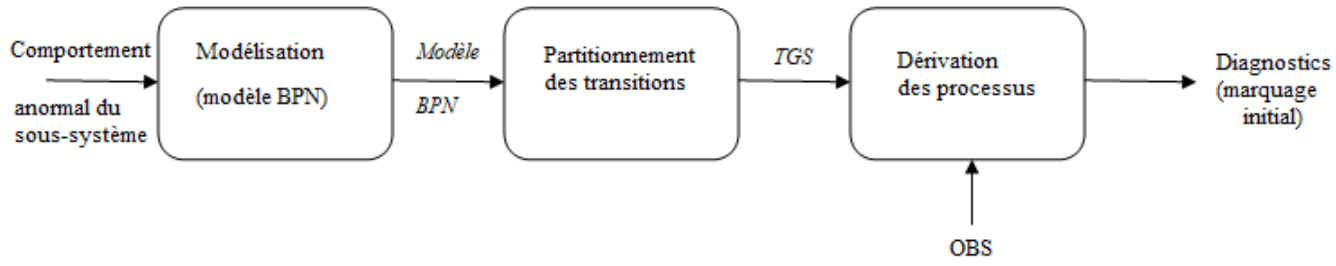

FIGURE 3.1 – Architecture générale d'interaction entre les modules du système.

3.2.1 Module de modélisation

La fonction principale de ce module est l'édition d'un modèle BPN qui représente le comportement anormal du système

3.2.2 Module de partitionnement des transitions

Ce module s'occupe de la partition du modèle BPN en sous-modèles BPNs. Il recçoit comme entrée le BPN initial et fournis comme sortie un ensemble de sous-réseaux générés par transitions (TGS).

3.2.3 Module de dérivation de processus

La principale tâche de ce module est la dérivation de processus associés a chaque TGS et appliquer les règles de franchissement en arrière sur chaque sous modèle.

3.3 Conception détaillée

La conception détaillée consiste à fournir pour chaque composant ses algorithmes et structure de données.

3.3.1 Structure de données

On va décrire les structures de données qui sont utilisées par tous les composants de notre outil.

3.3.1.1 Modélisation

Ce module fournis aux utilisateurs une interface graphique pour l'edition d'un modèle BPN.

Classe BPN

- name : chaîne de carectères disignant le nom de BPN
- Places : liste d'objets contenant tous les places de notre modèle BPN.
- OrTransition : liste d'objets contenant les transitions de type OR.
- AndTransition : liste d'objets contenant les transitions de type AND.
- Arcs : liste d'objets contenant les arcs.

Classe Place

- id : chaîne de carectères disignant l'identificateur de place.
- marq : chaîne de carectères indique le marquage de la place.
- inArc : liste d'objets contenant les arcs entrant.
- outArc : liste d'objets contenant les arcs sortant.
- inTransition : liste d'objets contenant les transitions entrant.
- outTransition : liste d'objets contenant les transitions sortant.
- posX,posY : nombres entiers représentant les coordonnées graphiques.

Classe orTransition

- id : chaîne de carectères disignant l'identificateur de transition.
- inArc : liste d'objets contenant les arcs entrant.
- outArc : liste d'objets contenant les arcs sortant.
- inPlace : liste d'objets contenant les places entrante.
- outArc : liste d'objets contenant les places sortante.
- posX,posY : nombres entiers représentaent les coordonnées graphiques.

Classe andTransition

- id : chaîne de carectères disignant l'identificateur de transition.
- inArc : liste d'objets contenant les arcs entrant.
- outArc : liste d'objets contenant les arcs sortant.
- inPlace : liste d'objets contenant les places entrante.
- outArc : liste d'objets contenant les places sortante.
- posX,posY : nombres entiers représentant les coordonnées graphiques.

Classe arc

- id : chaîne de carectères disignant l'identificateur d'arc.
- source : liste d'objet containe la source d'arc,cette source peut être une place ou transition.
- target : liste d'objet containe la destination d'arc,cette destination peut être une place ou transition.
- posX1,posY1,posX2,posY2 : nombres entier représentent les coordonnées graphique .

