# Preferential Discrete Model-based Diagnosis for Intermittent and Permanent Faults

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

# Preferential Discrete Model-based Diagnosis for Intermittent and Permanent Faults

**Valentin Bouziat**[1] and **Xavier Pucel**[1] and **Stéphanie Roussel**[1] and **Louise Travé-Massuyès**[2]
[1] ONERA / DTIS, Université de Toulouse, F-31055 Toulouse – France
e-mail: firstname.name@onera.fr
[2]LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
e-mail: louise@laas.fr

## 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 single diagnosis at each time step. Like all incomplete diagnosis approaches, ours is subject to deadlocks between the system and its diagnoser. In this paper, we address the detection of such deadlocks at design time with the rich semantic model-checker ELECTRUM.

## 1 Introduction

Using robots in situations where teleoperation is impossible or difficult requires the robots to have some level of decisional autonomy. In particular, an autonomous robot needs to detect and respond appropriately to abnormal losses of performance, due to degradations of the robotic system, or to external disturbances. This requires an elaborate fault management strategy, which can be challenging to design, especially when the mission is complex and the robot is subject to many faults. We are therefore interested in techniques that facilitate the identification of non-nominal operation modes, the detection of faults, and the estimation of the remaining capabilities of the robot.

We propose a modeling formalism constructed so that it meets three requirements. First, it should incrementally calculate a single diagnosis at each time step. This requirement stems from constraints on system performance: one can not afford to compute all possible diagnoses on a large system for a long period of time. Moreover, considering the complete robotic system, the diagnostic module is in our case associated with a deterministic planner which expects a single diagnosis as input. Second, we require our formalism to take into account both permanent and intermittent faults, in particular to distinguish disturbances (noise, false contact, etc.), supposedly intermittent, from degradations (breakdowns, etc.), often permanent. Third, since the diagnosis is used to make autonomous decisions, we want to avoid gooing back and modifying a diagnosis previously issued. We also forbid from producing a sequence of diagnoses that does not correspond to a possible evolution of the system (for example to remove a permanent fault). The principle is that the diagnosis should be a reliable piece of information, from which the robot can make a safe autonomous decision. If it is impossible to always produce such a diagnosis, then we want to detect it at design time so that the robot, the mission, or the fault management strategy can be modified.

In this paper, we describe how such requirements can create situations where the robotic system produces an observation sequence for which the diagnoser has no explanation. In this kind of situation, which we call a *deadlock*, the entire decision process supported by the diagnostic engine fails. This is why we are interested in detecting deadlock scenarios at the design time.

The paper is sturctured as follows. We start with a presentation of the state of the art in section 2. We then introduce our modeling formalism in section 3, and describe the deadlock issue in section 4. We propose a verification method at design time based on the model-checker ELECTRUM [1] in section 5, and present experimental results on small multi-robot systems in section 6. Perspectives are discussed in section 7.

## 2 Related Work

Our work addresses the diagnosis of discrete event systems. In [2] the authors define a framework for the diagnosis of systems modeled by finite state machines subject to permanent faults. The authors describe the construction of a diagnoser that allows an incremental computation of the diagnosis. However, their diagnoser requires memorizing all the possible state-diagnosis pairs, which severely limits its scalability. Their definition of diagnosability illustrates the importance of validating the performance of a diagnoser at design time.

Our formalism associates propositional logic constraints with a conditional preference theory from [3]. It is inspired by the formalism used in [4] to produce incremental diagnoses. In this related work, the authors just point at the risk of encountering a deadlock. In [5], the same authors propose to detect deadlock scenarios between a system and its diagnoser by an iterative model-checking approach but this approach is difficult to implement and no associated experimentations are provided.

The problem of deadlocks between a system and its diagnoser occurs in [6] where the diagnoser goes back in the execution of the diagnoser in order to find a consistent explanation. This solution violates our third requirement explained above. We want to produce reliable diagnoses, or no diagnoses at all.

A classic way to order diagnoses is to prefer the minimal ones, this approach knows several variants compared in [7]. Other approaches like [8] order faults according to some criterion, and deduce an order on diagnoses. In all

these approaches, the diagnosis ordering is unconditional, *i.e.* observations have no effect on the order of diagnoses. Our model uses conditional preferences precisely to remove this limitation, and to make it possible to prefer certain diagnoses based on present and past observations.

The diagnosis of intermittent faults is discussed in the literature from different points of view. In [9], the same function is called multiple times, but while faults manifest themselves in an intermittent manner, the underlying diagnosis is constant from one test to another. In [10], a repair event is associated with each fault, and the diagnosis task consists in detecting for each fault which event (fault or repair) happened last. A diagnoser is built to allow incremental evaluation, which involves the previously mentioned scaling problems since all diagnoses are kept in memory.

## 3 Diagnosis model

Our diagnosis model is a tuple $(s_0, \Delta, \Gamma)$, where $s_0$ is the initial system state, $\Delta$ is the behavioral model of the system and $\Gamma$ is the conditional preference model. We assume that the system evolves with discrete events dynamics and that all time steps have the same duration.

### 3.1 Variables

We use a set of propositional variables $\mathtt{P}$ to describe the state of the system. $\mathtt{P}$ is partitioned into two subsets $\mathtt{O}$ and $\mathtt{E}$, which respectively represent the elements of the system that are observed and those to be estimated. At each discrete time step, given a truth assignment to the variables from $\mathtt{O}$, our goal is to estimate the value for the variables from $\mathtt{E}$.

**Notations** For a variable set $\mathtt{X}$, we define an *assignment* as a function from $\mathtt{X}$ to $\{\top, \bot\}$ that associates to each variable $\mathtt{x}$ from $\mathtt{X}$ the boolean value *true* ($\top$) or *false* ($\bot$). We write $x$ and $\overline{x}$ the assignments to $\{\mathtt{x}\}$ such that $x(\mathtt{x}) = \top$ and $\overline{x}(\mathtt{x}) = \bot$. For a set of variables $\mathtt{X} = \{\mathtt{x_1}, \ldots, \mathtt{x_n}\}$, if for $i \in [1, n]$, $f_i$ is an assignment $\{\mathtt{x_i}\} \to \{\top, \bot\}$, then $f_1\ f_2 \ldots f_n$ denotes the $f$ assignment on $\mathtt{X}$ such that $\forall \mathtt{x_i} \in \mathtt{X}, f(\mathtt{x_i}) = f_i(\mathtt{x_i})$. For example, $a\ \overline{b}\ \overline{c}$ is the assignment to $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ that assigns true to $\mathtt{a}$ and false to $\mathtt{b}$ and $\mathtt{c}$.

