Conception d'un système de vérification de la diagnosticabilité par model checking à partir du modèle du système

Mehdi Chankate, Alexandre Philippot, Véronique Carré-Ménétrier, Pascale Marangé

CONCEPTION D'UN SYSTEME DE VERIFICATION DE LA DIAGNOSTICABILITE PAR MODEL CHECKING A PARTIR DU MODELE DU SYSTEME

M. CHANKATE, A. PHILIPPOT, V. CARRE-MENETRIER.

Université de Reims Champagne-Ardenne, CReSTIC (EA3804), Reims, France
mehdi.chankate.emsi.marrakech@gmail.com
alexandre.philippot@univ-reims.fr
veronique.carre@univ-reims.fr

P. MARANGE

Université de Lorraine, CRAN, UMR 7039, Campus Sciences, BP 70239, 54506 Vandoeuvre-lès-Nancy, France
CNRS, CRAN, UMR 7039, France
pascale.marange@univ-lorraine.fr

RESUME : *Dans ce papier, nous nous intéressons à la vérification de la diagnosticabilité par model-checking des Systèmes à Evénements Discrets (SED) modélisés à base d'automates à états finis. Nous proposons une méthode basée sur un algorithme permettant l'analyse de la diagnosticabilité du SED à partir de son modèle de base. Celui-ci constitue la donnée d'entrée de notre méthode. Une condition de la diagnosticabilité étant la présence d'un cycle indéterminé dans un système, nous proposons de rechercher l'ensemble des cycles puis de les analyser. Pour cela, notre méthodologie se découpe en 3 phases. Une phase de modélisation et instanciation de modèles, une phase de recherche de cycles et enfin une phase d'analyse de la non-diagnosticabilité. Nous illustrons notre approche autour d'une étude expérimentale.*

MOTS-CLES : *Diagnostic, Diagnosticabilité, Automate à états finis, Systèmes à Evénements Discrets.*

1 INTRODUCTION

Ces dernières années, les travaux autour du diagnostic ont pris de l'essor dans le monde académique et industriel du fait de l'augmentation de la complexité des systèmes, mais aussi des coûts d'interventions de maintenance. Pour améliorer la disponibilité et la fiabilité des installations, il est nécessaire de développer des approches systématiques de diagnostic afin de détecter et d'isoler des fautes. Par ailleurs, il est devenu primordial de mettre en place des approches permettant d'évaluer les performances de ces méthodes en termes de détection, localisation et identification d'une faute dans un délai fini. On parle alors de diagnosticabilité. Parmi les approches de diagnostic, la littérature a montré un intérêt particulier autour des approches à base de modèles pour le diagnostic des Systèmes à Evénements Discrets (SEDs) (Cassandra and Lafortune, 1999).

La notion de diagnostiqueur (ou diagnoser), observateur du système, est apparue dans les travaux de (Sampath et al., 1995). C'est également dans ces travaux que la notion de « diagnosticabilité » est apparue, permettant de garantir le diagnostic d'une faute dans un délai fini. Elle apparaît comme une information primordiale sur le système et sur le travail à réaliser sur l'observateur de fautes. En d'autres termes, il se pose la question de la nécessité de construire un diagnostiqueur si le système n'est pas diagnosticable à certains types de fautes. Différentes évolutions de la diagnosticabilités sont retrouvées dans la littérature notamment en fonction de la structure du système et de sa modélisation (Contant et al., 2004), (Debouk et al., 2000), (Qiu and Kumar, 2006). Mais quelle que soit cette diagnosticabilité, il est primordial de pouvoir évaluer sa capacité. (Sampath et al.1995) se base sur la construction d'un diagnostiqueur alors que (Jiang et al., 2001) construit un modèle vérificateur (par une méthode dite du twin-plant). D'autres méthodes, par la construction d'un automate vérificateur non déterministe dans (Yoo, et al., 2002) ou par le test du vide dans un automate de Büchi dans (Cassez and Tripakis, 2008), ont été traitées comme des approches centralisées. Dans le cadre des structures décentralisées, il y a très peu d'algorithmes concernent la vérification de diagnosticabilité locale et/ou modulaire. Dans (Pencolé, 2004), un vérificateur local est construit pour chaque sous-système. Dans (Boussif and Ghazel, 2017) une variante du diagnostiqueur classique est mise en place, en séparant explicitement les états normaux de ceux fautifs dans chaque nœud du diagnostiqueur.

La plupart de ces approches mettent en avant des algorithmes de résolution performants dans la mesure où le système reste de taille raisonnable et que la partition de fautes est très faible (souvent une seule faute considérée). En effet, en outre le problème d'explosion combinatoire de l'espace d'états que l'on pourrait retrouver sur le diagnostiqueur du système, la complexité combinatoire de l'algorithme d'évaluation en est un. Les différentes approches font très souvent appel à des modèles intermédiaires pouvant rendre la résolution parfois complexe.

