Kalpa Publications in Computing
Volume 4, 2018, Pages 234–247
28th International Workshop on
Principles of Diagnosis (DX'17)

Intermittent Fault Diagnosis as Discrete Signal Estimation: Trackability analysis*

Xavier Pucel[1] and Stéphanie Roussel[1]

Department of Information Processing and Systems (DTIS),
ONERA, Toulouse, France
Firstname.Lastname@onera.fr

Abstract

We address the problem of intermittent fault diagnosis as an instance of discrete signal estimation, in the context of fault management in autonomous systems and autonomous vehicles. We propose an estimation approach based on constrained optimization using conditional preference theories. We show that in some cases, our estimator can fail to find an estimation for the system. We provide a way to detect and eliminate these cases at design time.

1 Introduction

Autonomous systems and autonomous vehicles such as drones and rovers face the challenge of managing aleas such as physical degradation, software bugs, or environmental perturbations, that can prevent them from reaching their goal. More precisely, aleas degrade the performance of some of the system's functions, which impacts the success of the system's actions. For example, the action of moving to a specific place can either succeed properly, succeed with delay, or fail. This action's outcome depends on the propulsion function and on the self localization function. These functions themselves depend on other functions such as power supply, actuators and sensors, etc. The action's outcome also depends on environmental conditions such as ground quality (slippery, sandy), or GNSS signal quality.

Informally, alea management is usually decomposed into the following steps: (1) detection of abnormal behaviour, (2) explanation of this anomaly, (3) evaluation of the performance of each relevant system function and the expected outcome of each relevant system action, and (4) possible reconfiguration of the autonomous mission. In this paper, we address steps (1) to (3) as an estimation problem of discrete-valued signals.

In our experience, it is very impossible to assess with utmost certainty whether an alea is due to a cause internal to the system, or external. It is also usually impossible to assess with total certainty whether its effects will be temporary or permanent. Robot operators usually manage aleas with a "try and see" approach as long as it is safe for the robot and the environment, and their decisions are educated guesses more often than logical inferences. Alea simply disappearing with no reason are, in practice,

*The research leading to the presented results has been partially funded by the SWARMs European Project (Smart and Networking Underwater Robots in Cooperation Meshes), under Grant Agreement No. 662107-SWARMs-ECSEL-2014-1.

as frequent as them appearing. This is why be put a lot of emphasis on intermittent fault modelling and diagnosis.

Our approach consists in modelling the expected performance of each function with a discrete-valued signal (e.g. `nominal` / `degraded` / `lost`). We use a discrete-event model to describe the dynamics behind the performance variations of each function, that is used for finding possible explanations for the system's observable behaviour. This task, usually called model-based diagnosis, can be viewed as an instance of discrete-valued signal estimation: boolean or discrete signals are used to model the occurrence of fault events, the health status of components, the performance of each function, the outcome of each action, etc. The behavioural model constrains the values that each signal can take as function of each other's values, and of their own past values as well. The diagnosis problem consists in estimating the value of some signals while measuring the value of other signals. These signals can represent very different quantities:

- the system's internal state, and its inputs;
- the health status of the system components;
- the expected performance of the system's functions (power supply, movement, sensors, etc);
- the expected outcome of actions (data acquisition, communication, etc).

Thus, the estimation approach can be used for diagnosis purposes, but can also provide some forecast mechanism about the mission success, if the model contains some knowledge about this.

In most practical applications, there are constantly several consistent values for most signals to be estimated. Our approach is to select one value among the consistent ones using a preference model specified by the system's designer. This approach has two advantages. First, a single, clear situation assessment simplifies the decision process that is usually performed by a planner. Second, a preference model lets the designer control whether to select optimistic estimations that will let the system try to fulfil its mission, or pessimistic estimations that will encourage conservative behaviour.

The diagnosis approach presented here has been described in [1], as well as some limitations, the most important one being the possibility that the estimator diverges from the real system state and as a consequence fails to produce an estimation. In this paper we adress the problem of detecting and eliminating such failures at design time.

The paper is organized as follows. First, the estimation model is described, and the notions of preference is formally defined. Then, the state estimation problem is defined, and an estimator synthesis method is presented. Third, the problem of potential deadlocks is explained, and an approach based on model-checking is presented for detecting such deadlocks. Finally, general hints for deadlock elimination are discussed.

2 Estimation model

The estimation model is based on a set S of system states. The model is a triple (s_0, Δ, Γ) where $s_0 \in S$ is the initial state, Δ is the system's *behavioural model*, and Γ is the estimator's *preference model*. Δ describes the system's transition relation $\overset{\Delta}{\rightarrow} \subseteq S \times S$, and describes which states are possible successors to a state s. For estimation purposes, Γ orders the possible successors of every state s, and defines a unique preferred successor. Formally, Γ defines a total order $\preceq_\Gamma \subseteq \overset{\Delta}{\rightarrow} \times \overset{\Delta}{\rightarrow}$ on the transitions defined by Δ.

We represent the system states with a set $\mathtt{V_{now}}$ of discrete-valued variables that model the system's internal state, its inputs, and outputs. We assume a synchronous model where time steps all last the same predefined duration; discrete events are modeled by a dedicated boolean variable, true at the time step during which event occurs and false otherwise. For the sake of clarity, we assume all variables are boolean in the model.

We adopt the usual notations of propositional logic.

- Let $\mathtt{V}$ be a set of propositional variables, then an assignment a to $\mathtt{V}$ is a function from $\mathtt{V}$ to $\mathbb{B} = \{\mathit{true}, \mathit{false}\}$.
- If $\mathtt{x} \in \mathtt{X}$, then x (resp. $\overline{x}$) denotes the specific assignment to $\{\mathtt{x}\}$ that assigns value true (resp. false) to $\mathtt{x}$.
- If a and b are two assignments to two disjoint sets $\mathtt{A}$ and $\mathtt{B}$ then ab is the assignment to $\mathtt{A} \cup \mathtt{B}$ such that $\forall \mathtt{x} \in \mathtt{A}$, $ab(\mathtt{x}) = a(\mathtt{x})$ and $\forall \mathtt{x} \in B$, $ab(\mathtt{x}) = b(\mathtt{x})$.
- If a is an assignment to $\mathtt{X}$ and $\mathtt{A} \subseteq \mathtt{X}$, then $a{\downarrow_{|\mathtt{A}}}$ is the restriction of a to $\mathtt{A}$.
- The satisfaction relation $\models$ between assignments and propositional formulas is defined by: $a \models \mathit{true}$, $a \not\models \mathit{false}$, $a \models \mathtt{x}$ *iff* $a(\mathtt{x}) = \mathit{true}$, $a \models \neg\mathtt{F}$ *iff* $a \not\models \mathtt{F}$,
  $a \models \mathtt{F}_1\, \mathit{op}\, \mathtt{F}_2$ *iff* $(a \models \mathtt{F}_1)$ *op* $(a \models \mathtt{F}_2)$ *with op* $\in \{\wedge, \vee, \ldots\}$.
- $\mathtt{scope}(\mathtt{F})$ is the set of variables mentioned in $\mathtt{F}$.
- If a is an assignment, we note $\mathtt{form}(x)$ the formula representing x, e.g. $\mathtt{form}(v_1\overline{v_2}) = \mathtt{v}_1 \wedge \neg\mathtt{v}_2$.

