RAPPORT DE RECHERCHE

L R I

DIAGNOSTICABILITE DES SYSTEMES A EVENEMENTS DISCRETS : ETAT DE L'ART

NOUIOUA F / DAGUE P

Unité Mixte de Recherche 8623
CNRS-Université Paris Sud –LRI

03/2009

Rapport de Recherche N° 1517

CNRS – Université de Paris Sud
Centre d'Orsay
LABORATOIRE DE RECHERCHE EN INFORMATIQUE
Bâtiment 490
91405 ORSAY Cedex (France)

Diagnosticabilité des systèmes à événements discrets : État de l'art

Farid Nouioua et Philippe Dague
{Farid.Nouioua, Philippe.Dague}@lri.fr

LRI, Université Paris 11, Groupe GEMO, INRIA Saclay Ile-de-France

Parc club Université,
4, rue Jacques Monod, Bât G, 91893, Orsay

Tables des matières

1. Introduction……..… 4
2. Les « dimensions » de la diagnosticabilité……..……...……. 4
   2.1. Systèmes discrets, continus et hybrides……..…….....…….. 5
      2.1.1. Systèmes continus…...……...……... 5
      2.1.2. Systèmes discrets…...……...……... 5
      2.1.1. Systèmes hybrides…..……...……... 6
   2.2. Systèmes centralisés et systèmes distribués……... 6
3. Diagnosticabilité dans les systèmes à événements discrets……. 7
   3.1. Rappels sur les systèmes à événements discrets…….....…….. 7
      3.1.1. Notations de base……...……... 7
      3.1.2. Définition formelle de la diagnosticabilité…….…….. 8
   3.2. Approches centralisées…….…….. 9
      3.2.1. Approche à base d'un diagnostiqueur ……..……...…. 9
         3.2.1.1. Construction d'un diagnostiqueur ……...……..... 9
         3.2.1.2. Conditions de diagnosticabilité ……...……. 10
         3.2.1.3. Performance et complexité…….……... 11
      3.2.2. Une première approche à base de vérificateur ……..……..…. 12
         3.2.2.1. Construction d'un diagnostiqueur et d'un vérificateur……..... 12
         3.2.2.2. Condition de diagnosticabilité ……..……...……. 14
         3.2.2.3. Résultats de complexité …….……... 14
      3.2.3. Une deuxième approche à base d'un vérificateur ……...……... 15
         3.2.3.1. Construction des F_i-*vérifieurs*…...……..……..... 15
         3.2.3.2. Condition de diagnosticabilité ……..……...……. 15
         3.2.3.3. Résultats de complexité …….……... 16

3.3. Approches décentralisées/distribuées……..…….. 16

3.3.1. Une approche distribuée à base de vérificateurs locaux……...…. 17

3.3.1.1. Modèles locaux et modèle global……...……...……... 17

3.3.1.2. Diagnostiqueurs et vérificateurs locaux…..……...……..... 19

3.3.1.3. Un premier algorithme de vérification de la diagnosticabilité...…... 21

3.3.1.4. Un deuxième algorithme de vérification de la diagnosticabilité..... 22

4. Autres méthodes pour la vérification de la diagnosticabilité…….…….. 24

4.1. Model-checking à base d'une structure de Kripke et une logique LTL…..… 24

4.1.1. Le modèle……..……...…. 24

4.1.2. Conditions de diagnosticabilité…….……...…. 24

4.1.3. La diagnosticabilité comme problème d'atteingnabilité……..... 25

4.1.4. Expression et vérification en termes d'une Structure de Kripke…....…. 26

4.2. Utiliser un langage algébrique : PEPA……...… 27

4.2.1. Modélisation d'un système physique à l'aide de PEPA……..……. 28

4.2.1.1. Le langage PEPA……..... 28

4.2.1.2. Modélisation du système physique…… 29

4.2.2. Caractériser le diagnostic avec PEPA ……..... 30

4.2.3. Diagnosticabilité avec PEPA …… 31

4.3. Utiliser les algorithmes de satisfiabilité : SAT…….… 32

5. Conclusion….....…….. 35

6. Références……..….. 36

1. Introduction

Les systèmes automatiques complexes trouvent leurs places aujourd'hui dans de plus en plus d'applications réelles de la vie quotidienne par exemple dans l'industrie automobile et aéronautique entre autres. L'une des exigences pour ces systèmes, est qu'elle soient le plus autonomes que possible même dans des cas où de défaillances. Le problème de diagnostic automatique de défaillances dans des systèmes complexes est l'un des domaines de recherche qui ont attiré l'attention aussi bien de la communauté de contrôle de systèmes avec des approches issues de l'automatique que de la communauté d'Intelligence Artificielle (IA) avec les travaux sur le diagnostic à base de modèles initié par le papier de Reiter [Reiter, 87]. Ces approches ont débouché ces dernières années sur l'étude de la diagnosticabilité d'un système qui consiste à concevoir et implémenter des algorithmes de vérification de propriétés formelles du système garantissant qu'un modèle, dont on connaît à l'avance les événements observables, permet la détection et la discrimination d'un ensemble de défaillances possibles connues a priori et incorporées au modèle du système. L'étude de la diagnosticabilité intervient principalement dans la phase de conception du système et permet de savoir si les observables prévus pour le système (souvent, l'ajout d'observables dans un système revient à ajouter des capteurs dans ce système) suffisent à discriminer au bout d'un temps fini, d'une part, le fonctionnement normal de tout fonctionnement fautif et d'autre part, tout fonctionnement fautif de tout autre fonctionnement fautif.

Dans ce papier, nous présentons un état de l'art de travaux effectué dans le domaine de la diagnosticabilité. Nous commençons, dans la section, par présenter les différentes variantes que l'on peut trouver pour le problème de la diagnosticabilité selon la nature du système modélisé. Nous nous focalisons dans la section 3 sur la diagnosticabilité des systèmes à événements discrets, plus précisément le cas le plus communément utilisé où le système est modélisé par un automate fini. Nous passerons en revue ainsi quelques algorithmes de vérification de la diagnosticabilité aussi bien dans le cadre centralisé que dans le cadre distribué. Dans la section 4, nous présentons trois autres approches proposées pour traiter le problème de diagnosticabilité à savoir : l'utilisation de techniques de model-checking, la modélisation et la vérification par un langage algébrique et la réduction du problème de vérification de la diagnosticabilité à un problème SAT.

2. Les « dimensions » de la diagnosticabilité

Contrairement au diagnostic en ligne qui peut parfois être effectué même en l'absence d'un modèle du système (en s'appuyant généralement sur des comparaisons de grandeurs basées sur des redondances i.e., des grandeurs qui peuvent se calculer par différents moyens), l'étude de la diagnosticabilité présuppose en tout l'existence d'un modèle qui décrit les comportements normal et défectueux du systèmes. Par conséquent, l'étude de la diagnosticabilité présuppose de répertorier a priori un certain nombre fautes dont on veut vérifier la diagnosticabilité. En revanche, la nature du système modélisé pour conduire à des approches très différentes dans la définition des fautes et de l'observabilité du système et par conséquent dans la conception des algorithmes de vérification de la diagnosticabilité. Nous présentons brièvement dans ce qui suit quelques critères principaux qui permettent de catégoriser les systèmes étudiés selon leurs natures et qui détermine bien évidemment l'approche adéquate pour l'étude de leur diagnosticabilité.

2.1. Systèmes discrets, continus et hybrides

Selon la nature du système étudié, on utilise en pratique trois types de modèles avec des représentations différentes de la description de la dynamique du système, de ses fautes potentielles et de ses observations.

2.1.1. Systèmes continus

Pour les systèmes dont le comportement est continu dans le temps, on utilise des modèles à bases d'états (state based models). Dans ces modèles, le comportement du système est décrit par un ensemble de variables à domaines continues ainsi que les contraintes (par exemples des équations analytiques) reliant ces variables dans ses différents modes possibles de fonctionnement. L'observabilité dans un tel système est définie en déterminant, parmi ses variables, un sous ensemble de variables dits observables, i.e., dont la valeur peut être observée. Le système est donc caractérisé, à un instant donné, par l'ensemble des valeurs de ses variables observables à cet instant. L'observabilité du système est dans ce cas représentée par l'ensemble des n-uplets des valeurs possibles des variables observables de ce système. Les fautes dans un tel système sont vues comme des perturbations imprévues qui affectent le comportement correct du système et par conséquent les observations sur ce système.

La diagnosticabilité, comme d'autres notions voisines telles que la détectabilité et la discriminabilité des systèmes continus, est basée sur la notion de signature. Intuitivement, une signature d'une combinaison possible de fautes, consiste en l'ensemble d'observations pouvant être obtenu à partir du fonctionnement du système dans le cas où cette combinaison de fautes s'est produite. Deux fautes sont dits discriminables si leurs signatures sont complètement différentes et une faute est dite détectable, si elle est discriminable du mode normal, i.e. si sa signature est complètement différente de la signature du fonctionnement normal (ici on traite de la même manière le mode normal comme tout autre mode fautif résultant de la production d'une combinaison quelconque de fautes). Le système est dit diagnosticable, si toute paire de fautes est discriminable. En pratique, cette dernière condition de diagnosticabilité est assez forte, et on utilise souvent une notion mois forte qui s'appuie sur ce qu'on appel la discriminabilité faible : une faute f_1 est faiblement discriminable d'une faute f_2 si et seulement si la signature de f_1 n'est pas un sous-ensemble de la signature de f_2. Il est clair que la discriminabilité implique la discriminabilité faible et l'inverse n'est pas toujours vrai [WS-DIAMOND Group, 07].

2.1.2. Systèmes discrets

Les systèmes discrets (ou plus souvent systèmes à événement discrets) sont des systèmes qui peuvent êtres modélisé par des modèles à base d'événements (Event based approaches). Il s'agit de systèmes dont le comportement est vu comme des transitions possibles entre différents états suite à l'occurrence d'événements (considérés généralement ponctuels du point de vue temporel). Les automates finis et les réseaux de Petri et leurs variantes et extensions sont des exemples de modèles souvent utilisés pour représenter le comportement de tels systèmes (surtout les automates finis). L'observabilité dans ce type de systèmes est traité en partitionnant *a priori* dans le modèle l'ensemble des événements en deux sous ensemble : celui des événements observables (généralement ceux correspondants à des capteurs installé dans le système) et celui des événements non observables qui englobent justement les événements de fautes. Ainsi, ce qu'on observe d'un système à événements discrets sont des séquences d'événements observables.

La diagnosticabilité dans les systèmes à événements discrets est définie intuitivement comme suit : Un système est dit diagnosticable si et seulement si tout occurrence d'une faute est suivie par une séquence finie d'observables qui ne se produit pas en l'absence de la faute.

Cette définition se rapproche de la notion de signature définie pour les systèmes continue. On trouve d'ailleurs dans [Cordier et al., 06] une étude qui montre la notion de signature peut être formellement définie pour les systèmes à événements discrets de manière à ce que la diagnosticabilité peut être définie de la même manière dans les systèmes continus et discrets.

La plupart des travaux actuels, sur la diagnosticabilité des systèmes à événements discrets, utilisent les automates finis simples comme modèles de représentations. Il existe également d'autres travaux dans ce domaine qui utilisent d'autres modèles à événements discrets. Dans [Tripakis, 02], le diagnostic et la diagnosticabilité sont analysés dans le cadre des automates temporisés en suivant l'approche des « twin plants » proposée par [Jiang et al., 01]. La diagnosticabilité des réseaux de Petri en général est étudiée par [Wen et Jeng, 04]. Finalement, [Haar et al, 03] et [Haar, 05] étudient la diagnosticabilité des systèmes à événements discrets avec une sémantique d'ordre partielle en se fondant particulièrement sur la technique d'exécution de réseaux de Petri (Petri Net unfoldings) (voir à ce sujet [Esparza et al., 02].

2.1.3. *Systèmes hybride*

Dans les systèmes hybride, et comme le nom l'indique, les deux aspects continu et discret co-existent dans la modélisation. L'idée générale dans cette approche est de modéliser le système par un ensemble d'états et de transitions comme dans un système à événements discrets mais que les états du système, eux, sont modélisés par une approche continu. Ainsi un état du système peut être vue comme un système continu avec des variables continues reliées par des contraintes, mais la portée de ces contraintes est restreinte à l'état en question. La transition du système d'un état à un autre fait changer son mode de fonctionnement en le faisant subir d'autres lois continues propres au nouvel état. Il y a peu de travaux sur l'étude de la diagnosticabilité dans de tels systèmes (voir à ce sujet par exemple [Fourlas et al., 02] [Beyoudh et al., 06]).

2.2. Systèmes centralisés et systèmes distribués

En plus de la nature continu, discrète ou hybride, un autre facteur qu'il faut prendre en compte dans la conception d'algorithmes pour la vérification de la diagnosticabilité d'un système est son aspect centralisé ou distribué.

Dans l'approche centralisée, le système étudié est décrit par un seul modèle qui intègre l'intégralité de son comportement. Nous disposons ainsi dans un tel modèle d'une vision globale des paramètres pertinents du système. La plupart des travaux sur la diagnosticabilité s'inscrivent dans ce cadre. L'inconvénient de cette approche, est que souvent, il devient très coûteux voire impossible, pour des raisons de complexité, de représenter dans un modèle unique le fonctionnement de l'ensemble du système. En pratique, les systèmes réels sont vus comme un ensemble de sous-systèmes qui coopèrent à travers des modes de communications appropriés. Naturellement, ce dont on dispose donc réellement est un ensemble de sous-modèles pour ces différents sous-systèmes ainsi qu'une description de leurs façons de communiquer entre eux. Théoriquement, on peut toujours se ramener à un cas distribué en construisant un modèle globale du système à partir des différents sous-modèles (ce qui ce fait par exemple dans le cas des automates finis par l'opération de synchronisation), mais en pratique cette démarche peut devenir très coûteuse en cas de systèmes complexes de grande taille. L'enjeu est donc de concevoir des algorithmes qui permettent de faire émerger une décision globale sur la diagnosticabilité d'un système à partir de décisions locales sur la diagnosticabilité de ses différentes parties. Quelques travaux on été déjà réalisés dans ce contexte pour les systèmes à événements discrets modélisés par des automates (voir la section

3.3). Cependant ce domaine reste à notre connaissance vierge dans le cas de systèmes continus et hybrides.

Après ce bref aperçu sur les différentes facettes du problème de diagnosticabilité, dimensions Nous nous limitons dans ce qui suit aux travaux qui ont traité le problème de diagnosticabilité dans les systèmes à événements discrets

3. Diagnosticabilité dans les systèmes à événements discrets

3.1. Rappels sur les systèmes à événements discrets

3.1.1. Notations de base

Un SEF G est défini par le quadruplet : $G = (X, \Sigma, \delta, x_0)$ tel que :

X est un ensemble d'états, Σ est un ensemble d'événements, $\delta \subseteq X \times \Sigma \times X$ est un ensemble fini de transitions[1] et x_0 est l'état initial. Le langage généré par G est noté $L(G)$ ou tout simplement L. L est un sous-ensemble de Σ^* qui est la fermeture de Kleene de Σ. Il correspond à l'ensemble des traces (c-à-d les mots) qui peuvent être exécutées dans G à partir de l'état initial. $L(G)$ est donc préfixé i.e., $L(G) = pr(L(G))$, où $pr(L(G)) = \{u / \exists v \in \Sigma^*, uv \in L(G)\}$ est l'ensemble des préfixes des traces dans $L(G)$.