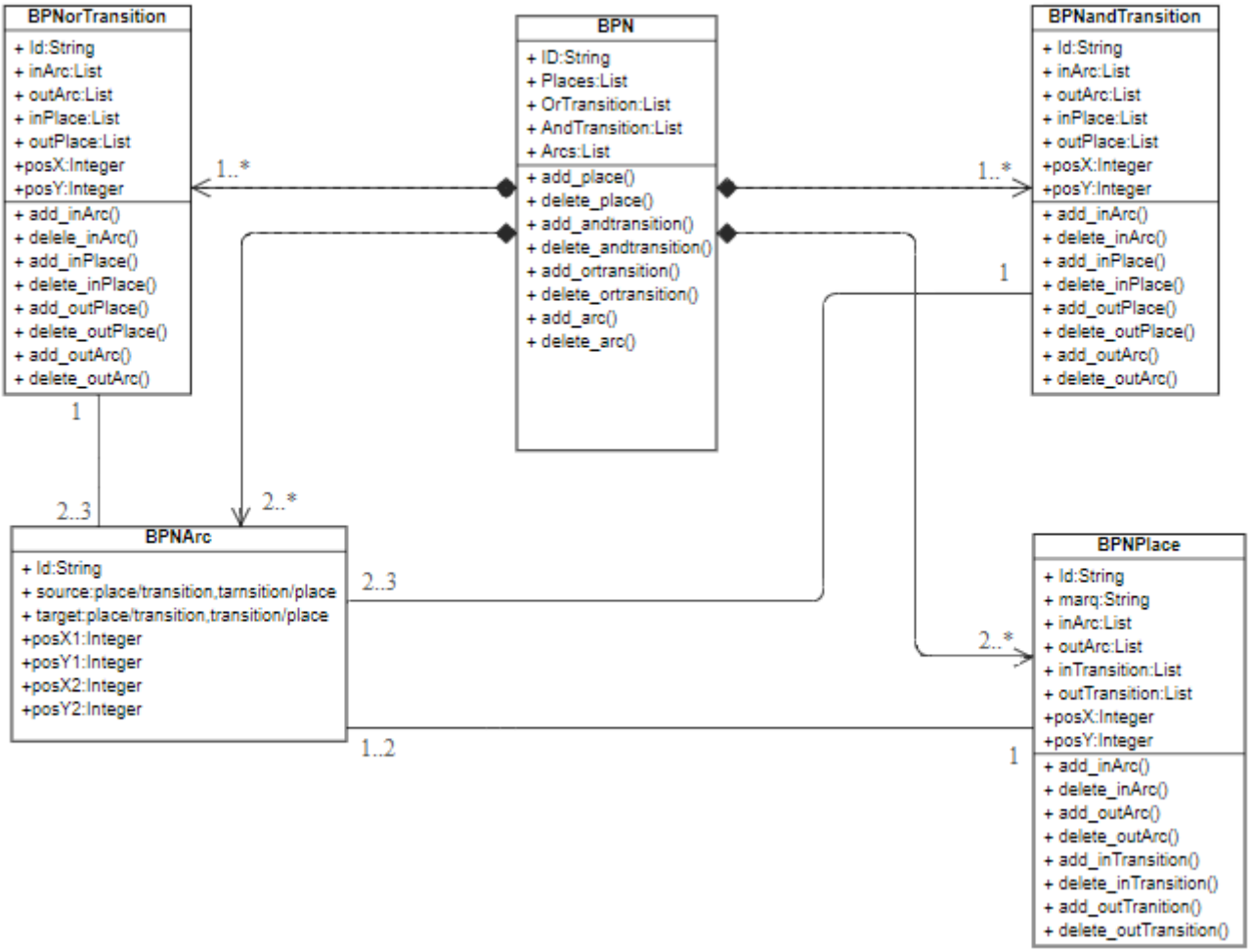

FIGURE 3.2 – Diagramme de classe.

Fichier d'enregistrement

Afin d'arriver aux résultats attendus, on a besoin de sauvgarder le modèle BPN saisie par l'utilisateur dans un fichier d'enregistrement, ce fichier doit contenir toutes les informations du modèle .
Dans notre travail on a utilisé les fichiers XML

```
<?xml version="1.0" ?>
<BPN>
    <place>
        <id name="pist_ring_state(worn)"/>
        <marq name="e"/>
        <outArcs name="a1"/>
        <outtransition name="t1"/>
        \</place>
    <place>
        <id name="pist_state(worn)"/>
        <marq name="e"/>
        <outArcs name="a2"/>
        <outtransition name="t1"/>
    </place>
    <place>
        <id name="oil_cons(incr)"/>
        <marq name="e"/>
        <inArcs name="a3"/>
        <intransition name="t1"/>
        <outtransition name="t2"/>
    </place>
    <place>
        <id name="A"/>
        <marq name="e"/>
        <inArcs name="a5"/>
        <outArcs name="a7"/>
        <intransition name="t2"/>
```

FIGURE 3.3 – Extrait d'un fichier XML.

3.3.1.2 Partitionneemnt des transitions

- bn-relation : liste contient tous les transitions qui ont une relation binaire entre eux.
- starting-transition : liste contient tous les transitions de départ du modèle.
- transition-subset : liste contient sous-ensemble des transitions généré par une transition de départ .
- TGS : liste contient les champs suivants :
  - bpn : objet de type BPN.
  - S : objet contient la transition de départ de ce bpn.
  - E : objet contient la transition de fin de ce bpn.