**Definition 1** (Observation)**.** *An observation, denoted $o$, is an assignment of the variables of $\mathtt{O}$.*

**Definition 2** (State)**.** *A state, denoted $s$, is an assignment of the variables of $\mathtt{P}$. $\mathcal{S}$ denotes the set of states.*

### 3.2 Behavioral Model

The behavioral model $\Delta \subseteq \mathcal{S}^2$ is the transition relation of the system. In order to refer to the state of the system at the previous time step, we introduce a bijective function $pre$ on the set $\mathtt{P}$. For a variable $\mathtt{p}$ in $\mathtt{P}$, $pre(\mathtt{p})$ represents the value of the variable $\mathtt{p}$ at the previous time step. Formally, we define a variable set $\mathtt{P_{pre}} = \{\mathtt{pre\_p} \,|\, \mathtt{p} \in \mathtt{P}\}$ such as $\forall \mathtt{p} \in \mathtt{P}, pre(\mathtt{p}) = \mathtt{pre\_p}$. $\Delta$ is represented by a set of propositional logic formulas $\Delta_p$ that can relate to both the variables of $\mathtt{P}$ and those of $\mathtt{P_{pre}}$.

A pair of states $(s_{pre}, s_{now})$ belongs to the transition relation $\Delta$ when the system can be in the state $s_{pre}$ at time $t-1$ and in the state $s_{now}$ at time $t$. To formally link $\Delta$ to $\Delta_p$, for any pair of states $(s_{pre}, s_{now})$, we define the assignment $\sigma_{s_{pre}, s_{now}}$ on variables from $\mathtt{P} \cup \mathtt{P_{pre}}$ such as $\forall \mathtt{p} \in \mathtt{P}, \sigma_{s_{pre}, s_{now}}(\mathtt{p}) = s_{now}(\mathtt{p})$ and $\sigma_{s_{pre}, s_{now}}(pre(\mathtt{p})) = s_{pre}(\mathtt{p})$. We consider that a pair of states belongs to the transition relation $\Delta$ if and only if the corresponding assignment satisfies the formulas of $\Delta_p$.

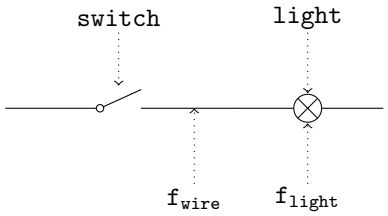


Figure 1: A simple system composed of a lamp and a switch

**Definition 3** (Transition)**.** *A pair of states $(s_{pre}, s_{now}) \in \mathcal{S}^2$ is a transition, denoted $(s_{pre}, s_{now}) \in \Delta$, if and only if $\sigma_{s_{pre}, s_{now}} \models \Delta_p$.*

In the remaining of this paper, we do not distinguish $\Delta$ et $\Delta_p$.

In order to reason about the possible evolutions of the system state, and the associated observation sequences, we define consistent state sequences and consistent observation sequences as follows.

**Definition 4** (Consistent state sequence)**.** *A state sequence $(s_0, s_1, \ldots, s_n) \in \mathcal{S}^n$ is consistent if and only if $\forall i \in [1, n]$, $(s_{i-1}, s_i) \in \Delta$.*

**Definition 5** (Consistent observation sequence)**.** *An observation sequence $(o_0, o_1, \ldots, o_n)$ is consistent if and only if there exist a consistent state sequence $(s_0, s_1, \ldots, s_n)$ such that $\forall i \in [0, n], \forall \mathtt{o} \in \mathtt{O}, s_i(\mathtt{o}) = o_i(\mathtt{o})$.*

In the following, when there is no ambiguity, state and observation sequences are assumed to be consistent.

Given a previous state and an observation of the system, we define the set of candidates for diagnosis as the set of states that are compatible with the previous state, the observation and the behavioral model $\Delta$.

**Definition 6** (Diagnosis candidates)**.** *The set $\mathcal{S}_\Delta(s_{pre}, o)$ of diagnosis candidates for a previous state $s_{pre}$ and an observation $o$ is defined by:*

$$\mathcal{S}_\Delta(s_{pre}, o) = \big\{ s_{now} \in \mathcal{S} \,|\, (s_{pre}, s_{now}) \in \Delta \text{ and } \forall \mathtt{o} \in \mathtt{O}, s_{now}(\mathtt{o}) = o(\mathtt{o}) \big\}$$

**Example 1.** *The system illustrated in Figure 1 is composed of a lamp controlled by a switch. This system may be subject to a permanent fault and an intermittent fault, our goal is to estimate their presence or absence. The set $\mathtt{O}$ is composed of* `light` *which is true when the light is on, false otherwise, and* `switch` *which is true when the switch is closed (current flows), false otherwise. The set $\mathtt{E}$ contains two variables representing the two faults that may occur: $\mathtt{f_{wire}}$ represents an intermittent loose contact in the wire and $\mathtt{f_{light}}$ a permanent lamp failure.*

*The two rules of $\Delta$ represent the following behavior: the lamp glows when the switch is closed and when there is no fault on the wire nor in the lamp ($\delta_1$). If the bulb of the lamp was broken at the previous state, then it is at the present time ($\delta_2$), i.e. the fault $\mathtt{f_{light}}$ is permanent. In the initial state $s_0 =$ light switch $\overline{f_{light}}\ \overline{f_{wire}}$, the lamp glows and there is no fault.*

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

*From the behavioral model $\Delta$, we can calculate diagnosis candidates. For instance, at time step 1, $s_0$ is the previous system state, let us consider the observation $o_1 = \overline{light}$ switch. The candidate states set is $\mathcal{S}_\Delta(s_0, o_1) = \{s_{now}, s'_{now}, s''_{now}\}$ with:*

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

*In the state $s_{now}$, the intermittent fault is present alone; in $s'_{now}$, the permanent fault is present alone; both faults are present in $s''_{now}$.*

### 3.3 Single diagnosis choice

For a given previous state and observation, in order to produce a single diagnosis, we have to choose a single state from all the candidates of $\mathcal{S}_\Delta(s_{pre}, o)$. This choice is dictated by the conditional preference model $\Gamma$ that consists in an ordered set of *conditional preferences*.