L'ensemble d'événements Σ est répartis en deux sous ensembles Σ_o et Σ_{uo} : $\Sigma = \Sigma_o \cup \Sigma_{uo}$. Σ_o contient les événements *observables* i.e., les événements dont l'occurrence peut être observée et qui peuvent être les commandes issues du contrôleur ou les valeurs obtenues des capteurs lors de l'exécution du système. Σ_{uo} contient les événements *non observables* i.e., les événements dont l'occurrence ne peut pas être observée et qui contiennent les événements de fautes et tout autre événement qui peut causer un changement dans le système mais dont les capteurs ne détectent pas la présence. L'ensemble des fautes du système est noté Σ_f et représente un sous-ensemble des non observables : $\Sigma_f \subseteq \Sigma_{uo}$. On partitionne l'ensemble de fautes Σ_f en m ensembles disjoints qui correspondent aux différents types de fautes dont le traitement est assuré par les mêmes actions consécutives. Nous avons ainsi : $\Sigma_f = \Sigma_{f1} \cup ... \cup \Sigma_{fm}$. $\Pi_f = \{1,...,m\}$ est l'ensemble des indices i correspondant aux types de fautes Σ_{fi}. On suppose aussi que le système modélisé vérifie deux hypothèses :

- (H_1) son langage L est vivant i.e., il existe une transition à partir de tout état x de X,
- (H_2) le système n'a pas de cycle formé exclusivement d'événements non observables.

Une trace vide est notée par ε. L/s désigne le post-langage de L après s c'est-à-dire : $L/s = \{t \in \Sigma^* / st \in L\}$. La fonction de projection $P : \Sigma^* \longrightarrow \Sigma_o^*$ associe à chaque trace s de Σ une trace ne contenant que les observables de s i.e., P supprime tout simplement tous les non observables d'une trace donnée. La projection inverse P^{-1} est donc définie par :

.

Soit s_f le dernier événement dans une trace s, on définit $\Psi(\Sigma_{fi}) = \{s\sigma_f \in L / \sigma_f \in \Sigma_{fi}\}$ qui désigne l'ensemble des traces de L qui se terminent avec une faute de la classe Σ_{fi}. Par abus de langage, on écrit $\Sigma_{fi} \in s$ pour exprimer le fait qu'il y a une faute σ_f de Σ_{fi} telle que $\sigma_f \in s$.

[1] Dans ce qui suit, nous étendons la définition de δ pour couvrir aussi les séquences d'événement i.e., nous considérons : $\delta \subseteq X \times \Sigma^* \times X$

Construction du générateur G'

G' résulte de *G* en ne décrivant que le comportement du système par rapport aux événements observables. *G'* est en général non déterministe et génère le langage *P(L)* qui résulte de l'application de la fonction de projection *P* sur toutes les trace de *L* i.e., *G'* exprime la partie observable su système modélisé. Avant de montrer comment construire *G'* nous commençons par introduire les définitions suivantes :

X_0 = *{x_0} ∪ {x∈X / il y a une transition étiquetée par un observable et entrante à x}.*

L(G, x) est l'ensemble des traces de *L* issues de l'état *x* de *G*.

L_0(G, x) est l'ensemble des traces de *L* issues de l'état *x* de *G* et se terminant par la première occurrence d'un observable : $L_0(G,x) = \{s \in L(G,x) / su\sigma, u \in \Sigma_{uo}^*, \sigma \in \Sigma_o\}$.

L_σ(G, x) l'ensemble des traces de *L_0(G, x)* qui se terminent par l'observable particulier σ :

.

Le générateur *G'* est défini par : *G' = (X_0, Σ_o, $\delta_{G'}$, x_0)* où X_0, Σ_o, x_0 ont été déjà définis et où $\delta_{G'}$ est la fonction de transition de *G'* telle que $\delta_{G'} \subseteq (X_0 \times \Sigma_o \times X_0)$ avec :

$(x, \sigma, x') \in \delta_{G'}$ ssi $(x, s, x') \in \delta$ pour un certain $s \in L_\sigma(G, x)$.

Exemple 1. (inspiré de [Jiang et al., 01])

La figure 1. montre le modèle d'un système à événements discrets *G* (la partie gauche) ainsi que son générateur G'.

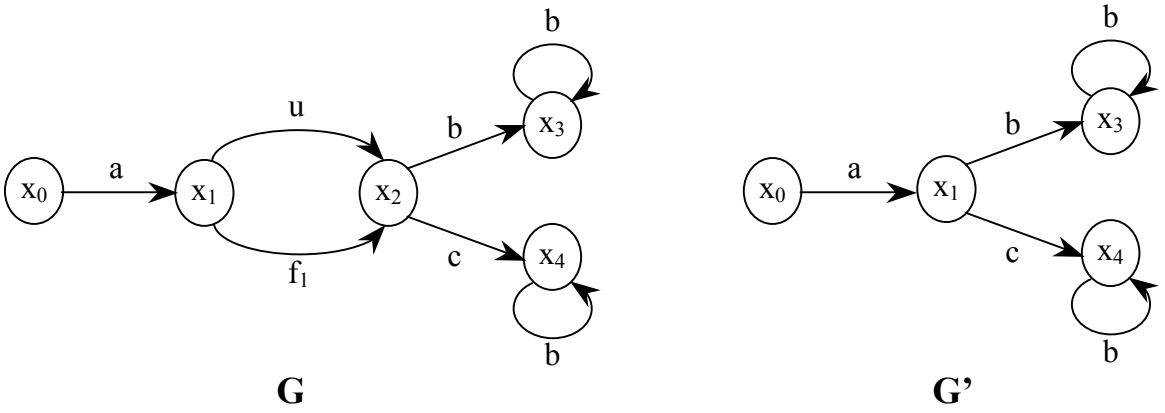


Figure 1. **Exemple d'un système à événement discret et de son générateur**

Le modèle initial *G* est défini par :

- l'ensemble des états est : $X = \{x_0, x_1, x_2, x_3, x_4\}$;
- l'ensemble des événements est : $\Sigma = \{a, b, c, u, f_1\}$, les sous ensembles des événements observables, non observables et de fautes sont respectivement :

$$\Sigma_0 = \{a, b, c\},\ \Sigma_{uo} = \{u, f_1\} \text{ et } \Sigma_f = \{f_1\};$$

- l'ensemble de transitions est :

$$\delta = \{(x_0, a, x_1), (x_1, f_1, x_2), (x_1, u, x_2), (x_2, b, x_3), (x_2, c, x_4), (x_3, b, x_3), (x_4, b, x_4)\};$$

- l'état initial est : x_0.

Le générateur correspondant G' est défini par : X_0, Σ_o, $\delta_{G'}$, x_0

- l'ensemble des états est : $X_0 = \{x_0, x_1, x_3, x_4\}$;
- l'ensemble des événements est : $\Sigma_0 = \{a, b, c\}$;
- l'ensemble de transitions est :

$$\delta_{G'} = \{(x_0, a, x_1), (x_1, b, x_3), (x_1, c, x_4), (x_3, b, x_3), (x_4, b, x_4)\};$$

- l'état initial est : x_0 .

3.1.2. Définition formelle de la diagnosticabilité

Un langage L préfixé et vivant est dit diagnosticable par rapport à une projection P et un ensemble de partitions de défauts Π_f si et seulement si [Sampath et al. 95]:

$$(\forall i \in \Pi_f)(\exists n_i \in N)[\forall s \in \Psi(\Sigma_{fi})](\forall t \in L/s)[\|t\| \geq n_i \Rightarrow D]$$

Tel que la condition de diagnosticabilité D est comme suit : $\omega \in P_L^{-1}[P(st)] \Rightarrow \Sigma_{fi} \in \omega$

3.2. Approches centralisées

3.2.1. Approche à base d'un diagnostiqueur [Sampath et al., 95]

3.2.1.1. Construction d'un diagnostiqueur

On définit l'ensemble des étiquettes de fautes $\Delta_f = \{F_1, ..., F_m\}$ où $|\Pi_f| = m$ (une étiquette F_i par type de fautes Σ_{fi}) et l'ensemble total des étiquettes possibles qui peuvent figurer dans les états du diagnostiqueur (voir ci-après) : $\Delta = (\{\{N\}\} \cup 2^{\{\Delta f \cup \{A\}\}})/\{\{\}\}$ *N* veut dire *« normal », A* veut dire *« ambigu » et* F_i veut dire qu'une faute de type F_i s'est produite. Les éléments l_i de Δ peuvent prendre les formes suivantes : $l_i = \{N\}$, $l_i = \{A\}$, $l_i = \{F_{i1}, ..., F_{ik}\}$ *ou* $l_i = \{A, F_{i1}, ..., F_{ik}\}$ $(\{i_1, ..., i_k\} \subseteq \{1, ..., m\})$.

Le diagnostiqueur G_d de G est défini par : $G_d = (Q_d, \Sigma_o, \delta_d, q_0)$ tel que :

a) $q_0 = \{(x_0, \{N\})\}$ i.e., on démarre d'un état normal du système.

b) L'espace d'état Q_d est un sous-ensemble de l'ensemble $Q_0 = 2^{X0 \times \Delta}$ (voir la définition de X_0 *plus* haut). Q_d contient les états de Q_0 atteignables par la fonction de transition δ_d (voir ci-après) à partir de l'état initial q_0. Un état q_d de Q_d est de la forme $\{(x_1,l_1), ..., (x_n, l_n)\}$ où $x_i \in X_0$, $l_i \in \Delta$ (voir plus haut les formes possibles de l_i).

c) Pour définir la fonction de transition δ_d du diagnostiqueur nous avons besoin de définir *les* fonctions suivantes :

i) La fonction de propagation d'étiquettes $LP : X_0 \times \Delta \times \Sigma^* \rightarrow \Delta$:

$$LP(x,l,s) = \begin{cases} \{N\} & si\ l = \{N\} \text{et } \forall i[\Sigma_{fi} \notin s] \\ \{A\} & \text{si } l = \{A\} \text{et } \forall i[\Sigma_{fi} \notin s] \\ \{F_i / F_i \in l \vee \Sigma_{fi} \in s\} & \text{sinon} \end{cases}$$

ii) La fonction de rang $R : Q_0 \times \Sigma_o \rightarrow Q_0$:

$$R(q,\sigma) = \bigcup_{(x,l)\in q} \bigcup_{s\in L_\sigma(G,x)} \{(\delta(x,s), LP(x,l,s))\}$$

iii) La fonction de correction d'étiquettes $LC : Q_0 \rightarrow Q_0$:

$$LC(q) = \{(x,l) \in q / x \text{ apparait une seule fois dans les couples de q}\}$$

$$\cup \{(x, \{A\} \cup l_{i1} \cap ... \cap l_{ik}) \text{ s'il } \exists \text{ deux ou plus de paires } (x,l_{i1}), ..., (x,l_{ik}) \text{ dans } q\}$$

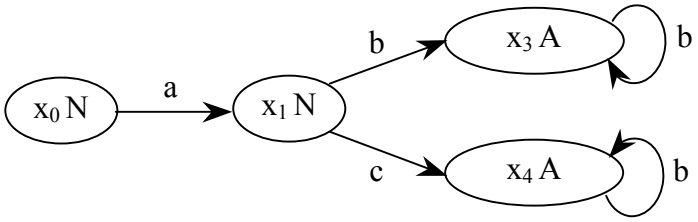
La fonction de transition $\delta_d : Q_0 \times \Sigma_o \rightarrow Q_0$ est définie comme suit :

$$q_2 = \delta_d(q_1,\sigma) \text{ ssi } q_2 = LC[R(q_1,\sigma)], \text{ avec :}$$

$$\sigma \in e_d(q_1) \quad \text{et} \quad e_d(q_1) = \bigcup_{(x,l)\in q_1} \{P(s) / s \in L_0(G,x)\}$$

Exemple 2.

La figure 2. montre le diagnostiqueur correspondant au système G présenté dans l'exemple 1.



G_d

Figure 2. **Diagnostiqueur du système présenté dans l'exemple 1**

3.2.1.2. Conditions de diagnosticabilité

Avant d'énoncer les condition nécessaires et suffisantes pour la diagnosticabilité d'un système à événements discrets G dont le diagnostiqueur est G_d et le générateur est G', donnons quelques définitions utiles :

- Un état q de Q_d est dit ***F_i-certain*** si : $\forall (x,l) \in q, F_i \in l.$
- Un état q de Q_d est dit ***F_i-incertain*** si : $\exists (x,l),(y,l') \in q, F_i \in l$ et $F_i \notin l'$.
- Un état q de Q_d est dit ***ambiguë*** si : $\exists (x,l) \in q, A \in l.$

Remarque : Différence entre un état Fi-incertain et un état ambigu.

- Un état q est ***F_i-certain*** si toute trace de q_0 à q contient obligatoirement la faute F_i.
- Un état q est ***F_i-incertain*** s'il y a deux traces s_1 et s_2 de q_0 à q ayant la même projection observable tel que s_1 contient F_i alors que s_2 ne contient pas F_i et que dans le système initial G, les traces s_1 et s_2 mènent vers deux états différents.
- Un état q est ***ambigu*** s'il y a deux traces s_1 et s_2 de q_0 à q ayant la même projection observable tel que s_1 contient F_i alors que s_2 ne contient pas F_i et que dans le système initial G, les traces s_1 et s_2 mènent vers un même état.

- Un ensemble d'états $x_1, ..., x_n \in X$ forme un cycle dans G si :

$$\exists s \in L(G,x_1) \text{ tel que } s = \sigma_1\sigma_2..\sigma_n \text{ et } \delta(x_i,\sigma_i) = x_{(i+1) \bmod n}, i = 1,2,...,n.$$

- Un ensemble d'états *F_i-incertains* $q_1, q_2, ..., q_n \in Q_d$ forme un cycle *F_i-indéterminé* si :

a) $q_1, q_2, ..., q_n$ forment un cycle dans G_d avec :

$$\delta_d(q_l,\sigma_l) = q_{(l+1) \bmod n} \text{ tel que } \sigma_l \in \Sigma_o, l = 1,2,...,n$$

b) Pour ce cycle dans G_d, il existe un cycle correspondant dans *G'* impliquant uniquement des états ayant F_i dans leurs étiquettes dans le cycle de G_d et un cycle similaire dans *G'* impliquant uniquement des états n'ayant pas F_i dans leurs étiquettes dans le cycle de G_d. Formellement :

$\exists (x_l^k, l_l^k), (y_l^r, \tilde{l}_l^r) \in q_l, l = 1,...,n, \quad k = 1,..,m, \quad r = 1,...,m'$ tel que

1. $F_i \in l_l^k, F_i \notin \tilde{l}_l^r$ pour tout l, k et r.
2. Les séquences d'états $\{x_l^k\} l = 1,...,n, k = 1,...,m$ et $\{y_1^r\} l = 1,...,n, r = 1,...,m'$ forment des cycles dans G' (le générateur) avec :

$$(x_l^k, \sigma_l, x_{l+1}^k) \in \delta_{G'}, \qquad l = 1,...,n, \qquad k = 1,...,m$$
$$(x_n^k, \sigma_n, x_1^{k+1}) \in \delta_{G'}, \qquad k = 1,...,m-1$$
$$(x_n^m, \sigma_l, x_1^1) \in \delta_{G'}$$

et

$$(y_l^r, \sigma_l, y_{l+1}^r) \in \delta_{G'}, \qquad l = 1,...,n, \qquad r = 1,...,m'$$
$$(y_n^r, \sigma_n, y_1^{r+1}) \in \delta_{G'}, \qquad r = 1,...,m'-1$$
$$(y_n^{m'}, \sigma_l, y_1^1) \in \delta_{G'}$$

Diagnosticabilité

Un langage L sans fautes multiples de même type est diagnosticable si et seulement si son diagnostiqueur G_d satisfait les conditions suivantes :

C1) Il n'y a pas de cycle *F_i-indéterminé* dans G_d pour tout type d'erreur F_i.