A state $s \in S$ is an assignment to $\mathtt{V}_{\mathtt{now}}$, which means $S = 2^{\mathtt{V}_{\mathtt{now}}}$. To model transitions, we need to represent relations between the current state and the previous state. We introduce an additional set of variables $\mathtt{V}_{\mathtt{pre}} = \{\mathtt{pre}(\mathtt{v}) | \mathtt{v} \in \mathtt{V}_{\mathtt{now}}\}$ that represent the system's state at the previous time step, and define $\mathtt{V} = \mathtt{V}_{\mathtt{now}} \cup \mathtt{V}_{\mathtt{pre}}$. $\mathtt{O} \subseteq \mathtt{V}_{\mathtt{now}}$ denotes the set of observable variables, and $\mathtt{E} \subseteq \mathtt{V}_{\mathtt{now}}$ the set of estimated variables. Without loss of generality we assume that $\mathtt{O} \cap \mathtt{E} = \emptyset$. We note $\mathit{Obs} = 2^{\mathtt{O}}$ the set of possible observations, and $\mathit{Est} = 2^{\mathtt{E}}$ the set of possible estimations.

For every state $s \in S$, $\mathit{pre}(s)$ represents the assignment s applied to the previous state variables, i.e. $\forall \mathtt{v} \in \mathtt{V}_{\mathtt{now}}, \mathit{pre}(s)\big(\mathtt{pre}(\mathtt{v})\big) = s(\mathtt{v})$. So in the following, if s and t are two assignments, then $\mathit{pre}(s)t$ is an assignment that applies s at the previous time step, and t at the current time step.

The first part of the model, Δ, consists in a set of propositional formulas over $\mathtt{V}$ that specifies a set of admissible transitions $\xrightarrow{\Delta}$[1]. A pair of states (s, t) belongs to $\xrightarrow{\Delta}$, which we note $s\xrightarrow{\Delta}t$, if and only if the assignment $\mathit{pre}(s)t$ satisfies all the formulas in Δ, i.e. $s\xrightarrow{\Delta}t$ if and only if $\mathit{pre}(s)t \models \Delta$. s_0 is the initial state. A state is admissible if and only if it is reachable from s_0 via a sequence of admissible transitions.

The second part of the model Γ is an ordered list $(\gamma_1, \cdots, \gamma_p)$ of preferences of the form $\gamma_i = (\mathtt{v}_{\gamma \mathtt{i}}, \mathtt{C}_{\gamma \mathtt{i}})$ where $\mathtt{v}_{\gamma \mathtt{i}} \in \mathtt{V}$ is the target variable and $\mathtt{C}_{\gamma \mathtt{i}}$ is a propositional formula over $\mathtt{V}$, and describes the condition for preferring $v_{\gamma i}$ to $\overline{v_{\gamma i}}$.

Definition 1 (Preference satisfaction). *A transition $s\xrightarrow{\Delta}t$ satisfies preference $\gamma_i = (\mathtt{v}_{\gamma \mathtt{i}}, \mathtt{C}_{\gamma \mathtt{i}})$ if and only if the assignment $\mathit{pre}(s)t$ satisfies the formula $(\mathtt{v}_{\gamma \mathtt{i}} \leftrightarrow \mathtt{C}_{\gamma \mathtt{i}})$. We note $s\xrightarrow{\Delta}t \models \gamma_i$.*

Due to the definition of satisfaction, and for the sake of clarity, a preference can equivalently be described as $\gamma_i = (\mathtt{v}_{\gamma \mathtt{i}}, \mathtt{C}_{\gamma_\mathtt{i}})$ or $\gamma_i = (\neg\mathtt{v}_{\gamma_\mathtt{i}}, \neg\mathtt{C}_{\gamma_\mathtt{i}})$. We assume that the preference dependences form no cycle, i.e. that for any two preferences γ_i and γ_j, $i < j$, the condition of γ_i does not depend on the target of γ_j, i.e. $\mathtt{v}_{\gamma_\mathtt{j}} \notin \mathtt{C}_{\gamma_\mathtt{i}}$. Informally, γ_i is considered more important than γ_j, so it must not depend on the outcome of γ_j. See [2] for a more in-depth discussion on the consistency of preference models.

Definition 2 (Transition preference order). *A transition $s\xrightarrow{\Delta}t$ is preferred to another $s'\xrightarrow{\Delta}t'$, noted $s\xrightarrow{\Delta}t \preceq_\Gamma s'\xrightarrow{\Delta}t'$, if and only if there exists an index i such that $s\xrightarrow{\Delta}t \models \gamma_i$, $s'\xrightarrow{\Delta}t' \not\models \gamma_i$, and for every $j \in [1..i]$, $s\xrightarrow{\Delta}t \models \gamma_j \leftrightarrow s'\xrightarrow{\Delta}t' \models \gamma_j$.*

[1] In a small abuse of notation, we use Δ to denote both the set of formulas $\{\delta_1, \cdots, \delta_n\}$ and the conjunction $\delta_1 \wedge \cdots \wedge \delta_n$.

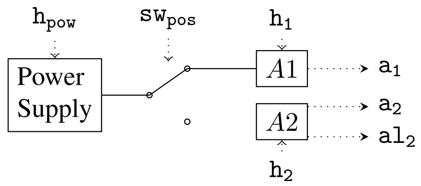

Figure 1: A system with one power supply and two redundant actuators. The power supply is subject to a permanent fault. The first actuator and the power supply are subject to random intermittent faults. When a fault occurs on actuator $A2$, it reboots, and sends an alarm just after.

As described in [1], the PTLTL logic (propositional logic enriched with past temporal operators Y for yesterday and S for since) is a convenient and efficient specification language for Δ. Γ can be described by a conditional preference theory, where the preference conditions are specified in PTLTL as well. When written using PTLTL sentences, the estimation model can be flattened into propositional logic formulas by introducing variables for aliasing temporal operators, and memoizing some variable's past values. The flattening operation is linear in time and memory [3].

With the aforementioned assumptions made on Γ, $\preceq_\Gamma$ may be only a partial order. A sufficient condition for it to be a total order is to assume there exists one preference for each variable in $\mathtt{V}$. This guarantees that for any state and any input, there exists a unique preferred successor state. However, as we will see in definition 4, there is no need for preferences on observable variables nor for previously estimated variables ($\mathtt{pre(v)}$ for $\mathtt{v} \in \mathtt{E}$) since there is no ambiguity on their value in the estimation process.

We naturally extend the observable projection from states to paths.

Definition 3 (System Path, Observable path)**.** *A* system path *is a sequence of states $p = (s_0, \cdots, s_n) \in S^*$ such that $s_i \xrightarrow{\Delta} s_{i+1}$ for $i \in [0..n[$. We note $L_\Delta \subseteq S^*$ the language of all system paths, $\|p\|$ the length of p, $p[i]$ the i-th state in p, and $p[i..j] = (s_i, \cdots s_j)$, with $0 \leq i \leq j \leq \|p\|$.*

The observable projection of a path $p \in S^*$ is the sequence $P_{obs}(p) = (p[0]{\downarrow_{\mathtt{O}}}, \cdots, p[n]{\downarrow_{\mathtt{O}}})$.

Example 1. *We consider a switch mechanism between two redundant actuators $A1$ and $A2$, powered by a common power supply, as represented in Figure 1. Our system is modeled by the following variables:*
$\mathtt{sw_{pos}}$ *: switch position (true when actuator $A1$ is active)*
$\mathtt{h_{pow}}$ *: power supply health status (true when nominal)*
$\mathtt{h_i}$ *: actuator Ai health status (true when nominal)*
$\mathtt{a_i}$ *: actuator Ai active (true when operating)* *The above variables correspond to set* $\mathtt{V_{now}}$.
$\mathtt{al_2}$ *: actuator $A2$ alarm event*
$\mathtt{y_i}$ *: symptom that Ai is not functioning*
$\mathtt{oy_i}$ *: symptom $\mathtt{y_i}$ has appeared in the past*
The set of observable variables is $\mathtt{O} = \{\mathtt{sw_{pos}}, \mathtt{a_1}, \mathtt{a_2}, \mathtt{al_2}\}$, and the set of estimated variables is $\mathtt{E} = \{\mathtt{h_{pow}}, \mathtt{h_1}, \mathtt{h_2}\}$.