A conditional preference relates to a variable to be estimated and indicates under which condition we prefer the diagnoses that assign true of false to this variable to the other diagnoses. A preference is a form of "soft constraint", it is only applied when there exists diagnoses with both values for the variable. A formal definition follows.

**Definition 7** (Conditional preference)**.** *A conditional preference $\gamma$ on a variable* e *of* E*, is denoted* $\texttt{cond} : e \prec \overline{e}$*. The preference's condition* cond *is a propositional formula on* $\texttt{P} \cup \texttt{P}_{\texttt{pre}}$*. The variable* e *is called the preference's target.*

Note that $\texttt{cond} : \overline{e} \prec e$ is equivalent to $\neg\texttt{cond} : e \prec \overline{e}$. For a variable e from E, the conditional preference $\texttt{cond} : e \prec \overline{e}$ expresses a preference for states in which e is true to those where e is false if and only if cond is satisfied.

Formally, a preference $\gamma = \texttt{cond} : e \prec \overline{e}$ defines a partial order $\prec_\gamma$ an equivalence relation $\approx_\gamma$ between pairs of transitions as follows. For all triples $s_{pre}, s, s' \in \mathcal{S}^3$, $\gamma$ strictly prefers the transition $(s_{pre}, s)$ to transition $(s_{pre}, s')$ (denoted $(s_{pre}, s) \prec_\gamma (s_{pre}, s')$) if and only if $\sigma_{s_{pre},s} \models \texttt{cond} \leftrightarrow \texttt{e}$ and $\sigma_{s_{pre},s'} \not\models \texttt{cond} \leftrightarrow \texttt{e}$. Transitions $(s_{pre}, s)$ et $(s_{pre}, s')$ are equivalent to $\gamma$ (denoted $(s_{pre}, s) \approx_\gamma (s_{pre}, s')$) if and only if $(\sigma_{s_{pre},s} \models \texttt{cond} \leftrightarrow \texttt{e}) \Leftrightarrow (\sigma_{s_{pre},s'} \models \texttt{cond} \leftrightarrow \texttt{e})$.

**Example 2.** *Let us consider the preference $\gamma = \neg\texttt{light} : f_{light} \prec \overline{f_{light}}$. This preference indicates that if both $f_{light}$ and $\overline{f_{light}}$ are part of some diagnosis candidate, then $f_{light}$ is preferred is and only if* $\neg\texttt{light}$ *holds. Formally, we prefer diagnoses that satisfy $\neg\texttt{light} \leftrightarrow \texttt{f}_{\texttt{light}}$. For the states $s_{now}$, $s'_{now}$ and $s''_{now}$ in example 1, as* $\neg\texttt{light}$ *holds, we prefer the states in which $\texttt{f}_{\texttt{light}}$ holds, i.e. the states $s'_{now}$ and $s''_{now}$. Formally, $s'_{now} \prec_\gamma s_{now}$, $s''_{now} \prec_\gamma s_{now}$ and $s''_{now} \approx_\gamma s'_{now}$.*

We assume that in $\Gamma$, all estimated variables are the target of a conditional preference. This raises the question of the order in which preferences are applied, that we address now.

**Definition 8** (Conditional preference model)**.** *A conditional preference model $\Gamma$ is a sequence of conditional preferences $(\gamma_1, \gamma_2, .., \gamma_n)$ with $\gamma_i = \texttt{cond}_\texttt{i} : e_i \prec \overline{e_i}$ for $i \in [1, n]$ such that each variable* e *from* E *is the target of exactly one preference.*

We require $\Gamma$ to be an acyclic preference model, which guarantees its consistence (see [11]). For example, the following two preferences form a cycle: $\texttt{a} : b \prec \overline{b}$ et $\texttt{b} : a \prec \overline{a}$. In the first preference the value of a depends on that of b, and in the second preference the exact opposite happens. While the literature contains work on cyclic preference networks [11], in this paper we assume that $\Gamma$ is acyclic. This means that the condition of a preference $\gamma_i$ cannot use variables that are the target of the following preferences in the sequence. Formally, $\forall i \in [1, n]$ , the scope of the condition $\texttt{cond}_\texttt{i}$ is a subset of $\texttt{P}_{\texttt{pre}} \cup \texttt{O} \cup \{\texttt{e}_\texttt{j} | 1 \leq j < i\}$.

From a preference model $\Gamma$, it is possible to define a partial order $\prec_\Gamma$ between pairs of states as follows. Intuitively, as in a lexicographic order, we consider preferences $\gamma_i$ in their index order in $\Gamma$ and we apply at each index the order relation $\prec_{\gamma_i}$ defined above. This order is partial because it compares only pairs of transitions $(s_{pre}, s_{now})$ and $(s_{pre}{}', s_{now}{}')$ that have the same previous state (*i.e.* $s_{pre} = s_{pre}{}'$) and whose successor states produce the same observation (*i.e.* $\forall \texttt{o} \in \texttt{O} s_{now}(\texttt{o}) = s_{now}{}'(\texttt{o})$).

Formally, $\forall s_{pre}, s, s' \in \mathcal{S}^3$, $(s_{pre}, s)$ is strictly preferred to $(s_{pre}, s')$ by $\Gamma$ (denoted $(s_{pre}, s) \prec_\Gamma (s_{pre}, s')$) if and only if there exists $i \in [1, n]$ such that for all $j < i$, $(s_{pre}, s) \approx_{\gamma_j} (s_{pre}, s')$ and $(s_{pre}, s) \prec_{\gamma_i} (s_{pre}, s')$.