C2) Il n'y a pas d'état q de Q_d qui soit ambigu.

Exemple 3.

Selon le diagnostiqueur (figure 2.) on voit bien que la condition C2) n'est pas vérifiée. Il existe deux états ambigus : $\{x_3, \{A\}\}$ et $\{x_4, \{A\}\}$. Le système n'est donc pas diagnosticable.

3.2.1.3. Performance et complexité

On trouve dans [Sampath et al., 95] les preuves:

- Qu'il y a une limite supérieure du délai de diagnosticabilité pour une faute donnée F_i (i.e., l'entier n_i de la définition de la diagnosticabilité section 3.1.2). En fait, on a :

$$n_i \leq C_i \times n_0 + n_0 \quad \text{où :}$$

n_0 est la longueur maximale des séquences formées exclusivement d'événements non observables (n_0 est fini selon l'hypothèse *(H_2)*, section 3.1.1)

C_i est le nombre total d'états x de X_0 qui figurent dans tous les états F_i-*incertains* du diagnostiqueur :

$$C_i = \sum_{q \in Q_d : q \text{ est } F_i - incertain} nombre\ d'états\ x\ dans\ q$$

- Que le diagnostiqueur G_d détecte l'occurrence d'une faute F_i dans un délai qui correspond au plus à la production de n_0+n_i événements après l'occurrence de la faute F_i.

L'algorithme proposé est ***exponentiel*** par rapport au nombre d'états du système G et ***doublement exponentiel*** par rapport au nombre de types de fautes.

3.2.2. Une première approche à base d'un vérificateur [Jiang et al. 01]

3.2.2.1. Construction d'un diagnostiqueur et d'un vérificateur

On construit le SEF non déterministe (diagnostiqueur) $G_0 = (X_0, \Sigma_o, \delta_0, x_0^o)$ dont le langage est $L(G_0) = P(L(G))$ avec :

L'espace d'états de G_0 est $X_0 = \{(x, f) / x \in X_1 \cup \{x_0\}, f \subseteq \Sigma_f\}$ où X_1 est l'ensemble des états de G atteignables par une chemin observable (au moins partiellement) : $X_1 = \{x \in X_1 / \exists (x_0, \sigma, x) \in \delta \text{ avec } P(\sigma) \neq \varepsilon\}$; Σ_o est l'ensemble des événements observables de G; $x_0^o = (x_0, \phi)$est l'état initial et $\delta_0 \subseteq X_0 \times \Sigma_o \times X_0$ est la fonction de transition telle que : $((x, f), \sigma, (x', f')) \in \delta_0$ si et seulement si :

- il existe un chemin $(x, \sigma_1, x_1, \ldots, \sigma_n, x_n, \sigma, x')$ $(n \geq 0)$ avec $P(\sigma_i) = \varepsilon$ $(i \in \{1, \ldots, n\})$, $P(\sigma) = \sigma$;
- f' = $\{\sigma_i / \sigma_i \in \Sigma_f\} \cup$ f .

On construit le SEF non déterministe (vérificateur) $G_d = G_0 \,||\, G_0$ (la composition stricte de G_0 avec lui-même) ; $G_d = (X_d, \Sigma_o, \delta_d, x_0^d)$ avec :

L'ensemble d'états de G_d est $X_d = \{(x_1^o, x_2^o) / x_1^o, x_2^o \in X_0\}$, l'état initial est $x_0^d = (x_0^o, x_0^o)$et $\delta_d \subseteq X_d \times \Sigma_o \times X_d$ est la fonction de transition telle que $((x_1^o, x_2^o), \sigma, (y_1^o, y_2^o)) \in \delta_d$ si et seulement (x_1^o, σ, y_1^o) et (x_2^o, σ, y_2^o) sont dans δ_0.

Exemple 4 ([Jiang et al., 01])

La figure 3. représente un exemple d'un système $G = (X, \Sigma, \delta, x_0)$ avec :

$\Sigma = \{a, b, c, u, f_1, f_2\}$, $\Sigma_o = \{a, b, c\}$, $\Sigma_{uo} = \{u, f_1, f_2\}$ et $\Sigma_f = \{f_1, f_2\}$. Les ensembles des états et des transitions sont clairement déduits de la figure.

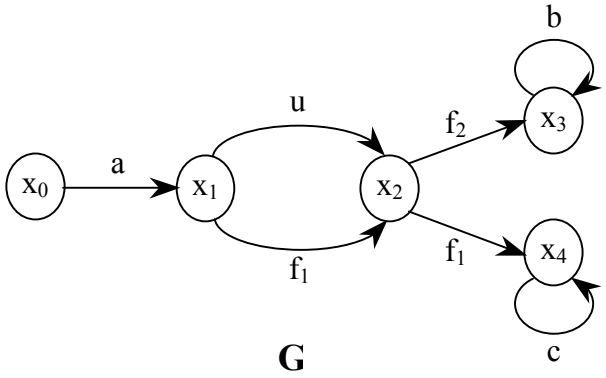


Figure 3. **Un autre exemple d'un système à événement discret**

Le diagnostiqueur G_0 et le vérificateur G_d correspondant sont représentés respectivement dans les figures 4 et 5 suivantes :

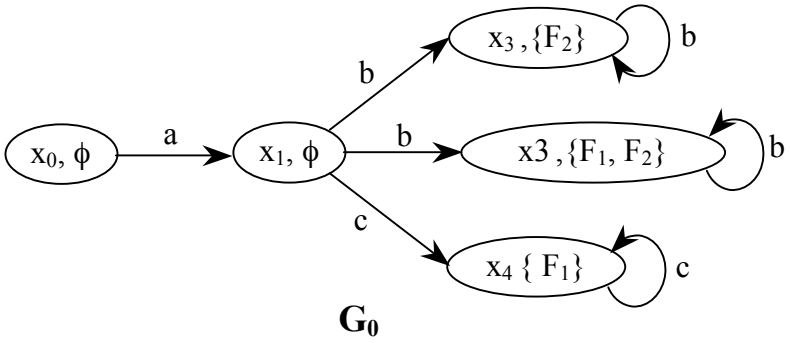


Figure 4. **Diagnostiqueur G_0 du système**

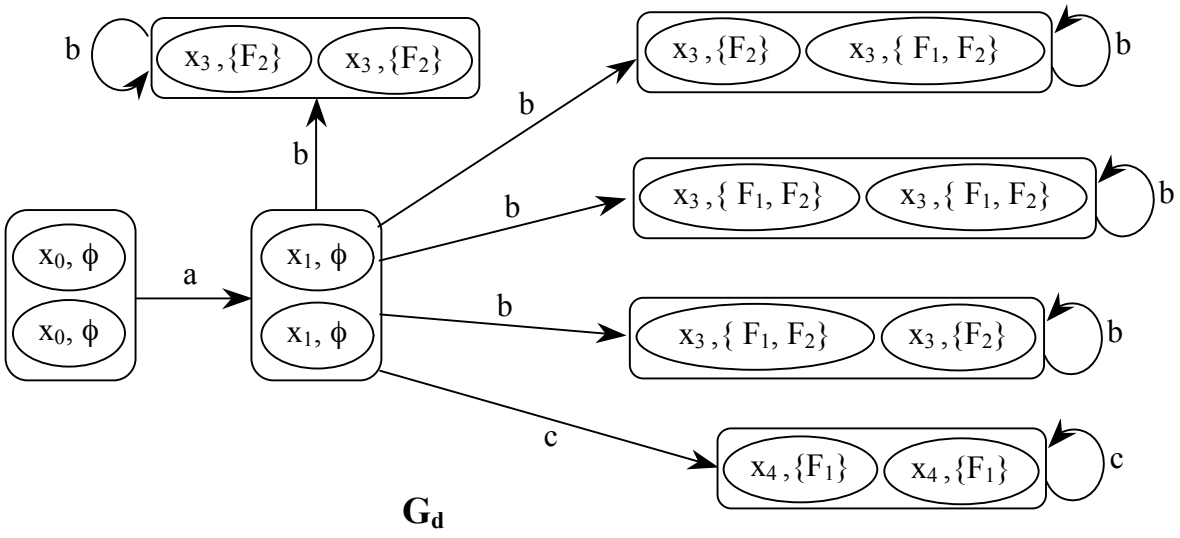


Figure 5. **Vérificateur G_d du système**

3.2.2.2. Condition de diagnosticabilité

L(G) est diagnosticable si et seulement si pour tout cycle *cl* dans G_d,

$cl = (x_1, \sigma_1, x_2, ..., x_n, \sigma_n, x_1), n \geq 1, \quad x_i = ((x_i^1, f_i^1), (x_i^2, f_i^2)), \ i = 1,...,n$, nous avons :

$$f_i^1 = f_i^2$$

puisque les f_i sont des ensembles croissants par construction et qu'on est dans un cycle, la condition $f_i^1 = f_i^2$ pour tout $i = 1,...,n$ se ramène à la condition : $f_1^1 = f_1^2$. Ainsi, si après la construction de G_d, on trouve un cycle

$cl = (x_1, \sigma_1, x_2, ..., x_n, \sigma_n, x_1), n \geq 1, \quad x_i = ((x_i^1, f_i^1), (x_i^2, f_i^2)), \ i = 1,...,n$ où $f_1^1 \neq f_1^2$

on déduit que le système n'est pas diagnosticable. Cette étape peut être améliorée en déterminant d'abord les états $((x^1, f^1), (x^2, f^2))$ de G_d tel que : $f^1 \neq f^2$ et supprimer tous les autres états ainsi que les transitions qui lui sont associées et ensuite vérifier si le graphe qui en résulte contient des cycles.

Exemple 5

Selon le vérificateur (figure 5) du système introduit ci-dessus, on constate l'existence de deux cycles : $cl_1 = (((x_3, \{F_2\}), (x_3, \{F_1, F_2\})), b, ((x_3, \{F_2\}), (x_3, \{F_1, F_2\})))$, avec $\{F_2\} \neq \{F_1, F_2\}$ et $cl_1 = (((x_3, \{F_1, F_2\}), (x_3, \{F_2\})), b, ((x_3, \{F_1, F_2\}), (x_3, \{F_2\})))$, avec $\{F_1, F_2\} \neq \{F_2\}$. Les système n'est donc pas diagnosticable.

3.2.2.3. Résultats de complexité

- Le nombre maximum d'états dans G_0 est : $|X| \times 2^{|\Sigma_f|}$.
- Le nombre maximum de transitions dans G_0 est : $|X|^2 \times 2^{2|\Sigma_f|} \times |\Sigma_o|$.
- Le nombre maximum d'états dans G_d est : $|X|^2 \times 2^{2|\Sigma_f|}$.
- Le nombre maximum de transitions dans G_d est : $|X|^4 \times 2^{4|\Sigma_f|} \times |\Sigma_o|$.
- La complexité de la construction de G_0 est : $O\left(|X|^2 \times 2^{2|\Sigma_f|} \times |\Sigma_o|\right)$
- La complexité de la construction de G_d est : $O\left(|X|^4 \times 2^{4|\Sigma_f|} \times |\Sigma_o|\right)$
- La complexité de vérifier s'il y a un cycle dans le résultat de la réduction de G_d (voir la section précédente) qui vérifie la condition de non diagnosticabilité est : $O\left(|X|^4 \times 2^{4|\Sigma_f|}\right)$
- La complexité globale de l'algorithme est donc: $O\left(|X|^4 \times 2^{4|\Sigma_f|} \times |\Sigma_o|\right)$. L'algorithme est donc ***polynomial*** par rapport au nombre d'états de *G* et ***exponentiel*** par rapport au nombre de types de fautes.
- On peut remarquer que le système est diagnosticable si et seulement s'il est diagnosticable pour chaque type de faute $i \in \Pi_f = \{1,...,m\}$ (le nombre de types de fautes est donc supposé égal à *m*). En se basant sur cette remarque, on peut rendre la complexité de l'algorithme de vérification du système polynomial par rapport au

nombre de types de fautes m en vérifiant séparément chaque type. La complexité de l'algorithme de vérification de chaque type de faute sera donc $O\left(|X|^4 \times 2^{4|\mathrm{l}|} \times |\Sigma_o|\right) = O\left(|X|^4 \times |\Sigma_o|\right)$ et la complexité de l'algorithme de vérification de l'ensemble des fautes est : $O\left(|X|^4 \times |\Sigma_o| \times m\right)$, qui est linéaire par rapport au nombre des fautes.

3.2.3. Une deuxième approche à base d'un vérificateur [Yoo et Lafortune, 02]

3.2.3.1. Construction des F_i-vérificateurs

Le *Fi-vérificateur* est défini par : $V_{F_i} = (Q^{V_{F_i}}, \Sigma, \delta^{V_{F_i}}, q_0^{V_{F_i}})$ avec :

L'espace d'état de V_{F_i} est $Q^{V_{F_i}} = X \times L_i \times X \times L_i$ où L_i est un ensemble d'étiquette relatif à F_i, $L_i = \{N, F_i\}$. L'état initial est $q_0^{V_{F_i}} = (x_0, N, x_0, N)$. La fonction de transition $\delta^{V_{F_i}}$ de V_{F_i} est non déterministe et est définie comme suit :

- Pour $\sigma \in \Sigma_{f_i}$ $\quad \delta^{V_{F_i}}((x_1,l_1,x_2,l_2),\sigma) = \begin{cases} (\delta(x_1,\sigma),F_i,x_2,l_2) \\ (x_1,l_1,\delta(x_2,\sigma),F_i) \\ (\delta(x_1,\sigma),F_i,\delta(x_2,\sigma),F_i) \end{cases}$
- Pour $\sigma \in \Sigma_{uo}/\Sigma_{f_i}$ $\quad \delta^{V_{F_i}}((x_1,l_1,x_2,l_2),\sigma) = \begin{cases} (\delta(x_1,\sigma),l_1,x_2,l_2) \\ (x_1,l_1,\delta(x_2,\sigma),l_2) \\ (\delta(x_1,\sigma),l_1,\delta(x_2,\sigma),l_2) \end{cases}$
- Pour $\sigma \in \Sigma_o$ $\quad \delta^{V_{F_i}}((x_1,l_1,x_2,l_2),\sigma) = (\delta(x_1,\sigma),l_1,\delta(x_2,\sigma),l_2)$

3.2.3.2. Condition de diagnosticabilité

Avant énoncer la condition de diagnosticabilité de G dans l'ensemble des *F_i-vérificateur,* nous introduisons la définition da la *Fi-confusion* d'un *F_i-vérificateur* :

Un *Fi-vérificateur* V_{F_i} est dit ***F_i-confondu*** s'il existe un cycle *(q_1, q_2, ..., q_n)* dans V_{F_i} tel que pour tout $q_i = (x_1^{q_i}, l_1^{q_i}, x_2^{q_i}, l_2^{q_i})$, $l_1^{q_i} = F_i$ et $l_2^{q_i} = N$ ou inversement. V_{F_i} est dit : ***F_i-libre-confusion*** s'il ne contient pas de tels cycles.

Diagnosticabilité

L(G) est diagnosticable par rapport à Σ_o et Π_f si et seulement si V_{F_i} est *F_i-libre-confusion* pour tout *i* dans Π_f.