3.3.1.3 Dérivation de processus

- process : liste de processus associé à chaque TGS.
- OBS : liste constitué par les champs suivants :
  - id : chaine de caractère,C'est l'identificateur de place observer.
  - marq : le marquage de cette place.
- Place-vide : liste des places vides sortants des transitions forcée.
- transition-franchissable : liste des transitions franchissable.
- input-var : liste constitué par les champs suivants :
  - id : chaine de caractère,c'est l'identificateur de place .
  - var : liste qui contient les champs suivant :
    - c : caractère indique le marquage de place{b,w}.
    - v : nombre entier représente l'identificateur d'evolution.
- Nets : liste des BPNs qui sont outplaces de transition de départ sont marquées.

3.3.2 Algorithmes

3.3.2.1 Module de modélisation

Supprimer une place au modèle BPN

Pendant l'edition du modèle BPN, un ensemble d'operation sont mis en service pour l'utilisateur,parmi ces opération la possibilité du supprimer une place .

Algorithm 1 Algorithme de supprission d'une place

```
function DELETE_PLACE()
    canvas.delete(p.objet)
    canvas.delete(p.idtext)
    canvas.delete(p.martext)
    bpn.delete_place(p)
end
```

3.3.2.2 Module de Partitionneemnt des transitions

Transitions de départ de modèle BPN

Une transition de départ a soit plusieurs places de sortie, soit exactement une seule place de sortie.

Algorithm 2 Algorithme transitions de départ

```
function STARTING TRANSITION(BPN)
    for t in transition do
        if len(t.outplace)>1 then
            starting_transition.append(t)
        if (len(t.outplace)==1 and len(t.outplace.outtransition)==0 ) then
            starting_transition.append(t)
end
```

Relation binaire

le but de catte algorithme est remplir la liste bn-relation

Algorithm 3 Algorithme relation binaire

```
function RELATION BINAIRE(BPN)
    for t in transition do
        if (len(t.outplace)==1 and len(t.outplace.outTransition)==1) then
            tj=t.outplace.outTransition
            if len(tj.inplace)==1 then
                bn_relation.append([t,tj])
end
```

Sous-ensembles de transition

A partir d'un transition de départ on peut remplir la liste transition-subset

Algorithm 4 Algorithme sous-ensembles de transition

```
function SOUS-ENSEMBLE-DE-TRANSITION(BPN)
    for t in starting_transition do
        T= []
        T.append(t)
        transition_subset.append(T)
        for i in rang(len(bn_relation)) do
            if t == bn_relation[i][0] or t == bn_relation[i][1] then
                T.remove(t)
                transition_subset.remove(T)
                for a in rang(len(bn_relation[i])) do
                    T.append(bn_relation[i][a])
                transition_subset.append(T)
end
```

3.3.2.3 Module de derivation de processus

Etat de simplification

Aprés l'etat de simplification, aucun processus n'attentra un message qui ne peut pas etre envoyé

Algorithm 5 Algorithme d'etat de simplification

```
function SIMPLIFICATION(BP)
    for bpn in TGS do
        t=bpn[1][0]
        # t c'est une transition de départ
        for p in t.outplace do
            if p.marq=="e" and len(p.outtra)== 0 then
                t.outplace.remove(p)
        if len(t.outplace)==0 then
            TGS.remove(bpn)
end
```

Observations

le but de cette algorithme est remplir la liste d'observation

Algorithm 6 Algorithme OBS

```
function GET_OBS(BPN)
    for p in bpn.Places do
        if (p.marq! ="e" and len(p.outtra) == 0) then
            OBS.append([p.id,p.marq])
    /* envoyer les oservations à la liste input_var
end
```

Règles de franchissement

L'objective de cette algorithme est marquée les places entrantes du transition

Algorithm 7 Algorithme Règles de franchissement

```
for t in transition'_franchissable do
    if t.__class__.__name__ == 'BPNandTransition' then
        if len(t.outplace) == 1 then
            if t.outplace.marq=="b" then
                /*marquer tous les places entrantes par un jeton noir en la liste input_var
            if t.outplace.marq=="w" then
                /*marquer tous les places entrantes par l'alternative choix en la liste input_var
                /* p1.marq="w" et p1.marq="w"
                /* p1.marq="w" et p1.marq="b"
                /* p1.marq="b" et p1.marq="w"
        if len(t.outplace) > 1 then
            for p in t.outplace do
                if p in place_vide then
                    /*marquer la place par un jeton de meme type du autre place
            if p1.marq ==p2.marq then
                /*marquer la place entranter par le meme type de jeton
            if p1.marq <>p2.marq then
                /*marquer la place entranter par "i"
                /*"i" signifié l'inconsistance
    if t.__class__.__name__ == 'BPNorTransition' then
        if t.outplace.marq=="w" then
            /*marquer tous les places entrantes par un jeton blanc en la liste input_var
        if t.outplace.marq=="b" then
            /*marquer tous les places entrantes par l'alternative choix en la liste input_var
            /* p1.marq="b" et p1.marq="b"
            /* p1.marq="b" et p1.marq="w"
            /* p1.marq="w" et p1.marq="b"

end
```

dérivation de processus

Algorithm 8 Algorithme dérivation de processus

```
function DERIV_PROCESS(BPN)
    with Manager()as manager :
    input-var=manager.list(input-var)
    for net in Nets do
        process=Process(target=firing_rule,args=(net,input-var))
        processes.append(process)
        process.start()
end
```