The transition relation is defined as:

$$\Delta = \left\{ \begin{array}{l} \mathtt{h_{pow}} \rightarrow \mathtt{pre}(\mathtt{h_{pow}}) \\ \mathtt{a_1} \leftrightarrow (\mathtt{h_{pow}} \wedge \mathtt{sw_{pos}} \wedge \mathtt{h_1}), \\ \mathtt{a_2} \leftrightarrow (\mathtt{h_{pow}} \wedge \neg\mathtt{sw_{pos}} \wedge \mathtt{h_2}), \\ \mathtt{al_2} \leftrightarrow (\neg\mathtt{pre}(\mathtt{h_2}) \wedge \mathtt{h_2}), \\ \mathtt{y_1} \leftrightarrow (\mathtt{sw_{pos}} \wedge \neg\mathtt{a_1}), \\ \mathtt{y_2} \leftrightarrow (\neg\mathtt{sw_{pos}} \wedge \neg\mathtt{a_2}), \\ \mathtt{oy_1} \leftrightarrow (\mathtt{pre}(\mathtt{oy_1}) \vee \mathtt{y_1}), \\ \mathtt{oy_2} \leftrightarrow (\mathtt{pre}(\mathtt{oy_2}) \vee \mathtt{y_2}) \end{array} \right\} \begin{array}{l} (\delta_1) \\ (\delta_2) \\ (\delta_3) \\ (\delta_4) \\ (\delta_5) \\ (\delta_6) \\ (\delta_7) \\ (\delta_8) \end{array}$$

A fault in the power supply is permanent (δ_1). Each actuator is active if and only if the power supply is healthy, the switch is in the appropriate position, and it is healthy $(\delta_2)(\delta_3)$. When actuator $A2$ fails, it tries to reboot. If it succeeds, it sends an alarm event indicating its status is back to ok (δ_4). For clarity purposes, we introduce "symptom" variables $\mathtt{y_i}$, indicating when we expect actuator i to be working and it is not $(\delta_5)(\delta_6)$. We memoize if each symptom has happened in the past $(\delta_7)(\delta_8)$. All components are subject to intermittent faults for which we have no model, this is why we model their status as unobserved input variables indicating their health status.

The initial state is:

$$s_0 = sw_{pos} h_{pow} h_1 h_2 a_1 \overline{a_2}\,\overline{al_2}\,\overline{y_1 y_2 oy_1 oy_2}$$

The preference model implements the following estimation strategy. When a symptom appears on only one actuator, we assume the cause is an actuator failure. However, once a symptom has appeared on each actuator, we blame the battery. Moreover, once we estimate an actuator to be faulty (or healthy), we keep this estimation until we have evidence its health status has changed.

$$\Gamma = \begin{pmatrix} (\neg\mathtt{h_{pow}}, & \mathtt{oy_1} \wedge \mathtt{oy_2}) \\ (\neg\mathtt{h_1}, & \neg\mathtt{pre}(\mathtt{h_1}) \vee (\mathtt{y_1} \wedge \mathtt{h_{pow}})) \\ (\neg\mathtt{h_2}, & \neg\mathtt{pre}(\mathtt{h_2}) \vee (\mathtt{y_2} \wedge \mathtt{h_{pow}})) \end{pmatrix} \begin{array}{l} (\gamma_1) \\ (\gamma_2) \\ (\gamma_3) \end{array}$$

When a symptom has occurred on both actuators, we prefer explanations in which the power supply is faulty (γ_1). We prefer explanations in which actuators are healthy when they were healthy at the previous time step, and when if there is a symptom, it is already explained by a power supply failure $(\gamma_2)(\gamma_3)$.

We did not specify any preference for $\mathtt{y_1}$, $\mathtt{y_2}$, $\mathtt{oy_1}$, $\mathtt{oy_2}$, since Δ will leave no ambiguity for these variables. It is possible to add a dummy preference $(\mathtt{v}, \top)$ for each of these variables, with no impact on the estimator's behaviour.

3 Estimation process

Despite the assumptions made on the preference model (at most one preference per variable, acyclic preference conditions), the preferred successor state of a given state for a given observation remains a difficult task.

Definition 4 (State estimation problem)**.** *Given a model (s_0, Δ, Γ), a previous estimated state for a previous observation $\widehat{s} \in Obs \times Est$ and a current observation $obs \in Obs$, the* estimation problem *consists in finding the successor of $\widehat{s}$ that produces obs and that is preferred by $\preceq_\Gamma$. Formally, if there exists a transition $s \overset{\Delta}{\to} t$ such that:*

$$s{\downarrow_{\mathtt{O} \cup \mathtt{E}}} = \widehat{s},\ t{\downarrow_{\mathtt{O}}} = obs,\ \textit{and} \qquad (1)$$

$$\forall s' \overset{\Delta}{\to} t' \textit{ satisfying } (1),\ s \overset{\Delta}{\to} t \preceq_\Gamma s' \overset{\Delta}{\to} t' \qquad (2)$$

$$\begin{tikzcd}
\text{System :} & & s_0 \arrow[r] \arrow[d, dotted] & s_1 \arrow[r] \arrow[d, dotted] & \cdots \\
\text{Estimator :} & \mathit{init} \arrow[r, "s_0{\downarrow}_{\mathtt{O}}"] & \widehat{s_0} \arrow[r, "s_1{\downarrow}_{\mathtt{O}}"] & \widehat{s_1} \arrow[r] & \cdots
\end{tikzcd}$$

Figure 2: System / Estimator synchronization. The estimator uses a special init state for startup.

then $\widehat{t} = t{\downarrow}_{\mathtt{O}\cup\mathtt{E}}$ *is the estimated state. If no transition satisfies* (1)*, the estimated state is undefined.*

The state estimation problem is an instance of constrained outcome optimisation in CP-theories, several algorithms have been suggested to solve it [4]. Our approach, detailed in [1], is to reduce it to a boolean optimization problem, encoded as a Partial Weighted MaxSAT query [5] as follows:

1. Assignments $\widehat{s}$ and obs are encoded into hard clauses;
2. Constraints in Δ are encoded into hard clauses;
3. For every preference $\gamma_i, i \in [1..p]$, the constraint $\mathtt{v}_{\gamma i} \leftrightarrow \mathtt{C}_{\gamma i}$ is encoded into soft clauses with weight 2^{p-i};

The solution to the MaxSAT query encodes the preferred successor state for $\widehat{s}$ for observation obs.

Definition 5 (Estimator)**.** *An estimator for a model* (s_0, Δ, Γ) *with respect to observable variables* $\mathtt{O}$ *and estimated variables* $\mathtt{E}$ *is a deterministic automaton* $(E^{est}, S^{est}, \mathit{init}, \mathit{estim})$ *where:*
- $E^{est} = \mathit{Obs}$ *is the input alphabet;*
- $S^{est} = (\mathit{Obs} \times \mathit{Est}) \cup \{\mathit{init}\}$ *is the set of estimator states;*
- init *is the initial state;*
- estim *is a partial transition function defined as:*
  - $\mathit{estim}(\mathit{init}, s_0{\downarrow}_{\mathtt{O}}) = s_0{\downarrow}_{\mathtt{O}\cup\mathtt{E}}$*, and*
  - $\mathit{estim}(\widehat{s}, \mathit{obs}) = \widehat{t}$ *if and only if* $\widehat{t}$ *is defined, and is the estimated state for* $\widehat{s}$ *and* obs*.*

The initialization of the estimator and its relation with the system are illustrated in Figure 2. We note $L_{estim} \subseteq \mathit{Obs}^*$ *the language accepted by the estimator.*

There are two important assumptions about the estimation process. The first assumption is that the estimated state $\mathit{estim}(\widehat{s}, \mathit{obs})$ only depends on the previous estimated state $\widehat{s}$, and not on the history of the system. However, it is possible to introduce variables for memoizing past features, and expressing preferences on these variables. This task is especially easy when using PTLTL for modeling (see [1]). The second assumption is that the estimated state $\mathit{estim}(\widehat{s}, \mathit{obs})$ only depends on the variables of $\mathtt{E} \cup \mathtt{O}$ at the previous time step, and not on other variables. This is a feature that is particularly helpful in managing divergence, as we will see in section 5.