Although the $\prec_\Gamma$ order is partial on $\mathcal{S}^2$, it is complete on any set of diagnosis candidates. We use this order relation to define the preferred diagnosis at each time step.

**Proposition 1.** *Let $s_{pre}$ be a state, $o$ an observation, $\Delta$ a behavioral model and $\Gamma$ a preference model. If $s$ and $s'$ are two candidate states for $s_{pre}$ et $o$ $(s, s' \in \mathcal{S}_\Delta(s_{pre}, o)^2)$ such that $s \neq s'$ then we have either $(s_{pre}, s) \prec_\Gamma (s_{pre}, s')$ or $(s_{pre}, s') \prec_\Gamma (s_{pre}, s)$.*

**Proof** By definition 6, $s$ and $s'$ belong to $\mathcal{S}_\Delta(s_{pre}, o)$ implies that $\forall \texttt{o} \in \texttt{O}, s(\texttt{o}) = s'(\texttt{o})$. Therefore, $s \neq s'$ means that there is a variable $\texttt{e} \in \texttt{E}$ such that $s(\texttt{e}) \neq s'(\texttt{e})$. So, there exists a preference $\gamma \in \Gamma$ such that $s \approx_\gamma s'$ does not hold. Let $i$ be the index of the first preference of the sequence in this case. Formally, $\gamma_i$ is such that $\forall j < i$, $(s_{pre}, s) \approx_{\gamma_j} (s_{pre}, s')$, $(s_{pre}, s) \approx_{\gamma_i} (s_{pre}, s')$ does not hold. This means that $(s_{pre}, s) \prec_{\gamma_i} (s_{pre}, s')$ or $(s_{pre}, s') \prec_{\gamma_i} (s_{pre}, s)$. From the order $\prec_\Gamma$, we then have $(s_{pre}, s) \prec_\Gamma (s_{pre}, s')$ or $(s_{pre}, s') \prec_\Gamma (s_{pre}, s)$. □

### 3.4 Estimation process

At each step, given the set of candidate states $\mathcal{S}_\Delta(s_{pre}, o)$ we select the state preferred by the conditional preference model $\Gamma$.

**Definition 9** (Estimated state)**.** *The estimated state $\hat{s} = estim(s_{pre}, o)$ for a previous state $s_{pre}$ and an observation $o$ is the element of $\mathcal{S}_\Delta(s_{pre}, o)$ preferred by $\prec_\Gamma$. Formally, $\hat{s} = estim(s_{pre}, o)$ if and only if:*

1. *$\hat{s} \in \mathcal{S}_\Delta(s_{pre}, o)$, and*
2. *$\forall s_{now} \in \mathcal{S}_\Delta(s_{pre}, o)$ such that $s_{now} \neq \hat{s}$, we have $(s_{pre}, \hat{s}) \prec_\Gamma (s_{pre}, s_{now})$.*

From the initial state $s_0$, we now define the sequence of estimated states that is produced by the diagnoser for a given observation sequence.

**Definition 10** (Estimated state sequence)**.** *A state sequence* $(s_0, \hat{s}_1, \hat{s}_2, ..., \hat{s}_k)$ *is the estimated state sequence for the observation sequence* $(o_0, o_1, o_2,...,o_k)$ *if and only if* $\forall i \in [1, k], \hat{s}_i = estim(\hat{s}_{i-1}, o_i)$.

**Example 3.** *Let us associate the lamp model* $\Delta$ *from example 1 to the following preferences:*

$$\Gamma = \left( \begin{array}{rl} \texttt{pre\_f}_{\texttt{wire}} : f_{wire} \prec \overline{f_{wire}} & (\gamma_1) \\ \bot : f_{light} \prec \overline{f_{light}} & (\gamma_2) \end{array} \right)$$

*The first preference* $(\gamma_1)$ *declares that we prefer the value for* $\texttt{f}_{\texttt{wire}}$ *that was estimated at the previous time step. This mechanism makes it possible to bring some stability to the diagnosis since* $\texttt{f}_{\texttt{wire}}$ *is an intermittent fault: if* $\Delta$ *allows both diagnoses for* $\texttt{f}_{\texttt{wire}}$*, we prefer to maintain the diagnosis that was chosen at the previous time step.*

*The preference* $(\gamma_2)$ *indicates that we always prefer to assume that* $\texttt{f}_{\texttt{light}}$ *is absent.*

*In example 1, with* $s_0 = light\ switch\ \overline{f_{light}}\ \overline{f_{wire}}$*) as the previous state and* $o_1 = \overline{light}$ *switch as the observation, the set of diagnosis candidates* $\mathcal{S}_\Delta(s_{pre}, o)$ *contains the three states* $s_{now} = \overline{light}$ *switch* $\overline{f_{light}}\ f_{wire}$*,* $s_{now}{}' = \overline{light}$ *switch* $f_{light}\ \overline{f_{wire}}$ *and* $s''_{now} = \overline{light}$ *switch* $f_{light}\ f_{wire}$*.*

*By applying* $(\gamma_1)$ *first, as* $\overline{pre\_f_{wire}}$ *holds in the three states, we prefer the states in which* $\overline{f_{wire}}$*. In our case, only* $s_{now}{}'$ *satisfies this criterion. As there is only one diagnosis left, we do not need to apply preference* $(\gamma_2)$*. The preferred single state is* $\hat{s}_1 = estim(s_0, o_1) = s_{now}{}' = \overline{light}\ switch\ f_{light}\ \overline{f_{wire}}$*.*

# 4 Deadlock

A deadlock situation occurs when the diagnoser has previously estimated a state $\hat{s}$ different from the current system state $s$, and receives an observation $o$ in contradiction with $\hat{s}$ and $\Delta$. In such a situation the candidates set for $\hat{s}$ and $o$ is empty and the diagnoser is unable to return a diagnosis.