3.4 Implémentation

3.4.1 Outils et langage de dévelopement

Dans cette section, nous présentons différents outils et langages, qui nous aident lors de la réalisation de notre projet (niveau de programmation et niveau théorique)

3.4.1.1 Langage de programmation python

Python est un langage de programmation intelligent que nous avons utilisé dans la mise en œuvre de notre application. Il est facile à apprendre, car il est flexible et sa syntaxe n'est pas difficile à apprendre. Un programme Python est court que les programmes d'autres langages, en raison de la disponibilité de nombreuses fonctions implémentées. Python est un langage de programmation open source. Il est disponible pour tous les systèmes d'exploitation (Windows, LINUX, Mac OS).

3.4.1.2 Editeur de programmation Pycharm

PyCharm est un environnement de développement intégré (IDE) open source, utilisé pour la programmation python. C'est un assistant de codage puissant, il peut mettre en évidence les erreurs et introduit des correctifs rapides basés sur un débogueur Python intégré. C'est un éditeur approprié pour écrire et tester de nombreuses lignes de code et de classes, car il offre une vue structurelle du projet et une navigation rapide dans les fichiers.

3.4.1.3 Package "Tkinter"

Tkinter [] est un package open source d'interface utilisateur graphique (GUI).Il est destiné au langage de programmation Python. Nous avons préféré l'outils *Tkinter* pour développer l'interface graphique de notre application, car elle est simple à apprendre, Il est disponible sur les deux systèmes d'exploitation (Windows, Linux et Mac OS).

3.4.1.4 Package "multiprocessing"

Multiprocessing [] est un package qui prend en charge les processus de génération en utilisant une API similaire au module de threading. Le package multitraitement offre à la fois la concurrence locale et distante, contournant efficacement le verrou d'interpréteur global en utilisant des sous-processus au lieu de threads. Pour cette raison, le module multiprocesseur permet au programmeur d'exploiter pleinement plusieurs processeurs sur une machine donnée. Il fonctionne à la fois sous Unix et Windows.

3.4.1.5 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 la langue 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 renvois et des bibliographies. Il est plus familier à un programmeur informatique, car il suit le cycle code-compilation-exécution.

3.4.1.6 Editeur TEX MAKER

TEX est un éditeur gratuit et open source pour la rédaction de 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 TEX MAKER pour rédiger notre rapport et faire notre présentation, car il produit des articles et des exposés de haute qualité.

3.4.2 Implémentation

les algorithmes sont implémentés en utilisant la programmation objet orientée (POO) et un ensemble de logiciels et de matériel qui sont résumés dans le tableau suivant

Software/Hardware	Version
OS	Microsoft Windows 10 , 64bits
CPU	Intel(R) Core(TM) i3-4005U CPU @1.70GHz 1.70GHz
RAM	4.00Go
Python Interpreter	3.7.2
PyCharm	2018.3.7
Tkinter	8.6

TABLE 3.1 – versions Software/Hardware

3.4.2.1 Application d'accueil

FIGURE 3.4 – Application d'accueil.

Dans notre implémentation, nous avons développé des interfaces utilisateurs graphiques (GUI) pour la facilité d'utilisation de l'application. L'application contient une barre de menus avec trois menus (File ,Edit et Aide). La figure 3.5 illustre le menu et ses commandes.

FIGURE 3.5 – Menue fichier.

Description :
File :menue de fichier containt :
New_BPN : créer un nouveau BPN.
Save : sauvgarder le modèl BPN avec l'extension .xml.
Import : importer un modèl BPN.
Barre d'outils :

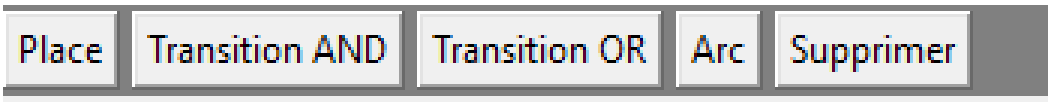

FIGURE 3.6 – Barre d'outils.

En barre d'outil, nous avons le bouton Place pour dessiner une place et le bouton Transition AND c'est pour dessiner transition de type AND,le bouton Transition OR c'est pour dessiner transition de type OR,et le bouton Arc pour dessiner un arc et le bouton Supprimer utiliser pour supprimer tous les objets existent dans la zone de modélisation.

Dessiner place Aprés le dessin de place et transitions, on peut changer id de chaque objet et entrer le marquage de place ou supprimer l'objet avec une click droite.