Exemple 6

Dans cet exemple, nous nous proposons de montrer la non diagnosticabilité de du système présenté dans l'exemple 4. Pour cela, nous construisons d'abord le *F_1-vérificateur* de ce système. La figure 6 présente une partie de ce vérificateur.

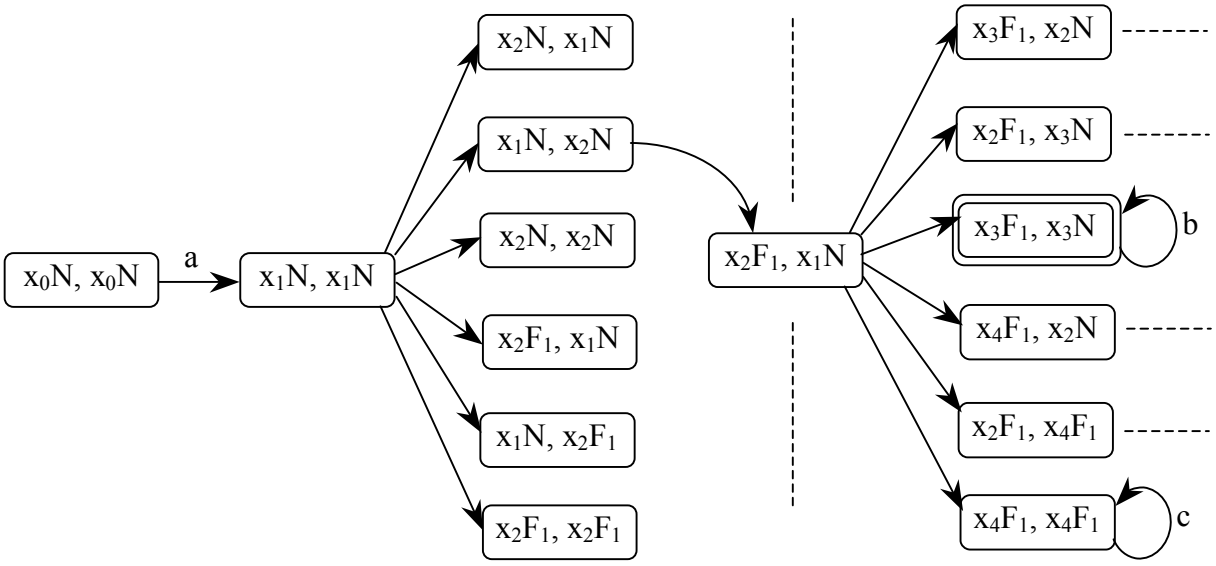


Figure 6. **Le F_1-Vérificateur de G**

Le *F_1-vérificateur* de la figure 6 est *F_1-confondu* car il contient (au moins) le cycle formé du seul état (ambigu): (x3, F1, x3, N). Le système n'est donc pas diagnosticable.

3.2.3.3. Résultats de complexité

- Etant donné $i \in \Pi_f$, la complexité de la construction de V_{F_i} est : .
- Etant donné V_{F_i}, la complexité de vérifier l'existence d'un cycle *F_i-confondu* dans V_{F_i} est : .
- La complexité de l'algorithme de vérification de diagnosticabilité de *L(G)* par rapport à Σ_o et Π_f est : $O\left(|X|^2 \times |\Sigma| \times |\Pi_f|\right)$. L'algorithme est donc ***polynomial*** par rapport à $|X|$, $|\Sigma|$ et $|\Pi_f|$.

3.3. Approches décentralisées / distribuées

Dans cette section, nous nous intéressons à la diagnosticabilité dans le cas de systèmes distribués. L'enjeu consiste à vérifier la diagnosticabilité globale d'un système en se fondant des modèles locaux des sous-systèmes qui le constituent et de la manière de communication entre ses sous-systèmes sans être obligé de calculer le modèle global du système. Quelques travaux ont commencé a être développés dans ce domaine, mais nous pensons que l'objectif de distribuer complètement la vérification de la diagnosticabilité n'est pas encore atteint. Comme nous allons le voir dans les deux algorithmes que nous allons présenter, le risque d'être obligé de calculer le modèle global (ce qui peut être parfois impossible à cause de la limitation des ressources système) n'est pas complètement écarté. Parmi les travaux dans ce domaine on peut cité [Sengupta, 99] qui se limite à donner une caractérisation formelle de la diagnosticabilité dans le cas distribué sans proposer d'algorithmes pour sa vérification. Le travail décrit dans [Contant et al., 06] peut être vu comme généralisation de l'algorithme initial de vérification de la diagnosticabilité proposé dans [Sampath et al., 95] au cas d'une architecture décentralisée modulaire. Enfin, les travaux décrits dans [Pencolé, 04] et [Schumann et Pencolé, 07] s'intéressent également au traitement de la diagnosticabilité de

SED dans le cas distribué en adaptant l'algorithme centralisé proposé dans [Jiang et al. 01] au contexte distribué. Leur adaptation est en gros basée sur la construction successive de modèles pour des sous-systèmes impliquant de plus en plus de modèles locaux des composants élémentaires jusqu'à ce que la diagnosticabilité est vérifié, le modèle global est calculé ou que les ressources systèmes sont épuisés. Nous présentons dans ce qui suit deux variantes assez proche de cet algorithmes présentées dans [Pencolé, 04] et [Schumann et Pencolé, 07].

3.3.1. Une approche distribuée à base de vérificateurs locaux

3.3.1.1. Modèle local et modèle global

Modèle local d'un composant

Le modèle local d'un composant C_i est le système d'états finis : $G_i = (X^i, \Sigma^i, \delta^i, x_0^i) où\ X^i$ est l'ensemble des états du composant *i*, x_0^i est son état initial, Σ^i est l'ensemble de ses événements et $\delta^i \subseteq X^i \times \Sigma^i \times X^i$ est sa fonction de transition. L'ensemble Σ^i des événements du composant *i* est partitionné en trois sous-ensembles : $\Sigma^i = \Sigma_o^i \cup \Sigma_{uo}^i \cup \Sigma_s^i$. Σ_o^i (resp. Σ_{uo}^i) regroupe les événements observables (resp. non observables) du composant *i*. L'ensemble $\Sigma_f^i \subseteq \Sigma_{uo}^i$ est l'ensemble des fautes (non observables) pouvant se produire dans le composant *i*. L'ensemble Σ_s^i est l'ensemble des événements de communication que le composant *i* partage avec les autres composants du système (ils sont donc toujours supposés non observables ?, jusqu'à présent, je n'ai pas vu d'indication explicite sur le statut d'observabilité des événements de communication, mais il me semble plausible qu'il soient non observables et qu'on n'observe que ce qui se passe dans chaque composent).

Exemple 7. [Schumann et Pencolé, 07]

La figure 7. montre le modèle d'un système composé de deux modèles locaux C_1, C_2.

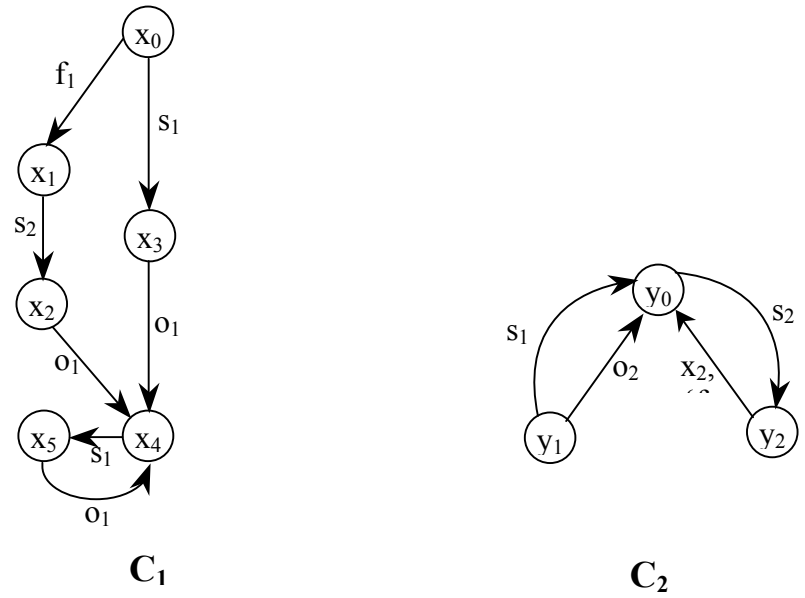


Figure 7. **Un modèle d'un système avec trois sous modèles**

Modèle global d'un sous-système

On appelle sous-système un ensemble non vide des composants du système. Le comportement global d'un sous-système Γ de k composants est décrit par son modèle global G qui résulte de l'opération de *synchronisation* (notée *Synch*) des différents modèles locaux $G_{i_{(i=1..k)}}$ par rapport à leurs événements de communication : $Synch(G_1,...,G_k,\bigcup_{j=1}^{k}\Sigma_s^j) = (X,\Sigma,\delta,x_0)$ avec[2] :

- $X \subseteq X^1 \times ... \times X^k$ est l'ensemble des états de G.
- $\Sigma = \bigcup_{j=1}^{k}\Sigma^j$ est l'ensemble des événements de G.
- $x_0 = (x_0^1,...,x_0^k)$ est l'état initial de G.
- $\delta \subseteq \delta^1 \times ... \times \delta^k$ est la fonction de transition de G définie comme suit :
  - si $s \in \bigcup_{j=1}^{k}\Sigma_s^j$, alors pour tout couple d'états $x = (x^1,...,x^k)$ et $x' = (x'^1,...,x'^k)$, $(x,s,x') \in \delta$ ssi $(\forall j \in \{1,...,k\})((x^j,s,x'^j) \in \delta^k)$ [3]
  - si $s \notin \bigcup_{j=1}^{k}\Sigma_s^j$, alors pour tout couple d'états $x = (x^1,...,x^k)$ et $x' = (x'^1,...,x'^k)$, $(x,s,x') \in \delta$ ssi : $(\exists j \in \{1,...,k\})((x^j,s,x'^j) \in \delta^k) et (\forall l \in \{1,...,k\} \setminus \{j\})((x^l = x'^l) ou (x^l,s,x'^l) \in \delta)$ [4]

Diagnosticabilité locale d'une faute

Dans le cas d'un système modélisé par un ensemble de *SEFs*, une faute F est dite localement diagnosticable dans un sous-système si et seulement s'il existe une séquence finie d'observables se produisant dans le sous-système après l'occurrence de F tel qu'on est sûr que la faute F a eu lieu. Il s'agit en fait d'appliquer la définition générale de la diagnosticabilité donnée dans §3.1.2. en se limitant seulement à un sous-système. Si le sous-système en question est le résultat de la synchronisation de l'ensemble des composants, et représente ainsi le système global, on retombe sur le cas centralisé.

On peut montrer que si une faute est localement diagnosticable dans un sous-système, alors elle est globalement diagnosticable. Si une faute n'est pas diagnosticable dans un sous-système, la synchronisation de son modèle avec des modèles d'autres sous-systèmes peut éventuellement conduire à un modèle ou la faute devienne diagnosticable. La raison est tout simplement que cette combinaison élargit le nombre d'observables impliqués et par conséquent peut ajouter de l'information permettant de détecter une faute en rendant visibles des séquences pertinentes d'événements observables qui ne faisaient pas partie du sous-système initial.

---

[2] L'opération de synchronisation peut être appliquée en général par rapport à un sous-ensemble quelconque d'événements.

[3] Puisque s fait partie des événements de synchronisation, on s'assure que tous les sous-états évoluent simultanément par rapport à s.

[4] Pour s ne faisant pas partie des événements de synchronisation, on envisage (au maximum) autant de transitions qu'il y a de sous-ensembles de sous-états qui peuvent évoluer par rapport à s.

3.3.1.2. Diagnostiqueurs et vérificateurs locaux

Dans cette approche, on sera ramené à construire des diagnostiqueurs et des vérificateurs similaires à ceux présentés dans l'algorithme de Jiang et al (2001) décrit dans la section §3.2.2.1. Cependant, dans cette approche, on se focalise à chaque fois sur une faute donnée *F* qui se produit dans un composant donné C_i. On parle ainsi de *F-diagnostiqueur* (resp. *diagnostiqueur*) local et de *F-vérificateur* (resp. *vérificateur*) local pour le composant dans lequel la faute se produit (resp. pour les autres composants). Un autre paramètre qui doit être géré dans le cas distribué est la présence des événements de communication.

Construction d'un F-diagnostiqueur local

Un *F-diagnostiqueur* local d'un composant C_i est défini par le SEF $\tilde{G}_i = (\tilde{X}^i, \tilde{\Sigma}^i, \tilde{\delta}^i, \tilde{x}_0^i)$ tel que : $\tilde{X}^i \subseteq X^i \times \{\{F\}, \phi\}$ est l'ensemble des états de $\tilde{G}_i$, $\tilde{\Sigma}^i = \Sigma_i^o \cup \Sigma_s^i$ est l'ensemble des événements de $\tilde{G}_i$, $\tilde{x}_0^i = (x_0^1, \phi)$ est l'état initial de $\tilde{G}_i$ et $\tilde{\delta}_i \subseteq \tilde{X}^i \times \tilde{\Sigma}^i \times \tilde{X}^i$ est la fonction de transition de $\tilde{G}_i$ définie comme suit : $((x,f),\sigma,(x',f')) \in \tilde{\delta}_i$ si et seulement si :

- il existe un chemin : $(x,\sigma_1,x_1,...,\sigma_n,x_n,\sigma,x')$ *avec :* $\sigma_j \in \Sigma_{uo}^i$ $(1 \le j \le n)$ *et* $\sigma \in \Sigma_o^i \bigcup \Sigma_s^i$
- $f' = \varnothing$ *si* $f = \varnothing$ *et* $F \notin \{\sigma_j \mid 1 \le j \le n\}$, $= \{F\}$ sinon
- enfin, on ne garde que les états *(x,f)* atteignables à partir de $\tilde{x}_0^i$.

Construction d'un F-vérificateur local

Pour construire le *F-vérificateur* local $\hat{G}_i$ du composant C_i nous prenons deux exemplaires $g:\tilde{G}_i$ et $\mathrm{d}:\tilde{G}_i$ du *F-diagnostiqueur* (exemplaire de gauche et exemplaire de droite). Les messages de communication sont alors renommés dans les deux exemplaires: tout message de communication *s* est renommé dans $g:\tilde{G}_i$ (resp. $\mathrm{d}:\tilde{G}_i$) par $g:s$ (resp. $d:s$). Le *F-vérificateur* est donc le résultat de synchronisation des deux exemplaires $g:\tilde{G}_i$ et $\mathrm{d}:\tilde{G}_i$ par rapport aux événements observables Σ_o^i : $\hat{G}_i = Synch(g:\tilde{G}_i, d:\tilde{G}_i, \Sigma_o^i)$ (voir la section précédente pour le détail technique de l'opération de synchronisation). Le vérificateur ainsi obtenu est exprimé par : $\hat{G}_i = (\hat{X}_i, \hat{\Sigma}, \hat{\delta}, \hat{x}_0)$.

La construction des diagnostiqueurs et des vérificateurs locaux des composants $C_k (k \neq i)$ i.e., les composants autres que celui dans lequel la faute s'est produite se fait exactement de la même manière que pour C_i. La différence est que ces diagnostiqueurs et vérificateurs ne contiennent aucune information sur la faute *F* (toutes les étiquettes de fautes sont vides).