L'approche présentée dans ce papier se base sur une méthode de vérification de la diagnosticabilité à travers l'utilisation du Model-Checking. Cette proposition permet notamment l'analyse de la diagnosticabilité à partir du modèle global du système sans construction de modèles intermédiaires. Nous montrons qu'une telle méthode permet aussi la diagnosticabilité d'un système caractérisé par l'apparition de plusieurs types de fautes. Nous nous intéressons dans ce travail à déterminer une solution dans une architecture centralisée afin de prouver la faisabilité de l'approche. Nos travaux futurs se focaliseront sur l'étude de la diagnosticabilité dans une architecture distribuée.

Dans ce qui suit nous rappelons dans un premier temps quelques notions sur la modélisation des SEDs et les opérations associées, la définition de la diagnosticabilité et les deux principales méthodes de diagnosticabilité de la littérature. La section 3 se focalise sur la méthode de vérification de la diagnosticabilité par Model-Checking que nous proposons. Une étude expérimentale illustre l'algorithme en section 4. Enfin nous conclurons le papier avec des perspectives d'amélioration de l'algorithme.

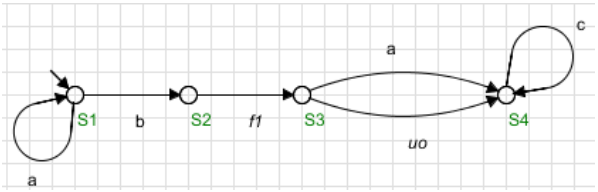
2 PRELIMINAIRES

Un des moyens formels pour étudier le comportement logique des SED est basé sur la théorie des langages et les automates. Dans ces travaux, l'aspect temporel du système est ignoré, seuls l'ordre et l'identité des événements générés par le système sont pris en compte. Par conséquent, un modèle logique où seules les séquences d'événements sont manipulées, est parfaitement adapté à ce type de problème.

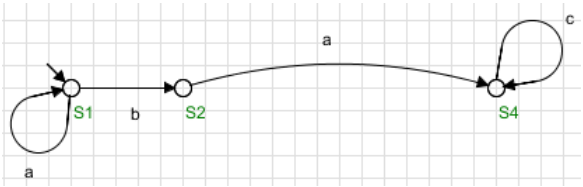
2.1 Modélisation des SEDs

Soit un système modélisé par un automate à états finis $G = (\mathrm{Q}, \Sigma, \delta, q_0)$ avec Q l'ensemble des états composant le modèle, Σ l'ensemble fini des événements avec deux sous-ensembles Σ_o : l'ensemble des événements observables, et Σ_{uo} : celui des événements non observables tel que $\Sigma = \Sigma_o \cup \Sigma_{uo}$. δ est la relation de transition $(q, \sigma, q') \in Q \times \Sigma \times \mathrm{Q}$ avec $q' \in \delta(\mathrm{q}, \sigma)$ et enfin $q_0 \in Q$ est l'état initial.
On peut dénoter par $\Sigma_f \subseteq \Sigma_{uo}$ l'ensemble des événements fautifs à diagnostiquer. L'ensemble des fautes Σ_f forme une partition de m sous-ensembles de fautes $\Sigma_f = \Sigma_{f1} \cup \Sigma_{f2} \cup \ldots \cup \Sigma_{fm}$ notée $\Pi_f = \{\Sigma_{f1}, \ldots, \Sigma_{fm}\}$, avec $m \geq 1$.

Un diagnostiqueur étant un modèle des événements observables du système, il convient alors de définir la fonction de projection sur un modèle. On définit donc la fonction de projection sur l'ensemble des événements observables par $P: \Sigma^* \to \Sigma_o^*$, i.e, P retire tous les événements non observables d'une séquence d'événements donnés. De manière générale, $P(\sigma) = \sigma$ si $\sigma \in \Sigma_o$ et $P(\sigma) = \epsilon$ si $\sigma \in \Sigma_{uo}$; $P(s.\sigma) = P(s)P(\sigma)$ avec $s \in \Sigma^*$et $\sigma \in \Sigma$. Par déploiement, la fonction de projection aux langages est $P(\mathrm{L}) = \{\mathrm{P(s)}/s \in \mathrm{L}\}$ avec L le langage régulier clos par préfixe $L \subseteq \Sigma^*$ produit par le modèle G. La figure 1 donne l'effet de la projection P avec : $\Sigma = \{a, b, c, uo, f1\}, \Sigma_o = \{a, b, c\}, \Sigma_{uo} = \{uo, f1\}, \Sigma_f = \{f1\}$



Modèle G



Projection observable du modèle G

Figure 1. Exemple de la projection observable du modèle G

Par ailleurs, une fonction de composition synchrone de deux modèles G_1 et G_2 est définie par $G_1 \| G_2 = (\mathrm{Q}_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta, q_1 \times q_2)$. Elle permet notamment d'obtenir l'évolution du modèle du système global.

2.2 Définition de la Diagnosticabilité

La notion de diagnosticabilité d'un système a été proposée sur la base de la construction d'un diagnostiqueur global dans (Sampath, 1995). Cette vérification consiste à évaluer les deux conditions nécessaires suivantes pour que le diagnostiqueur soit diagnosticable :
C1. Il existe au moins un état du diagnostiqueur pour lequel le diagnostiqueur décide avec certitude l'occurrence d'une faute appartenant à une partition Π_{fi}.