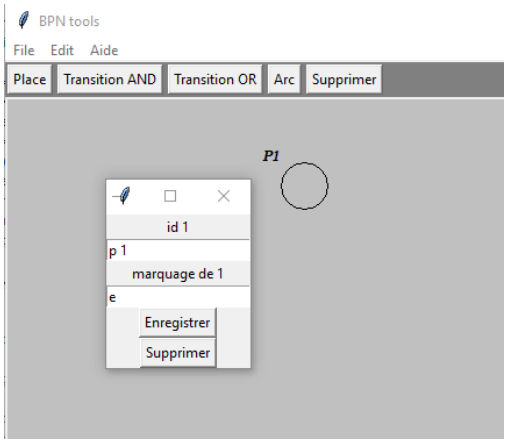

FIGURE 3.7 – Ajouter marquage.

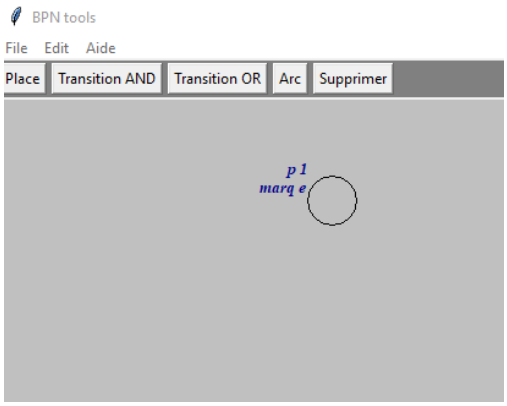

FIGURE 3.8 – Objet place

Exemple d'un modèle BPN

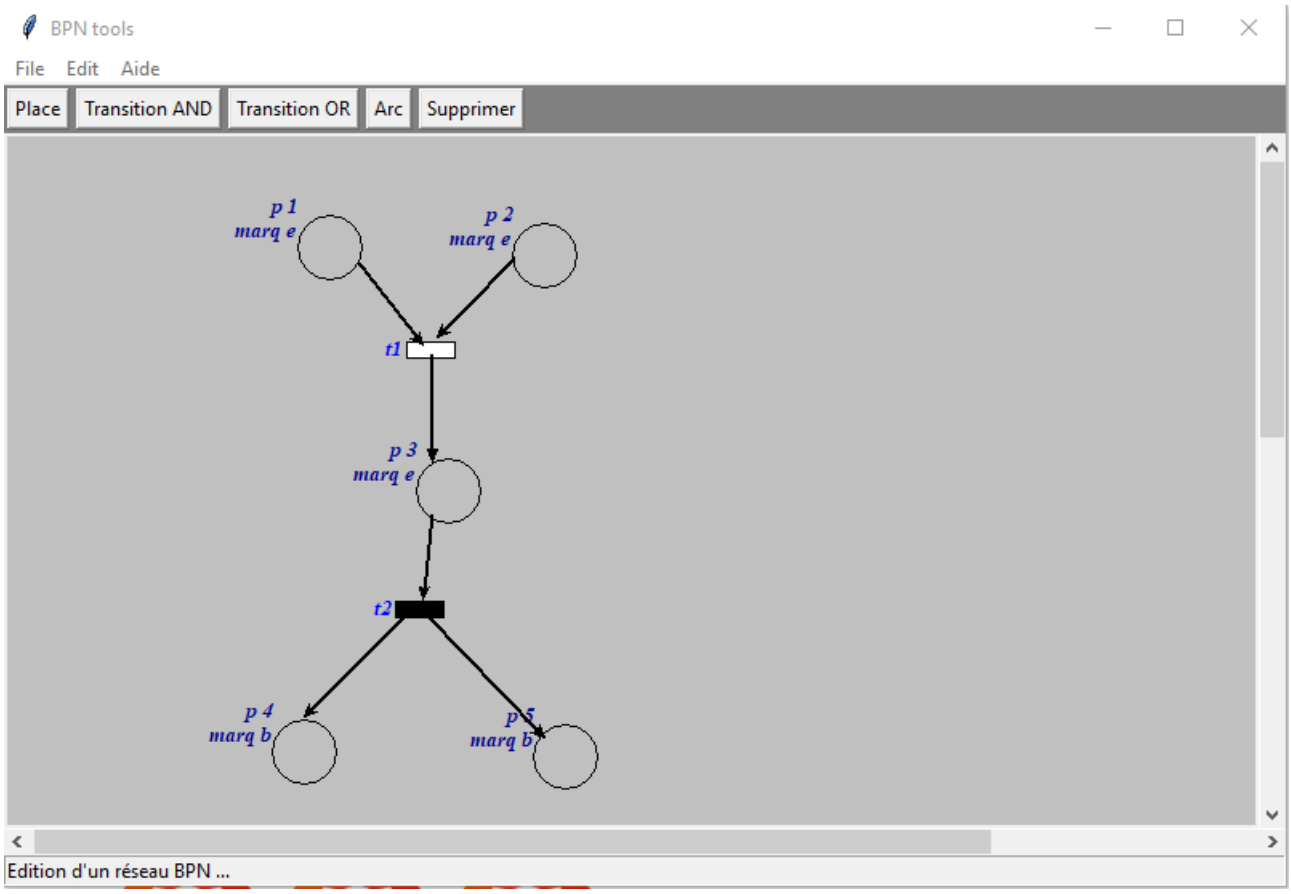

FIGURE 3.9 – Exemple d'un modèle BPN

Si nous appliquons les algorithmes précédentes sur l'exemple illustrer dans la figure 2.2 (BPN représente le comportement anormal d'un moteur d'une voiture) On a la resultat suivante

