A Theory of Abstraction for Diagnosis of Discrete-Event Systems

Alban Grastien[*]
NICTA and the Australian National University,
Canberra, Australia

Gianluca Torta
Università di Torino, Torino, Italy

Abstract

We propose a theory of abstraction of discrete-event systems (DES) formulated at the semantic level, i.e., as a function that maps event traces at the original (ground) level to traces at the abstract level.

We study how diagnosis of DES can be performed using an abstract model, and under which conditions this process leads to a correct solution (i.e., a set of alternative diagnoses that include the real status of the system).

Finally, we study how the use of an abstract model can affect the precision of diagnosis, i.e., the presence of spurious system states in the solution. To this end, we introduce the notion of diagnosability with abstract models, which ensures the precision of abstract diagnoses, and we discuss a practical way to test it.

1 Introduction

Diagnosis is the problem of detecting fault occurrences in a system and determining which specific fault occurred. In model-based diagnosis of discrete-event systems (DES, (Cassandras and Lafortune 1999)), this is done by deciding whether the system model allows for traces of a given fault mode consistent with the observation. There exist different techniques for diagnosis of DES, based on different modeling tools: automata, Petri nets, process algebras, propositional logic, etc. They all suffer the same major drawback which is that reasoning on a DES model is exponentially complex in the number of components in the system it is modeling: the system state space is roughly the Cartesian product of its component state spaces.

An important point, which is the underlying motivation of the present work, is that the DES model available for diagnosis is often at the wrong level of abstraction and, in particular, it contains too many irrelevant details. This can usually be explained by the fact that the model is built by assemblying pre-existing component models, or by the fact that it has been designed to support different tasks (including diagnosing different types of faults). *Model abstraction* is a way to move from a model that is too detailed to the correct level of abstraction for the current purpose. Therefore, model abstraction, implicitly or explicitly, has become, and will become, increasingly important to diagnose large systems.

In this paper we propose a theory of abstraction for the diagnosis of DES, formulated at the semantic level as a transformation of the set of possible behaviors of the DES. This transformation indirectly leads to syntactic transformations of the specific representation of the DES as a finite automaton, and therefore also such a representation is considered.

The goal of the present proposal is not that of representing an exhaustive account on the abstraction of DES. Instead, we would like to provide a formal framework for abstraction in diagnosis of DES, and foster further resarch on the topic based on solid ground.

We discuss the properties useful abstractions should satisfy and identify the well-known model increasing property (Giunchiglia and Walsh 1992; Nayak and Levy 1995) as essential. We also demonstrate that observation- and fault-consistency are important in order to preserve the correctness of the diagnosis. Finally, we consider the fundamental property of diagnosability, which states that a fault will always eventually be diagnosed; we show that diagnosability checking for an abstract model should be defined with respect to the original model.

The paper is structured as follows. We introduce diagnosis of DES in the next section. We define abstraction and related properties in Section 3. In Section 4 we show how abstract models may be built. Finally, diagnosis and diagnosability using abstract models are presented, respectively, in Sections 5 and 6.

2 Model-Based Diagnosis

Model-based diagnosis (MBD) is a diagnosis technique performed by comparing a system model with the observation emitted by the system.

In this paper, we are interested in MBD of DES. A

[*]NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

DES is a model for dynamic systems where the system evolution is modeled by the occurrence of discrete events. We will discuss about the DES at two different levels: the *semantic level* describes what the DES models, while the *syntactic level* describes how the DES is represented. Most definitions (notably, the definition of *abstraction function* itself) will be given at the semantic level, although it will often be useful to also discuss their meaning and impact at the syntactic level.

2.1 Semantic Level

Given a set of events Σ, a finite sequence of events will be called a trace and denoted by $u \in \Sigma^*$. The *prefix* relationship will be denoted by $u \sqsubset v$, i.e., u is a prefix of v. To simplify the definitions, we consider that the system can only be in one of two modes $\Upsilon = \{N, F\}$: the nominal mode N and the faulty mode F; this restriction can be lifted without affecting much of our discussion.