**Definition 11** (Deadlock)**.** *An estimation model* $(s_0, \Delta, \Gamma)$ *is in a deadlock situation for an observation sequence* $(o_0, o_1, o_2, \ldots, o_k)$ *with* $k > 1$ [1] *if and only if:*

1. *there exists an estimated state sequence for* $(o_0, \ldots, o_{k-1})$*, and*
2. *there exist no estimated state sequence for* $(o_0, \ldots, o_k)$

**Example 4.** *Let* $(o_0, o_1, o_2, o_3)$ *be the observation sequence produced by the system and along with the associated state sequence* $(s_0, s_1, s_2, s_3)$ *described in Figure 2.*

[1] There can be no deadlock at $\hat{s}_1$ because we assume that the diagnoser is correctly initialized at $s_0$.

| Step $i$ | $o_i$ | $s_i$ | $\hat{s}_i$ |
|---|---|---|---|
| 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: Deadlock scenario for example 4. In columns $s_i$ and $\hat{s}_i$, we omit variables from O whose values are identical to the column $o_i$, and we only represent variables from E).

*When observation* $o_1$ *is received,* $\Gamma$ *selects the preferred state* $\hat{s_1}$ *=* $light\ switch\ \overline{f_{light}}\ \overline{f_{wire}}$*. Then for observation* $o_2$*, we are in the situation described in Example 3 and the preferred state is* $\hat{s}_2$ *=* $\overline{light}\ switch\ f_{light}\ \overline{f_{wire}}$*. Observation* $o_3$ *is in contradiction with the previously estimated state. In fact the estimator has estimated* $f_{light}$ *in the previous state, meaning that variable* $\texttt{pre\_f}_{\texttt{light}}$ *is true. Rules of* $\Delta$ *are not consistent with such a configuration:* $(\delta_1)$ *requires* $\texttt{f}_{\texttt{light}}$ *to be false because the lamp glows while* $(\delta_2)$ *requires* $\texttt{f}_{\texttt{light}}$ *to be true since the associated fault is permanent.*

*Therefore, there is an estimated state sequence for* $(o_0, o_1, o_2)$*: this is the sequence* $(s_0, \hat{s}_1, \hat{s}_2)$*. However, since there is no estimated state sequence for* $(o_0, o_1, o_2, o_3)$*, the pair system-diagnoser is in a deadlock situation for this observation sequence.*

In the previous example, the deadlock situation could be easily avoided by changing the order or conditions of preferences. However, as soon as one is interested in more complex systems, it becomes difficult to anticipate the deadlock situations and even more to solve them. In this paper, we focus on detecting these deadlocks and leave their resolution for future work.

# 5 Deadlock Checking

In this section, we show how to use a model-checker to identify deadlock scenarios at the design phase of the diagnostic model.

## 5.1 Deadlock checker

The verification method we propose is inspired from the Twin-Plant [12] technique, which makes it possible to verify the diagnosability of a system by constructing the synchronous product of two finite state machines.

Our method is similar since it involves constructing a deadlock verifier by synchronizing two state machines. The first state machine represents the system and is constrained only by the transition relation $\Delta$. We use it to generate consistent observation sequences. The second state machine is the diagnoser built from the whole estimation model. It is also constrained by $\Delta$, but it deterministically selects the next state by applying $\Gamma$ preferences. The verifier is the product of these two state machines synchronized on observable variables at each time step. Deadlock checking then consists in checking whether there exists an observation sequence that leads the verifier to a state in which the system has a successor state, but the diagnoser has none.

In order to distinguish the states of the two state machines presented above, *i.e.* the state of the system and that of the diagnoser, we introduce two variable sets $\texttt{P}^{\texttt{sys}}$ and $\texttt{P}^{\texttt{est}}$ that are direct copies of P. We also use a est_sat variable that indicates whether the diagnoser has a successor state.

**Definition 12** (Verifier variables)**.** *The set of the verifier variables is defined by* $\texttt{P}^{\texttt{verif}} = \texttt{P}^{\texttt{sys}} \cup \texttt{P}^{\texttt{est}} \cup \{\texttt{est\_sat}\}$*, where :*

- $\texttt{P}^{\texttt{sys}} = \{\texttt{p\_sys} \mid \texttt{p} \in \texttt{P}\}$ *is the set of variables describing the system state ;*
- $\texttt{P}^{\texttt{est}} = \{\texttt{p\_est} \mid \texttt{p} \in \texttt{P}\}$ *is the set of variables describing the diagnoser state;*
- est_sat *indicates whether there is an estimated state for the diagnoser.*

*A state of the verifier is an assignment on variables of* $\texttt{P}^{\texttt{verif}}$*.*

In order to define the state machine resulting from the synchronous product, we successively define the initial state of the verifier (definition 13) and its transition relation (definition 14).

**Definition 13** (Verifier initial state)**.** *The initial state $s_0^{verif}$ of the verifier is such that:*

- $\forall \mathtt{p} \in \mathtt{P},\ s_0^{verif}(\mathtt{p\_sys}) = s_0(\mathtt{p})$,
- $\forall \mathtt{p} \in \mathtt{P},\ s_0^{verif}(\mathtt{p\_est}) = s_0(\mathtt{p})$,
- $s_0^{verif}(\mathtt{est\_sat}) = \top$.

**Definition 14** (Verifier transition relation)**.** *Let $s_{pre}^{verif}$ and $s_{now}^{verif}$ be two verifier states. The pair $(s_{pre}^{verif}, s_{now}^{verif})$ is a verifier transition if and only if:*

*(1) variables associated with observations take the same value on system and diagnoser sides,*

*(2) the system transition described by variables of $\mathtt{P^{sys}}$ satisfies $\Delta$,*

*(3) the variable* est_sat *indicates whether there exists a possible estimated state (i.e. if the candidates set is not empty), and*

*(4) if* est_sat *is true, then the diagnoser transition described by variables of $\mathtt{P^{est}}$ satisfies $\Delta$ and $\Gamma$.*

*Formally :*

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

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

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

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

**Proposition 2.** *An estimation model $(s_0, \Delta, \Gamma)$ is subject to a deadlock if and only if the associated verifier contains a path in which* est_sat *is false. The deadlock scenario is the observation sequence corresponding to the states of the verifier.*