Résultat d'algorithme transition de départ

```
starting transition  ['t20', 't2', 't5', 't9', 't10', 't8', 't17', 't18', 't19']

Process finished with exit code 0
```

FIGURE 3.10 – Résultat d'algorithme transition de départ

Résultat d'algorithme relation binaire

```
****************************
binary relation  [['t1', 't2'], ['t6', 't9'], ['t11', 't10'], ['t12', 't18'], ['t21', 't22']]

Process finished with exit code 0
```

Run | 6: TODO | Terminal | Python Console

FIGURE 3.11 – Résultat d'algorithme relation binaire

Sous-ensemble générés par transition

BPN_module

```
N 1 = {
Places  ['G', 'spark_ign(irr)', 'accel_resp(del)']
andTransitions []
orTransitions ['t20']
Arcs  ['a21', 'a35', 'a36']
}
N 2 = {
Places  ['pist_ring_state(worn)', 'pist_state(worn)', 'oil_cons(incr)', 'A', 'B']
andTransitions ['t2']
orTransitions ['t1']
Arcs  ['a1', 'a2', 'a3', 'a4', 'a5', 'a6']
}
N 3 = {
Places  ['B', 'ex_smoke(black)']
andTransitions ['t5']
orTransitions []
Arcs  ['a37', 'a38']
}
N 4 = {
Places  ['A', 'C', 'lack_of_oil(sev)', 'E', 'F']
andTransitions ['t9']
orTransitions ['t6']
Arcs  ['a7', 'a8', 'a9', 'a10', 'a11', 'a12']
}
N 5 = {
Places  ['road_cond(poor)', 'ground_clear(low)', 'oil_sump(holed)', 'C', 'D']
andTransitions ['t11', 't10']
orTransitions []
Arcs  ['a23', 'a24', 'a25', 'a26', 'a28', 'a27']
}
```