A language $\mathcal{B}$ of finite traces is a subset of Σ^*; it is *prefix-closed* if it contains all the prefixes of each trace in it: $\forall u \in \mathcal{B}$, $\forall u' \sqsubset u$, $u' \in \mathcal{B}$; it is *live* if it contains at least a continuation of each trace in it: $\forall u \in \mathcal{B}$, $\exists u' \in \mathcal{B} : u \sqsubset u'$.

Definition 1 *At the semantic level, a* DES M *is defined by a language* $\mathcal{B}(M) \subseteq \Sigma^*$ *such that* $u \in \mathcal{B}(M)$ *is a possible finite evolution of the system according to the model. We require* $\mathcal{B}(M)$ *to be prefix-closed and live.*

The set of possible system evolutions $\mathcal{B}(M)$ *is the union of the sets* $\mathcal{B}_N(M)$ *and* $\mathcal{B}_F(M)$*, which respectively contain the nominal and the faulty evolutions. We require that the set* $\mathcal{B}_N(M)$ *is prefix-closed while the set* $\mathcal{B}_F(M)$ *is live.*

Finally, we assume that a subset $\Sigma_o \subseteq \Sigma$ *of events are* observable.

Note that, because the model may be imprecise or abstract, the sets of nominal $\mathcal{B}_N(M)$ and faulty behaviors $\mathcal{B}_F(M)$ may intersect, i.e., there may exist traces $u \in \mathcal{B}(M)$ that belong to $\mathcal{B}_N(M) \cap \mathcal{B}_F(M)$.

2.2 Syntactic Level

The syntactic level describes how the DES model is represented. We focus on DES models that can be represented as finite automata.

Definition 2 *At the syntactic level, a* DES M *is defined by a tuple* $A(M) = \langle Q, \Sigma, T, \Sigma_o, q_0, \mathcal{Q}\rangle$ *where*

- Q *is a set of* states *and* $q_0 \in Q$ *is the* initial state*,*
- Σ *is a set of* events *and* $\Sigma_o \subseteq \Sigma$ *is the set of* observable events*,*
- $T \subseteq Q \times \Sigma \times Q$ *is a set of* transitions*,*
- $\mathcal{Q} : Q \to 2^{\Upsilon} \setminus \{\emptyset\}$ *is a function labeling the DES states.*

The link between the semantic and the syntactic levels is the following one. The language $\mathcal{B}(M)$ is defined as the list of traces u that label a path on the DES automaton from the initial state. A trace u that leads to a nominal state q (i.e., s.t. $N \in \mathcal{Q}(q)$) is a nominal trace in $\mathcal{B}_N(M)$; a trace u that leads to a faulty state q (i.e., s.t. $F \in \mathcal{Q}(q)$), is a faulty trace in $\mathcal{B}_F(M)$. Since a state q may be labeled by $\{N, F\}$, the traces that lead to it are both in $\mathcal{B}_N(M)$ and $\mathcal{B}_F(M)$.

2.3 Diagnosis

A system evolution modeled by trace $u \in \Sigma^*$ generates observations defined as the sequence of observable events in u. Such a sequence is denoted by $\sigma = obs_{\Sigma_o}(u)$ or simply $obs(u)$. A diagnosis problem is defined by a system model and an observation of a finite trace.

Definition 3 *A model-based diagnosis problem (*MBD problem*) is a pair* $P = \langle M, \sigma\rangle$ *where* M *is a DES and* $\sigma \in {\Sigma_o}^*$ *is the observed (and assumed exact) sequence of observable events that took place in the system.*

The purpose of diagnosis is to estimate whether the fault mode was reached. In MBD, the estimate is done by determining the fault mode of the behaviors predicted by the model for these observations.

Definition 4 *The model-based diagnosis (*MBD*) of problem* $P = \langle M, \sigma\rangle$ *is the set of labels* $\Delta(P) \subseteq \Upsilon$ *defined by* $\Delta(P) = \{l \in \Upsilon \mid \exists u \in \mathcal{B}_l(M) : obs(u) = \sigma\}$.