Example 2 (Estimation)**.** *In our example, we have* $\mathtt{E} \cup \mathtt{O} = \{\mathtt{h_{pow}}, \mathtt{h_1}, \mathtt{h_2}, \mathtt{sw_{pos}}, \mathtt{a_1}, \mathtt{a_2}, \mathtt{al_2}\}$*. Let* $\widehat{s}_0 = s_0{\downarrow}_{\mathtt{O}\cup\mathtt{E}}$*, and let nothing particular happen, we have* $obs_1 = sw_{pos}\, a_1\, \overline{a_2}\, al_2$*. The estimated state is* $\widehat{s}_1 = obs_1 h_{pow} h_1 h_2$*.* h_{pow} *and* h_1 *are enforced by* δ_2*, and* h_2 *is deduced by applying preference* γ_3*.*

From $\widehat{s}_1$*, let* $A1$ *fail. We receive observation* $obs_2 = sw_{pos}\, \overline{a_1\, a_2}\, \overline{al_2}$*. The estimated state is* $\widehat{s}_2 = obs_2 h_{pow} \overline{h_1} h_2$*.* h_{pow} *is obtained by applying preference* γ_1 *(since* $\mathtt{oy_2}$ *is false by* δ_6 *and* δ_8*), then preference* γ_2 *imposes* $\overline{h_1}$ *and* γ_3 *gives* h_2*.*

If, in response to estimation $\widehat{s}_1$*, the operator switches to actuator* $A2$*, and we receive observation* $obs_3 = \overline{sw_{pos}\, a_1}\, a_2\, \overline{al_2}$*, the estimated state is* $\widehat{s}_3 = obs_3 h_{pow} \overline{h_1} h_2$*.* δ_3 *enforces* $h_{pow} h_2$*, and* γ_2 *prefers* $\overline{h_1}$*, since in* $\widehat{s}_2$*,* $\mathtt{h_1}$ *is false.*

Finally, let the operator switch back to $A1$*, and let us observe* $obs_4 = sw_{pos}\, a_1\, \overline{a_2}\, \overline{al_2}$*. The estimated state* $\widehat{s}_4 = obs_4 h_{pow} h_1 h_2$ *is returned. Note that in* $\widehat{s}_4$*, preference* γ_2 *is violated: the condition is falsified (since* $\mathtt{pre}(\mathtt{h_1})$ *is false in* $\widehat{s}_3$*), so for* γ_2 *to be satisfied we need* $\overline{h_1}$*. However,* δ_2 *enforces* h_1*, and wins* *over γ_2. This illustrates how preferences are soft constaints that are only applied when Δ allows both values for some variable.*

4 Divergence, Deadlocks and Trackability

At each time step, for a given state and input, there may be several possible successor states, which means that an estimator may choose the wrong branch. In this case, the estimator *diverges* from the real system state. This is the source of several potential problems that should be at least detected at design time.

Example 3 (Divergence and deadlock)**.** *From $\widehat{s}_0 = s_0{\downarrow}_{\mathtt{O}\cup\mathtt{E}}$, let us receive observation $obs_1 = sw_{pos}\, a_1\, \overline{a_2}\, \overline{al_2}$, which yields estimation $\widehat{s}_1 = obs_1 h_{pow}\, h_1\, h_2$ as described in Example 2.*

Let the next observation be $obs_2 = sw_{pos}\, a_1\, \overline{a_2}\, al_2$: actuator $A2$ sends a "reboot finished" alarm. The assignment $pre(\widehat{s}_1)$ entails $pre(h_2)$, which, along with observation al_2, violates constraint δ_4. There is no estimated state for this scenario.

Informally, at time step 1, actuator $A2$ failed silently, but the estimator diverged and decided that $\mathtt{h}_2$ *was true because of preference γ_3. At time step 2, observation al_2 and rule δ_4 are inconsistent with the previous estimated state. Consequently by definition 4 there is no estimated state at time step 2. The artefact composed of the system and the estimator cannot evolve synchronously, and deadlocks.*

This deadlock phenomenon is due to the fact that the system and the estimator can take two execution branches that display the same observations at first, then different observations after some time. Whatever branch the system takes, the estimator may choose according to its preference model. If the estimator chooses the wrong branch, then later on the system will produce observations that are inconsistent with the previous choices of the estimator.

This problem exists as well when diagnosing permanent faults, and has been addressed in the litterature [6, 7]. Some online solutions have been proposed: backtrack in time to identify the correct execution branch, or reset the estimator to a "any state" estimation. In this paper, we attempt to detect and eliminate this problem at design time.

4.1 Trackability

The intuition behind trackability is the same as in [6]. A system is trackable by an estimator if and only if the estimator for the variables in $\mathtt{E}$ accepts all the observation sequences from the system. As illustrated in example 3, this is not always the case.

Definition 6 (Trackability)**.** *A system with language $L_\Delta \subseteq S^*$ is trackable if and only if $\forall p \in L_\Delta, P_{obs}(p) \in L_{estim}$.*

Checking trackability can be done in a way that resembles so-called "Twin-Plant" approach to diagnosability [8] or regular language difference algorithms for automata. We construct a tracking plant by introducing new variables $\widehat{\mathtt{v}}$ not in $\mathtt{V}$ and build a transition system as follows.

Definition 7 (Tracking Plant)**.** *Let (s_0, Δ, Γ) be an estimation model, and let $\mathtt{O}$ and $\mathtt{E}$ be the sets of observable and estimated variables respectively. The tracking plant (TP) is the Kripke Structure (S^T, I^T, R^T, L^T) defined as follows:*

- *The set of states $S^T = S \times (Obs \times Est) \times \mathbb{B}$ is encoded with the variables $\mathtt{V} \cup \{\widehat{\mathtt{v}} \mid \mathtt{v} \in \mathtt{O}\cup\mathtt{E}\} \cup \{\mathtt{track}\}$. The initial state is unique: $I^T = \{(s_0, s_0{\downarrow}_{\mathtt{O}\cup\mathtt{E}}, \top)\}$.*
- *The labelling function $L^T : S^T \to \mathbb{B}$, defined by $L^T\big((s, \widehat{s}, k)\big) = k$, simply records the value of variable* $\mathtt{track}$.

– *The transition relation* $R^T \subseteq S^T \times S^T$ *is defined by the following conditions:*

$$\big((s, \widehat{s}, \top), (t, \widehat{t}, \top)\big) \in R^T \textit{ if and only if } s \overset{\Delta}{\to} t \textit{ and } \widehat{t} = estim(\widehat{s}, t{\downarrow}_{\mathtt{O}}) \tag{3}$$

$$\big((s, \widehat{s}, \top), (t, \widehat{s}, \bot)\big) \in R^T \textit{ if and only if } s \overset{\Delta}{\to} t \textit{ and } estim(\widehat{s}, t{\downarrow}_{\mathtt{O}}) \textit{ is undefined} \tag{4}$$

$$\big((s, \widehat{s}, \bot), (s, \widehat{s}, \bot)\big) \in R^T \tag{5}$$

Informally, the tracking plant runs the system model Δ and the estimator side by side, and annotates each step with a variable `track` initially set to $\top$. (3) requires that when the estimator accepts the observation $t{\downarrow}_{\mathtt{O}}$, it estimates the next state, and `track` is kept to $\top$. When $estim$ is undefined, (4) requires that `track` switches to false, indicating that the system has generated an observation that the estimator does not accept. (5) lets the TP be alive for consistency with Kripke Structure axioms.