FIGURE 3.12 – Sous-ensemble générés par transition 1

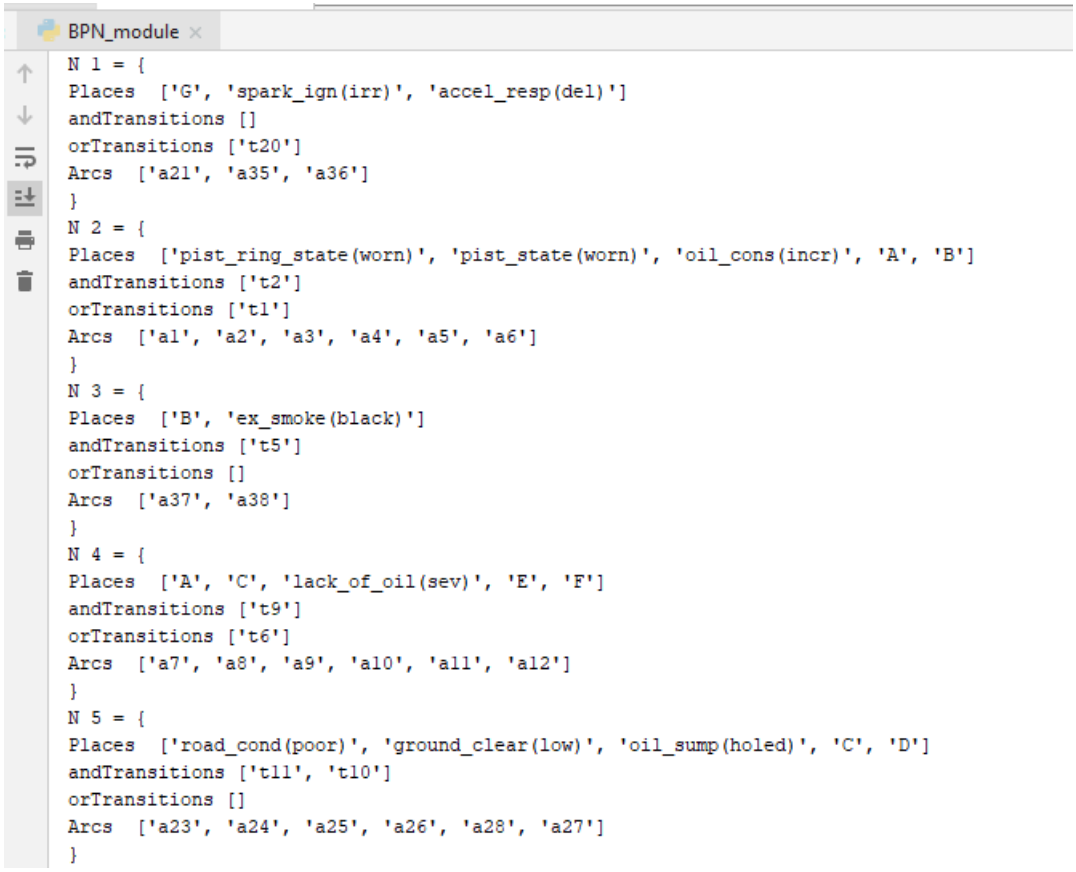

FIGURE 3.13 – Sous-ensemble générés par transition 2

Résultat d'algorithme observation

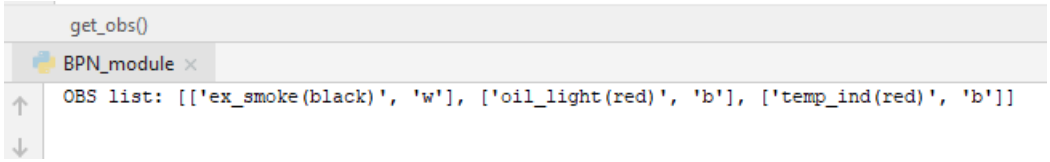

FIGURE 3.14 – Résultat d'algorithme observation

Contenu de la liste input-var

cette liste contient l'identificateur de place d'entrée et l'élément de donnée de type $< c, v >$ de chaque transition.
Lorsque nous recevons l'observation,le contenu de la liste input-var est le suivant :

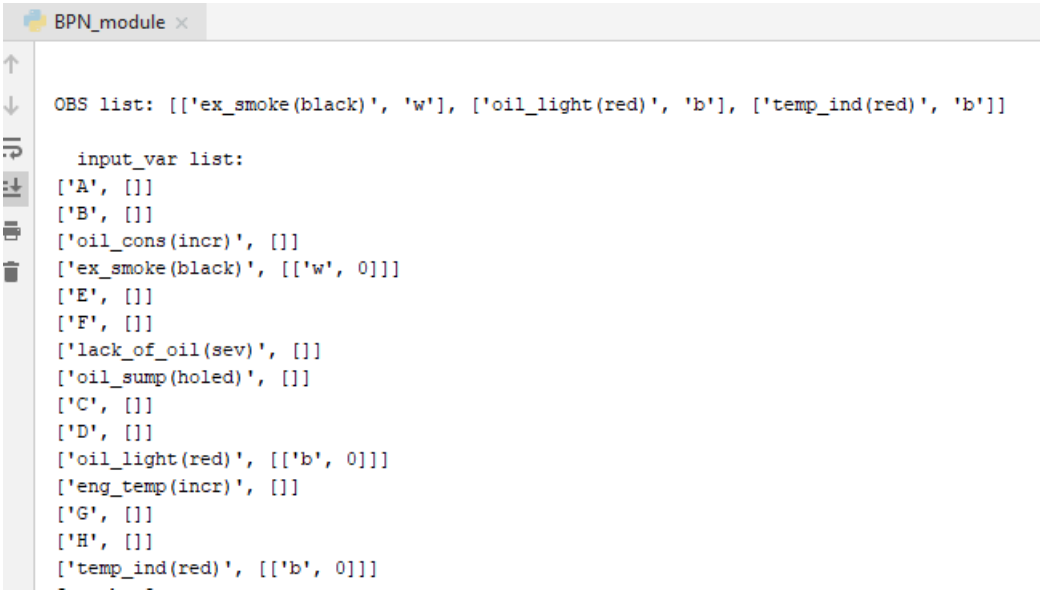

FIGURE 3.15 – La liste input-var

Dérivation de processus

Premièrement,nous devons identifier les sous-réseaux qui sont transition de départ franchissable,Ensuite en applique les règles de ftanchissement.
Les premiesr processus qui exécutent en parallèle