**Proof** Suppose that the estimation model is subject to a deadlock. According to Definition 11, there exists an observation sequence $(o_0, o_1, \ldots, o_n)$ generated by a consistent state sequence $(s_0, s_1, \ldots, s_n)$, such that there is an estimated state sequence $(\hat{s}_0, \hat{s}_1, \ldots, \hat{s}_{n-1})$ for the partial sequence $(o_0, o_1, \ldots, o_{n-1})$ and that there does not exist an estimated state sequence $(\hat{s}_0, \hat{s}_1, \ldots, \hat{s}_n)$ for the complete sequence. We then build the verifier's state sequence $(s_0^{verif}, s_1^{verif}, \ldots, s_n^{verif})$ as follows:

- $\forall i \in [0, n], \forall p \in \mathtt{P},\ s_i^{verif}(\mathtt{p\_sys}) = s_i(\mathtt{p})$,
- $\forall i \in [0, n-1], \forall p \in \mathtt{P},\ s_i^{verif}(\mathtt{p\_est}) = \hat{s}_i(\mathtt{p})$,
- $\forall i \in [0, n-1],\ s_i^{verif}(\mathtt{est\_sat}) = \top$,
- $\forall \mathtt{p} \in \mathtt{O},\ s_n^{verif}(\mathtt{p\_est}) = o_n(\mathtt{p})$,
- $\forall \mathtt{p} \in \mathtt{E},\ s_n^{verif}(\mathtt{p\_est}) = \top$,
- $s_n^{verif}(\mathtt{est\_sat}) = \bot$.

We show that for all $i \in [0, n-1]$, $(s_i^{verif}, s_{i+1}^{verif})$ is a transition of the checker: let $i$ be an integer in $[0, n-1]$, we show that the four parts of Definition 14 are satisfied:

- $\forall \mathtt{o} \in \mathtt{O},\ \forall p \in \mathtt{P}, s_i(\mathtt{p}) = \hat{s}_i(\mathtt{p})$ and $s_{i+1}(\mathtt{p}) = \hat{s}_{i+1}(\mathtt{p})$. Equation (1) holds;
- equation (2) holds by construction of the verifier;
- for $i < n-1$, $s_i^{verif}(\mathtt{est\_sat}) = \top$ and $(\hat{s}_i, \hat{s}_{i+1})$ is the pair of states in $\Delta$ satisfying equation (3);
- for $i = n-1$, the definition of a deadlock implies that there does not exist a pair of states satisfying $\Delta$: equation (3) is satisfied;
- for $i < n-1$, $s_i^{verif}(\mathtt{est\_sat}) = \top$ and $(\hat{s}_i, \hat{s}_{i+1})$ is the pair of states in $\Delta$ satisfying equation (4);

For the reciprocal, let us assume that the checker has a path in which est_sat is false. We consider such a path $(s_0^{verif}, s_1^{verif}, \ldots, s_n^{verif})$ in wich $s_n^{verif}$ is the only state in which false is assigned to est_sat. We then build two states sequences $(s_0, s_1, \ldots, s_n)$ and $(s_0, \hat{s}_1, \ldots, \hat{s}_{n-1})$ as follows:

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

We prove that these state sequences are consistent with $\Delta$. From definition 14, est_sat is false (last time step) implies that no state $s_{now}$ allows to satisfy the right part of the condition (equation 3) for $s_{pre} = \hat{s}_{n-1}$. This means that no state meets the definition 6. There is a deadlock for the observation sequence generated by $(s_0, s_1, \ldots, s_n)$. □

## 5.2 Choice of the model-checker

Model-checking is a set of techniques and computer tools for checking properties on systems. Among the many model-checkers in the literature, NuSMV [13] and NuXMV are known for their effectiveness in checking temporal properties on dynamic systems. They are based on temporal logic like LTL or CTL. Another family of model checkers is specialized in checking properties on structurally complex models, but without temporal dynamics. This is the case of the Alloy Analyzer model-checker [14] in which the models are based on the Alloy language, itself based on first order logic.

In the case of deadlock verification, we need to express the temporal dynamics of the system but also the preferences of the diagnoser, which are not easily expressed in propositional logic. We therefore chose to use the model-checker ELECTRUM [1] which is based on the Alloy language and integrates temporal dynamics as in NuSMV.

## 5.3 Quantified form of preference

The concept of preference is part of the verifier definition. This means that the model-checker language must allow to express them. To do so, we define a *quantified form* for preferences.

**Definition 15** (Preference quantified form)**.** *Let be* $\Gamma = (\gamma_1, \ldots, \gamma_n)$ *a preference model in which each preference has form* $\gamma_i = \mathtt{cond_i} : e_i \prec \overline{e_i}$*. The quantified form of* $\gamma_i$ *is the formula* $\psi_i$ *defined by:*

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

The quantified form $\psi_i$ is composed of two parts. The left part is only satisfied when for both truth values variable $\mathtt{e_i}$, there is a way to assign the target variables of the following preferences and still satisfy $\Delta$. It represents the cases when the preference is actually applied. The right part expresses the effect of the preference, *i.e* that $\mathtt{e_i}$ is assigned to true if and only if the condition $\mathtt{cond_i}$ is satisfied. Through the following proposition, we show that $\Gamma$ preferences and their quantified form are equivalent from the point of view of the estimated state for a previous state and an observation.

**Proposition 3.** *For a previous estimated state* $\hat{s}_{pre}$*, an observation* $o$*, a candidate* $\hat{s} \in \mathcal{S}_\Delta(\hat{s}_{pre}, o)$*,* $\hat{s} = estim(\hat{s}_{pre}, o)$ *if and only if the assignment* $\sigma_{\hat{s}_{pre},\hat{s}}$ *satisfies conjunction* $\bigwedge_{i\in[1,n]} \psi_i$*.*

**Proof.** Let $\hat{s}_{pre}$ be a previous estimated state, $o$ an observation and $\hat{s}$ a state in $\mathcal{S}_\Delta(\hat{s}_{pre}, o)$. We inductively show that $\hat{s}$ is preferred for the preference sequence $(\gamma_1, \ldots, \gamma_k)$ if and only if the assignment $\sigma_{\hat{s}_{pre},\hat{s}}$ satisfies $\bigwedge_{i\in[1,k]} \psi_i$.