État non diagnosticable (ambigü)

Un état $\hat{x}$ de $\hat{G}_i$ est dit non diagnosticable (ou ambigu) si et seulement s'il est de la forme *((x, A), (x,B))* avec : $A \neq B$ où *A* et *B* sont deux ensembles d'étiquettes de fautes[5].

La faute *F* est diagnosticable localement par rapport au sous-système composé du seul composant C_i s i et seulement s'il n'existe dans $\hat{G}_i$ aucun cycle observable et non diagnosticable (*CON*). Un cycle est dit observable et non diagnosticable si et seulement s'il comporte au moins : un état non diagnosticable et un événement observable.

[5] Puisqu'on se focalise sur une seule faute F, l'état est ambigu ssi $A = \Phi$ et $B = \{F\}$ ou $A = \{F\}$ et $B = \Phi$,

Il s'agit bien ici de la même condition discutée dans §3.2.2. appliquée au modèle local d'un composant. Cependant, la non diagnosticabilité locale d'une faute par rapport au modèle d'un composant (ou d'un sous-système de composants) n'implique pas forcément que la faute n'est pas diagnosticable globalement. La synchronisation du vérificateur d'un sous-système avec les vérificateurs d'autres sous-systèmes par rapport à leurs événements de communication peut conduire à l'élimination de cycles observables non diagnosticables. Nous présentons dans ce qui suit deux algorithmes pour résoudre le problème de diagnosticabilité dans le cadre distribué. Les deux algorithmes se basent sur l'opération de synchronisation incrémentale des vérificateurs locaux des différents composants dans l'espoir d'arriver à des sous-systèmes où la faute soit diagnosticable sans calculer forcément un vérificateur global du système[6]. Afin de gagner en termes d'efficacité, les deux algorithmes ont recours au fur et à mesure de leurs déroulements à des réductions des vérificateurs en ne gardant que les parties jugées pertinentes pour la diagnosticabilité de la faute. En plus de l'objectif de résoudre le problème de diagnosticabilité à partir d'un ensemble de modèles locaux sans calculer forcément un modèle global, les deux algorithmes essaient de retourner, en cas où le système n'est pas diagnosticable, une information qui permet de mieux comprendre les causes de la non diagnosticabilité. Cette information consiste dans le cas du premier algorithme en un langage d'observables qui correspond au déroulement du système pour lequel une faute ne peut être diagnostiquée. Dans le cas du deuxième algorithme, cette information consiste en l'ensemble des cycles qui causent la non diagnosticabilité d'une faute donnée.

Exemple 8.

La figure 8. (resp. figure 9) ci-dessous, montre le diagnostiqueur local (resp. une partie du f_1-vérificteur local) du modèle local C_1.

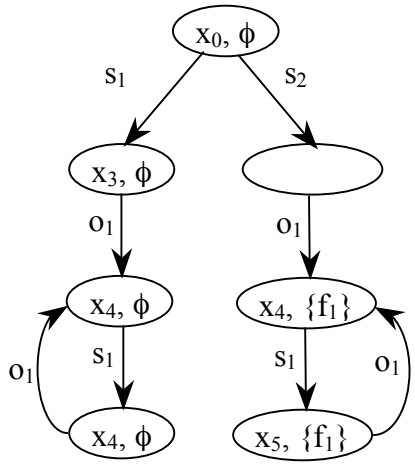


Figure 7. **Le f_1-diagnostiqueur** $\widetilde{G}_1$

[6] Les algorithmes ne calculent pas forcément un vérificateur global, mais ceci n'est pas pour autant entièrement exclu notamment si la faute n'est pas diagnosticable globalement (si les ressources système le permettent bien entendu).

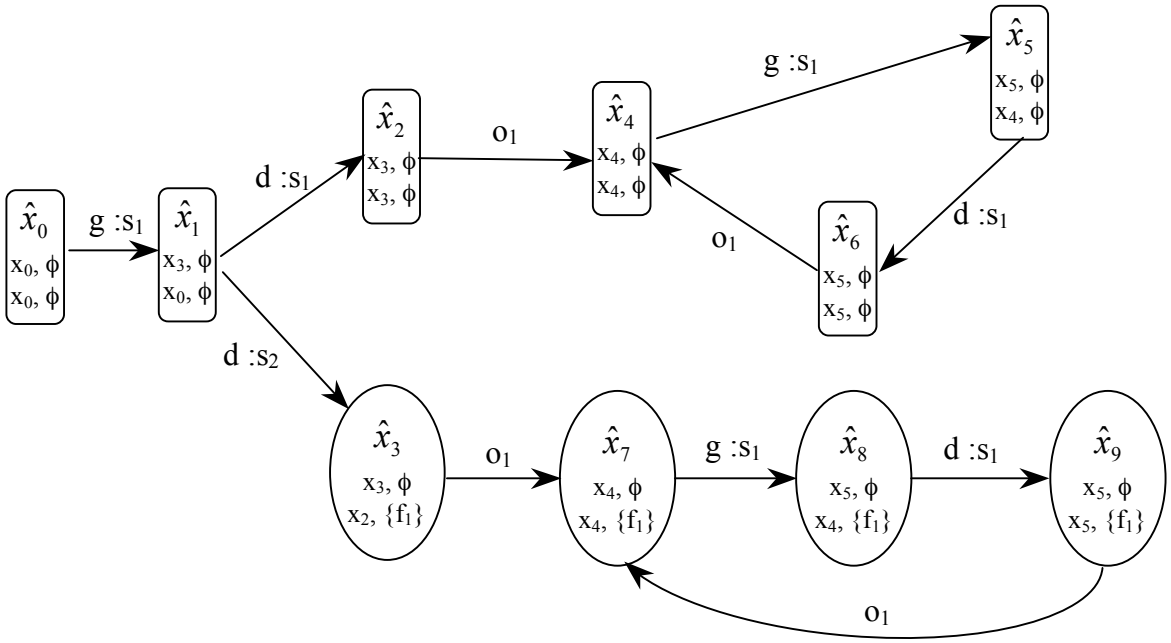


Figure 9. **Une partie du f_1-vérificateur** $\hat{G}_1$

3.3.1.3. Un premier algorithme de vérification de la diagnosticabilité [Y. Pencolé, 04]

Etant donné un ensemble de composants $C_1, ..., C_n$ et une faute F qui se produit dans C_i. L'algorithme se déroule comme suit :

- Calculer $\hat{G}_i$, le *F-vérificateur* du composant C_i
- $R \leftarrow \{C_1,...,C_n\} / \{C_i\}$, $S \leftarrow \{C_i\}$
- $V \leftarrow Red_1(\hat{G}_i)$ où $Red_1(V)$ est le résultat de la réduction d'un vérificateur V en ne gardant que les chemins issus de l'état initial et menant vers un cycle contenant au moins un état non diagnosticable et un événement observable de chaque composant de S.
- ***Tant que*** $V \neq \Phi$ et il n'y a pas de chemin de V ne contenant que des observables et se terminant par un cycle ayant un état non diagnosticable :
  - sélectionner un composant C_j de R partageant des événements de communication avec C_i, $R \leftarrow R - \{C_j\}$, $S \leftarrow S \cup \{C_j\}$
  - calculer $\hat{G}_j$, le vérificateur du composant C_j
  - $V \leftarrow Synch(V, \hat{G}_j, \Sigma_s^j \cup \Sigma_s^v)$ où Σ_s^v est l'ensemble d'événements de communication de V.
  - $V = Red_2(V)$ où Red_2 est l'opération qui consiste à enlever de V tous les événements de communication partagés uniquement entre composants de S (ces événements ne seront plus utiles par la suite)
  - $V = Red_1(V)$

- En sortant de la boucle, on examine la situation :
  - si $V = \Phi$, F est diagnosticable dans le sous-sytème formé des composants de S.
  - sinon, F n'est pas diagnosticable (si $R = \Phi$ alors, l'algorithme a calculé le modèle global et F est toujours non diagnosticable sinon, l'algorithme a trouvé un chemin ne contenant que des observables et se terminant par un cycle ayant un état non diagnosticable, ce qui veut dire qu'aucune autre synchronisation possible ne permettra de supprimer ce cycle, et la on peut décider que la faute n'est pas diagnosticable)

3.3.1.4. Un deuxième algorithme de vérification de la diagnosticabilité [A. Schumann et Y. Pencolé, 07]

Avant de décrire le déroulement de cet algorithme, nous commençons par expliquer le mécanisme qu'il utilise pour la réduction des vérificateurs des sous-systèmes intermédiaires construits en cours de son exécution. L'idée est que l'information sur une faute F qui se produit dans un composant C_i est propagée aux autres composants, ce qui permettra d'identifier les parties de leurs vérificateurs pertinentes au problème de diagnosticabilité. La propagation est basée sur une mesure de connectivité entre le composant C_i et l'ensemble des autres composants.

Distance de communication entre composants et états possiblement non diagnosticables

La *α-connectivité* notée *Con(α, G)* désigne l'ensemble des *SEFs* connectés au *SEF G* par une distance de connectivité qui vaut *α*. Cet ensemble est défini récursivement comme suit :

$Con(0, G) = \{G\}$

$Con(\alpha, G) = \{D / \exists H \in Con(\alpha-1, G)$ *tq. D et H partagent au moins un événement de communication et* $D \notin Con(\beta, G)$ *pour tout* $\beta < \alpha\}$

La *α-connectivité* est utilisée dans le processus de détermination des ***états possiblement non diagnosticables (EPNs)***. Intuitivement les *EPNs* sont les états qui peuvent être impliqués dans la non diagnosticabilité d'une faute dans le vérificateur global. En supposant que la faute F se produit dans G_i , l'ensemble des *EPNs* d'un vérificateur $\hat{G}_j$ noté $EPN(\hat{G}_j)$ est défini formellement comme suit :

- $EPN(\hat{G}_j) = \{x \in \hat{X}_j / x \text{ est non diagnosticable}\}$ si $G_j \in Con(0, G_i)$ *i,e.,* $G_j = G_i$.
- $EPN(\hat{G}_j) = \hat{Y}_j$ si $G_j \in Con(\alpha, G_i)$, $\alpha > 0$, *et pour tout* $\hat{y}_j \in \hat{Y}_j$ et pour tous les vérificateurs connectés $G_k \in con(\alpha - 1, G_i)$ il existe un état $(\hat{y}_j, \hat{x}_k)$ dans le vérificateur $Synch(\hat{G}_j, \hat{G}_k)$ tel que $\hat{x}_k$ est possiblement non diagnosticable.
- $EPN(\hat{G}_j) = \hat{X}_j$ si $G_j \notin Con(\alpha, \hat{G}_i)$ pour tout α.

Soit $w = \{C_1, ..., C_{|w|}\}$ un ensemble de w composants et $\hat{G}_w$ le vérificateur sous-jacent. Un état $\hat{x} = (\hat{x}_{k_1}, ..., \hat{x}_{k_{|w|}})$ est dit possiblement non diagnosticable si et seulement si :

$$\forall i \in \{1, ..., |w|\}, \hat{x}_{k_i} \in EPN(\hat{G}_{k_i}) .$$

On prouve par ailleurs, qu'un état du vérificateur global est non diagnosticable si et seulement s'il est possiblement non diagnosticable. Ce résultat permettra d'un côté de réduire les vérificateurs locaux et d'un autre côté de distribuer le test de la diagnosticabilité.

Réduction des vérificateurs

Un vérificateur est dit réduit si et seulement s'il ne contient que l'état initial, les états possiblement non diagnosticables ou les états menant à des états possiblement non diagnosticables à partir de l'état initial. On fait appel à l'opération de réduction au début de l'algorithme pour réduire les vérificateurs des différents composants du système. Aussi cette opération est appliquée à chaque fois où un vérificateur local d'un sous-système est calculé.

La distribution de la vérification de la diagnosticabilité du système est fondée sur le résultat suivant:

Une faute *F* se produisant dans le système *G* est diagnosticable si et seulement s'il existe un ensemble de vérificateurs réduits $\hat{G}_R = \{\hat{G}_{w1},...,\hat{G}_{w_k}\}$ avec :

- $\{w_1,...,w_k\}$ est une partition de l'ensemble des composants du système ($\bigcup_{i=1}^{k} w_i = C$ (l'ensemble des composants du système) et $w_i \cap w_j = \phi$ pour tout $w_i \neq w_j$)
- Aucun vérificateur dans $\hat{G}_R$ ne contient un cycle avec au moins un observable et un état possiblement non diagnosticable (un tel cycle est noté *COPN)*.

L'algorithme

L'enjeu de l'algorithme est de trouver une partition des composants du système où le vérificateur local sous-jacent à chaque partie (sous-ensemble de composants) ne contient aucun *COPN* sachant que l'opération de synchronisation peut éventuellement éliminer des *COPN*. Etant donné un ensemble de composants $C_1, ..., C_n$ et une faute *F* qui se produit dans C_i, Les grandes lignes de l'algorithme proposé sont résumées comme suit :

- Calculer le F-vérificateur $\hat{G}_i$ du composant C_i ainsi que les vérificateurs $\hat{G}_j$ des différents autres composants .
- $\hat{G} = \{R(\hat{G}_k) / 1 \leq k \leq n\}$ où $R(\hat{G}_k)$ est le résultat de la réduction du vérificateur $\hat{G}_k$.
- Tant que (il y a des *COPNs* dans $\hat{G}$ et Mémoire-Suffisante et Nombre d'éléments de $\hat{G} > 1$)
  - Synchroniser les éléments de $\hat{G}$ deux à deux et réduire les vérificateurs obtenus qui remplaceront alors les éléments de $\hat{G}$.[7]
- En sortant de la boucle on examine la situation :
  - Si aucun vérificateur de $\hat{G}$ ne contient de *COPN*, la faute *F* est diagnosticable puisqu'une partition $\hat{G}_R$ sans *COPN* est trouvée.
  - Un vérificateur contient un *COPN* qui ne peut plus être supprimé par d'autres synchronisations. C'est le cas où le vérificateur global réduit a été calculé. *F* est dans ce cas non diagnosticable. Retourner alors l'ensemble des états impliqués dans des *COPN* qui permet d'avoir une vue synthétique des raisons de la non diagnosticabilité de la faute *F* dans le système.

[7] Il est clair que le choix des paires à synchroniser est tout à fait pertinent pour l'efficacité de l'algorithme. Dans tous les cas, il faut qu'au moins l'un des deux vérificateurs contienne un *COPN* et que les deux vérificateurs soient connectés.

- L'algorithme se termine faute de ressources en mémoire. Dans ce cas, on n'a pas la certitude que la faute n'est pas diagnosticable; L'algorithme retourne l'ensemble des vérificateurs réduits avec des *COPNs*, ce qui donne une vue globale sur les raisons de non diagnosticabilité potentielle de la faute *F*. On peut confirmer en tout cas que tout sous-ensemble de composants impliqués dans un élément donné de la partition actuelle $\hat{G}$ n'est pas suffisant pour assurer la diagnosticabilité de *F* avec certitude.

4. Autres méthodes pour la vérification de la diagnosticabilité

La propriété de diagnosticabilité dans les SED peut être vue comme toute propriété formelle que l'on veut vérifier qu'un modèle formel donné satisfait. C'est pourquoi la vérification de la diagnosticabilité a attiré également des chercheurs qui s'intéressent à ce type de problèmes, notamment ceux qui travaillent sur le model-checking mais aussi ceux qui s'intéressent à la modélisation de systèmes au moyen de langages algébriques ou qui cherchent à vérifier des propriétés formelles en les modélisant dans un cadre logique à l'aide de formules SAT et en utilisant ensuite des solveurs adaptés pour la vérification.