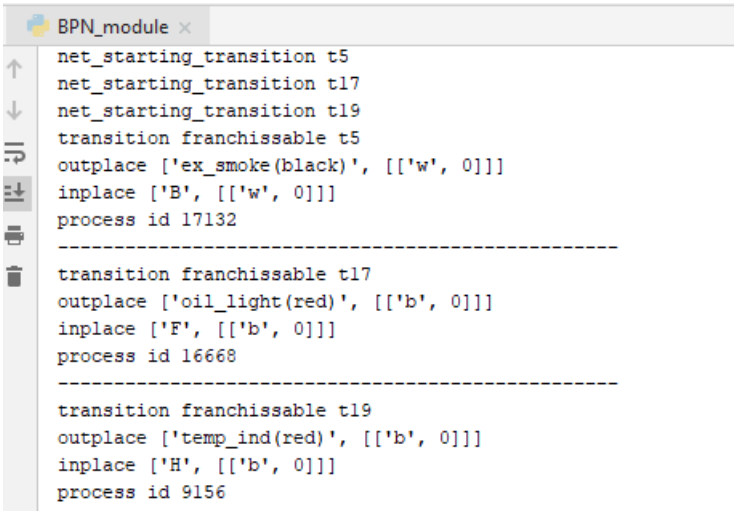

FIGURE 3.16 – Exécution de processus

Conclusion

Durant ce chapitre, nous avons essayé de décrire la démarche que nous avons suivi pour implémenter la technique.
Nous avons présenté les fontionnalités rendues aux utilisateurs dans l'analyse des besoins, suivi par la conception globale qui présente les composants constituant notre outil. Finalement, nous avons présenté les structures des données et les algorithmes utilisés par chaque composant aisni que les outils et le langage de programmation puis les résultats.

Conclusion générale

Conclusion générale

Tout système peut tomber en panne. Pour rétablir son comportement normal, il est nécessaire de localiser la vraie cause de la défaillance observée (diagnostic).

Le diagnostic basé-modèle nécessite de décrire le comportement du système à diagnostiquer sous forme d'un modèle. Plusieurs outils de représentation d'un tel modèle ont été proposés et dans ce travail on a utilisé les BPNs comme outil de représentation et de raisonnement. , En fait, l'analyse en arrière de marquages accessibles (Analyse BW) est exploitée pour réaliser le proceesus de résolution d'un problème de diagnostic à base de modèles BPNs. Ceco est accompli via l'application d'un ensemble de regèles de franchissement en arrière.

Le souci de ce travail était l'implémentation parallèle de la technique d'analyse B-W par la dérivation d'un ensemble de processus en se basant sur la structure du modèle BPN donné. Les processus dérivés sont formés de processus indépendants et des processus coopératifs. Tandis que ceux indépendants peuvent étre lancés en parallèle, les processus coopératifs nécessitent une certaine synchronisation via les places communes entre les sous-ensembles générés par transtion (point de syncronisation).

On a montré dans ce mémoire les différentes étape de développement d'un outil de dérivation de proccessus pour une implémentation parallèle de l'analyse BW.

Bibliographie

[Rei87] Raymond Reiter. "A theory of diagnosis from first principles". In : *Artificial intelligence* 32.1 (1987), p. 57-95.

[Poo89] David Poole. "Normality and Faults in Logic-Based Diagnosis." In : *IJCAI*. T. 89. Citeseer. 1989, p. 1304-1310.

[CT90] Luca Console et Pietro Torasso. "Hypothetical reasoning in causal models". In : *International Journal of Intelligent Systems* 5.1 (1990), p. 83-124.

[CT91] Luca Console et Pietro Torasso. "A spectrum of logical definitions of model-based diagnosis 1". In : *Computational intelligence* 7.3 (1991), p. 133-141.

[Ham91] Walter C Hamscher. "Modeling digital circuits for troubleshooting". In : *Artificial Intelligence* 51.1-3 (1991), p. 223-271.

[DMR92] Johan De Kleer, Alan K Mackworth et Raymond Reiter. "Characterizing diagnoses and systems". In : *Artificial intelligence* 56.2-3 (1992), p. 197-222.

[Moz92] Igor Mozetič. "Model-based diagnosis: an overview". In : *Advanced Topics in Artificial Intelligence*. Springer, 1992, p. 419-430.

[Por93] Luigi Portinale. "Petri net models for diagnostic knowledge representation and reasoning". In : *PhD Tesis, Dipartamento di Informatica–Universita'di Torino* (1993).

[VK94] Robert Valette et Luis Allan Künzle. "Réseaux de Petri pour la détection et le diagnostic". In : (1994).

[SLA04] BEKHOUCHE HAMIDA SLATNIA SIHAM. "diagnostic basé-mod'le par analyse B-W dans les BPNs". Thèse de doct. UNIVERSITE DE MOHAMED KHIDER BISKRA, 2004.

[Bla+06] Mogens Blanke et al. *Diagnosis and fault-tolerant control*. T. 2. Springer, 2006.

[Ham12] Bennoui Hammadi. "Distributed Causal Model-based Diagnosis: An Approach by Interacting Petri Nets". Thèse de doct. UNIVERSITE DE MOHAMED KHIDER BISKRA, 2012.

[] *Latex*. `https://www.latex-project.org/`. Accessed: 2020-08-28.

[] *multiprocessing*. `https://docs.python.org/3/library/multiprocessing.html`. Accessed: 2020-08-28.

[] *tkinter*. `https://docs.python.org/3/library/tkinter.html`. Accessed: 2020-08-28.