C2. Il ne doit pas y avoir de cycles dits indéterminés pour lesquels le diagnostiqueur est incapable de décider avec certitude l'occurrence d'une faute appartenant à une partition Π_{fi}.

Ainsi, il est possible de savoir si un système est diagnosticable avec certitude, incertitude ou ambiguïté. Ci-après nous rappelons les notions relatives à une faute (Fi-certain, Fi-incertain, Fi-ambigu).

Définition 1. (Fi-certain dans G)
Un état q est Fi-certain si toute trace de q_0 à q contient obligatoirement la faute Fi.

Définition 2. (Fi-incertain dans G)
Un état du système est Fi-incertain s'il y a deux traces s_1 et s_2 partant de q_0 jusqu'à respectivement q_1 et q_2 et ayant la même projection observable tel que s_1 contienne Fi alors que s_2 ne contienne pas Fi et que dans le système

initial G, les traces s_1 et s_2 mènent vers deux états différents (q_1, q_2).

Définition 3. (Fi-ambigu dans G)
Un état du système est Fi-ambigu s'il y a deux traces s_1 et s_2 partant de q_0 jusqu'à q_1 et ayant la même projection observable tel que s_1 contienne Fi alors que s_2 ne contienne pas Fi et que dans le système initial G, les traces s_1 et s_2 mènent vers un même état (q_1).

De plus, il convient d'admettre les deux hypothèses suivantes :

- L'automate *G* est vivant c'est à dire qu'à partir de chaque état q de G, il existe au moins une transition franchissable ;
- Il n'y a pas de cycle contenant uniquement des événements non-observables.

L'absence de cycles d'événements inobservables permet de garantir que l'on dispose d'observations régulières pour observer les effets d'une faute. L(G) est donc diagnosticable par rapport à la projection P : $\Sigma \rightarrow \Sigma_o$ et par rapport à la partition Π_f de Σ_f, si la condition suivante est satisfaite (éq. 1):

$(\forall i \in \Pi_f)(\exists ni \in \mathbb{N})[\forall s \in \Psi(\Sigma_{fi})](\forall t \in L/s) : [||t|| \geq ni$

$\Rightarrow \omega \in P^{-1}L(G)\,[P(st)] \Rightarrow \Sigma_{fi} \in \omega]$ (1)

Avec :

- $\Psi(\Sigma_{fi})$ l'ensemble des traces se terminant par une faute de Σ_{fi};
- L/s l'ensemble des séquences suffixes de s dans L(G) ;
- ||t|| la longueur de la séquence t ;
- P^{-1}L(G) [P(u)] toutes les traces v dans L(G) telles que P(v)=u.

$\Psi(\Sigma_f)$ est l'ensemble des séquences finies d'événements qui se terminent par un événement fautif (de Σ_{fi}). Cette définition signifie qu'un langage L est diagnosticable si et seulement si, pour toute séquence 's' se terminant par un événement fautif et toute suite d'événements 't' de 's' suffisamment longue (|t| ≥ ni), alors toute autre séquence d'événements 'ω' ayant la même projection observable que 's.t (P(s.t) = P (ω))', contient nécessairement un événement fautif de Σ_f.

Propriété 1. (Cycle dans G)
Soit CL l'ensemble des cycles cl possibles dans un système G tel que : $(q \in Q, \sigma \in \Sigma)$
$cl_1 = (q_0, \sigma_0, q_1, \sigma_1, \dots, q_n, \sigma_n, q_m)$ avec $m[0,n]$
$cl_2 = (q_0, \sigma'_0, q'_1, \sigma'_1, \dots, q'_p, \sigma'_p, q'_o)$ avec $o \in [0,p]$

Pour vérifier la diagnosticabilité dans le modèle G du système il convient de vérifier la non-présence de deux cas de figure :

i. Le système G est non diagnosticable s'il existe un cycle cl_1 dans G avec Fi-certain, un autre cycle cl_2 ne contenant pas la faute Fi et ayant la projection observable (éq. 2).

$(\exists\, cl_1 \subseteq CL\,/\exists(q_j,\sigma_j) \in L: \sigma_j \in F_i)(\exists\, cl_2 \subseteq CL\,/\forall(q_k,\sigma_k) \in L: \sigma_k \in \Sigma_o)/P(cl_1) = P(cl_2)\ avec\{j \in [0,n],\ k \in [0,p]\}$ (éq. 2).

ii. Le système G est non diagnosticable s'il existe un cycle cl_1 dans G avec Fi-certain, un autre cycle cl_2 avec Fj-certain (Fi≠Fj) et cl_1 et cl_2 partageant la même projection observable (éq. 3)

$(\exists\, cl_1 \subseteq CL\,/\exists(q_j,\sigma_j) \in L: \sigma_j \in F_i)(\exists\, cl_2 \subseteq CL\,/\exists(q_k,\sigma_k) \in L: \sigma_k \in F_{s\neq i})/P(cl_1) = P(cl_2)\ avec\{j \in [0,n], k \in [0,p]\}$ (éq. 3).

