Diagnostic préférentiel à base de modèles discrets pour des fautes intermittentes et permanentes

Valentin Bouziat, Xavier Pucel, Stéphanie Roussel, Louise Travé-Massuyès

Diagnostic préférentiel à base de modèles discrets pour des fautes intermittentes et permanentes

Valentin Bouziat[1] **Xavier Pucel**[1] **Stéphanie Roussel**[1] **Louise Travé-Massuyès**[2]

[1] ONERA, Toulouse, France

[2] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France

`prénom.nom@onera.fr` `louise@laas.fr`

Résumé

Nous nous intéressons dans cet article au diagnostic de pannes intermittentes et permanentes dans des systèmes à événements discrets. Nous proposons un formalisme de modélisation logique associé à des préférences conditionnelles destiné à produire un diagnostic unique à chaque pas de temps. Cette approche est susceptible de provoquer des interblocages entre le système surveillé et son diagnostiqueur. L'objet de cet article est de détecter ces interblocages dès la conception, par une méthode basée sur le model-checking.

Abstract

In this paper we consider the diagnosis of intermittent and permanent faults in discrete event systems. We present a logic based modeling approach associated with conditional preferences in order to produce a unique diagnosis at each time step. This approach is subject to deadlocks between the monitored system and its diagnoser. The goal of this paper is to detect at design time such deadlocks, using a model-checking approach.

1 Introduction

Dans de nombreuses applications, les robots ne peuvent être téléopérés et nécessitent un certain degré d'autonomie décisionnelle. Cette autonomie requiert notamment de détecter et de réagir de manière appropriée à des baisses anormales de performance, dûes à des dégradations du système robotique, ou à des perturbations extérieures. Nous nous intéressons donc à la détection automatique et en temps réel de ces aléas pouvant survenir dans un système autonome tel qu'une flotte de robots par exemple.

Nous proposons un formalisme de modélisation construit de telle sorte qu'il répond à trois exigences en termes de diagnostic. Premièrement, il calcule de manière incrémentale un diagnostic unique à chaque pas de temps. Cette exigence découle de contraintes sur les performances du système : on ne peut se permettre de traiter tous les diagnostics possibles sur des problèmes réels de grande taille pendant une longue période. De plus, si on considère un système robotique complet, le module de diagnostic est dans notre cas associé à un planificateur déterministe qui attend un diagnostic unique en entrée. Deuxièmement, on souhaite que notre formalisme prenne en compte à la fois les fautes permanentes et intermittentes pour notamment différencier les perturbations (bruit, faux-contact, etc..) des dégradations (pannes, etc..). Troisièmement, puisque le diagnostic est utilisé pour prendre des décisions autonomes, on s'interdit de revenir en arrière pour modifier un diagnostic précédemment émis. On s'interdit également de produire une séquence de diagnostics qui ne correspond pas à une évolution possible dans le système (par exemple faire disparaître une faute permanente). Le principe est de n'émettre comme diagnostic que des informations fiables, à partir desquelles le robot pourra prendre une décision autonome sûre.

Nous détaillons dans l'article comment de telles exigences entraînent des situations dans lesquelles le système robotique produit une observation que le diagnostiqueur est incapable d'expliquer. Dans ce genre de situation, que nous appelons un *interblocage*, toute la chaîne de décision supportée par le diagnostiqueur est inopérante. C'est pourquoi nous nous intéressons à la détection des interblocages dès la phase de conception du système.

Pour détecter ces scénarios d'interblocage, nous commençons par présenter l'état de l'art en section 2. Nous présentons ensuite notre formalisme de modélisation en section 3, puis nous décrivons le problème de l'interblocage en section 4. Nous proposons une méthode de vérification en phase de conception utilisant le model-checker Electrum [13] en section 5, dont les tests sur des exemples de systèmes multi-robots sont décrits en section 6. Nous décrivons les perspectives de ce travail en section 7.

2 Travaux connexes

Nous nous plaçons dans le cadre du diagnostic de systèmes à événements discrets. Dans [16] les auteurs définissent un cadre pour le diagnostic de systèmes modélisés par des automates à états finis sujets à des fautes permanentes. Les auteurs décrivent la construction d'un diagnostiqueur qui permet un calcul incrémental du diagnostic. Toutefois, leur diagnostiqueur nécessite de mémoriser l'ensemble des paires état-diagnostic possibles, ce qui pose de graves problèmes de passage à l'échelle. Leur définition de la diagnosticabilité illustre l'importance de garantir dès la conception les performances d'un diagnostiqueur.

Notre formalisme associe des contraintes de logique propositionnelle à une théorie de préférences conditionnelles issue de [1]. Il est inspiré du formalisme utilisé dans [14] pour produire des diagnostics incrémentaux. Dans ces travaux les auteurs décrivent le risque d'interblocage mais ne proposent pas de solution. Dans [15], ils tentent de détecter les scénarios d'interblocage entre système et diagnostiqueur par une approche de model-checking itératif difficile à mettre en œuvre et sans expérimentation associée.

Dans la littérature, d'autres travaux traitent également du problème de l'interblocage (deadlock) entre système et diagnostiqueur. Certains reviennent en arrière dans l'exécution du diagnostiqueur afin de retrouver un branchement cohérent [11]. D'autres autorisent le diagnostiqueur à emprunter des transitions qui ne respectent pas la dynamique temporelle du système par exemple en réinitialisant le diagnostiqueur [8]. Dans les deux cas, nous excluons ces solutions pour les raisons d'interaction avec le planificateur expliquées précédemment.

Une manière classique d'ordonner les diagnostics est de préférer les diagnostics minimaux, cette démarche connaît plusieurs variantes comparées dans [5]. D'autres approches comme [7] ordonnent les fautes selon un critère quelconque, et en déduisent un ordre sur les diagnostics. Dans toutes ces approches, l'ordre des diagnostics est inconditionnel, c'est à dire que les observations n'ont pas d'effet sur l'ordre des diagnostics. À l'inverse, notre modèle utilise des préférences conditionnelles précisément afin de lever cette limitation, et permet de privilégier certains diagnostics en fonction des observations présentes et passées.

Le diagnostic des fautes intermittentes est abordé dans la littérature, sous différents angles. Dans [6], la même fonction est rejouée de multiples fois, mais si les fautes se manifestent de manière intermittente, elles n'évoluent pas d'un test à l'autre. Dans [4], un événement de réparation est associé à chaque faute, et le diagnostic consiste à détecter pour chaque faute quel événement (faute ou réparation) est arrivé en dernier. Un diagnostiqueur est construit pour permettre une évaluation incrémentale, qui reproduit les problèmes de passage à l'échelle mentionnés précédemment puisque tous les diagnostics sont gardés en mémoire.

3 Modèle de diagnostic

Notre modèle de diagnostic est un tuple (s_0, Δ, Γ) dans lequel s_0 est l'état initial du système, Δ le modèle comportemental du système et Γ le modèle de préférences conditionnelles. Nous supposons que le système évolue suivant une dynamique discrète et que tous les pas de temps ont la même durée.