Proposition 1 (Trackability condition)**.** *A* deadlocking path *is a path of the tracking plant* $p \in S^{T*}$ *ending in a state where* `track` *is false.*

The estimation model (s_0, Δ, Γ) *is trackable with respect to* `O` *and* `E` *if and only if the tracking plant contains no deadlocking path.*

Proof by induction. Let there be a model (s_0, Δ, Γ) with $\Delta = \bot$. We have $\overset{\Delta}{\to} = \emptyset$, $L_\Delta = s_0$, and the estimator only contains the initial transition, i.e. $L_{estim} = \{s_0{\downarrow}_{\mathtt{O}}\}$, thus by definition 6 the system is trackable. Moreover, the TP has no transition either, its only state is $(s_0, s_0{\downarrow}_{\mathtt{O}\cup\mathtt{E}}, \top)$, it contains no deadlocking path. Proposition 1 is verified.

Let a model (s_0, Δ, Γ) satisfy proposition 1, and let us add a single transition from s to t. We note $\overset{\Delta}{\to}' = \overset{\Delta}{\to} \cup (s, t)$, $\Delta' = \Delta \vee \mathtt{form}(pre(s)t)$, and $L'_\Delta = L_\Delta \cup \{p.s.t \,|\, p.s \in L_\Delta\}$. The model$(s_0\Delta', \Gamma)$ has an estimator that accepts language L'_{estim} and its tracking plant is noted TP′.

Let $q \in L'_\Delta$ denote any path of length n ending with s, and such that $P_{obs}(q) \in L'_{estim}$. This means that there exists a unique path in the estimator $\widehat{q} \in Est^*$ such that $\widehat{q}[i+1] = estim'(\widehat{q}[i], q{\downarrow}_{\mathtt{O}})$ for $i \in [1..n[$ (def. 5). Consequently, the state $(q[n], \widehat{q}_n, \top)$ is reachable in the TP′ via transitions of type (3).

Let us assume that for all such q, $estim'(\widehat{q}[n], t{\downarrow}_{\mathtt{O}})$ is defined and equal to $\widehat{t}$. Then $P_{obs}(q.t) \in L'_{estim}$ (def. 5) and the system is trackable (def. 6). Also, TP′ contains the transitions of TP plus $\big((q[n], \widehat{q}_n, \top), (t, \widehat{t}, \top)\big)$ by rule (3), which introduces no deadlocking path.

Let us now assume that there exists a q such that $estim'(\widehat{q}[n], t{\downarrow}_{\mathtt{O}})$ is undefined. Then $\widehat{q}[n]$ has no transition labelled $t{\downarrow}_{\mathtt{O}}$ (def. 5), thus $P_{obs}(q.t) \notin L'_{estim}$ and the system is not trackable. Moreover, TP′ contains the transitions of TP plus $\big((q[n], \widehat{q}_n, \top), (t, \widehat{q}_n, \bot)\big)$ by rule (4), which introduces a deadlocking path. □

In [8], diagnosability checking is done by model-checking a similar construction called “coupled twin plant”. In our approach, the system is trackable if and only if the tracking plant satisfies the CTL formula $AG(\mathtt{track})$ (or equivalently $\neg EF(\neg\mathtt{track})$). Thus, a model-checker for CTL logic [9] can check a system’s trackability if we can feed it with a description of the Tracking Plant.

However, as stated in section 3, the estimator’s transition function $estim$ is expressed as a boolean *optimization* problem called Partial Weighed MaxSAT. The framework of symbolic model checking only accepts transition functions expressed as a constraint *satisfaction* problem (with hard constraints only). In the next section, we attempt to translate the optimization problem into a satisfaction problem.

4.2 Preferences as hard constraints

It is possible to translate each preference into a constraint that will reject suboptimal transitions. The constraints encode the following idea: "for each preference γ_i, for each state such that both v_{γ_i} and $\overline{v_{\gamma_i}}$ are admissible, independently of preferences $\gamma_j, j > i$, choose the one value that satisfies γ_i". To express such constraints, we need to define quantifiers as follows. $\mathtt{F}_{\mathtt{v}=\top}$ (resp. $\mathtt{F}_{\mathtt{v}=\bot}$) denotes the formula $\mathtt{F}$ in which every occurrence of $\mathtt{v}$ is substituted by $\top$ (resp. $\bot$). We define $\exists\mathtt{v}, \mathtt{F} = \mathtt{F}_{\mathtt{v}=\top} \vee \mathtt{F}_{\mathtt{v}=\bot}$, and $\forall\mathtt{v}, \mathtt{F} = \mathtt{F}_{\mathtt{v}=\top} \wedge \mathtt{F}_{\mathtt{v}=\bot}$.

Definition 8 (Unconditional optimality constraint)**.** *The optimality constraints $\Phi_1 \dots \Phi_p$ respectively associated to preferences $\gamma_1 \dots \gamma_p$ are defined as follows:*

$$\begin{aligned}
\Phi_1 &= \Delta \quad \wedge \Big(\big(\forall \mathtt{v}_{\gamma\mathtt{1}}, (\exists \mathtt{v}_{\gamma\mathtt{i}})_{\mathtt{i}\in[\mathtt{2..p}]}, \Delta\big) \qquad \rightarrow (\mathtt{v}_{\gamma\mathtt{1}} \leftrightarrow \mathtt{C}_{\gamma\mathtt{1}})\Big)\\
&\dots\\
\Phi_j &= \Phi_{j-1} \wedge \Big(\big(\forall \mathtt{v}_{\gamma\mathtt{j}}, (\exists \mathtt{v}_{\gamma\mathtt{i}})_{\mathtt{i}\in[\mathtt{j+1..p}]}, \Phi_{j-1}\big) \rightarrow (\mathtt{v}_{\gamma\mathtt{j}} \leftrightarrow \mathtt{C}_{\gamma\mathtt{j}})\Big)\\
&\dots\\
\Phi_p &= \Phi_{p-1} \wedge \Big(\big(\forall \mathtt{v}_{\gamma\mathtt{p}}, \Phi_{p-1}\big) \qquad\qquad \rightarrow (\mathtt{v}_{\gamma\mathtt{p}} \leftrightarrow \mathtt{C}_{\gamma\mathtt{p}})\Big)
\end{aligned}$$

A transition $s \overset{\triangle}{\rightarrow} t$ that satisfies Φ_p is optimal with respect to $\preceq_\Gamma$. Hence, for a previous estimated state $\widehat{s}$ and an observation obs, the state $pre(\widehat{s})obs\widehat{t}$ that satisfies Φ_p is unique and we have $estim(\widehat{s}, obs) = \widehat{t}$.