2.3 Analyse de la diagnosticabilité selon Sampath et Jiang

(Sampath et al., 1995) et (Jiang et al., 2001) ont proposé chacun un algorithme de vérification de la diagnosticabilité d'un système G (fig. 2), sur le modèle du diagnostiqueur.
Dans la première approche, l'auteur effectue une projection du système G pour obtenir un modèle observateur G' déterministe. A partir de là, et en associant un label à chaque état (N : normal, Fi : Fautif i), un diagnoser G_d est obtenu. L'algorithme de recherche de cycles indéterminés est appliqué sur G_d. L'algorithme de vérification est exponentiel par rapport au nombre d'états du système G et doublement exponentiel par rapport au nombre de types de fautes. Le risque d'explosion combinatoire peut apparaître dans cette étape.
Dans la seconde approche la construction du modèle intermédiaire « vérificateur » de (Jiang et al., 2001) est obtenue par composition synchrone du modèle observateur labellisé G_o avec lui-même « twin-plant » (G_v=G_o||G_o). Cette approche présente donc un risque d'explosion de l'espace d'états dès la construction du vérificateur. L'algorithme de vérification consiste alors à chercher l'absence ou non de cycle *cl* avec (Fi ≠ Fj) dans G_v. Il est donc polynomial par rapport au nombre d'états de G et exponentiel par rapport au nombre de types de fautes.

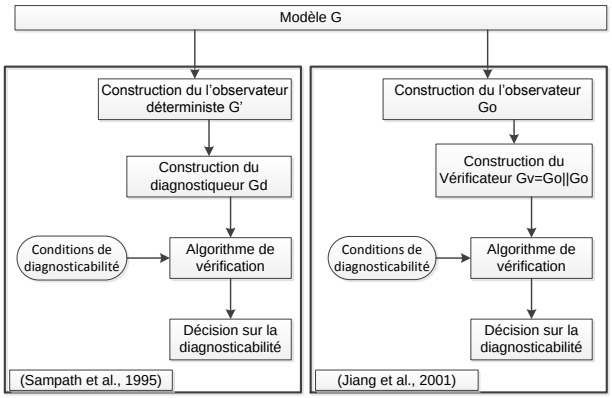


Figure 2. Etapes des approches de Sampath et Jiang

L'approche proposée consiste à éviter la construction de modèles intermédiaires et par ailleurs les compositions d'automates en utilisant des instances de classe du modèle G. La première instanciation G_0 permettra notamment de rechercher un cycle indéterminé et de la comparer son observation à un autre cycle indéterminé issu de la seconde instanciation G_1. Ainsi, sans composition parallèle, le Model-Checker permettra de vérifier par équivalence en termes de parcours de graphe les conditions de diagnosticabilité (Fig. 3). L'approche est détaillée dans la partie 3.2 mais auparavant il convient de rappeler quelques principes sur ce dernier.

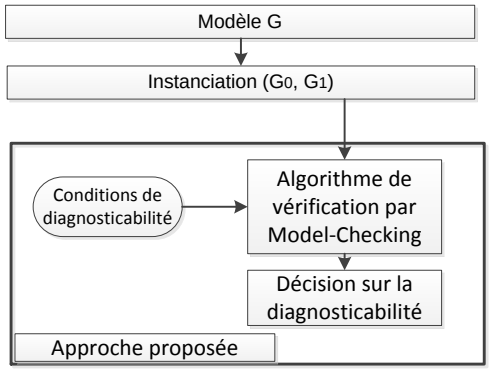


Figure 3. Structure de notre approche

2.4 Principe du Model-Checking

Le Model-Checking est une méthode algorithmique formelle généralement mis en opposition avec le test et la simulation (Schnoebelen *et al.*, 1999). Au même titre que l'« Automated Theorem Proving » basée sur le raisonnement automatisé et qui tente de prouver un ensemble limité de théorèmes mathématiques par des programmes informatiques, le Model-Checking permet de vérifier formellement qu'un système à états fini modélisé satisfait une spécification logique. Ces méthodes peuvent être automatisées et parcourir exhaustivement tous les comportements mais impliquent que le modèle et la spécification soient correctement exprimés. Le Model-Checking nécessite l'utilisation de :

- Un modèle *M* du système,
- Une spécification φ des propriétés à vérifier,
- Un algorithme vérifiant automatiquement si le modèle satisfait la formule noté $M \models \varphi$.