For $k = 1$, the set of free variables in formula $lhs_1 = \forall \mathtt{e_1}, \exists \mathtt{e_2}, \ldots \mathtt{e_n}, \Delta$ is the set $\mathtt{O}$, since all the variables in $\mathtt{E}$ are quantified. The formula $lhs_1$ is satisfied by an observation $o$ if and only if both $oe_1$ and $o\overline{e_1}$ have an extension in $\mathtt{P} \cup \mathtt{P_{pre}}$ that satisfies $\Delta$. This means $\mathcal{S}_\Delta(\hat{s}_{pre}, o)$ contains at least two states $s$ and $s'$ with $s(\mathtt{e_1}) \neq s'(\mathtt{e_1})$. Among these two states, the one that satisfies $\mathtt{cond_1} \leftrightarrow \mathtt{e_1}$ is the preferred state by preference $\gamma_1$, and also satisfies $\psi_1$. As the states preferred by $\gamma_1$ are exactly the ones for which the assignment $\sigma_{\hat{s}_{pre},\hat{s}}$ satisfies $\psi_1$, then the proposition holds for $k = 1$.

Let $k > 1$ and let us assume that the induction is valid at index $k-1$, *i.e.* that $\hat{s}$ is preferred for the preference sequence $(\gamma_1, \ldots, \gamma_{k-1})$ if and only if the assignment $\sigma_{\hat{s}_{pre},\hat{s}}$ satisfies $\bigwedge_{i\in[1,k-1]} \psi_i$. Since $s$ and $s'$ belong to $\mathcal{S}_\Delta(\hat{s}_{pre}, o)$, since $s$ is a state preferred for $(\gamma_1, \ldots, \gamma_{k-1})$ and since $\sigma_{\hat{s}_{pre},s'}$ satisfies $\bigwedge_{i\in[1,k-1]} \psi_i$, then for all $i < k$, $s(\mathtt{e_i}) = s'(\mathtt{e_i})$.

The set of free variables in formula $lhs_k = \forall \mathtt{e_k}, \exists \mathtt{e_{k+1}}, \ldots \mathtt{e_n}, \Delta$ is the set $\mathtt{O} \cup \{\mathtt{e_i} \mid i < k\}$. An assignment *op* to these variables satisfies $lhs_k$ if and only if *op* $e_k$ and *op* $\overline{e_k}$ both have an extension to $\mathtt{P} \cup \mathtt{P_{pre}}$ that satisfies $\Delta$.

If for a given assignment *op* to $\mathtt{O} \cup \{\mathtt{e_i} \mid i < k\}$ the formula $lhs_k$ does not hold, then for all $s$ and $s'$ in $\mathcal{S}_\Delta(\hat{s}_{pre}, o)$, $s(\mathtt{e_k}) = s'(\mathtt{e_k})$. Thus, the preference $\gamma_k$ is not applied and the preferred states for $(\gamma_1, \ldots, \gamma_{k-1})$ are the also preferred for $(\gamma_1, \ldots, \gamma_k)$. In this case, $\psi_k = \top$ and the states $s'$ such that $\sigma_{\hat{s}_{pre},s'}$ satisfy $\bigwedge_{i\in[1,k-1]} \psi_i$ are the same ones for which $\sigma_{\hat{s}_{pre},s'}$ satisfy $\bigwedge_{i\in[1,k]} \psi_i$ and the induction is valid.

If for a given assignment *op* to $\mathtt{O} \cup \{\mathtt{e_i} \mid i < k\}$ the formula $lhs_k$ holds, then $\mathcal{S}_\Delta(\hat{s}_{pre}, o)$ contains at least two states $s$ and $s'$ such that for all $i < k$, $s(\mathtt{e_i}) = s'(\mathtt{e_i})$ and $s(\mathtt{e_k}) \neq s'(\mathtt{e_k})$. Between these two states, the one in which $\mathtt{cond_k} \leftrightarrow \mathtt{e_k}$ is true is the preferred state by preference $\gamma_k$. As the states preferred by $\gamma_k$ are exactly the ones for which the assignment $\sigma_{\hat{s}_{pre},\hat{s}}$ satisfies $\psi_k$, thus the induction is valid at index $k$. □

```
pred delta[fwire,light,flight,
           switch,preF_flight : Bool] {
  (isTrue[preF_flight] => isTrue[flight])
  and
  (isTrue[light] <=> ( isTrue[switch] and
                       !isTrue[flight] and
                       !isTrue[fwire]))
}
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: The observed system and observation synchronization in ELECTRUM.

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

Figure 4: Temporal dynamics in ELECTRUM.

## 5.4 Model Encoding with ELECTRUM

In this section, we illustrate how the verifier defined in Definition 13 and 14 is encoded in ELECTRUM. We illustrate the encoding of the system described in examples 1 and 3.

To model the system's state machine, we declare a predicate corresponding exactly to the model $\Delta$ then we declare a constraint (a `fact` in ELECTRUM) that specifies that the system respects the predicate at all time steps (`always` in ELECTRUM), as illustrated in figure 3. A similar constraint is declared for the diagnoser's state machine. Finally, we synchronize the variables from `O` of the observed system with those of the second state machine representing our estimator, as expressed in condition (1) of definition 14.

To represent the temporal aspect of our formalism, we use the operator `'` from ELECTRUM which describes a variable at the next time step. Figure 4 reproduces the intention of the bijective function $pre$ (`p_pre` value at next state is equal to `p` value at current state).

ELECTRUM supports the use ofrelational algebra operators. It is possible for each preference to write a predicate