3.1 Variables

Nous utilisons un ensemble de variables propositionnelles $\mathtt{P}$ afin de décrire l'état du système. $\mathtt{P}$ est partitionné en deux sous-ensembles $\mathtt{O}$ et $\mathtt{E}$, qui représentent respectivement les éléments observables du système et ceux qui doivent être estimés. À chaque pas de temps discret, connaissant la valeur des variables de $\mathtt{O}$ on cherche à estimer la valeur des variables de $\mathtt{E}$.

Notations Pour un ensemble de variables $\mathtt{X}$, on définit une *affectation* comme une fonction de $\mathtt{X}$ vers $\{\top, \bot\}$ qui associe à chaque variable $\mathtt{x}$ de $\mathtt{X}$ la valeur booléenne vrai ($\top$) ou faux ($\bot$). On note x (resp. $\overline{x}$) l'affectation qui à $\mathtt{x}$ associe $\top$ (resp. $\bot$). Pour un ensemble variables $\mathtt{X} = \{\mathtt{x_1},..,\mathtt{x_n}\}$ on note $f_1 \dots f_n$ avec $f_i : \{x_i\} \rightarrow \{\top, \bot\}$ l'affectation f telle que $\forall \mathtt{x_i} \in \mathtt{X}, f(\mathtt{x_i}) = f_i(\mathtt{x_i})$. Par exemple, $a\ \overline{b}\ \overline{c}$ est une affectation qui associe vrai à $\mathtt{a}$ et faux à $\mathtt{b}$ et $\mathtt{c}$.

Définition 1 (Observation) *Une observation, notée o, est une affectation des variables de* $\mathtt{O}$.

Définition 2 (État) *Un état, noté s, est une affectation des variables de* $\mathtt{P}$. *On note* $\mathcal{S}$ *l'ensemble des états.*

3.2 Modèle comportemental

Le modèle comportemental Δ décrit la relation de transition du système. Pour faire référence à l'état précédent du système, on utilise l'ensemble P auquel on associe une fonction bijective *pre* permettant d'accéder à la valeur d'une variable de $\mathtt{p}$ à l'instant précédent. Formellement, pour chaque état s de $\mathcal{S}$ on définit un ensemble de variables $\mathtt{P_{pre}} = \{\mathtt{pre\_p} \mid \mathtt{p} \in \mathtt{P}\}$ tel que $\forall \mathtt{p} \in \mathtt{P}, pre(\mathtt{p}) = \mathtt{pre\_p}$. Δ est représenté par un ensemble Δ_p de formules de logique propositionnelle pouvant porter à la fois sur les variables de $\mathtt{P}$ et celles de $\mathtt{P_{pre}}$.

Une paire d'état (s_{pre}, s_{now}) appartient à la relation de transition Δ lorsque le système peut être dans l'état s_{pre} à l'instant $t-1$ et dans l'état s_{now} à l'instant t. Pour lier formellement Δ à Δ_p, pour toute paire d'états (s_{pre}, s_{now}) on définit l'affectation $\sigma_{s_{pre},s_{now}}$ portant sur les variables de $\mathtt{P} \cup \mathtt{P_{pre}}$ telle que $\forall \mathtt{p} \in \mathtt{P}, \sigma_{s_{pre},s_{now}}(\mathtt{p}) = s_{now}(\mathtt{p})$ et $\sigma_{s_{pre},s_{now}}(pre(\mathtt{p})) = s_{pre}(\mathtt{p})$. Nous considérons qu'une affectation appartient à la relation de transition Δ si et seulement si celle-ci est cohérente avec les contraintes de Δ_p.

Définition 3 (Transition) *Une paire d'états* $(s_{pre}, s_{now}) \in \mathcal{S}^2$ *est une transition, noté* $(s_{pre}, s_{now}) \in \Delta$*, si et seulement si* $\sigma_{s_{pre},s_{now}} \models \Delta_p$.

Par la suite, on confondra Δ et Δ_p.

De manière à représenter les évolutions des états du système, du diagnostiqueur ou des observations générées, nous définissons les séquences d'états et d'observations cohérentes de la manière suivante.

Définition 4 (Séquence d'états cohérente) *Une séquence d'états* $(s_0, s_1, \ldots, s_n)$ *est cohérente si et seulement si* $\forall i \in [1, n], (s_{i-1}, s_i) \in \Delta$.

Définition 5 (Séquence d'observations cohérente) *Une séquence d'observations* $(o_0, o_1, \ldots, o_n)$ *est cohérente si et seulement si il existe une séquence d'états cohérente* $(s_0, s_1, \ldots, s_n)$ *telle que* $\forall i \in [0, n], \forall \mathtt{o} \in \mathtt{O}, s_i(\mathtt{o}) = o_i(\mathtt{o})$.

Dans la suite, sauf indication contraire, les séquences d'états et d'observations sont supposées cohérentes.

Étant donné l'état précédent et une observation du système, on définit l'ensemble des candidats au diagnostic comme l'ensemble des états cohérents avec l'état précédent, les observations et le modèle comportemental Δ.

Définition 6 (Candidats au diagnostic) *L'ensemble* $\mathcal{S}_\Delta(s_{pre}, o)$ *des états candidats au diagnostic pour un état précédent* s_{pre} *et une observation* o *est défini par :*

$$\mathcal{S}_\Delta(s_{pre}, o) = \Big\{ s_{now} \in \mathcal{S} \mid \begin{array}{l} (s_{pre}, s_{now}) \in \Delta \text{ et} \\ \forall \mathtt{o} \in \mathtt{O}, s_{now}(\mathtt{o}) = o(\mathtt{o}) \end{array} \Big\}$$

Exemple 1 Le système illustré en Figure 1 est composé d'une ampoule contrôlée par un interrupteur. Ce système peut-être sujet à une faute permanente et une faute intermittente, dont nous cherchons à estimer la présence ou l'absence.

L'ensemble $\mathtt{O}$ est composé de `light` qui est vrai lorsqu'on observe de la lumière, faux sinon, et `switch` qui est vrai lorsque l'interrupteur est en position fermée (le courant passe), faux sinon. L'ensemble $\mathtt{E}$ contient deux variables représentant les deux fautes pouvant survenir : $\mathtt{f}_{\mathtt{wire}}$ représente une faute intermittente (elle peut disparaître au cours du temps) et $\mathtt{f}_{\mathtt{light}}$ une faute permanente, qui peut par exemple nécessiter une intervention (remplacer le filament/l'ampoule) pour disparaître.

Les deux règles de Δ représentent le comportement suivant : l'ampoule éclaire lorsque l'interrupteur est en position basse, qu'il n'y a pas de faute sur le fil ni sur le filament de l'ampoule (δ_1). Si le filament de l'ampoule était rompu à l'état précédent, alors il l'est à l'instant présent (δ_2), autrement dit la faute $\mathtt{f}_{\mathtt{light}}$ est permanente. L'état initial est $s_0 = \mathit{light}\ \mathit{switch}\ \overline{f_{light}}\ \overline{f_{wire}}$ dans lequel la lumière est allumée et aucune faute n'est présente.