4.1. Model-checking à base d'une structure de Kripke et une logique LTL

Dans cette approche [Cimatti et al. 03], le rôle du processus de diagnostic est d'analyser les entrées (y compris celles provenant d'un contrôleur relié au système en boucle fermée) et les sorties du système à diagnostiquer afin d'estimer un état de croyances qui contient un ensemble d'états internes du système contenant forcément son état réel. L'état de croyances estimé par le diagnostiqueur alimente à son tour le contrôleur en plus des sorties du système. Le système physique est modélisé à l'aide d'un système de transition (à états finis) partiellement observable.

4.1.1. Le modèle

Un automate (partiellement observable) est défini par la structure $P = (X, U, Y, \delta, \lambda)$. X, U, Y sont des ensembles finis désignant respectivement : l'espace d'états, l'espace des entrées et l'espace de sorties. $\delta \subset X \times U \times X$ est la fonction de transition et $\lambda \subset X \times Y$ est une relation d'observation. P exprime les différents comportements possible du systèmes (corrects et fautifs) et peut être non déterministe. On écrit $x \xrightarrow{u} x'$ pour $(x,u,x') \in \delta$ et x / y pour $(x,y) \in \lambda$. Les états du système (les éléments de l'ensemble X) ne sont pas observables. Seules les séquences des entrées et des sorties du système (les éléments de U et Y) sont observables.

Une exécution faisable de $k \geq 0$ étapes dans P est une séquence $\sigma = x_0, y_0, u_1, x_1, y_1, ..., u_k, x_k, y_k$ tel que $x_{i-1} \xrightarrow{u_i} x_i'$ pour $1 \leq i \leq k$ et x_i / y_i pour $0 \leq i \leq k$. L'ensemble des exécutions faisables de P est noté Σ_P. La partie observable d'une exécution faisable σ est $w = y_0, u_1, y_1, ..., u_k, y_k$. On écrit alors $\sigma : x_0 \xrightarrow{w} x_k$ [8].

4.1.2. Conditions de diagnosticabilité

Le diagnostic en ligne démarre de connaissances initiales (éventuellement partielles) et observe les séquences d'entrées et de sorties du système pour mettre à jour, à l'aide d'une ***fonction de diagnostic,*** son état de croyances qui contient les états courants possibles du système. La fonction de diagnostique doit retourner l'ensemble le plus réduit possible contenant l'état actuel réel du système.

---

[8] Pour une exécution σ de longueur k, la trace w des observable est un élément de $Y \times (U \times Y)^k$. Dans le cas général, nous avons $Y \times (U \times Y)^*$.

Une ***fonction de diagnostic*** pour P est : $\Delta : 2^X \times Y \times (U \times Y)^* \to 2^X$. Pour $x_0 \in 2^X$, une valeur de diagnostic $\hat{x} = \Delta(\hat{x}_0, w)$ est dite correcte par rapport à $\hat{x}_0$ et w si et seulement si pour tout $x_0 \in \hat{x}_0$ et x tel que $x_0 \xrightarrow{w} x$ on a $x \in \hat{x}$. Δ est correcte si et seulement si $\Delta(\hat{x}_0, w)$ est correcte pour tout $\hat{x}_0$ et w.

Puisqu'une fonction de diagnostic correcte englobe tous les états courants possibles, on est sûr que l'état courant réel en fait partie.

Une ***condition de diagnostic*** pour un automate P est une paire d'ensembles non vides $c_1, c_2 \subset X$ notée $c_1 \perp c_2$.

On peut utiliser une condition de diagnostic pour exprimer la détection d'une faute ($fault \perp \neg fault$) ou la discrimination entre deux fautes ($fault_a \perp fault_b$). Intuitivement, une valeur de diagnostic est considérée non satisfaisante si elle partage des éléments avec les deux ensembles constituant la condition de diagnostic. Cependant, il n'est pas réaliste d'exiger de la fonction de diagnostic de donner une estimation exacte dans toutes les circonstances. On veut plutôt que le diagnostic soit capable de distinguer les deux alternatives de la condition de diagnostic quand cela est pertinent. On défini alors la notion de contexte.

Un ***contexte de diagnostic*** pour P est une structure $C = \langle \theta, \Sigma_{12} \rangle$ où θ est une relation d'équivalence sur X et $\Sigma_{12} \subset \Sigma_P \times \Sigma_P$. $\hat{x}_0$ satisfait θ ($\hat{x}_0 \models \theta$) ssi $\hat{x}_0 \times \hat{x}_0 \subset \theta$. $(\hat{x}_0, w) \models C$ ssi $\hat{x}_0 \models \theta$ et il existe $(\sigma_1, \sigma_2) \in \Sigma_{12}, x_{0_1}, x_{0_2} \in \hat{x}_0$ tel que $\sigma_1 : x_{0_1} \xrightarrow{w} x_1$ et $\sigma_1 : x_{0_2} \xrightarrow{w} x_2$.

On dit qu'une valeur de diagnostic $\hat{x}$ satisfait $c_1 \perp c_2$ et on note $\hat{x} \models c_1 \perp c_2$ ssi soit $\hat{x} \cap c_1 = \phi$ ou $\hat{x} \cap c_2 = \phi$.

On dit qu'une fonction de diagnostic Δ satisfait $c_1 \perp c_2$ sur le contexte C et on note $\Delta, C \models c_1 \perp c_2$ ssi pour tout $(\hat{x}_0, w) \models C$ on a : $\Delta(\hat{x}_0, w) \models c_1 \perp c_2$.

Enfin, Une condition $c_1 \perp c_2$ est dite ***diagnosticable*** dans P sur C ss'il existe une fonction de diagnostic correcte qui la satisfait.

4.1.3. La diagnosticabilité comme problème d'atteignabilité

L vérification de la diagnosticabilité d'une condition $c_1 \perp c_2$ consiste à vérifier qu'il n'existe pas de paire de chemins critiques i.e., deux exécutions ayant les mêmes traces d'observables, l'une conduisant à c_1 et l'autre conduisant à c_2.

Formellement, une ***paire critique*** d'un automate P avec la trace w pour la condition $c_1 \perp c_2$ est une paire d'exécutions faisables $\sigma_1 : x_{0_1} \xrightarrow{w} x_1$ et $\sigma_2 : x_{0_2} \xrightarrow{w} x_2$ que l'on note $\sigma_1 \| \sigma_2 : x_{0_1} \| x_{0_2} \xrightarrow{w} x_1 \| x_2$ tel que $x_1 \in c_1$ et $x_2 \in c_2$. La paire $\sigma_1 \| \sigma_2$ est dite couverte par un contexte $C = \langle \theta, \Sigma_{12} \rangle$ ssi $(x_{0_1}, x_{0_2}) \in \theta, (\sigma_1, \sigma_2) \in \Sigma_{12}$.

$c_1 \perp c_2$ est ***diagnosticable*** dans P sur C si et seulement si P ne contient aucune paire critique pour $c_1 \perp c_2$ dans C.

Pratiquement, pour vérifier l'existence de paires critiques, on construit l'automate couplé P^2 défini comme suit :

L'automate couplé P^2 est définie par : $P^2 = (X.X, U, Y, \delta.\delta, \lambda.\lambda)$ tel que :

- $X.X \subset X^2$ et pour tout $x_1, x_2 \in X$, $(x_1, x_2) \in X.X$ ss'il existe $y \in Y$ tel que $(x_1, y) \in \lambda$ et $(x_2, y) \in \lambda$.
- $((x_1, x_2), u, (x_1', x_2')) \in \delta.\delta$ ssi $(x_1, u, x_1') \in \delta$ et $(x_2, u, x_2') \in \delta$..
- $((x_1, x_2), y) \in \lambda.\lambda$ ssi $(x_1, y) \in \lambda$ et $(x_2, y) \in \lambda$.

Etant donné un automate couplé P^2, $\sigma_1 : x_{0_1} \xrightarrow{w} x_1$ et $\sigma_2 : x_{0_2} \xrightarrow{w} x_2$ sont deux exécutions faisables de P ssi $(x_{0_1}, x_{0_2}) \xrightarrow{w} (x_1, x_2)$ est une exécution faisable dans P^2.

4.1.4. Expression et vérification en termes d'une structure de Kripke

La vérification proposée dans ce travail est basée sur l'application de techniques de model-checking sur une structure de Kripke correspondant à l'automate analysée. Il s'agit de vérifier la vérité d'une formule exprimée en logique temporelle dans la structure de Kripke. L'intérêt des méthodes basées sur le model-checking est qu'elles exhibent des exemples d'exécutions où la formule à tester est vérifiée. Puisqu'il s'agit de tester une formule qui exprime la non diagnosticabilité du système, cela veut dire que le model-checker peut nous donner des contre-exemples à la diagnosticabilité du système dans le cas où celui-ci n'est pas diagnosticable.

Un e structure de Kripke peut être vue comme un système de transitions, en général non déterministe, et dont les transitions ne sont pas libellées. Les informations du système sont contenues dans les états et consistent en des interprétations des variables d'états de la structure.

Un automate P peut être facilement traduit en structure de Kripke K_P en se basant sur l'idée que les espaces d'entrées et de sorties peuvent êtres incorporées aux états de la structure. Formellement, chaque état s de K_P est associé à une interprétation qui caractérise le triplet : $x / y \xrightarrow{u}$ (P est dans l'état x, la sortie y est observée et l'entrée u est reçue). Pour toute exécution faisable σ de P telle que $\sigma = x_0, y_0, u_1, x_1, y_1, ..., u_k, x_k, y_k$, on associe un chemin de K_P, $\pi = s_0, s_1, ..., s_k$ où chaque s_i est associée au triplet $x_i / y_i \xrightarrow{u_{i+1}}$. Cette procédure de construction s'applique pour le cas de l'automate couplé P^2.

Après la construction de la structure de Kripke, le deuxième ingrédient du model-checking est *la représentation symbolique*. Un état de la structure de Kripke est défini par le vecteur de variables (x_1, x_2, u, y) $(x_1, x_2 \in X, u \in U, y \in Y)$. On peut utiliser des formules pour caractériser des ensembles d'états. Supposons que nous avons les variables v_1, v_2 dans le système et que v_1 peut prendre les valeurs val_{11}, val_{12} et v_2 peut prendre les valeurs val_{21}, val_{22} et notons par P_1 (resp. P_2) la première occurrence de l'automate P qui contribue à l'automate couplé P^2.On peut caractériser par exemple les états où des variables ont certaines valeurs (e.g. $P_1.v_1 = val_{12}$) ou les états dont des variables de même types sont égales (e.g. $P_1.v_2 = P_2.v_2$). Tout sous-ensemble de $P_1.v_1 = val_{12}$ peut être exprimée par une formule $P_1.v_1 = val_{12}$. Par exemple, la formule $c_1(x_1) \wedge c_2(x_2)$ exprime les états de l'automate couplé où la première instance est dans c_1 tandis que la deuxième est dans c_2.

Le dernier ingrédient du model-checking est d'utiliser des formules de logique temporelle (ici c'est la logique temporelle linéaire (*LTL*) qui est utilisée) pour exprimer les propriétés qu'on veut verifier.

Une formule de *LTL* est une combinaison de propositions atomiques à l'aide de connecteurs logiques classiques ou de modalités temporelles. Si ϕ et ψ sont deux formules de *LTL* alors, c'est le cas aussi pour : $F\phi$ (une fois dans le futur ϕ), $G\phi$ (toujours dans le futur ϕ), $\psi \cup \phi$ (une fois dans le futur ϕ et jusque là ψ) et $X\phi$ (dans l'instant suivant ϕ). La sémantique de *LTL* est donnée brièvement comme suit :

- Une formule ϕ de LTL est vérifiée dans un chemin π ssi elle est vérifiée dans π dans l'étape 0 (notée $\pi^0 \models \phi$).
- Si p est une proposition atomique, $\pi^i \models p$ ssi p est vraie par rapport à l'interprétation associée au i^{ème} état de π.
- Les connecteurs logiques sont interprétés comme d'habitude.
- $\pi^i \models F\phi$ ssi $(\exists j \geq i, \pi^j \models \phi)$.
- $\pi^i \models G\phi$ ssi $(\forall j \geq i, \pi^j \models \phi)$.
- $\pi^i \models \psi \cup \phi$ ssi $(\exists j \geq i, \pi^j \models \phi \wedge (\forall \text{ i} \leq \text{h} \leq j, \pi^h \models \psi))$.
- $\pi^i \models X\phi$ ssi $\pi^{i+1} \models \phi$.

Un problème « existentiel » de model-checking $G\phi$ ($G\phi$ est existentiellement vérifiée dans $G\phi$) consiste à détecter si $G\phi$ contient un chemin où $G\phi$ est vérifiée.

Regardons à présent l'utilisation de LTL pour exprimer la condition de non diagnosticabilité.

- On exprime l'atteingnabilité dans P^2 d'une paire critique pour une condition de diagnostic $c_1 \perp c_2$ par la formule $F(c_1(x_1) \wedge c_2(x_2))$. Le model-checker peut être alors exécuté sur le problème : $K_{P^2} \models F(c_1(x_1) \wedge c_2(x_2))$ (si la réponse est oui, cela veut dire qu'on est en présence d'une paire critique).
- Pour prendre en compte un contexte donné $C = (\theta, \Sigma_{12})$, on vérifie le problème $K_{P^2} \models \theta(x_1, x_2) \wedge \Sigma_{12}(x_1, x_2, u, y) \wedge F(c_1(x_1) \wedge c_2(x_2))$. Il s'agit en fait de se limiter uniquement aux chemins de l'automate couplé qui satisfont le contexte C. Souvent $\Sigma_{12}(x_1, x_2, u, y)$ est simplifiée en termes de contraintes propositionnelles $\phi_{12}(x_1, x_2, u, y)$ qui doivent être vérifiées dans tus les états de l'exécution. Dans ce cas le problème à testé devient : $K_{P^2} \models \theta(x_1, x_2) \wedge \phi_{12}(x_1, x_2, u, y) \wedge F(c_1(x_1) \wedge c_2(x_2))$.

Différents outils efficaces ont été développé déjà pour le model-checking symbolique. Ces outils sont le plus souvent basés sur les diagrammes binaires de décision (BDDs) ou les solveurs de type SAT.