Le modèle considéré peut être exprimé au moyen d'un graphe orienté de type automates à états (Kripte, 1963). La spécification peut se formuler en logique temporelle (LTL : Logique Temporelle Linéaire, CTL : Computation Tree Logic ou structure de Kripke) (Clarke, Emerson and Sistle, 1986). LTL et CTL sont des logiques modales où la notion de vérité dépend de l'évolution du monde, c'est-à-dire qu'une proposition peut-être, à un moment, fausse puis, plus tard, devenir vraie. Elles sont définies sur un ensemble de propositions atomiques et combinées par un certain nombre de connecteurs logiques : non, ou, et, implications, vrai et faux ($\neg$, $\vee$, $\wedge$, $\Rightarrow$, $\Leftrightarrow$, *true*, *false*), ainsi que d'autres opérateurs temporels appelés « modalités » ou « quantificateurs ». L'ensemble permettant d'exprimer des formules exprimant l'atteignabilité, la sécurité ou la vivacité d'un système (E: possibly exists a path, A: Inevitably for all paths, F: some state in a path, and G: all states in a path).

3 VERIFICATION DE LA DIAGNOSTICABILITE PAR MODEL CHECKING

Afin d'éviter la construction de modèle intermédiaire, nous proposons d'exploiter la recherche de propriétés sur le modèle G par Model-Checking.
Une condition de la diagnosticabilité définit dans (Sampath, 1995) étant la présence d'un cycle indéterminé à base des événements non observables dans le système G, nous proposons de rechercher l'ensemble des cycles (avec ou sans faute), le choix des cycles revient au Model-Checker, car pour l'instant aucune condition de choix n'est imposée dans la recherche des cycles. Dans les travaux futurs nous prévoyons un choix spécifique à cette recherche notamment sur la seconde instanciation G_1. Une fois les deux cycles trouvés, une comparaison est mise en place, il faut alors analyser à la fois si le cycle possède une faute dans sa séquence, mais aussi si celui-ci est distinguable des autres avec certitude. La Fig. 4 illustre les différentes étapes de la méthodologie proposée :

- Modélisation et Instanciation du système
- Recherche de deux cycles
- Analyse de la non-diagnosticabilité

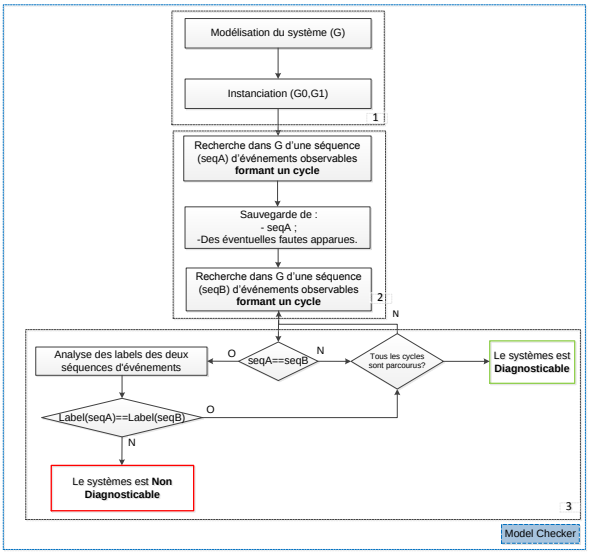


Figure 4. Logigramme de la solution proposée.

3.1 Modélisation et Instanciation

L'étape de modélisation consiste à donner une interprétation abstraite du système. Au même titre que (Sampath, 1995) ou (Jiang et al., 2001), ce modèle G est représenté comme à automate à états finis comme définit dans la section 2. Cependant, afin de décomposer l'information contenue dans une transition, nous allons interpréter chaque événement par un ensemble de labels {P[i], E[j], F[k]} avec :

- P[i] : mémorisation de l'état atteint i,
- E[j] : mémorisation de l'événement et de son observation (j=0 : événement non-observable, j>0 : événement observable),
- F[k] : mémorisation du type de faute (k=0 : aucune faute, k>0 : faute f_k)

Afin de chercher différents cycles et de pouvoir les comparer par la suite, nous proposons d'instancier le modèle G deux fois (G_0 et G_1). Ces deux automates sont donc identiques mais n'évolueront pas en même temps dans notre vérification.

3.2 Recherche d'un cycle dans G_0 et G_1

Le model-checker va commencer par parcourir le modèle G_0 en enregistrant les séquences d'événements à partir de son état initial. C'est cet enregistrement qui va lui permettre de vérifier à chaque occurrence si un cycle d'événements apparaît ou non. Lorsque celui-ci trouve une séquence observable (seqA) avec un cycle cl_1, une sauvegarde de ses informations est effectuée à travers la mémorisation des états visités et des fautes apparues. La seconde recherche peut alors débuter, mais cette fois-ci sur le second modèle G1. De la même façon, cette séquence observable (seqB) avec un cycle cl_2 est sauvegardée et l'ensemble peut être analysé.

3.3 Analyse de la non-diagnosticabilité

L'analyse commence par vérifier si les deux séquences sont identiques en termes d'événements observables. Si les 2 séquences ne disposent pas de la même observation $P(cl_1) \neq P(cl_2)$, il convient alors au model-checker de rechercher l'existence dans G_1 d'un autre cycle qui possèderait la même observation. S'il n'en trouve pas, alors le système sera considéré comme diagnosticable sous les hypothèses de la section 2.2.