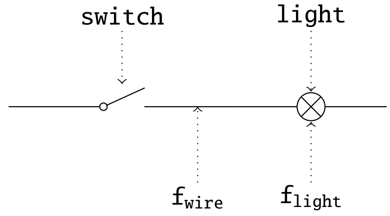

FIGURE 1 – Un système composé d'une ampoule et d'un interrupteur

$$\Delta = \left\{ \begin{array}{lr} \mathtt{light} \leftrightarrow \mathtt{switch} \wedge \neg \mathtt{f}_{\mathtt{light}} \wedge \neg \mathtt{f}_{\mathtt{wire}} & (\delta_1) \\ \mathtt{pre\_f}_{\mathtt{light}} \rightarrow \mathtt{f}_{\mathtt{light}} & (\delta_2) \end{array} \right\}$$

Le modèle comportemental Δ décrit la dynamique du système et nous permet de calculer à chaque instant les candidats au diagnostic. Par exemple, pour un état précédent $s_{pre} = \mathit{light}\ \mathit{switch}\ \overline{f_{light}}\ \overline{f_{wire}}$ et une observation $o_1 = \overline{\mathit{light}}\ \mathit{switch}$, l'ensemble des états candidats est $\mathcal{S}_\Delta(s_{pre}, o) = \{s_{now}, s'_{now}, s''_{now}\}$ avec :

$$\begin{array}{rcl} s_{now} & = & \overline{\mathit{light}}\ \mathit{switch}\ \overline{f_{light}}\ f_{wire} \\ s'_{now} & = & \overline{\mathit{light}}\ \mathit{switch}\ f_{light}\ \overline{f_{wire}} \\ s''_{now} & = & \overline{\mathit{light}}\ \mathit{switch}\ f_{light}\ f_{wire} \end{array}$$

Les trois états candidats au diagnostic sont celui où la faute intermittente est la seule faute présente (s_{now}), celui où la faute permanente est la seule faute présente (s'_{now}) et enfin celui où les fautes sont toutes deux présentes (s''_{now}).

3.3 Choix du diagnostic unique

Afin de produire un diagnostic unique, nous devons choisir un unique état s_{now} parmi tous les éléments de $\mathcal{S}_\Delta(s_{pre}, o)$. Ce choix est réalisé d'après le modèle de préférences conditionnelles Γ qui décrit un ensemble ordonné de préférences.

Une préférence conditionnelle porte sur une variable à estimer et permet d'indiquer sous quelle condition on préfère assigner vrai ou faux à cette variable dans le diagnostic. Cette préférence n'est appliquée que lorsque le diagnostiqueur a le choix entre les deux affectations possibles pour la variable.

Définition 7 (Préférence conditionnelle) *Une préférence conditionnelle portant sur une variable* $\mathtt{e}$ *de* $\mathtt{E}$*, est notée* $\gamma = \mathtt{cond} : e \prec \overline{e}$. $\mathtt{cond}$ *(la condition de la préférence) est une formule propositionnelle pouvant porter sur* $\mathtt{P} \cup \mathtt{P}_{\mathtt{pre}}$*. La variable* $\mathtt{e}$ *est appelée la cible de la préférence.*

Notons que $\mathtt{cond} : \overline{e} \prec e$ est équivalent à $\neg\mathtt{cond} : e \prec \overline{e}$. Pour une variable $\mathtt{e}$ dans $\mathtt{E}$, la préférence conditionnelle $\mathtt{cond} : e \prec \overline{e}$ exprime une préférence pour les états dans lesquels $\mathtt{e}$ est vrai aux états dans lesquels $\mathtt{e}$ est faux si et seulement si $\mathtt{cond}$ est satisfaite. Formellement, une préférence $\gamma = \mathtt{cond} : e \prec \overline{e}$ définit un ordre partiel $\prec_\gamma$ et une relation d'équivalence $\approx_\gamma$ entre des paires d'états de la manière suivante. Pour tout triplet $s_{pre}, s, s' \in \mathcal{S}^3$, la transition (s_{pre}, s) est strictement préférée à la transition (s_{pre}, s') par rapport à γ (noté $(s_{pre}, s) \prec_\gamma (s_{pre}, s')$) si et seulement si $\sigma_{s_{pre},s} \models \mathtt{cond} \leftrightarrow \mathtt{e}$ et $\sigma_{s_{pre},s'} \not\models \mathtt{cond} \leftrightarrow \mathtt{e}$. Les transitions (s_{pre}, s) et (s_{pre}, s') sont équivalentes par rapport à γ (noté $(s_{pre}, s) \approx_\gamma (s_{pre}, s')$) si et seulement si $(\sigma_{s_{pre},s} \models \mathtt{cond} \leftrightarrow \mathtt{e}) \Leftrightarrow (\sigma_{s_{pre},s'} \models \mathtt{cond} \leftrightarrow \mathtt{e})$.

Le modèle de préférences Γ impose que toutes les variables à estimer soient la cible d'une préférence conditionnelle. Il permet également au concepteur du système de définir l'ordre dans lesquelles les préférences sont appliquées. Il est défini formellement comme suit.

Définition 8 (Modèle de préférences conditionnelles) *Un modèle de préférences conditionnelles Γ est une séquence de préférences conditionnelles $(\gamma_1, \gamma_2, .., \gamma_n)$ avec $\gamma_i = \mathtt{cond_i} : e_i \prec \overline{e_i}$ pour $i \in [1, n]$ telle que chaque variable $\mathtt{e}$ de $\mathtt{E}$ est la cible d'exactement une préférence.*

Nous imposons que Γ soit un modèle de préférences *cohérent*, c'est-à-dire qu'il n'y ait pas de cycle dans la séquence de préférences. Un cycle peut par exemple être construit avec les deux préférences suivantes : $\mathtt{a} : b \prec \overline{b}$ et $\mathtt{b} : a \prec \overline{a}$. La première préférence utilise la valeur de $\mathtt{a}$ pour choisir la valeur préférée de $\mathtt{b}$ et la deuxième fait exactement l'inverse. Dans la littérature, il existe des travaux portant sur les réseaux de préférences cycliques [2]. Dans ce papier, nous faisons l'hypothèse que le modèle Γ est acyclique. Cela signifie que la condition d'une préférence γ_i ne peut pas porter sur des variables qui sont la cible des préférences suivantes dans la séquence. Formellement, $\forall i \in [1, n]$, la condition $\mathtt{cond_i}$ porte uniquement sur des variables de $\mathtt{P_{pre}} \cup \mathtt{O} \cup \{\mathtt{e_j} | 1 \leq j < i\}$.

A partir d'un modèle de préférences Γ, il est possible de définir un ordre partiel $\prec_\Gamma$ entre les paires d'états de la manière suivante. Intuitivement, à la manière d'un ordre lexicographique, on considère les préférences γ_i dans l'ordre de la séquence et on utilise à chaque fois la relation d'ordre $\prec_{\gamma_i}$ définie précédemment. Cet ordre est partiel car il ne compare que les paires de transitions (s_{pre}, s_{now}), (s_{pre}', s_{now}') ayant le même état précédent (*i.e.* $s_{pre} = s_{pre}'$) et dont les états successeurs produisent la même observation (*i.e.* $\forall \mathtt{o} \in \mathtt{O} s_{now}(\mathtt{o}) = s_{now}'(\mathtt{o})$).