4.2. Utiliser un langage algébrique : PEPA

Ce travail [Console et al., 00] se propose d'utiliser un langage algébrique (PEPA pour Performance Evaluation Process Algebra) pour la modélisation de systèmes physiques, la caractérisation de leur diagnostic et l'analyse de leur propriété de diagnosticabilité. On va se focaliser dans ce qui suit sur un exemple d'un système physique (voir figure 10). Ce système comporte une pompe P qui délivre de l'eau à une citerne TA via un tube PI. Une deuxième citerne CO est utilisée comme collecteur de l'eau qui peut fuir du tube. On suppose que la pompe reçoit continuellement et normalement de l'eau. On fait les hypothèses suivantes :

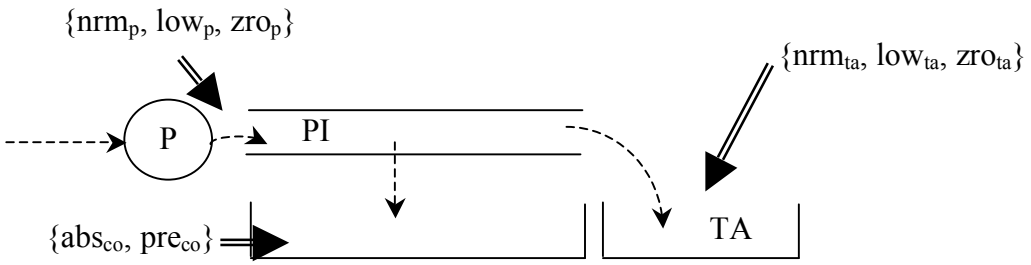


Figure 10. **Le système physique**

- La pompe possède trois modes de fonctionnement : le mode ***ok*** (elle produit un flux normal de sortie), le mode ***avec_fuite*** (elle produit un flux faible de sortie) et le mode ***bloqué*** (elle ne produit rien).
- Le tube deux modes de fonctionnement : le mode ***ok*** (il délivre à la citerne TA toute l'eau reçue de la pompe), le mode ***avec_fuite*** (il délivre un flux faible quand il reçoit un flux normal ou faible de la pompe et rien quand il ne reçoit rien).
- Les citernes TA et CO sont supposées avoir un seul mode qui est le mode ***ok***.
- On dispose de trois capteurs :
  - Le capteur $flux_p$: mesure le flux de sortie de la pompe qui peut être : normal (nrm_p), faible (low_p) ou nul (zro_p).
  - Le capteur $niveau_{TA}$: mesure le niveau de l'eau dans TA qui peut être : normal (nrm_{TA}), faible (low_{TA}) ou nul (zro_{TA}).
  - Le capteur $niveau_{CO}$: indique la présence de l'eau dans CO et peut prendre les valeurs : présent (pre_{CO}) ou absent (abs_{CO}).

4.2.1. Modélisation d'un système physique à l'aide de PEPA

4.2.1.1. Le langage PEPA

Le langage algébrique PEPA permet une modélisation modulaire d'un système : chaque composant est d'abord modélisé séparément, ensuite la structure global du système est décrite en considérant la manière avec laquelle les composants sont connectés. Les composants exécutes des activités ayant la forme $a = (\alpha, r)$. α désigne une action et r est un paramètre temporel exprimant sa durée (ce travail s'intéresse uniquement aux actions). La syntaxe de PEPA est définie par la grammaire suivante :

$$S ::= A \mid (\alpha, r).S \mid S + S$$
$$P ::= S \mid P \bowtie_L P \mid P / H$$

S (resp. P) définie les comportements des composants (resp. la structure du système). Nous avons les cas suivants :

- ***Constante :*** $S \overset{def}{=} A$ sert à associer à S le comportement du composant A.
- ***Préfixe :*** c'est le mécanisme de spécification des comportements $\alpha.S$ veut dire : exécuter α et se comportre comme S. Par exemple, dans la spécification du comportement

correct de la pompe, on peut écrire $Pok_1 ::= nrm_0.nrm_P.P$ pour exprimer que la pompe reçoit un flux normal, produit un flux normal et se comporte comme P.

- ***Choix :*** c'est pour capter la possibilité de choix entre plusieurs alternatives.
- ***Coopération :*** $C_1 \triangleright\triangleleft_L C_2$ modélise une communication entre C_1 et C_2. Cette communication est synchrone par rapport aux actions contenues dans L (appelé ensemble de coopération) et asynchrone pour les autres actions. Si l'ensemble L de coopération est vide, les deux composants fonctionnent indépendamment l'un de l'autre, on note dans ce cas la coopération par : $C_1 \| C_2$. Par exemple, pour dire que la pompe passe de l'eau au tube, on peut écrire $P \triangleright\triangleleft_L PI$ avec $L = \{nrm_P, low_P\}$. Les actions de L (qui sont des actions partagées) sont activées dans $P \triangleright\triangleleft_L PI$ s'il sont activées dans P et dans PI .
- ***Masquage :*** consiste à ne rendre observable de l'extérieur ou d'un autre composant que des actions jugées pertinentes. Dans P / H , c'est les actions de H qui seront masquées.

4.2.1.2. Modélisation du système physique

On utilise un ensemble d'équations pour décrire le comportement de chaque composant ainsi que la structure globale du système. L'exemple précédent est modélisé en PEPA comme suit:

(1) La pompe

$$P \overset{def}{=} Pok_1 + Plk_1 + Pbl_1 + End$$

$$Pok_1 \overset{def}{=} nrm_0.nrm_P.P$$

$$Plk_1 \overset{def}{=} nrm_0.low_P.P$$

$$Pbl_1 \overset{def}{=} nrm_0.zro_P$$

(2) Le tube

$$PI \overset{def}{=} PIok_1 + PIlk_1 + End$$

$$PIok_1 \overset{def}{=} nrm_P PIok_2 + low_P PIok_3 + zro_P PIok_4$$

$$PIk_2 \overset{def}{=} nrm_1.abs_2.PI$$

$$PIk_3 \overset{def}{=} \mathrm{low}_1.abs_2.PI$$

$$PIk_4 \overset{def}{=} \mathrm{zro}_1.abs_2.PI$$

$$PIlk_1 \overset{def}{=} nrm_P PIlk_2 + low_P PIlk_2 + zro_P PIlk_3$$

$$PIlk_2 \overset{def}{=} \mathrm{low}_1.pre_2.PI$$

$$PIlk_3 \overset{def}{=} \mathrm{zro}_1.abs_2.PI$$

(3) La citerne TA

$$TA \overset{def}{=} \mathrm{TA}ok_1 + End$$

$$TAok_1 \overset{def}{=} nrm_1.nrm_{ta}.TA + low_1.low_{ta}.T$$
$$+ zro_1.zro_{ta}.TA$$

(4) La citerne CO

$$CO \overset{def}{=} \mathrm{CO}ok_1 + End$$

$$COok_1 \overset{def}{=} \mathrm{pre}_2.pre_{co}.CO$$
$$+ abs_2.abs_{co}.CO$$

(5) Le End

$$End \overset{def}{=} \mathrm{end}.End$$

(6) Structure du système

a. Les instances des composants

$$P^{(1)} : P; PI^{(1)} : PI; TA^{(1)} : TA; CO^{(1)} : CO$$

b. Les connexions

$$SD_1 \overset{def}{=} (P^{(1)} \rhd\lhd_{L_1 \cup \{end\}} (PI^{(1)} \rhd\lhd_{L_2 \cup \{end\}} (TA^{(1)} \rhd\lhd_{\{end\}} CO^{(1)}))) / H$$
$$L_1 = \{nrm_P, low_P, zro_P\}, L_2 = \{nrm_1, low_1, zro_1, abs_2, pre_2\},$$
$$H = \{nrm_0, nrm_1, low_1, zro_1, abs_2, pre_2\}$$

Prenons par exemple le tube, l'équation PI dit que celui-ci peut être dans le mode $PIok_1$ (normal) ou dans le mode $PIlk_1$ (avec_fuite). L'identificateur additionnel ***End*** est utilisé pour exprimer que composant évolue vers un état final (absorbant) dont l'intérêt sera montré plus tard. $PIok_1$ est à son tour définie en termes des différents comportements normaux du tube. Ces comportements sont définis par la nature du flux que le tube reçoit de la pompe (normal, faible ou nul). L'équation $PIok_3$ par exemple, correspond au comportement normal du tube dans le cas où il a reçu un flux faible de la pompe (low_P): il délivre alors un flux faible low_1 et il n'y aura pas d'eau qui fuit du tube (abs_2). Le tube se comporte ensuite comme PI .

$P^{(1)}, PI^{(1)}, TA^{(1)}$ et $CO^{(1)}$ désignent des instances des composants du système et SD_1 décrit sa structure. On rappelle que H contient les actions présentes dans les équations mais qui seront masquées pour un observateur externe.

La sémantique d'un terme de PEPA est donnée par un $LTS = (C, A, \xrightarrow{\alpha})$ (Labeled Transition System) où C est un ensemble d'états (correspondant aux termes syntaxiques du langage), A est un ensemble d'actions et $\xrightarrow{\alpha}$ est une relation de transition indiquant quelle action conduit d'un état à un autre. L'évolution des composants individuellement ou en collaboration est décrite à l'aide de LTSs. Le système atteint un état final lorsque tous ses composants atteignent des états finaux. Le système de transition sous-jacent à un système physique *SD* sera noté LTS_{SD}.

4.2.2. Caractériser le diagnostic avec PEPA

Le diagnostic est effectué à partir d'un ensemble d'observations sur un ou plusieurs lapses de temps (un lapse de temps correspond à la prise des mesures des différents capteurs une fois chacun). Le résultat du diagnostic consiste à affecter, pour chaque lapse de temps, un mode de fonctionnement à chaque composant. Les observations sont exprimées en PEPA à l'aide d'équations. Par exemple:

L'équation $Obs_1 \overset{def}{=} nrm_P.abs_{CO}.nrm_{ta}.end.End$ correspond à l'observation de chaque capteur pendant un lapse de temps, les équations :

$Obs_1 \overset{def}{=} nrm_P.O_2.low_P.O_2; O_2 \overset{def}{=} abs_{CO}.nrm_{ta}.end.End$ correspondent à un cas où l'observation du flux d'entrée du tube n'est pas certaine mais peut être soit nrmP ou lowP.

Enfin l'équation $Obs_1 \overset{def}{=} nrm_P.abs_{CO}.nrm_{ta}.nrm_P.pre_{CO}.low_{ta}.end.End$ correspond à l'observation pour deux lapses de temps.

Pour un système DS et une observation Obs, le diagnostic sera basé sur l'équation : $Diag \overset{def}{=} (Obs \rhd\lhd_{S\cup'\{end\}} SD)/H$ où *S* contient toutes les actions que les capteurs peuvent mesurer et H l'ensemble des actions non observables[9]. Cette équation correspond à un LTS (noté LTS_{diag}) dans lequel seulement quelques chemins conduisent à un état final, c'est justement ces chemins qui expliquent l'observation. On peut trouver dans LTS_{diag} des interblocages i.e., des états dans lesquels on n'a pas de transitions. Considérons à titre d'exemple le système SD_1 l'observation *Obs*1. Dans Obs_1, le premier capteur indique un flux normal provenant de la pompe, de ce fait, on coupe de LTS_{SD_1} les chemins commençant par low_p ou par zro_p. D'autres coupures peuvent aussi être envisagées. On ne garde enfin du compte que les évolutions menant à un état final.

Formellement, pour tout composant C_i, on note par $\boldsymbol{B}_i$ l'ensemble des moides de fonctionnement de C_i. dans LTS_{diag} qui est le système de transition sous-jacent à $Diag \overset{def}{=} Obs \rhd\lhd_{S\cup\{end\}} SD$, chaque état est de la forme $\widetilde{O}\|\widetilde{C}_1\|\ldots\|\widetilde{C}_n$ où $\widetilde{O}$ est l'état de l'observation et $\widetilde{C}_i$ est l'état du composant C_i. L'état final est $End\|\ldots\|End \quad (\mathrm{n}+1 \text{ fois})$.

Un chemin $\sigma = s_1 s_2 \ldots s_n$ de LTS_{diag} est dit ***consistant*** s'il mène de l'état initial à l'état final.

Plaçons nous dans le cas d'une observation sur un seul lapse de temps et soit $\sigma = s_1 s_2 \ldots s_n$ un chemin consistant. Un diagnostic $\Delta(\sigma)$ est un ensemble de modes de fonctionnement $b_i \in B_i$ tel que b_i correspond à un certain état $\widetilde{C}_i$ dans s_i. Pour un diagnostic donné $\Delta(\sigma)$, $\Delta^F(\sigma)$ est l'ensemble de modes avec fautes dans $\Delta(\sigma)$.

Dans le cas d'une observation sur plusieurs laps de temps, un chemin σ est décomposé en une suite de sous-chemins $\sigma_1,\ldots\sigma_k$ où chaque sous-chemin correspond à un laps de temps. Le diagnostic $\Delta(\sigma)$ est dons une séquences de diagnostics sur des lapes de temps uniques $\Delta_1(\sigma),\ldots,\Delta_k(\sigma)$. Chaque $\Delta_i(\sigma)$ est obtenu à partir de σ_i de la même manière décrite dans le paragraphe précédent.

4.2.3. Diagnosticabilité avec PEPA

Dans ce cadre, le problème de diagnosticabilité consiste à vérifier si le système peut être diagnostiqué étant donnés les capteurs disponibles, la modélisation et un certain nombre de laps de temps. Dans ce travail, les auteurs s'intéressent uniquement au problème de déterminer un ensemble optimal de capteurs pour la diagnosticabilité.

Le système est dit diagnosticable par rapport à un ensemble Se de capteurs si et seulement si : pour toute combinaison de valeurs lues des capteurs, il y a uniquement un seul candidat

[9] Pour simplifier, *H* sera omis dans ce qui suit

minimal de diagnostic et tout faute du système correspond à un candidat de diagnostic pour une certaine lecture des capteurs.

En PEPA, l'idée est donc d'étudier l'équation $Diag \overset{def}{=} Obs \rhd\lhd_{S\cup\{end\}} SD$ pour les différentes ensembles Se de capteurs et les ensembles correspondants S des actions observables et les observations pertinentes Obs_i pour l'ensemble choisi de capteurs jusqu'à l'obtention d'un ensemble de capteurs pour lequel le système est diagnosticable.

Le système est diagnosticable par rapport à un ensemble de capteurs Se avec l'ensemble correspondant d'actions observables S si pour toute observation Obs_i, les chemins consistants du LTS_{Diag_i} de $Obs \rhd\lhd_{S\cup\{end\}} SD$ sont *d-équivalents* et toutes les fautes correspondent à au moins un chemin d'une certaine observation Obs_i. Sachant que les chemin d'un ensemble $S = \{\sigma_1,...,\sigma_2\}$ sont dits *d-équivalents* ssi $\exists j$ tel que : $D^F(\sigma_j) = \cap_{i=1..n} D^F(\sigma_j)$.

Pour l'exemple précédent, on a trois capteurs $flow_P, level_{ta}$ et level_{co} et on veut déterminer le sous-ensemble minimal de ces capteurs qui garantie la diagnosticabilité du système. On commence par étudier si un seul des trois capteurs suffit. On trouve par exemple pour le capteur $level_{ta}$ que les trois observations possibles sont : $Obs_1 \overset{def}{=} nrm_{ta}.end.End$, $Obs_2 \overset{def}{=} low_{ta}.end.End$ et $Obs_3 \overset{def}{=} zro_{ta}.end.End$. Les équations qui en résultent sont donc : $Diag_i \overset{def}{=} Obs_i \rhd\lhd_{S\cup\{end\}} SD_1$, $S = \{nrm_{ta}, low_{ta}, zro_{ta}\}$. Pour Obs_2 le système n'est pas diagnosticable car LTS_{Diag_2} contient deux chemins non d-équivalents correspondant au diagnostics minimums $D_1 = \{P^{(1)} leaking\}$ et $D_2 = \{PI^{(1)} leaking\}$. D'une manière analogue, on peut prouver que si on prend les deux capteurs $level_{ta}$ et $level_{co}$ le système sera diagnosticable.

4.3. **Utiliser les algorithmes de satisfiabilité : SAT**

En s'inspirant de l'algorithme proposé dans de [Jiang et al, 01], le travail décrit dans [Rintanen et Grasien, 07], ce travail modélise la vérification de la diagnosticabilité dans des systèmes succincts à événements discrets par une formule SAT propositionnelle. Les systèmes succincts à événements discrets sont des systèmes qui permettent de présenter d'une manière succincte des systèmes à événements discrets de grande taille en caractérisant les états par des variables d'états d'un ensemble A, qui dans le travail décrit ici, sont à valeurs booléennes. Les états du système correspondent aux différentes valuations possibles des variables de A. Un état s du système est défini par une fonction : $s : A \longrightarrow \{0,1\}$. L'ensemble des littéraux à base des variables d'états est $L = A \cup \{\neg a | a \in A\}$. Le langage Π est défini sur A et contient toutes les formules formées à partir des éléments de A et des connecteurs ∨ et ¬ (les autres connecteurs logiques sont définis par ces deux connecteurs de la façon habituelle)..