Example 4. *Preference $\gamma_1 = (\mathtt{h}_{\mathtt{pow}}, \neg\mathtt{oy}_1 \vee \neg\mathtt{oy}_2)$ is associated with constraint:*

$$\begin{aligned}
\Phi_1 &= \Delta \wedge \Big((\forall \mathtt{h}_{\mathtt{pow}}, \exists \mathtt{h}_1, \exists \mathtt{h}_2, \Delta) \rightarrow \big(\neg \mathtt{h}_{\mathtt{pow}} \leftrightarrow (\mathtt{oy}_1 \wedge \mathtt{oy}_2)\big)\Big)\\
&= \Delta \wedge \Big(\big((\forall \mathtt{h}_{\mathtt{pow}}, \delta_1) \wedge (\forall \mathtt{h}_{\mathtt{pow}}, \exists \mathtt{h}_1, \delta_2) \wedge (\forall \mathtt{h}_{\mathtt{pow}}, \exists \mathtt{h}_2, \delta_3) \wedge\\
&\qquad (\exists \mathtt{h}_2, \delta_4) \wedge \delta_5 \wedge \delta_6 \wedge \delta_7 \wedge \delta_8\big)\\
&\qquad\qquad \rightarrow \big(\neg \mathtt{h}_{\mathtt{pow}} \leftrightarrow (\mathtt{oy}_1 \wedge \mathtt{oy}_2)\big)\Big)\\
&= \Delta \wedge \Big(\big((\forall \mathtt{h}_{\mathtt{pow}}, \exists \mathtt{h}_1, \delta_2) \wedge (\forall \mathtt{h}_{\mathtt{pow}}, \exists \mathtt{h}_2, \delta_3)\big)\\
&\qquad\qquad \rightarrow \big(\neg \mathtt{h}_{\mathtt{pow}} \leftrightarrow (\mathtt{oy}_1 \wedge \mathtt{oy}_2)\big)\Big)^{2}\\
&= \Delta \wedge \Big(\big(\mathtt{pre}(\mathtt{h}_{\mathtt{pow}}) \wedge \neg \mathtt{a}_1 \wedge \neg \mathtt{a}_2\big)\\
&\qquad\qquad \rightarrow \big(\neg \mathtt{h}_{\mathtt{pow}} \leftrightarrow (\mathtt{oy}_1 \wedge \mathtt{oy}_2)\big)\Big)
\end{aligned}$$

The constraint can be read as follows. If the power supply was working at the last time step and no actuator is working (this means at least one component is failing), then let $\mathtt{h}_{\mathtt{pow}}$ be false only if there has been a symptom on both actuators. This exactly implements the preference γ_1.

In practice, the constraint Φ_p is computationally very difficult to encode. Experiments with BDDs [10], or QBF [11] failed to scale up to moderately complex examples. While the scalability problems of BDDs are well known, in the case of QBF, the difficulty resides mainly in normalizing of the formulas into QCNF. Since the quantifiers are nested deeply into equivalence formulas, each of which must be expanded twice for normalization, the problem quickly explodes in the number of preferences.

As a consequence, we cannot check trackability directly on the tracking plant. Instead, we use an approximation of the estimator's transition function to construct an approximate tracking plant.

[2] Δ implies δ_4 which implies $\exists\mathtt{h}_2, \delta_4$. Since Δ holds, $\exists\mathtt{h}_2, \delta_4$ can be removed from the condition. The same applies to δ_5 to δ_8.

4.3 Approximation of Preferences

In order to model-check the tracking plant and verify if the system is trackable, we now describe an approach where the transition function of the estimator $estim$ is overapproximated. Informally, this means that some spurious transitions are added to the estimator in order to simplify the expression of its transition relation. As a consequence, the tracking plant also contains spurious transitions, and the model-checker may discover spurious deadlocking paths.

We assume we are using a model-checker that returns a counter example when the property is not valid on the system. In our case, this means that when checking $AG(\texttt{track})$, a counter example consists in a deadlocking path. Our approach consists in checking if the counter example is spurious by checking that all its transitions are optimal with respect to $\preceq_\gamma$. In the case the deadlocking path is spurious, it violates at least one preference that could have been satisfied. We encode this knowedge in a constraint, use it to refine the approximation of the estimator's transition relation and iterate the process.

Upon discovery of a proper deadlocking path, it must be eliminated. Various elimination techniques are discussed in section 5.

Definition 9 (Approximate Tracking Plant)**.** *The Approximate Tracking Plant (ATP) for a model* (s_0, Δ, Γ)*, respective observable and estimated variable sets* $\texttt{O}$ *and* $\texttt{E}$*, and a set* Ξ *of learned constraints, is the Kripke Structure* (S^T, I^T, R^T_Ξ, L^T) *with the same states, initial states and labelling function as the tracking plant, and such that* R^T_Ξ *is defined by the following conditions:*

$$\big((s, \widehat{s}, \top), (t, \widehat{t}, \top)\big) \in R^T_\Xi \textit{ if and only if } s \overset{\Delta}{\Rightarrow} t \textit{ and } pre(\widehat{s})\widehat{t}t{\downarrow_{\texttt{O}}} \models \Delta \cup \Xi \tag{6}$$

$$\big((s, \widehat{s}, \top), (t, \widehat{s}, \bot)\big) \in R^T_\Xi \textit{ if and only if } s \overset{\Delta}{\Rightarrow} t \textit{ and } \forall \widehat{t} \in Est, pre(\widehat{s})\widehat{t}t{\downarrow_{\texttt{O}}} \not\models \Delta \cup \Xi \tag{7}$$

$$\big((s, \widehat{s}, \bot), (s, \widehat{s}, \bot)\big) \in R^T_\Xi \tag{8}$$

An approximate deadlocking path is a path in the ATP ending in a state where $\texttt{track}$ *is false.*

When Ξ is empty, the ATP ignores the preferences, and assumes that the estimator can follow any path accepted by Δ. Thus, it accepts more paths than the tracking plant. When it is model checked for the property $AG(\texttt{track})$, two outcomes are possible. On the one hand, if the property is valid on the ATP, it is thus valid on the TP, and the system is trackable. On the other hand, if the property is not valid on the ATP, a counter example is produced, under the form of an approximate deadlocking path. The next step is to validate whether this approximate deadlocking path is a valid deadlocking path.

Definition 10 (Deadlocking path validation)**.** *An approximate deadlocking path* $p = \big((s_0, \widehat{s}_0, k_0), \cdots, (s_n, \widehat{s}_n, k_n)\big)$ *is a valid deadlocking path if and only if* $\widehat{s}_{i+1} = estim(\widehat{s}_i, s_{i+1}{\downarrow_{\texttt{O}}})$ *for* $i \in [1..n[$*. It is spurious otherwise.*

Note that computing $estim(\widehat{s}, s_{i+1}{\downarrow_{\texttt{O}}})$ is done by calling a MaxSAT solver as described in section 3. Validating a path thus requires n MaxSAT calls, which is quite tractable in our experiments.

A spurious approximate deadlocking path is such that at some index i, we have $\widehat{s}_{i+1} \neq estim(\widehat{s}_i, s_{i+1}{\downarrow_{\texttt{O}}})$. We note $e_i = estim(\widehat{s}_i, s_{i+1}{\downarrow_{\texttt{O}}})$. Consequently there exists a preference that is violated by the estimator's transition while it could have been satisfied. This preference $\gamma_j = (\texttt{v}_\texttt{j}, \texttt{C}_\texttt{j})$ is actually the first preference (by index) such that $\widehat{s}_{i+1}(\texttt{v}_\texttt{j}) \neq e_i(\texttt{v}_\texttt{j})$.

The final step in our approximation refinement is to identify from γ_j what can be learned and added into Ξ. In definition 8, the sufficient condition for γ_j to be applicable is defined as $cond_j = (\exists \texttt{v}_{\gamma_k})_{k \in [j+1..p]}, \forall \texttt{v}_\texttt{j}, \Phi_{j-1}$. By construction the scope of $cond_j$ does not contain any variable $\texttt{v}_{\gamma\texttt{k}}, j \leq$

k, since they have been quantified out. The assignment $pre(\widehat{s}_i)s_{i+1}{\downarrow_{\mathtt{O}}}$ necessarily satisfies this condition, since γ_j should have been applied. Moreover, variables in $\{\mathtt{v}_{\gamma_k} \mid j \leq k\})$ do not impact this preference. Thus, the we have learned one model of $cond_j$, that, when it occurs, is sufficient to trigger the application of γ_j.