Formellement, $\forall s_{pre}, s, s' \in \mathcal{S}^3$, (s_{pre}, s) est strictement préféré à (s_{pre}, s') par rapport à Γ (noté $(s_{pre}, s) \prec_\Gamma (s_{pre}, s')$) si et seulement si il existe $i \in [1, n]$ tel que pour tout $j < i$, $(s_{pre}, s) \approx_{\gamma_j} (s_{pre}, s')$ et $(s_{pre}, s) \prec_{\gamma_i} (s_{pre}, s')$.

Bien que la relation d'ordre $\prec_\Gamma$ soit partielle sur $\mathcal{S}^2$, elle est totale sur n'importe quel ensemble de candidats au diagnostic. C'est grâce à cette relation d'ordre que nous allons définir le diagnostic préféré à chaque instant.

Proposition 1 *Soient s_{pre} un état, o une observation, Δ un modèle comportemental et Γ un modèle de préférences. Si s et s' sont deux états candidats pour s_{pre} et o $(s, s' \in \mathcal{S}_\Delta(s_{pre}, o)^2)$ tels que $s \neq s'$ alors on a soit $(s_{pre}, s) \prec_\Gamma (s_{pre}, s')$ soit $(s_{pre}, s') \prec_\Gamma (s_{pre}, s)$.*

Preuve Par la définition 6, l'appartenance de s et s' à $\mathcal{S}_\Delta(s_{pre}, o)$ permet d'affirmer que $\forall \mathtt{o} \in \mathtt{O}, s(\mathtt{o}) = s'(\mathtt{o})$. $s \neq s'$ signifie donc qu'il existe une variable $\mathtt{e} \in \mathtt{E}$ telle que $s(\mathtt{e}) \neq s'(\mathtt{e})$ et donc une préférence $\gamma_\mathtt{e} \in \Gamma$ telle que $s \approx_\gamma s'$ est faux. Soit i l'index de la première préférence de la séquence dans ce cas . Formellement, γ_i est telle que $\forall j < i$, $(s_{pre}, s) \approx_{\gamma_j} (s_{pre}, s')$ et non $(s_{pre}, s) \approx_{\gamma_i} (s_{pre}, s')$ et que $(s_{pre}, s) \prec_{\gamma_i} (s_{pre}, s')$ ou $(s_{pre}, s') \prec_{\gamma_i} (s_{pre}, s)$. Du point de vue de l'ordre $\prec_\Gamma$, on a donc $(s_{pre}, s) \prec_\Gamma (s_{pre}, s')$ ou $(s_{pre}, s') \prec_\Gamma (s_{pre}, s)$. ■

3.4 Processus d'estimation

A chaque étape, étant donné l'ensemble des états candidats $\mathcal{S}_\Delta(s_{pre}, o)$ on sélectionne l'état préféré par le modèle de préférences conditionnelles Γ.

Définition 9 (État estimé) *L'état estimé $\hat{s} = estim(s_{pre}, o)$ pour un état précédent s_{pre} et une observation o est l'élément de $\mathcal{S}_\Delta(s_{pre}, o)$ préféré par $\prec_\Gamma$. Formellement :*

$$\begin{aligned} &\hat{s} = estim(s_{pre}, o) \text{ si et seulement si} \\ &\quad \hat{s} \in \mathcal{S}_\Delta(s_{pre}, o) \text{ et} \\ &\quad \forall s_{now} \in \mathcal{S}_\Delta(s_{pre}, o) \text{ t.q. } s_{now} \neq \hat{s}, (s_{pre}, \hat{s}) \prec_\Gamma (s_{pre}, s_{now}) \end{aligned}$$

En partant de l'état initial s_0, on peut définir la séquence d'états estimés par le diagnostiqueur pour une séquence d'observations donnée.

Définition 10 (Séquence d'états estimés) *La séquence d'états $(s_0, \hat{s}_1, \hat{s}_2, ..., \hat{s}_k)$ est la séquence d'états estimés pour une séquence d'observations $(o_0, o_1, o_2, ..., o_k)$ si et seulement si $\forall i \in [1, k], \hat{s}_i = estim(\hat{s}_{i-1}, o_i)$.*

Exemple 2 Reprenons le modèle comportemental de l'ampoule et associons-le aux préférences suivantes :

$$\Gamma = \begin{pmatrix} \mathtt{pre\_f_{wire}} : f_{wire} \prec \overline{f_{wire}} & (\gamma_1) \\ \neg\text{light} : f_{light} \prec \overline{f_{light}} & (\gamma_2) \end{pmatrix}$$

La première préférence (γ_1) déclare que nous préférons toujours la valeur de $\mathtt{f_{wire}}$ à l'instant précédent. Ce mécanisme permet d'apporter une certaine stabilité au diagnostic puisque rappelons-le $\mathtt{f_{wire}}$ est une faute intermittente. La préférence (γ_2) indique que lorsque la lumière est éteinte nous préférons estimer que $\mathtt{f_{light}}$ est présente.

Pour l'état précédent $s_{pre} = s_0 = light\ switch\ \overline{f_{light}}\ \overline{f_{wire}}$ et l'observation $o_1 = \overline{light}\ switch$ nous avons vu dans l'Exemple 1 que $\mathcal{S}_\Delta(s_{pre}, o)$ contient les trois éléments

$s_{now} = \overline{light}\ switch\ \overline{f_{light}}\ f_{wire}$, $s_{now}' = \overline{light}\ switch\ f_{light}\ \overline{f_{wire}}$ et $s''_{now} = \overline{light}\ switch\ f_{light}\ f_{wire}$

En évaluant d'abord (γ_1), d'après $\overline{pre\_f_{wire}}$ on choisit $\overline{f_{wire}}$. Par (γ_2), on observe $\overline{light}$ et on en déduit f_{light}. L'état unique préféré sera donc $\hat{s}_1 = estim(s_0, o_1) = s_{now}' = \overline{light}\ switch\ f_{light}\ \overline{f_{wire}}$.

4 Interblocage

Une situation d'interblocage survient lorsque le modèle a précédemment estimé un état $\hat{s}$ différent de l'état réel s du système et reçoit une observation o en contradiction avec $\hat{s}$ et Δ. Dans une telle situation l'ensemble des candidats pour $\hat{s}$ et o est vide et le diagnostiqueur est incapable de renvoyer un diagnostic.