Dans le cas où les 2 séquences observables sont identiques $P(cl_1) = P(cl_2)$, les deux labels doivent alors être analysés. Si les deux labels sont égaux (Label(seqA)) = (Label(seqB)), cela signifie qu'on est en train d'analyser la même trace/chemin et qu'il faut rechercher alors une autre séquence observable avec cycle. Par contre, si les deux labels sont différents (Label(seqA)) ≠ (Label(seqB)), alors cela signifie que deux séquences disposent de la même observation mais possèdent pour l'une une faute et pour l'autre non ou une faute différente. Le model-checker répondra donc par la non-diagnosticabilité du système.

Cette phase d'analyse se base sur la mémorisation des fautes (ou non) du cycle cl_1 et des fautes (ou non) du cycle cl_2. Pour cela on effectue deux opérations : le produit ($Pdtc_i$) et la somme (Smc_i) de toutes les fautes apparues dans les cycles tel que:

$$cl_1 \quad \begin{cases} Pdtc_1 = \prod_{i=1}^{m} (F_i) \,/\, (F_i \in Cycle1) \\ Smc_1 = \sum_{i=1}^{m} (F_i) / \, (F_i \in Cycle1) \end{cases} \tag{4}$$

$$cl_2 \quad \begin{cases} Pdtc_2 = \prod_{i=1}^{m} (F_i) \,/\, (F_i \in Cycle2) \\ Smc_2 = \sum_{i=1}^{m} (F_i) / \, (F_i \in Cycle2) \end{cases} \tag{5}$$

Les produits ($Pdtc_1$, $Pdtc_2$) et les sommes (Smc_1, Smc_2) jouent le rôle de décision sur le label de faute. Comparer *Smc1* à *Smc2* et *Pdtc1* à *Pdtc2* pour arriver à un des deux cas possibles :

- Si (*Pdtc1= Pdtc2* **ET** *Smc1= Smc2*) alors les labels de faute sont les mêmes entre les deux cycles trouvés et par conséquent les deux cycles ont la même ou les mêmes partitions de fautes,
- Si (*Pdtc1≠ Pdtc2*) **OU** (*Smc1≠ Smc2*) alors les deux cycles ont deux partitions de fautes différentes et donc le système G est Non-diagnosticable.

Cette recherche de 2 cycles dans 2 instances du modèle G n'est pas une composition synchrone à l'image de (Jiang et al., 2001) mais implique au pire le parcours

d'autant d'états pour la recherche de cycles par le model-checker.

4 EVALUATION EXPERIMENTALE

Pour évaluer et comparer notre approche, nous prenons l'exemple illustré par la figure 5 tiré des travaux de (Sampath et al., 1995). Les sous-ensembles des événements sont $\Sigma_o = \{a, b, c\}$, $\Sigma_{uo} = \{uo, f_1\}$, $\Sigma_f = \{f_1\}$.

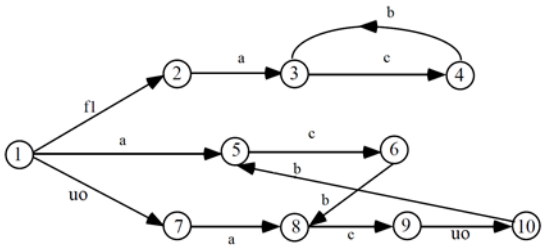


Figure 5. Modèle G de l'exemple

4.1 Diagnosticabilité de (Sampath et al., 1995)

Pour construire le diagnostiqueur de l'exemple, nous appliquons la projection observable pour obtenir l'observateur G' (fig. 6), puis nous rendons déterministe et nous labélisons les états par (N, F) pour obtenir le diagnostiqueur de la figure 7.

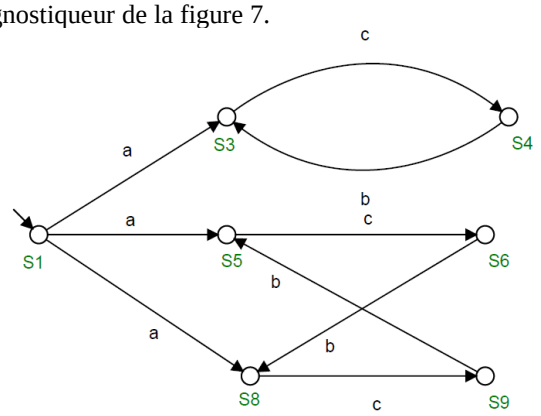


Figure 6. Observateur G' de G

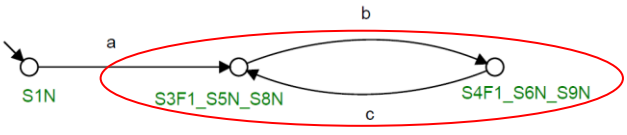


Figure 7. Diagnostiqueur G_d de G

D'après l'analyse de la figure 7, nous remarquons qu'il existe un cycle caractérisé par l'incertitude à cause de la présence d'une boucle avec l'étiquette (N et F). La faute F1 est indéterminée et par conséquent le système G est non diagnosticable.