Definition 11 (Learned condition)**.** *Let a spurious approximate deadlocking path* $p = \big((s_0, \widehat{s}_0, k_0), \cdots,$ $(s_n, \widehat{s}_n, k_n)\big)$ *violate preference* γ_j *at time step* i*. The* learned condition ξ *is defined by:*

$$cond = \big(pre(\widehat{s}_i)s_{i+1}{\downarrow_{\mathtt{O}}}\big){\downarrow_{\mathtt{V}-\{\mathtt{v}_{\gamma_k} \mid j \leq k\}}}$$
$$\xi = \begin{cases} \mathtt{form}(cond) \to \mathtt{v}_{\gamma \mathtt{j}} & \textit{if } cond \models \mathtt{C}_{\gamma \mathtt{j}} \\ \mathtt{form}(cond) \to \neg \mathtt{v}_{\gamma \mathtt{j}} & \textit{if } cond \not\models \mathtt{C}_{\gamma \mathtt{j}} \end{cases}$$

Where $\mathtt{form}(x)$ *is a formula accepting only* x *as a model.*

We add the learned condition ξ to Ξ and iterate the deadlock search until a proper deadlock is found, or until it is proved that the estimator will not deadlock.

Example 5. *The ATP can detect a deadlocking path as follows (variables* $\widehat{\mathtt{a}}_1$,$\widehat{\mathtt{a}}_2$,$\widehat{\mathtt{sw}}_{\mathtt{pos}}$,$\widehat{\mathtt{al}}_2$*, are omitted as they have the same value as their system counterpart):*

$$\begin{array}{lll} 0. \; sw_{pos} h_{pow} h_1 h_2 a_1 \overline{a_2}\,\overline{al_2}\,\overline{y_1 y_2 oy_1 oy_2} & \widehat{h}_{pow} \widehat{h}_1 \widehat{h}_2 & \top \\ 1. \; sw_{pos} h_{pow} \overline{h_1} h_2 \overline{a_1 a_2}\,\overline{al_2} y_1 \overline{y_2} oy_1 \overline{oy_2} & \overline{\widehat{h}_{pow}} \widehat{h}_1 \widehat{h}_2 & \top \\ 2. \; sw_{pos} h_{pow} h_1 h_2 a_1 \overline{a_2}\,\overline{al_2}\,\overline{y_1 y_2} oy_1 \overline{oy_2} & \overline{\widehat{h}_{pow}} \widehat{h}_1 \widehat{h}_2 & \bot \end{array}$$

At step 1, actuator $A1$ *fails, and the estimator has an explanation for it, thus a transition of type* (6) *is fired. In this path the approximate estimator explains observation* $\overline{a_1}$ *by blaming the power supply. In reality, this explanation violates preference* γ_1*, which could have been satisfied (see Example 2). At step 2,* $A1$ *is back to normal and produces observation* a_1*. The approximate estimator cannot explain this since it assumed the power supply has permanently failed. Thus a transition of type* (7) *is fired, and* `track` *is set to* $\bot$*.*

Since preference γ_1 *has been violated at step 1 while it could have been satisfied, this deadlocking path is spurious. We learn the clause:*

$$\begin{aligned} \xi &= \mathtt{form}(sw_{pos}\, a_1 \overline{a_2}\, y_1 \overline{y_2}\, oy_1 \overline{oy_2}) \to \mathtt{h_{pow}} \\ &= (\mathtt{sw_{pos}} \wedge \mathtt{a_1} \wedge \neg \mathtt{a_2} \wedge \mathtt{y_1} \wedge \neg \mathtt{y_2} \wedge \mathtt{oy_1} \wedge \neg \mathtt{oy_2}) \to \mathtt{h_{pow}} \end{aligned}$$

The most expensive operation in the deadlock search is the model-checking phase, since symbolic model-checking of CTL formulas is PSPACE [12] in $\|\mathtt{V}\|$, while the other steps are at most NP-HARD in $\|\mathtt{V}\|$. Since the analysis takes place at design time, it is acceptable to consider somewhat time-consuming algorithms.

5 Deadlock elimination

A proper deadlock is an actual problem for the estimator, and should be eliminated at design time. The most straightforward way is to modify the system, either by making some variables observable, or by modifying the system behaviour. However, another way to look at deadlocks is that they are caused by an attempt to estimate signals that cannot be estimated with certainty, or at least not with the fault management policy specified by the preference model. From this point of view, the correct way to eliminate a deadlock is to try and estimate another signal. We present here two examples that illustrate two generic elimination solutions: delayed estimation, and conditional estimation.

Example 6 (Delayed estimation). *The deadlock illustrated in Example 3 can be eliminated by introducing an additional variable* $\mathtt{ph}_2$ *(for previous* $\mathtt{h}_2$*) initialized to true, an additional hard constraint* $\delta_9 = \mathtt{ph}_2 \leftrightarrow \mathtt{pre}(\mathtt{h}_2)$ *in* Δ*, an additional dummy preference* $\gamma_4 = (\mathtt{ph}_2, \top)$ *in the last position of* Γ*, and setting* $\mathtt{E} = \{\mathtt{h}_{\mathtt{pow}}, \mathtt{h}_1, \mathtt{ph}_2\}$*.*

This eliminates the deadlock because the estimator remembers only the values for variables in $\mathtt{E} \cup \mathtt{O}$*. We removed* $\mathtt{h}_2$ *from this set and replaced it with* $\mathtt{ph}_2$*. Now, this means that the estimator is not bound by the value it assigned* $\mathtt{h}_2$ *by preference* γ_3*: it can assign* $\overline{ph_2}$ *even if it assumed* h_2 *at the previous time step.*

In detail, the scenario from Example 3 now unfolds as follows. From $\widehat{s}_0 = s_0{\downarrow_{\mathtt{O}\cup\mathtt{E}}} ph_2$*, we receive observation* $obs_1 = sw_{pos}\, a_1\, \overline{a_2}\, \overline{al_2}$*. This prompts estimation* $\widehat{s}_1 = obs_1 h_{pow}\, h_1\, ph_2$*, in this case* $\mathtt{ph}_2$ *is determined by preference* γ_3 *and rule* δ_9*, and preference* γ_4 *is not applied. Then, observation* $obs_2 = sw_{pos}\, a_1\, \overline{a_2}\, al_2$*, the estimator successfully finds the state* $\widehat{s}_2 = obs_2 h_{pow}\, h_1\, \overline{ph_2}$*.*

This example illustrates how the value of $\mathtt{h}_2$ is not trackable, but its previous value is. This notion is similar in intent to the concept of diagnosability introduced in [13], where a permanent fault is diagnosable when an observer can deduce its occurrence after a bounded delay. The definition of trackability in [6] is more similar to ours, even though it addresses only permanent faults as well. The discussion about the size of the "prediction window" corresponds precisely to a delayed diagnosis.