Définition 11 (Interblocage) *Un modèle d'estimation (s_0, Δ, Γ) est en situation d'interblocage pour une séquence d'observations $(o_0, o_1, o_2, \ldots, o_k)$ avec $k > 1$* [1] *telle que :*

— *il existe une séquence d'états estimés pour $(o_0, o_1, o_2, \ldots, o_{k-1})$*

— *et il n'existe aucune séquence d'états estimés pour $(o_0, o_1, o_2, \ldots, o_k)$*

Exemple 3 Soient la séquence d'observations (o_0, o_1, o_2, o_3) produite par le système et la séquence d'états associés (s_0, s_1, s_2, s_3) représentés en Figure 2. La séquence d'états estimés $(s_0, \hat{s}_1, \hat{s}_2)$ débouche sur un interblocage pour l'observation o_3.

i	o_i	s_i	$\hat{s}_i$
0	$light\ switch$	$\overline{f_{light}}\ \overline{f_{wire}}$	$\overline{f_{light}}\ \overline{f_{wire}}$
1	$light\ switch$	$\overline{f_{light}}\ \overline{f_{wire}}$	$\overline{f_{light}}\ \overline{f_{wire}}$
2	$\overline{light}\ switch$	$\overline{f_{light}}\ f_{wire}$	$f_{light}\ \overline{f_{wire}}$
3	$light\ switch$	$\overline{f_{light}}\ \overline{f_{wire}}$	/

FIGURE 2 – Scenario d'interblocage de l'exemple 3. Dans les colonnes de s_i et $\hat{s}_i$, on omet les variables de `O` identiques à la colonne o_i, et on ne représente que les variables de `E`).

Lorsque l'observation o_1 est reçue, les préférences sélectionnent l'état préféré $\hat{s}_1 = light\ switch\ \overline{f_{light}}\ \overline{f_{wire}}$. Ensuite pour l'observation o_2, nous sommes dans la situation décrite dans l'exemple 2 et l'état préféré est $\hat{s}_2 = \overline{light}\ switch\ f_{light}\ \overline{f_{wire}}$. L'observation o_3 est en contradiction avec l'état précédemment estimé, en effet l'estimateur ayant estimé f_{light} à l'état précédent, la variable $\texttt{pre\_f}_{\texttt{light}}$ est à vrai et les deux règles de Δ ne sont pas satisfaisables avec une telle configuration : (δ_1) nous impose que $\texttt{f}_{\texttt{light}}$ soit faux puisqu'on observe de la lumière tandis que (δ_2) nous impose que $\texttt{f}_{\texttt{light}}$ soit vrai puisque c'est une panne permanente.

Il existe donc une séquence d'états estimés pour (o_0, o_1, o_2). Il s'agit de la séquence $(s_0, \hat{s}_1, \hat{s}_2)$. Mais il n'existe pas de séquence d'états estimés pour (o_0, o_1, o_2, o_3). Le modèle est donc en situation d'interblocage pour cette séquence d'observations.

Dans l'exemple simple précédent, la situation d'interblocage pourrait être facilement évitée en modifiant l'ordre ou les conditions des préférences. Cependant, dès que l'on s'intéresse à des systèmes plus complexes, il devient difficile d'anticiper les situations d'interblocage et encore plus de les résoudre. Dans ce papier, nous nous concentrons sur la détection de ces interblocages et laissons leur résolution pour des travaux futurs.

5 Vérification des interblocages

Dans cette partie, nous montrons comment utiliser un model-checker pour identifier les scénarios d'interblocage dès la phase de conception du modèle de diagnostic.

5.1 Définition du vérificateur

La méthode de vérification que nous proposons s'inspire de la technique du Twin-Plant [10] qui permet de vérifier la diagnosticabilité d'un système en construisant le produit synchrone de deux machines à états finis.

Notre méthode est similaire puisqu'elle consiste à construire un vérificateur des interblocages en synchronisant deux machines à états. La première machine à états représente le système et est uniquement contrainte par la relation de transition Δ. Elle nous permet notamment de générer des séquences d'observations cohérentes du point de vue du système. La deuxième machine à états correspond au diagnostiqueur(ou estimateur) construit à partir du modèle de diagnostic (s_0, Δ, Γ). Elle est également contrainte par Δ, mais elle sélectionne de manière déterministe l'état suivant en appliquant les préférences de Γ. Le vérificateur est le produit de ces deux machines à états synchronisées sur les observations du système à chaque pas de temps. La vérification d'interblocage consiste alors à vérifier s'il existe une séquence d'observations qui conduit le vérificateur à être dans un état dans lequel la machine à états du système a un successeur et la machine à états du diagnostiqueur n'en a pas.

De manière à distinguer l'état du système de celui du diagnostiqueur, nous introduisons deux ensembles de variables $\texttt{P}^{\texttt{sys}}$ et $\texttt{P}^{\texttt{est}}$ qui sont des copies de `P`. Nous utilisons également une variable `est_sat` qui indique si le diagnostiqueur a un état successeur.

Définition 12 (Variables du vérificateur) *L'ensemble des variables du vérificateur est défini par*

1. Il ne peut pas y avoir d'interblocage à $\hat{s}_1$ car on suppose que l'estimateur est correctement initialisé à s_0.

$\mathtt{P}^{\mathtt{verif}} = \mathtt{P}^{\mathtt{sys}} \cup \mathtt{P}^{\mathtt{est}} \cup \{\mathtt{est\_sat}\}$, *où :*

— $\mathtt{P}^{\mathtt{sys}} = \{\mathtt{p\_sys} \mid \mathtt{p} \in \mathtt{P}\}$ *est l'ensemble des variables d'état du système ;*

— $\mathtt{P}^{\mathtt{est}} = \{\mathtt{p\_sys} \mid \mathtt{p} \in \mathtt{P}\}$ *est l'ensemble des variables d'état du diagnostiqueur ;*

— `est_sat` *indique s'il existe un état estimé.*

Un état du vérificateur est une affectation des variables de $\mathtt{P}^{\mathtt{verif}}$.

Pour définir la machine à états résultant du produit synchrone, nous commençons par définir l'état initial du vérificateur (Définition 13) puis sa relation de transition (Définition 14).

Définition 13 (Etat initial du vérificateur) *On suppose que l'état initial du vérificateur est l'état* s_0^{verif} *tel que :*

— $\forall \mathtt{p} \in \mathtt{P},\ s_0^{verif}(\mathtt{p\_sys}) = s_0(\mathtt{p})$

— $\forall \mathtt{p} \in \mathtt{P},\ s_0^{verif}(\mathtt{p\_est}) = s_0(\mathtt{p})$

— $s_0^{verif}(\mathtt{est\_sat}) = \top$

Définition 14 (Relation de transition du vérificateur) *Soient* s_{pre}^{verif} *et* s_{now}^{verif} *deux états du vérificateur. La paire* $(s_{pre}^{verif}, s_{now}^{verif})$ *est une transition du vérificateur si et seulement si :*

(1) les variables observables prennent la même valeur côté système et estimateur,

(2) les variables de $\mathtt{P}^{\mathtt{sys}}$ *respectent* Δ*,*

(3) la variable `est_sat` *indique si il existe un état estimé possible (i.e. si l'ensemble des candidats est non vide), et*

(4) si `est_sat` *est vrai, alors les variables de* $\mathtt{P}^{\mathtt{est}}$ *respectent* Δ *et* Γ.