4.2 Diagnosticabilité de (Jiang et al, .2001)

La méthode de (Jiang et al., 2001) consiste à construire dans un premier temps le pseudo-diagnostiqueur G_o labélisé (fig. 8). Dans un second temps, le vérificateur est obtenu en utilisant le « twin plant » du pseudo-diagnostiqueur ($G_v = G_o || G_o$) illustré en figure 9.

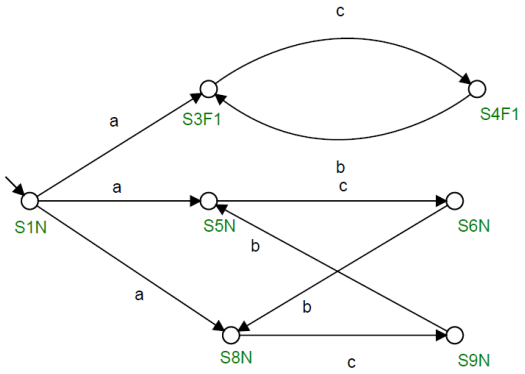


Figure 8. Pseudo-diagnostiqueur G_o de G

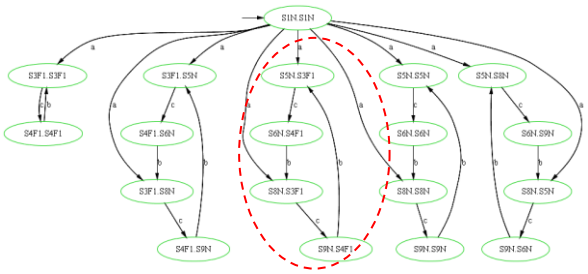


Figure 9. Vérificateur G_v de G

D'après le vérificateur G_v, nous constatons la présence d'un cycle incertain « offending cycle », et par conséquent le modèle G est bien non diagnosticable.

4.3 Diagnosticabilité par Model-Checking

Pour illustrer notre approche par Model-Checking, nous avons utilisé le model-checker UppAal (Behrmann et al., 2006). Pour la première étape de modélisation, il convient d'interpréter les événements du modèle initial par un ensemble de labels {P[i], E[j], F[k]}. Il en résulte la table 1. Le modèle est ensuite obtenu sous la forme de la figure 10. Celui-ci est alors instancié 2 fois dans le model-checker (G0 et G1).

Type d'événement	Evénement	Indice
Observables	a	E=1
	b	E=2
	c	E=3
Non Observables	uo	E=0
	f1	E=0 et F=1

Table 1. Affectation événement-indice

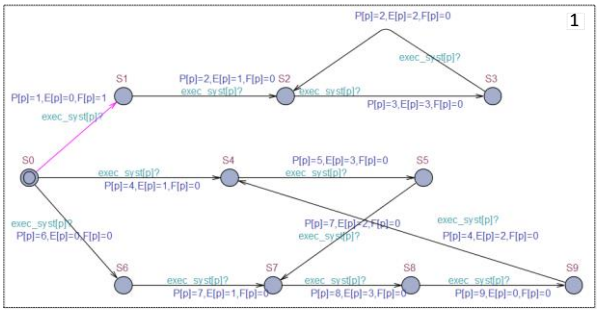


Figure 10. Modélisation sous Uppaal du modèle G

Afin d'exécuter automatiquement la deuxième étape de recherche de cycles de l'algorithme proposé dans la figure 4, un modèle de synchronisation est nécessaire (fig. 11). Cet automate commence par la recherche d'un premier cycle dans G_0 par une boucle :

- L'évolution de l'automate G_0 par synchronisation (*exec_var[ ]*),
- Appel d'une fonction de recherche de cycle avec mémorisation de la séquence (*evol( )*).

Une fois le cycle trouvé (*cycle ==1*) et sauvegardé (*save_init( )*), l'automate passe à la recherche d'un second cycle sur G_1 par une deuxième boucle sur le même principe de la première (zone 2 de la figure 11).
La troisième étape d'analyse (zone 3 de la figure 11) consiste à appeler une fonction de calcul des équations 4 et 5 (*cal_pdt_somme( )*) qui permettra ensuite de comparer l'observation des cycles et l'analyse des labels de fautes par une fonction (*test_NDiag( )*). Si la variable NDiag==1, alors cela signifie que nous avons deux cycles avec la même projection observable mais qu'ils n'ont pas le même label de faute.

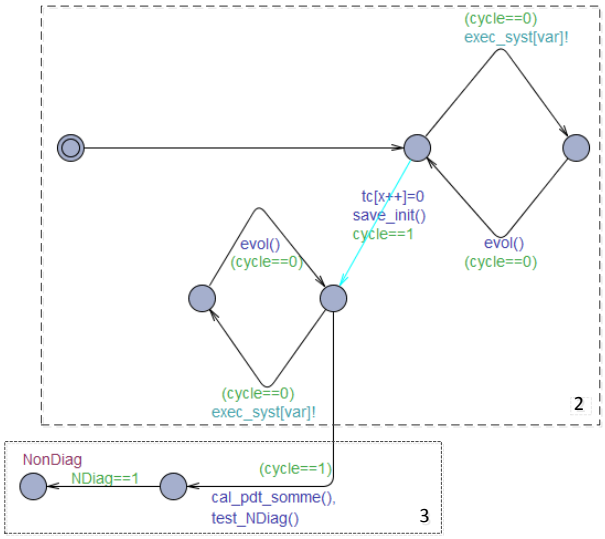


Figure 11. Modèle automate à états de synchronisation et contrôle (Synch)

Pour effectuer la vérification, nous avons implanté nos modèles dans le model-checker UppAal et nous avons vérifié s'il existait un chemin permettant d'atteindre la localité Nondiag du modèle de la figure 11. Cette propriété est notée dans la logique CTL : EF Nondiag.