```
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_fwire_applied{
  isTrue[Estimator.fwire] <=>
    isTrue[Estimator.preF_fwire]
}
pred pref_flight_possible{
  all flight : Bool |
    delta[Estimator.fwire,
    Estimator.preF_fwire,
    Estimator.light,
    flight,
    Estimator.switch,
    Estimator.preF_flight]
}
pred pref_flight_applied{
  isTrue[Estimator.flight] <=>
    isTrue[False]
}
fact prefs {
  always {
    pref_flight_possible implies
                  pref_flight_applied
    pref_fwire_possible implies
                  pref_fwire_applied
  }
}
```

Figure 5: Implementing preferences in ELECTRUM

telling us if a preference $\gamma_i$ is applicable, that is, if the two valuations for variable $\mathtt{e_i}$ (operator `all`), and at least one possible valuation for variables $\mathtt{e_j}, i < j \leq n$ (operator `some`) are consistent with $\Delta$ an the observation. This corresponds to the left part of the quantified form of a preference. Figure 5 illustrates the definition of predicates that implement these conditions with ELECTRUM operators. The rest of the preferences are described by a fact.

The order in which preferences are applied in $\Gamma$, is completely induced by the quantification overlays in the quantified forms of preferences (see definitions 8 and 15).

## 6 Experimental results

We experimented our approach on the model of a multi-robot mission of customizable size and complexity.

In this mission, one or more robots move on a rectangular grid of variable size, each robot moves towards its destination. The destination can change during the mission according to unspecified dynamics.

Behavioral constraints are declared in $\Delta$ for each robot. robots should move unless they are at their destination. Moreover, robots may be subject to permanent and / or intermittent faults. An intermittent fault slows down or immobilizes the robot for a few moments which is modelled by the fact that a robot takes several time steps to cross a cell. A permanent fault immobilizes the robot permanently. We also estimate the state of the terrain. A robot crosses a *normal* cell in one time step, but we introduce *difficult* cells (the robot takes several time steps to go through the cell) and *dangerous* cells (robot stays forever on the cell).

We observe at every moment the position of robots on the grid and their destination. We estimate the presence of intermittent and permanent faults on each robot, a variable indicating if it can move, as well as the dangerousness of each grid cell. We implement two estimation strategies, one subject to deadlock, the other not (for example always preferring the absence of a permanent fault).

Models are written in Scala language and are automatically translated into ELECTRUM. ELECTRUM proceeds to the verification through different solvers. For our experiments, we set ELECTRUM to use the Sat4j solver [15]. Verifications were performed on a 3.5GHz Intel Core i5-7600 quad-core processor. Figure 6 shows the results of our experiments for different mission parameters.

The first conclusion is that verification is faster for models that are subject to deadlock scenarios than for those that do not encounter such scenarios. This difference is directly related to the fact that when the model-checker finds a deadlock scenario in the model, the search ends immediately and it returns a counterexample corresponding to the sequence of observations. On the contrary, to prove that a model contains no deadlock scenarios, ELECTRUM explores many more possible executions paths for the verifier.

We can then see that the verification is very fast for the first models, but as we enrich P, $\Delta$ et $\Gamma$, the number of variables generated for Sat4j and the time required for verification increases exponentially. For examples of larger sizes, it is interesting to note that Sat4j did not have enough memory to process them.

Deadlock detection as proposed in the paper is carried out in the design phase. Hence, high calculation times are not necessarily unacceptable and do not call into question the approach we propose. Let us notice that the results we present here are preliminary results and we are currently working on different ways to improve them. Specifically, we aim to integrate ELECTRUM more closely to avoid the creation of boolean variables where ELECTRUM could handle enumerated variables.

## 7 Conclusion and future work

We have developed a generic method to validate the proper behavior of a diagnoser at design stage. In particular, we have showed that it is possible to anticipate deadlock situations.

In this paper, we do not introduce any mechanism to identify the causes and modify the estimation model in order to eliminate deadlock scenarios, but this remains a goal for future work.

In addition, we plan to reuse the automation of model-checking allowing us to verify the existence of deadlock scenarios for our diagnosers to check other properties related to diagnosis such as diagnosability.

Finally, progress on model-checker performance is needed to apply it to multi-robot missions involving large models. QBF solvers [16] are potential tools for finding deadlock scenarios of bounded length.

## References

[1] Nuno Macedo, Julien Brunel, David Chemouil, Alcino Cunha, and Denis Kuperberg. Lightweight specification and analysis of dynamic systems with rich con-

| ID | nb of robots | grid size | $\|\mathtt{P}\|$ | $\|\Delta\|$ | $\|\Gamma\|$ | CNF vars | CNF clauses | deadlock | no deadlock |
|---|---|---|---|---|---|---|---|---|---|
| 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: ELECTRUM model checking. Columns $\|\mathtt{P}\|$, $\|\Delta\|$ et $\|\Gamma\|$ indicate respectively the number of variables of P, the number of rules of $\Delta$ and the number of preferences in $\Gamma$. Columns 7 and 8 indicate respectively the number of variables and clauses sent to solver Sat4j by ELECTRUM. The last two columns indicate the verification time for a version of the model subject to a deadlock scenario and another version for which such a scenario does not exist.


figurations. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2016, pages 373–383, New York, NY, USA, 2016.

[2] Meera Sampath, Raja Sengupta, Stéphane Lafortune, Kasim Sinnamohideen, and Demosthenis Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on automatic control*, 40(9):1555–1575, 1995.

[3] Craig Boutilier, Ronen I Brafman, Carmel Domshlak, Holger H Hoos, and David Poole. Preference-based constrained optimization with cp-nets. *Computational Intelligence*, 20(2):137–157, 2004.

[4] Cedric Pralet, Xavier Pucel, and Stéphanie Roussel. Diagnosis of intermittent faults with conditional preferences. In *Proceedings of the 27th International Workshop on Principles of Diagnosis (DX'16)*, 2016.

[5] Xavier Pucel and Stéphanie Roussel. Intermittent fault diagnosis as discrete signal estimation: Trackability analysis. In *Proceedings of the 28th International Workshop on Principles of Diagnosis (DX'17)*, 2017.

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

[7] Marie-Odile Cordier, Philippe Dague, François Lévy, Jacky Montmain, Marcel Staroswiecki, and 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.

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

[9] Johan De Kleer. Diagnosing multiple persistent and intermittent faults. In *IJCAI*, pages 733–738, 2009.

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

[11] Craig Boutilier, Ronen I Brafman, Carmel Domshlak, Holger H Hoos, and 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.

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

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

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

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

[16] Massimo Narizzano, Luca Pulina, and Armando Tacchella. The qbfeval web portal. In *Logics in Artificial Intelligence*, pages 494–497. Springer Berlin Heidelberg, 2006.