Formellement :

$$\begin{array}{l} \forall \mathtt{o} \in \mathtt{O},\ s_{pre}^{verif}(\mathtt{sys\_o}) = s_{pre}^{verif}(\mathtt{est\_o}) \\ \forall \mathtt{o} \in \mathtt{O},\ s_{now}^{verif}(\mathtt{sys\_o}) = s_{now}^{verif}(\mathtt{est\_o}) \end{array} \tag{1}$$

$$\begin{array}{l} \exists (s_{pre}, s_{now}) \in \Delta, \forall \mathtt{p} \in \mathtt{P}, \\ \quad s_{pre}(\mathtt{p}) = s_{pre}^{verif}(\mathtt{sys\_p})\ et \\ \quad s_{now}(\mathtt{p}) = s_{now}^{verif}(\mathtt{sys\_p}) \end{array} \tag{2}$$

$$\mathtt{est\_sat} \leftrightarrow \left(\begin{array}{l} \exists (s_{pre}, s_{now}) \in \Delta, \\ \quad \forall \mathtt{p} \in \mathtt{P}, s_{pre}(\mathtt{p}) = s_{pre}^{verif}(\mathtt{est\_p})\ et \\ \quad \forall \mathtt{p} \in \mathtt{O}, s_{now}(\mathtt{p}) = s_{now}^{verif}(\mathtt{est\_p}) \end{array} \right) \tag{3}$$

$$\mathtt{est\_sat} \rightarrow \left(\begin{array}{l} \exists (s_{pre}, s_{now}) \in \Delta, \forall \mathtt{p} \in \mathtt{P}, \\ \quad s_{pre}(\mathtt{p}) = s_{pre}^{verif}(\mathtt{est\_p})\ et \\ \quad s_{now}(\mathtt{p}) = s_{now}^{verif}(\mathtt{est\_p})\ et \\ \quad \nexists s_{best} \in \mathcal{S}, \\ \quad\quad \forall \mathtt{o} \in \mathtt{O}, s_{best}(\mathtt{o}) = s_{now}(\mathtt{o}),\ et \\ \quad\quad (s_{pre}, s_{best}) \in \Delta,\ et \\ \quad\quad (s_{pre}, s_{best}) \prec_\Gamma (s_{pre}, s_{now}) \end{array} \right) \tag{4}$$

Proposition 2 *Un modèle d'estimation* (s_0, Δ, Γ) *est sujet à un interblocage si et seulement si le vérificateur associé contient un chemin dans lequel* `est_sat` *est faux. Le scénario d'interblocage est la séquence d'observations suivie par le vérificateur dans ce scénario.*

Preuve Supposons que le modèle d'estimation est sujet à un interblocage. Alors, d'après la Définition 11, il existe une séquence d'observations $(o_0, o_1, \ldots, o_n)$ générée par une séquence cohérente d'états $(s_0, s_1, \ldots, s_n)$, telle qu'il existe une séquence d'états estimés $(\hat{s}_0, \hat{s}_1, \ldots, \hat{s}_{n-1})$ pour la séquence partielle $(o_0, o_1, \ldots, o_{n-1})$ mais qu'il n'existe pas de séquences d'états estimés $(\hat{s}_0, \hat{s}_1, \ldots, \hat{s}_n)$ pour la séquence complète. On construit alors la séquence d'états du vérificateur $(s_0^{verif}, s_1^{verif}, \ldots, s_n^{verif})$ telle que :

— $\forall i \in [0, n], \forall p \in \mathtt{P}, s_i^{verif}(\mathtt{p\_sys}) = s_i(\mathtt{p})$

— $\forall i \in [0, n-1], \forall p \in \mathtt{P}, s_i^{verif}(\mathtt{p\_est}) = \hat{s}_i(\mathtt{p})$

— $\forall i \in [0, n-1], s_i^{verif}(\mathtt{est\_sat}) = \top$

— $\forall \mathtt{p} \in \mathtt{O}, s_n^{verif}(\mathtt{p\_est}) = o_n(\mathtt{p})$,

— $\forall \mathtt{p} \in \mathtt{E}, s_n^{verif}(\mathtt{p\_est}) = \top$,

— $s_n^{verif}(\mathtt{est\_sat}) = \bot$

On peut alors montrer que pour tout $i \in [0, n-1]$, $(s_i^{verif}, s_{i+1}^{verif})$ est une transition du vérificateur.

Pour la réciproque, on suppose que le vérificateur a un chemin dans lequel `est_sat` est faux. On considère un tel chemin $(s_0^{verif}, s_1^{verif}, \ldots, s_n^{verif})$ dans lequel s_n^{verif} est le seul état avec `est_sat` assigné à faux. On construit alors deux séquences d'états $(s_0, s_1, \ldots, s_n)$ et $(s_0, \hat{s}_1, \ldots, \hat{s}_{n-1})$ de la manière suivante :

— $\forall i \in [0, n], \forall \mathtt{p} \in \mathtt{P}, s_i(\mathtt{p}) = s_i^{verif}(\mathtt{p\_sys})$,

— $\forall i \in [0, n-1], \forall \mathtt{p} \in \mathtt{P}, \hat{s}_i(\mathtt{p}) = s_i^{verif}(\mathtt{p\_est})$

On prouve que ces séquences d'états sont cohérentes avec Δ. On montre également que si `est_sat` est à faux (dernier pas de temps), aucun état s_{now} ne permet de satisfaire la partie droite de la condition (3) pour $s_{pre} = \hat{s}_{n-1}$. Cela signifie qu'aucun état ne satisfait la Définition 6. Il y a un interblocage pour la séquence d'observations générées par $(s_0, s_1, \ldots, s_n)$. ■

5.2 Choix du model-checker

Le model-checking est un ensemble de techniques et d'outils informatique permettant de vérifier des propriétés sur des systèmes. Parmi les nombreux model-checkers de la littérature, NuSMV [3] et NuXMV sont reconnus pour leur efficacité sur la vérification de propriétés temporelles sur des systèmes dynamiques. Ils sont basés sur des logiques temporelles comme LTL ou CTL. D'autres outils sont plutôt voués à la vérification de propriétés sur des modèles structurellement complexes mais non-dynamiques. C'est le cas du model-checker Alloy Analyzer [9] dans lesquels les modèles sont basés sur le langage Alloy, lui-même basé sur la logique du premier ordre.

Dans le cas de la vérification des interblocages, il est nécessaire d'exprimer à la fois la dynamique du système mais également les préférences du diagnostiqueur. Nous nous sommes donc tournés vers le model-checker Electrum [13] qui s'appuie sur le langage Alloy et permet d'y ajouter une dynamique comme dans NuXMV.

5.3 Forme quantifiée des préférences

Le système de transition du vérificateur est en partie défini par les préférences. De manière à pouvoir exprimer ces préférences dans le langage du model-checker, nous définissons une forme quantifiée des préférences.

Définition 15 (Forme quantifiée d'une préférence) *Soit* $\Gamma = (\gamma_1, \ldots, \gamma_n)$ *où chaque préférence est de la forme* $\gamma_i = \mathtt{cond_i} : e_i \prec \overline{e_i}$*, pour* $i \in [1, n]$*. On définit la forme quantifiée de* γ_i *par la formule* ψ_i *comme suit :*

$$\psi_i = (\forall \mathtt{e_i}, \exists \mathtt{e_{i+1}}, \ldots, \mathtt{e_n},\ \Delta) \rightarrow (\mathtt{e_i} \leftrightarrow \mathtt{cond_i})$$

La forme quantifiée ψ_i est composée de deux parties. La parte gauche exprime que quelle que soit la valeur affectée à la variable $\mathtt{e_i}$, il existe une manière d'affecter les variables cibles des préférences suivantes tout en étant cohérent avec Δ. La partie droite exprime que $\mathtt{e_i}$ est affectée à vrai si et seulement si la condition $\mathtt{cond_i}$ est satisfaite. Au travers de la proposition suivante, nous montrons que les préférences de Γ et leur forme quantifiée sont équivalentes du point de vue de l'état estimé pour un état précédent et une observation.