Dans l'exemple étudié, le model-checker nous retourne une trace vérifiant la possibilité d'atteindre la localité NonDiag. Cette trace nous permet de connaître les 2 cycles qui ont la même séquence d'événements observables mais qui ne remontent pas la même détection de fautes. Le modèle G est donc bien non-diagnosticable, ce qui est cohérant avec les deux autres approches.

4.4 Discussion

Concernant l'analyse de la diagnosticabilité, nous remarquons que les deux approches nécessitent la construction d'un modèle intermédiaire alors que l'approche proposée dans ce papier ne l'exige pas. La présence de plusieurs types de fautes rend la vérification de la diagnosticabilité plus complexe pour les approches classiques, par contre notre approche n'est pas influencée. L'originalité de notre travail est d'utiliser le model-checking comme un outil pour vérifier l'atteignabilité, et de travailler sur le modèle G directement.

Dans les approches de Sampath et Jiang, la complexité se trouve dans la modélisation du modèle intermédiaire et peut engendrer une explosion combinatoire. Dans notre cas, cette complexité ne se trouve pas dans la modélisation mais dans la recherche des cycles dans G_0 et G_1. La complexité de recherche est liée au nombre de cycles dans G. Actuellement nous n'avons pas évalué cette complexité et cela fera l'objet de travaux futurs.

5 CONCLUSION

Cet article présente une formulation de la vérification de la diagnosticabilité dans le cadre du model-checking. Nous avons proposé un algorithme pour vérifier la diagnosticabilité directement à partir du modèle du système. Cet algorithme permet notamment de tenir compte de plusieurs fautes.
Ce début de travail de recherche a permis une première comparaison avec deux approches reconnues dans la littérature. Cette étude devra être poursuivie pour montrer la possibilité de passer sur des systèmes de plus grande taille et pour comparer nos approches en termes de temps de calcul et d'espace d'états explorés.
Une amélioration à court terme est de diminuer l'espace d'états explorés de notre algorithme en vérifiant l'équivalence des séquences d'événements dès le début de la recherche du second cycle. Ainsi, le modèle ne parcourra que les séquences équivalentes en termes d'événements observables. A plus long terme, nous aimerions utiliser les contre-exemples remontés par le model-checker pour proposer des ajouts d'informations sur le système et ainsi proposer une aide à la conception du diagnostiqueur.

REFERENCES

Behrmann, G., David, A., Larsen, K.G., Hakansson, J., Petterson, P., Yi, W., Hendriks, M.: Uppaal 4.0. In: Third International Conference on the Quantitative Evaluation of Systems - (QEST'06). pp. 125–126 (Sept 2006)

Boussif, A., and Ghazel, M. (2017). A diagnoser-based approach for intermittent fault diagnosis of discrete-event systems. In 2017 American Control Conference (ACC), pp. 3860–3867.

Cassandra C.G. and Lafortune S. (1999). Introduction to discrete event systems. Kluwer Academic Publishers, Dordrecht, 1999.

Cassez, F., and Tripakis, S. (2008). Diagnostic des systèmes temporisés. 145--176.

Contant, O., Lafortune, S., and Teneketzis, D. (2004). Diagnosis of modular discrete event systems1. IFAC Proc. Vol. *37*, 327–332.

Debouk, R., Lafortune, S., and Teneketzis, D. (2000). Coordinated Decentralized Protocols for Failure Diagnosis of Discrete Event Systems. Discrete Event Dyn. Syst. *10*, 33–86.

Jiang, S., Huang, Z., Chandra, V., and Kumar, R. (2001). A polynomial algorithm for testing diagnosability of discrete-event systems. IEEE Trans. Autom. Control *46*, 1318–1321.

Pencolé, Y. (2004). Diagnosability Analysis of Distributed Discrete Event Systems. In Proceedings of the 16th European Conference on Artificial Intelligence, (Amsterdam, The Netherlands, The Netherlands: IOS Press), pp. 38–42.

Qiu, W., and Kumar, R. (2006). Decentralized failure diagnosis of discrete event systems. IEEE Trans. Syst. Man Cybern. - Part Syst. Hum. *36*, 384–395.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. (1995). Diagnosability of discrete-event systems. IEEE Trans. Autom. Control *40*, 1555–1575.

T. Yoo and S. Lafortune. Polynomial-time verification of diagnosability of partially-observed discrete-event systems. IEEE Trans. on Automatic Control, 47(9):1491–1495, 2002.