This corresponds to the following, more familiar, formula defined at the syntactic level on automaton $A(M)$.

$$\begin{aligned}\Delta(P) \quad = \quad & \{l \in \Upsilon \mid \exists q \in Q, \exists u \in \mathcal{B}(M) : \\ & (obs(u) = \sigma) \wedge (l \in \mathcal{Q}(q)) \wedge (q_0 \xrightarrow{u} q)\}\end{aligned}$$

This computation can be implemented by unfolding the model according to the observations (Zanella and Lamperti 2003).

Two important properties that a diagnosis $\Delta(P)$ can exhibit are correctness and precision.

Definition 5 *Let* $\delta \in \Upsilon$ *be the actual mode of the system evolution being monitored. Diagnosis* $\Delta(P)$ *is* correct *if* $\delta \in \Delta(P)$*; it is* precise *if* $\Delta(P) \subseteq \{\delta\}$.

Correctness means that fault mode δ is correctly included in diagnosis $\Delta(P)$; precision means that other fault modes are precisely excluded from diagnosis $\Delta(P)$. It is often impossible to have both correctness and precision because the model is incomplete and the system is only partially observable; in general, however, it is required that the diagnosis is at least correct (Krysander and Nyberg 2008).

We illustrate the above concepts with the simple DES shown in Figure 1. The states are represented by nodes and the transitions by arrows; the nodes are labeled with the associated fault mode(s).

In this example, a user $\underline{\text{r}}$equests for access, and her request may be $\underline{\text{g}}$ranted (in which case she may $\underline{\text{w}}$rite or $\underline{\text{c}}$ancel her action) or $\underline{\text{d}}$enied, in which case she should not be able to write. A $\underline{\text{f}}$ault occurs when the user writes despite a denial.

Assume the (faulty) trace $u_1 = \mathtt{r} \cdot \mathtt{d} \cdot \mathtt{f} \cdot \mathtt{w} \cdot \mathtt{r} \cdot \mathtt{g} \cdot \mathtt{w}$. If the set of observable events is $\Sigma_o = \{\mathtt{r}, \mathtt{d}, \mathtt{c}\}$, then

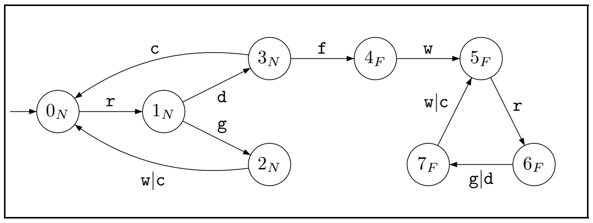

Figure 1: Example DES model automaton.

the observation is $obs(u_1) = \mathtt{r} \cdot \mathtt{d} \cdot \mathtt{r}$ and the MBD is $\Delta = \{F\}$ which is both correct and precise; if the set of observable events is $\Sigma'_o = \{\mathtt{r}, \mathtt{w}, \mathtt{c}\}$, then the observation is $\mathtt{r} \cdot \mathtt{w} \cdot \mathtt{r} \cdot \mathtt{w}$ and the MBD is $\Delta = \{N, F\}$ which is correct but imprecise. Assume on the other hand that the model is incomplete, and that the nominal trace $u_2 = \mathtt{r} \cdot \mathtt{d} \cdot \mathtt{r} \cdot \mathtt{g} \cdot \mathtt{w}$ (which is not allowed by the model) is the real system behavior. Assuming the observable events are $\Sigma_o = \{\mathtt{r}, \mathtt{d}, \mathtt{c}\}$, the observation is $obs(u_2) = obs(u_1) = \mathtt{r} \cdot \mathtt{d} \cdot \mathtt{r}$ and the MBD is $\{F\}$ which is both imprecise and incorrect.

It is usually assumed that all possible traces of the system are in the model, and are labeled correctly with one or more modes $l \in \Upsilon$; in such a case it is easy to see that the MBD is always correct. We shall see that, even when we can make this assumption for the original DES model M, some important conditions are required in order for it to hold also for an abstraction of M.

3 Abstraction

Generally, abstraction is an operation on models that removes irrelevant details. The goal of this section is to provide a sound formalization of the abstraction of DES models suitable for the MBD task.

3.1 Abstraction Definitions

When an abstraction is performed on a DES, the abstract model may be defined on a different set of events. Those new events are usually of higher (abstracted) level. Without loss of generality, we assume the original set and the abstract set of events are disjoint.