Proposition 3 *Pour un état estimé précédent* $\hat{s}_{pre}$*, une observation* o*, un candidat* $\hat{s} \in \mathcal{S}_\Delta(\hat{s}_{pre}, o)$*,* $\hat{s} = estim(\hat{s}_{pre}, o)$ *si et seulement si l'affectation* $\sigma_{\hat{s}_{pre},\hat{s}}$ *satisfait la conjonction* $\bigwedge_{i\in[1,n]} \psi_i$*.*

Preuve (grandes lignes) On montre par induction qu'un état est l'état estimé pour la séquence de préférences $(\gamma_1, \ldots, \gamma_k)$ si et seulement si il satisfait $\bigwedge_{i\in[1,k]} \psi_i$. Pour $k = 1$, on montre que la formule $\forall \mathtt{e_1}, \exists \mathtt{e_2}, \ldots \mathtt{e_n}, \Delta$ est vérifiée si et seulement si e_1 et $\overline{e_1}$ sont toutes deux cohérentes avec Δ. Cela signifie que $\mathcal{S}_\Delta(\hat{s}_{pre}, o)$ contient au moins deux états s et s' avec $s(\mathtt{e_1}) \neq s'(\mathtt{e_1})$. Entre ces deux états, celui dans lequel $\mathtt{cond_1} \leftrightarrow \mathtt{e_1}$ est vrai est l'état préféré par la préférence γ_1. C'est également l'état qui vérifie la formule ψ_1. Pour $k > 1$, en supposant que la propriété est vraie pour tout $i < k$, on réalise un raisonnement similaire pour montrer l'équivalence. ■

5.3.1 Encodage du modèle en Electrum

Dans cette partie, nous détaillons l'encodage du vérificateur tel que défini dans les Définitions 13 et 14. Pour cela, nous reprenons l'estimateur (s_0, Δ, Γ) illustré en Figure 1.

Nous nous intéressons d'abord à la modélisation du système observé. Pour modéliser la première machine à état, nous déclarons un prédicat correspondant exactement au

```
var one sig RealSystem {
  var fwire : Bool,
  var preF_fwire : Bool,
  var light : Bool,
  var flight : Bool,
  var switch : Bool,
  var preF_flight : Bool,
}
fact always_delta_RealSystem {
  always {
    delta[RealSystem.fwire,RealSystem.preF_fwire,
    RealSystem.light,RealSystem.flight,
    RealSystem.switch,RealSystem.preF_flight]}
}
fact synch_obs {
  always {
    Estimator.light = RealSystem.light
    Estimator.switch = RealSystem.switch}
}
```

FIGURE 3 – Le système observé et la synchronisation des observations en Electrum.

```
fact next_Estimator {
  always {
    Estimator.preF_flight' = Estimator.flight
    Estimator.preF_fwire' = Estimator.fwire }
  }
```

FIGURE 4 – La dynamique temporelle en Electrum.

modèle Δ puis nous ajoutons une contrainte (`fact` en Electrum) qui indique que le système observé respectera toujours (`always` en Electrum) les contraintes de Δ (voir Figure 3). Enfin, nous synchronisons les variables de `O` du système observé avec celles de la deuxième machine à états représentant notre estimateur, comme exprimé dans la condition (1) de la Définition 14.

Pour représenter l'aspect temporel de notre formalisme, on utilise l'opérateur `'` d'Electrum qui décrit une variable à l'instant suivant. La Figure 4 reproduit l'intention de la fonction bijective *pre* (la valeur de `p_pre` à l'état suivant est égale à la valeur de `p` dans l'état courant).

Electrum étant doté des opérateurs de l'algèbre relationnelle, il nous est possible pour chaque préférence d'écrire un prédicat nous indiquant si une préférence γ_i est applicable, c'est-à-dire si les deux valuations pour la variable $\mathtt{e_i}$ (opérateur `all`), et au moins une valuation possible des variables $\mathtt{e_j}, i < j \leq n$ (opérateur `some`) sont cohérentes avec Δ et l'observation. Cela correspond à la partie gauche de la forme quantifiée de la préférence. La Figure 5 illustre la définition des prédicats qui mettent en œuvre ces conditions avec les opérateurs d'Electrum. Le reste des préférences est

```
pred pref_fwire_possible{
  all fwire : Bool |some flight : Bool |
    delta[fwire,Estimator.preF_fwire,
    Estimator.light,flight,Estimator.switch,
    Estimator.preF_flight]
}
pred pref_flight_possible{
  all flight : Bool |
    delta[Estimator.fwire,Estimator.preF_fwire,
    Estimator.light,flight,Estimator.switch,
    Estimator.preF_flight]
}
```

FIGURE 5 – Implémentation des préférences en Electrum

décrit par un fait.

L'ordre dans lequel les préférences sont appliquées dans Γ, est indiqué par les imbrications de quantificateurs dans les formes quantifiées des préférences (voir Définitions 8 et 15).

6 Résultats Expérimentaux

Pour réaliser les expérimentations, nous avons défini plusieurs modèles basés sur notre formalisme pour effectuer le diagnostic d'une mission multi-robots.

Un ou plusieurs robots doivent se déplacer sur une grille rectangulaire de taille variable, chaque robot se déplace vers la destination qui lui est assignée, et qui peut changer au cours de la mission. On déclare des contraintes comportementales dans Δ pour chaque robot. Par exemple les robots peuvent être sujets à des fautes permanentes et/ou intermittentes. Une faute intermittente empêche le robot de se déplacer pendant quelques instants, alors qu'une faute permanente l'immobilise définitivement. Nous cherchons également à estimer l'état des différentes cases de la grille. Un robot traverse une case *normale* en un pas de temps, mais on introduit des cases *difficiles à traverser* (le robot reste sur la case pendant plusieurs instants) et *impossible à traverser* (le robot est immobilisé sur la case). Nous observons à chaque instant la position des robots sur la grille ainsi que leur destination. Nous estimons la présence des fautes intermittentes et permanentes sur chaque robots, une variable indiquant s'ils peuvent se déplacer, ainsi que la dangerosité de chaque case de la grille. Nous implémentons deux stratégies d'estimation, une sujette à interblocages, l'autre non (par exemple en préférant toujours l'absence d'une faute permanente).

Les modèles sont écrits en langage Scala et sont automatiquement traduits en Electrum. Electrum procède à la vérification par l'intermédiaire de différents solveurs. Lors de nos expérimentations, nous avons paramétré Electrum pour qu'il utilise le solveur Sat4j [12]. Les vérifications ont été effectuées sur un processeur quadricœur Intel Core i5-7600 de 3.5GHz.

La Figure 6 montre les résultats de nos expérimentations pour différentes tailles de jeux de données.