Un système succinct est représenté par la structure $(A, \Sigma_o, \Sigma_u, \Sigma_f, \delta, s_0)$ où :

A est un ensemble de variables booléennes ; Σ_o (resp. Σ_u, Σ_f) est l'ensemble des événements observables (resp. non observables, de faute) ; $\delta : \Sigma_o, \Sigma_u, \Sigma_f \longrightarrow 2^{\Pi \times 2^L}$ associe à chaque événement un ensemble de couples (ϕ, c) ; s_0 est l'état initial.

Un événement e est possible dans tout état s tel que $s \| = \phi$ four un certain $(\phi, c) \in \delta(e)$; lorsque e se produit dans un état s, l'une des couples (ϕ, c) est choisi est l'effet de e sera donc que les littéraux de c seront satisfaits dans le nouvel état $s' = succ(s, c)$ (s' est le successeur de s selon c). Nous avons :

- $s'(a) = 1$ pour tout $a \in A$ tel que $a \in c$.
- $s'(a) = 0$ pour tout $a \in A$ tel que $\neg a \in c$.
- $s'(a) = s(a)$ pour tout $a \in A$ qui ne se produit pas dans s.

La notion de plans parallèles ou partiellement ordonnés permet d'améliorer la performance des techniques SAT pour la planification. Cette idée est appliquée ici en regroupant considérant les événements qui peuvent s'exécuter simultanément et considérer le comportement du système par rapport à des ensembles d'événements indépendants. Cette indépendance est définie par la notion d'interférence. Deux couples (ϕ_1, c_1) et (ϕ_2, c_2) interfèrent s'il existe $a \in A$ tel que : soit $(a \in c_1)$ (resp. $(\neg a \in c_1)$) et $((\neg a \in c_2)$ ou $(\neg a \in \phi_2))$ resp. $(((a \in c_2)$ ou $(a \in \phi_2)))$, soit $(a \in c_2)$ (resp. $(\neg a \in c_2)$) et $(\neg a \in \phi_1)$(resp. $(a \in \phi_1)$). Les événements $e_1, ..., e_n$ peuvent se produire simultanément avec $o_1 \in \delta(e_1), ..., o_n \in \delta(e_n)$ si $(\forall (o, o') \in \{o_1, ..., o_n\})$, o et o' n'interfèrent pas.

La diagnosticabilité est définie alors sur des séquences $E_1, ..., E_n$ où E_i est un ensemble (potentiellement vide) d'événements simultanés. Le successeur s' d'un état s selon un ensemble E est défini par : $s' = succ(s, \cup_{(\phi, c) \in E} c)$. La projection $\pi(\sigma)$ d'une séquence d'ensembles d'événements σ est donnée par : $\pi(\varepsilon) = \varepsilon$ et $\pi(E\sigma) = (E \cap \Sigma_o)\pi(\sigma)$.

La définition de la diagnosticabilité pour les systèmes de transition succincts avec des événements simultané est analogue à la définition classique de la diagnosticabilité.

Pour vérifier la diagnosticabilité par des algorithmes de satisfiabilité, on construit une formule dont les interprétations qui la satisfassent correspondent à l'existence de couples $[(s_0, ..., s_n), (\hat{s}_0, ..., \hat{s}_n)]$ de séquences d'états avec $s_0 = \hat{s}_0$ et qui correspondent au couple de séquences d'ensemble d'événements $[(E_1, ..., E_{n-1}), (\hat{E}_1, ..., \hat{E}_{n-1})]$ tel que $\pi(E_1, ..., E_n) = \pi(\hat{E}_1, ..., \hat{E}_n)$, l'une des séquences du couple contient une faute mais pas l'autre séquence et pour les deux séquences, il existe une boucle, i.e. il existe $i \in \{1, ..., n-1\}$ tel que $s_n = s_i$ et $\hat{s}_n = \hat{s}_i$. La formule est satisfiable pour une longueur donnée des séquences d'événements si et seulement s'il est impossible de détecter l'occurrence d'une faute pour ses séquences.

Pour définir la formule permettant de tester la diagnosticabilité d'un système de transition succinct, chaque événement, à chaque instant de temps $\pi(\varepsilon) = \varepsilon$ est décrit par une formule $\pi(\varepsilon) = \varepsilon$. Les variables utilisées (paramétrés par les instants de temps) sont les suivantes :

- a^t et $\hat{a}^t$ pour tout $a \in A$ et $t \in \{0,...,n\}$.
- e_o^t pour tout $e \in \Sigma_o \cup \Sigma_u \cup \Sigma_f$, $o \in \delta(e)$ et $t \in \{0,...,n-1\}$.
- $\hat{e}_o^t$ pour tout $e \in \Sigma_o \cup \Sigma_u$, $o \in \delta(e)$ et $t \in \{0,...,n-1\}$.
- e^t pour tout $e \in \Sigma_o$ et $t \in \{0,...,n-1\}$.

Un événement ne se produit que si sa production est possible et s'il se produit, ses effets se produisent également :

$$e_0^t \rightarrow \phi^t \text{ pour tout } o = (\phi, c) \in \delta(e), \; e_0^t \rightarrow \Lambda_{l \in c} l^{t+1} \text{ pour tout } o = (\phi, c) \in \delta(e).$$

La valeur d'une variable ne change que s'il y a une raison pour cela :

$$(a^t \wedge \neg a^{t+1}) \rightarrow (e_{1o_1}^t \vee ... \vee e_{ko_k}^t)$$

pour tout $a \in A$ où $o_1 = (\phi_1, c_1), ..., o_k = (\phi_k, c_k)$ représentent toutes les occurrences d'événements avec $\neg a \in c_i$ et $e_1, ..., e_k$ sont les événements respectives avec $o = (\phi, c) \in \delta(e)$. (le changement du faux au vrai se fait d'une manière analogue).

Un événement se produit d'une seule façon et deux événements interférant ne se produisent pas simultanément :

$$\neg(e_o^t \wedge e_{o'}^t) \text{ pour tout } e \in \Sigma_o \cup \Sigma_u \cup \Sigma_f \text{ et } \{o, o'\} \subseteq \delta(e) \text{ tel que } o \neq o'$$

$$\neg(e_o^t \wedge e_{o'}'^t) \text{ pour tout } \{e, e'\} \in \Sigma_o \cup \Sigma_u \cup \Sigma_f \text{ et } o \subseteq \delta(e) \text{ , } o' \subseteq \delta(e') \text{ tel que } o, o' \text{ interfèrent.}$$

Les formules ci-dessus décrivent une étape pour la séquence avec faute. Une copie de ces formules doit être ajoutée pour la séquence sans faute en remplaçant les variables a^t et e_o^t par $\hat{a}^t$ et $\hat{e}_o^t$ et en restreignant les événements à l'ensemble $\Sigma_o \cup \Sigma_u$.

Les mêmes observables se produisent dans les deux séquences :

$(V_{o \in \delta(e)} e_0^t) \leftrightarrow e^t$ pour tout $e \in \Sigma_o$ et $(V_{o \in \delta(e)} \hat{e}_0^t) \leftrightarrow e^t$ pour tout $e \in \Sigma_o$

A tout instant, au moins un événement se produit :

$$V_{e \in \Sigma_o} e^t \vee V_{e \in \Sigma_u \cup \Sigma_f, o \in \delta(e)} e_o^t \vee V_{e \in \Sigma_u, o \in \delta(e)} \hat{e}_o^t$$

La conjonction de toutes les formules ci-dessus est notée $\mathrm{T}(t, t+1)$. Pour l'état initial s_0 on a la formule : $\mathrm{I}_0 = \Lambda\{a^0 \wedge \hat{a}^0 | a \in A, s_0(a) = 1\} \wedge \Lambda\{\neg a^0 \wedge \neg \hat{a}_0 | a \in A, s_0(a) = 0\}$.

La formule Φ_n^T suivante exprime l'existence d'une paire d'exécutions avec des cycles avec les mêmes événements observables et avec une faute dans l'une des exécutions mais pas dans l'autre :

$$\Phi_n^T = \mathrm{I}_0 \wedge \mathrm{T}(0,1),...,\mathrm{T}(n-1,n) \wedge V_{t=0}^{n-1} V_{e\in\Sigma_f} V_{o\in\delta(e)} e_o^t \wedge V_{m=0}^{n-1} \left(\Lambda_{a\in A} \left(\left(a^n \leftrightarrow a^m \right) \wedge \left(\hat{a}^n \leftrightarrow \hat{a}^m \right) \right) \right)$$

On montre alors qu'un système succinct est non diagnosticable si et seulement si la formule Φ_n^T est satisfiable pour un certain $n \geq 1$.

Ce travail introduit brièvement des techniques particulières pour plus d'efficacité dans la vérification de la diagnosticabilité d'un système à l'aide d'une formalisation SAT. De telles techniques sont particulièrement importance dans le cas d'un système diagnosticable. En fait, dans un tel cas, on peut être amené à vérifier la satisfiabilité de la formule Φ_n^T pour toutes les valeurs possibles de n. (pour q variables d'état cas, n varie entre 1 et 2^{2q}).

5. Conclusion

Nous avons présenté dans ce travail un état de l'art des méthodes de diagnosticabilité dans les systèmes à événements discrets. Plusieurs efforts restent à fournir dans ce domaine afin de tenir compte de plus en plus de contraintes de différentes natures imposées généralement dans le cas d'applications réelles. Parmi les perspectives ouvertes dans ce domaine, on peut citer :

- Adapter les algorithmes de diagnosticabilité à des modèles étendus à événements discrets. Parmi les cas qu'on peut envisager :
  - Etudier la diagnosticabilité de systèmes munis d'informations supplémentaires sur les probabilités d'occurrences des événements.
  - Etendre la diagnosticabilité à des modèles qui tiennent compte de contraintes temporelles.
  - Etudier des cas où l'information sur les événements et leurs natures (observables, non observables, faute) est incertaine, incomplète et/ou imprécise. On doit tester l'adéquation de théories communément utilisé en Intelligence Artificielle pour rendre compte de tels connaissances (logique floue, logique possibiliste, réseaux bayésiens, logiques non monotones, …).
  - Etudier des cas où la faute n'est pas simplement un événement élémentaire mais un pattern d'événement ;
- Développer des outils d'aide à la décision autour de la propriété de diagnosticabilité. Ces outils permettent à l'industriel de prendre des décisions sur le choix optimal des observables permettant d'assurer, à moindre coût, des systèmes diagnosticables ou du moins dont la non diagnosticabilité n'est pas critique en pratique.
- Lier la diagnosticabilité au diagnostic. L'étude de la diagnosticabilité permet de mettre en relation des types de défaillances prédéfinies avec des signatures particulières qui sont des séquences d'observations qui caractérisent les occurrences de ces défaillances. Ces relations peuvent être exploitées par un processus de diagnostic en ligne fondé sur la reconnaissance de chroniques qui sont dans notre cas les signatures des défaillances. Il est important ensuite d'étudier comment synchroniser et faire coopérer un tel diagnostic par reconnaissance de chroniques avec un autre diagnostic en ligne basé, lui, sur la surveillance continue du système en comparant son comportement avec son modèle de fonctionnement correct, afin de détecter des défaillances non répertoriées à l'avance

6. Références

M. Beyoudh, L. Travé-Massuyès et X. Olive, "hybrid systems diagnosability by abstracting faulty continuous Dynamics", *Proceedings of the 17th International Workshop on Principles of Diagnosis (DX'06)*, pp. 9-15, 2006.

A. Cimatti, C. Pecheur et R. Cavada, "Formal Verification of Diagnosability via Symbolic Model Checking", *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, pp. 363-369, 2003.

L.Console, C. Picardi et M. Ribaudo, "Diagnosing and diagnosability analysis using pepa", *Proceedings of the 14th European Conference on Artificial Intelligence, (ECAI-02)*, pp. 131-135, 2000.

O.Contant, S. Lafortune et D. Teneketzis, "Diagnosability of Discrete Event Systems with modular structure*", Discrete Event Dynamic Systems*, 16(1)-9-37, 2006.

M.O. Cordier, L Travé-Massuyès et X. Pucel, "Comparing diagnosability in continuous and discrete-events systems", *Proceedings of the 17th International workshop on principles of Diagnosis (DX'06)*, pp. 55-60, 2006.

J. Esparza, S. Römer, W. Vogler, "An improvement of McMillan's Unfolding Algorithm", Formal Methods in System Design, 20(3) : 285-310, 2002.

S. Haar. "Diagnosability Of Asynchronous Discrete Event Systems in Partial Order Semantics". Research Report INRIA, No 0, 2005.

S. Haar, A. Benveniste, E. Fabre, C. Jard. "Partial Order Diagnosability Of Discrete Event Sustems Using Petri Net Unfoldings", Proceedings of the 42th IEEE Conference on Decision and Control, volume 4(9-12), pages 3748-3753, 2003.

G.K. Fourlas, K.J. Kyriakopoulos et N.J. Krikelis, "Diagnosability of hybrid systems", *Proceedings of the 10th IEEE Mediteranean Conference on Control and Automation (MED2002)*, 2002.

S. Jiang, Z. Hiang, V. Chandra et R. Kumar, "A polynomial Algorithm for Testing Diagnosability of Discrete Event Systems", *IEEE Transactions on Automatic Control*, 46(8) : 1318 – 1321, 2001.

Y. Pencolé, "Diagnosability analysis of ditributed discrete event systems", *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI-04)*, pp. 43-47, 2004.

R. Reiter, "A Theory of diagnosis from First Principles", *Artificial Intelligence*, 32(1) : 57-95, 1987.

J. Rintanen et A. Grasien, "Diagnosability Testing with Satisfiability Algorithms", *Proceedings of the 20th International Joint Conference on Artificial Intelligence(IJCAI-07)*, pp. 532-537, 2007.

M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen et D. Teneketzis, "Diagnosability of Discrete-Event Systems", *IEEE Transactions on Automatic Control*, 40(9) : 1555 – 1575, 1995.

A. Schumann et Y. Pencolé, "Scalable Diagnosability Checking of Event-Driven Systems", *Proceedings of the 20th International Joint Conference on Artificial Intelligence(IJCAI-07)*, pp. 575-580, 2007.

R. Sengupta, "Diagnosis and communication in distributed system", *California Partners for Advanced Transit and Highways (PATH) research report, UCB-ITS-PRR-99-16,* 1999.

S. Tripakis, “Fault Diagnosis for Timed Automata”, Proceedings of the 7th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems, LNCS, vol. 2469, pages 205-224, 2002.

Y.L. Wen, M. Jeng, “Diagnosability of Petri nets”, IEEE International Conference on Systems, Man and Cybernetics, 5(10-13) : 4891-4896, 2004.

WS-DIAMOND Group, “Characterization of diagnosability and repairability for delf-healing Web Services”, *Delivrable N°. D5.1 (IST-516933) of the WS-DIAMOND european project (Web Services, DIagnosability, MONitoring and Diagnosis)*, 2007.

T. Yoo et S. Lafortune : “Polynomial-Time Verification of Diagnosability of Partially-Observed Discrete-Event Systems”, *IEEE Transactions on Automatic Control*, 47(9) : 1491 – 1495, 2002.