Example 7 (Conditional estimation). *Another proper deadlocking path exists, when a temporary fault occurs on each actuator during operation, then everything is back to normal. The estimator incorrectly blames the power supply, and cannot explain that everything comes back to normal. Formally, the proper deadlocking path is:*

$$\begin{array}{lll}
0. & sw_{pos} h_{pow} h_1\, h_2\, a_1\, \overline{a_2}\, \overline{al_2}\, \overline{y_1\, y_2\, oy_1\, oy_2} & \widehat{h}_{pow} \widehat{h}_1 \widehat{h}_2 \quad \top \\
1. & sw_{pos} h_{pow} \overline{h_1}\, h_2\, \overline{a_1\, a_2}\, \overline{al_2}\, y_1\, \overline{y_2}\, oy_1\, \overline{oy_2} & \widehat{h}_{pow} \overline{\widehat{h}_1} \widehat{h}_2 \quad \top \\
2. & \overline{sw_{pos}} h_{pow} h_1\, \overline{h_2}\, \overline{a_1\, a_2}\, \overline{al_2}\, \overline{y_1}\, y_2\, oy_1\, oy_2 & \overline{\widehat{h}_{pow}} \widehat{h}_1 \widehat{h}_2 \quad \top \\
3. & \overline{sw_{pos}} h_{pow} h_1\, h_2\, \overline{a_1}\, a_2\, \overline{al_2}\, \overline{y_1\, y_2}\, oy_1\, oy_2 & \overline{\widehat{h}_{pow}} \widehat{h}_1 \widehat{h}_2 \quad \bot
\end{array}$$

At time step 1, actuator 1 fails. At time step 2, the switch is flipped, actuator 1 is back to normal, and actuator 2 fails. There have now been symptoms on both actuators, the estimator blames the power supply. At time step 3, actuator 2 is back to normal. This deadlocking path is due to the fact that our fault management policy chooses a permanent fault over an intermittent one.

There are times when a power supply failure can be detected with certainty: when $A2$ *just rebooted, we know* $A2$ *is working. If in this instant the switch is positioned towards* $A2$*, we can deduce the value of* $\mathtt{h}_{\mathtt{pow}}$*. Thus if we introduce a variable* $\mathtt{surefail}_{\mathtt{pow}}$*, a constraint* $\mathtt{surefail}_{\mathtt{pow}} \leftrightarrow (\neg \mathtt{sw}_{\mathtt{pos}} \wedge \mathtt{pre}(\mathtt{al}_2) \wedge \neg \mathtt{h}_{\mathtt{pow}})$*, some dummy preference* $(\mathtt{surefail}_{\mathtt{pow}}, \bot)$*, and estimate this variable instead of* $\mathtt{h}_{\mathtt{pow}}$ *(i.e.* $\mathtt{E} = \{\mathtt{surefail}_{\mathtt{pow}}, \mathtt{h}_1, \mathtt{h}_2\}$*), the deadlock is eliminated. It is a form of conditional estimation, since the value of* $\mathtt{h}_{\mathtt{pow}}$ *can be deduced from that of* $\mathtt{surefail}_{\mathtt{pow}}$ *only when some conditions are met, namely when* $\mathtt{surefail}_{\mathtt{pow}}$ *is true.*

Another way to eliminate this deadlock is to model and estimate the "confidence" we have in the power supply's health. While a fault in the power supply is permanent, our confidence in it can be intermittent. With a variable $\mathtt{c}_{\mathtt{pow}}$*, a rule* $\mathtt{c}_{\mathtt{pow}} \rightarrow \mathtt{h}_{\mathtt{pow}}$ *a preference* $\gamma_5 = (\mathtt{c}_{\mathtt{pow}}, \mathtt{pre}(\mathtt{c}_{\mathtt{pow}}))$*, it we estimate* $\mathtt{c}_{\mathtt{pow}}$ *instead of* $\mathtt{h}_{\mathtt{pow}}$ *(i.e.* $\mathtt{E} = \{\mathtt{c}_{\mathtt{pow}}, \mathtt{h}_1, \mathtt{h}_2\}$*), the deadlock is eliminated as well, since at step 3, instead of violating constraint* δ_1*, we will be violating preference* γ_5*.*

The last example illustrates how a way to manage deadlocks can be to estimate other signals. This is very important, since it also impacts which reconfigurations can be made. When a designer chooses to eliminate a deadlock by estimating a delayed or conditional version of a signal, they can adapt the reconfiguration policy adequately.

6 Conclusion

Adressing the problem of fault diagnosis as a discrete signal estimation problem increases the expressiveness of the model: we are not limited only to permanent physical degradations, but can model and estimate intermittent perturbations, function performance (past, present), action outcomes (past, present), and even make some form of prediction, for example via a confidence model, about function performance and action outcomes. This is very convenient for specifying fault management policies for autonomous systems and vehicles.

In terms of modeling expressiveness, the diagnosis approaches most similar to ours are based on probabilistic models such as [7]. In these approaches, the preferred transition(s) are the most probable ones. The main, well known limitation of probabilistic approaches is the difficulty to set the probabilities. This is especially true for autonomous systems: during the inevitable testing phase, the problem of adjusting the probabilities so that the system passes a given set of unit tests is very difficult. We claim that the modeling framework of conditional preferences is more designer-friendly when it comes to validating the system against a set of unit tests. For a comparison with other diagnosis modelling approaches, see [1].

With greater expressiveness comes greater computational difficulty. Our estimator executes a MaxSAT query at each time step, which has poor worst case execution time. Moreover, verifying that the model does not deadlock is a computationally difficult task. Our approach of model-checking overapproximations or the estimator and iteratively refining them is inspired by [14], who initially developped this method for software systems. While our approach will ultimately find deadlocks if they exist, proving an estimation model is deadlock free can only be done on finite horizons with bounded model-checking [15], unless the model-checker handles the estimator's transition function, which is unlikely on realistic examples. It would be interesting to define restricted models for which more efficient algorithms exist.

There are multiple directions for the future development of this approach. First, experimental results should be produced for non-trivial examples. Second, the definition of the Approximate Tracking Plant and of the learned conditions could take advantage of the structure of the transition relation in order to minimize the number of iterations, which is the most important factor in the performance of our analysis. Third, properties other than trackability shall be defined and checked. In particular, the correctness and the stability of the estimation are important properties when designing a fault management system. By correctness we mean that the estimator should converge to the exact value in a given amount of time, and by stability we mean that the estimator should not introduce spurious fault events, which may trigger useless system reconfigurations.

References

[1] 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.

[2] 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.

[3] E. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B:Formal Models and Semantics*, pages 995–1072. Elsevier, 1990.

[4] Nic Wilson. Computational techniques for a simple theory of conditional preferences. *Artificial Intelligence*, 175(7):1053–1091, 2011.

[5] J. Marques-Silva, J. Argelich, A. Graça, and I. Lynce. Boolean lexicographic optimization: algorithms & applications. *Ann. Math. Artif. Intell.*, 62(3-4):317–343, 2011.

[6] Alban Grastien, Anbu Anbulagan, et al. Incremental diagnosis of DES with a non-exhaustive diagnosis engine. In *Proceedings of the 20th International Workshop on Principles of Diagnosis*. Linkoping University Institute of Technology, 2009.

[7] James Kurien and P Pandurang Nayak. Back to the future for consistency-based trajectory tracking. In *AAAI/IAAI*, pages 370–377, 2000.

[8] Alessandro Cimatti, Charles Pecheur, and Roberto Cavada. Formal verification of diagnosability via symbolic model checking. In Georg Gottlob and Toby Walsh, editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 363–369. Morgan Kaufmann, 2003.

[9] Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In *International Conference on Computer Aided Verification*, pages 359–364. Springer, 2002.

[10] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17(1):229–264, 2002.

[11] Mikolas Janota, Charles Jordan, Will Klieber, Florian Lonsing, Martina Seidl, and Allen Van Gelder. The QBFGallery 2014: The QBF competition at the FLoC olympic games. *Journal on Satisfiability, Boolean Modeling and Computation*, 9:187–206, 2016.

[12] Philippe Schnoebelen. The complexity of temporal logic model checking. *Advances in modal logic*, 4(393-436):35, 2002.

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

[14] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In *Computer aided verification*, pages 154–169. Springer, 2000.

[15] Armin Biere, Alessandro Cimatti, Edmund M Clarke, Ofer Strichman, and Yunshan Zhu. Bounded model checking. *Advances in computers*, 58:117–148, 2003.