Le premier constat est que la vérification est plus rapide pour des modèles sujets aux interblocages que pour ceux qui n'en rencontrent pas. Cette différence est directement liée au fait que lorsque le model-checker trouve un scénario d'interblocage dans le modèle, la vérification s'interrompt et celui-ci renvoie un contre exemple explicitant la séquence d'observations associée. Au contraire, pour un modèle qui ne rencontre pas d'interblocage, Electrum explore beaucoup plus d'exécutions possibles pour l'estimateur.

Nous pouvons ensuite constater que la vérification est très rapide pour les premiers modèles, mais à mesure que l'on enrichit $\mathtt{P}$, Δ et Γ, le nombre de variables générées pour Sat4j ainsi que le temps nécessaire à la vérification augmente de manière exponentielle. Pour des exemples de tailles supérieures, il est intéressant de noter que Sat4j ne disposait pas de suffisamment de mémoire pour les traiter.

La détection des interblocages que nous proposons est réalisée en phase de conception. Aussi, des durées de calcul élevées ne sont pas forcément rédhibitoires et ne remettent pas en cause l'approche que nous proposons. De plus, les résultats que nous présentons ici sont des résultats préliminaires et nous travaillons actuellement sur différentes pistes pour les améliorer. Plus précisément, nous visons à intégrer plus étroitement Electrum de manière à éviter la création de variables booléennes là où Electrum pourrait gérer des variables entières.

7 Conclusion et perspectives scientifiques

Nous avons développé une méthode générique permettant de valider le bon fonctionnement d'un diagnostiqueur dès la phase de conception. Nous avons vu qu'il est possible d'anticiper les situations d'interblocage pour les diagnostiqueur que nous construisons.

Cependant, nous n'introduisons dans cet article aucun mécanisme permettant de réparer un diagnostiqueur sujet aux scénarios d'interblocage mais cela reste un objectif pour nos travaux futurs.

De plus, l'automatisation du model-checking nous permettant de vérifier l'existence de scénarios d'interblocage sur nos diagnostiqueurs sera certainement réutilisée pour vérifier automatiquement d'autres propriétés relatives au diagnostic telles que la diagnosticabilité.

Enfin, des progrès sur les performances du model-checker sont nécessaires afin de l'appliquer sur des missions multi-robots impliquant des modèles de grande taille.

ID	nb robots	taille grille	$\|\|P\|\|$	$\|\|\Delta\|\|$	$\|\|\Gamma\|\|$	CNF vars	CNF clauses	avec	sans
1	1	1 x 2	13	17	7	1083	1586	0,1s	0,8s
2	1	2 x 2	21	27	11	3400	3925	0,5s	4,8s
3	1	2 x 3	29	37	15	13757	10593	7,3s	52,9s
4	2	2 x 2	34	50	14	12631	13561	6,7s	60,5s
5	2	2 x 3	46	68	18	65 030	47 669	144,6s	1001,4s

FIGURE 6 – Vérification de modèles en Electrum. Les colonnes $||P||$, $||\Delta||$ et $||\Gamma||$ indiquent respectivement le nombre de variables de P, le nombre de règles de Δ et le nombre de préférences dans Γ. Les colonnes 7 et 8 indiquent respectivement le nombre de variables et de clauses envoyées au solveur Sat4j par Electrum. Les deux dernières colonnes indiquent le temps de vérification pour une version du modèle sujette à un scénario d'interblocage et une autre version pour laquelle un tel scénario n'existe pas.

Références

[1] Boutilier, C., R. I. Brafman, C. Domshlak, H. H. Hoos et D.Poole: *Preference-based constrained optimization with cp-nets*. Dans *Computational Intelligence*, pages 137–157. 2004.

[2] Boutilier, Craig, Ronen I Brafman, Carmel Domshlak, Holger H Hoos et David Poole: *CP-nets : A tool for representing and reasoning with conditional ceteris paribus preference statements*. J. Artif. Intell. Res.(JAIR), 21 :135–191, 2004.

[3] Cimatti, Alessandro, Edmund Clarke, Fausto Giunchiglia et Marco Roveri: *NUSMV : a new symbolic model checker*. International Journal on Software Tools for Technology Transfer, 2(4) :410–425, Mar 2000.

[4] Contant, Olivier, Stéphane Lafortune et Demosthenis Teneketzis: *Diagnosis of intermittent faults*. Discrete Event Dynamic Systems, 14(2) :171–202, 2004.

[5] Cordier, M O, Philippe Dague, François Lévy, Jacky Montmain, Marcel Staroswiecki et Louise Travé-Massuyès: *Conflicts versus analytical redundancy relations : a comparative analysis of the model based diagnosis approach from the artificial intelligence and automatic control perspectives*. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 34(5) :2163–2177, 2004.

[6] De Kleer, Johan: *Diagnosing Multiple Persistent and Intermittent Faults*. Dans *IJCAI*, pages 733–738, 2009.

[7] Felfernig, Alexander et Monika Schubert: *Fastdiag : A diagnosis algorithm for inconsistent constraint sets*. Dans *Proceedings of the 21st International Workshop on the Principles of Diagnosis (DX 2010), Portland, OR, USA*, pages 31–38, 2010.

[8] Grastien, Alban et Anbulagan: *Incremental Diagnosis of DES with a Non-Exhaustive Diagnosis Engine*. Dans *Proceedings of the 20th International Workshop on Principles of Diagnosis*, 2009.

[9] Jackson, Daniel: *Software Abstractions : Logic, Language, and Analysis*. The MIT Press, 2006, ISBN 0262101149.

[10] Jiang, Shengbing, Zhongdong Huang, V. Chandra et R. Kumar: *A polynomial algorithm for testing diagnosability of discrete-event systems*. IEEE Transactions on Automatic Control, 46(8) :1318–1321, Aug 2001, ISSN 0018-9286.

[11] Kurien, James et P Pandurang Nayak: *Back to the future for consistency-based trajectory tracking*. Dans *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 370–377, 2000.

[12] Le Berre, Daniel et Anne Parrain: *The sat4j library, release 2.2, system description*. Journal on Satisfiability, Boolean Modeling and Computation, 7 :59–64, 2010.

[13] Macedo, Nuno, Julien Brunel, David Chemouil, Alcino Cunha et Denis Kuperberg: *Lightweight Specification and Analysis of Dynamic Systems with Rich Configurations*. Dans *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2016, pages 373–383, New York, NY, USA, 2016.

[14] Pralet, Cedric, Xavier Pucel et Stéphanie Roussel: *Diagnosis of Intermittent Faults with Conditional Preferences*. Dans *Proceedings of the 27th International Workshop on Principles of Diagnosis (DX'16)*, 2016.

[15] Pucel, Xavier et Stéphanie Roussel: *Intermittent Fault diagnosis as Discrete Signal Estimation : Trackability Analysis*. Dans *Proceedings of the 28th International Workshop on Principles of Diagnosis (DX'17)*, 2017.

[16] Sampath, M., R. Sengupta, S. Lafortune, K. Sinnamohideen et D. Teneketzis: *Diagnosability of discrete-event systems*. IEEE Transactions on Automatic Control, 40(9) :1555–1575, Sep 1995, ISSN 0018-9286.