The abstraction function presented below defines how a trace is mapped in the new abstract model.

Definition 6 *Let Σ and Σ' be two sets of events. An* abstraction function *α from Σ to Σ' is a total function from Σ^* to Σ'^* such that*

- *$\alpha(\varepsilon) = \varepsilon$ and*
- *$\forall u, v \in \Sigma^*,\ u \sqsubseteq v \Rightarrow \alpha(u) \sqsubseteq \alpha(v)$.*

Given a trace u_1 defined on events of Σ, the abstraction $\alpha(u_1)$ returns a trace on events of Σ'. If trace u_1 is extended with additional events to a trace $v = u_1 \cdot u_2$, the abstraction of v is an extension of $\alpha(u_1)$, i.e., a trace $\alpha(u_1) \cdot u'_2$. Notice however that, in general, it is not the case that $u'_2 = \alpha(u_2)$, i.e., the abstraction of a trace u that can be decomposed as a concatenation of sub-traces $u_1 \cdot \ldots \cdot u_k$ is not necessarily the concatenation of the abstractions of the sub-traces $u_i, i = 1, \ldots, k$.

The abstraction function is defined at the semantic level; we now give a corresponding definition at the syntactic level.

Definition 7 *An* abstraction automaton *is a deterministic finite state machine $A = \langle Q, \Sigma, \Sigma', T, q_0 \rangle$ where*

- *Q is a set of states partitioned in* input states *Q_I and* output states *Q_O, and q_0 is an input state, and*
- *$T \subseteq Q \times (\Sigma \cup \Sigma') \times Q$ is a set of transitions partitioned into* input transitions *T_I and* output transitions *T_O where*
  - *$T_I \subseteq Q_I \times \Sigma \times Q$ such that $\forall q \in Q_I,\ \forall \mathtt{e} \in \Sigma, \exists! q' \in Q : \langle q, \mathtt{e}, q' \rangle \in T_I$, and*
  - *$T_O \subseteq Q_O \times \Sigma' \times Q$ such that $\forall q \in Q_O,\ \exists! \langle \mathtt{e}, q' \rangle \in \Sigma' \times Q : \langle q, \mathtt{e}, q' \rangle \in T_O$.*

In this definition, $\exists!$ stands for *there exists exactly one*. The automaton is partitioned into two sub-automata. The input sub-automaton reads a trace from Σ^*; because any trace can be emitted, this sub-automaton is complete, i.e., it defines a successor for any event. The output sub-automaton outputs a trace from Σ'^* which corresponds to the abstract trace.

The traces of an abstraction automaton are defined similarly to traces of regular automata except that we only consider traces that end up in an input state, i.e. we define the language of A as:

$$\mathcal{B}(A) = \{v \in (\Sigma \cup \Sigma')^* \mid \exists q \in Q_I : q_0 \xrightarrow{v} q\}$$

Given a trace $v \in (\Sigma_1 \cup \Sigma_2)^*$, we denote by $Pr_{\Sigma_i}(v)$ the restriction of v to the events of Σ_i.

We now link the notion of abstraction automaton to the notion of abstraction function.

Definition 8 *An abstraction automaton $A = \langle Q, \Sigma, \Sigma', T, q_0 \rangle$ is an* automaton representation *of α (denoted as $A(\alpha)$) iff:*

- *for each trace $v \in \mathcal{B}(A)$, then $\alpha(Pr_\Sigma(v)) = Pr_{\Sigma'}(v)$, and*
- *for each trace $u \in \Sigma^*$, there exists a trace $v \in \mathcal{B}(A)$ such that $Pr_\Sigma(v) = u$ (and therefore $Pr_{\Sigma'}(v) = \alpha(u)$).*

It should be noted that not every abstraction function can be represented by a finite automaton; however in this paper we focus on abstraction functions that have such a representation.

Definition 7 is illustrated in Figure 2 for an abstraction function α where $\Sigma = \{\mathtt{c}, \mathtt{d}, \mathtt{f}, \mathtt{g}, \mathtt{r}, \mathtt{w}\}$, $\Sigma' = \{\mathtt{d}', \mathtt{g}', \mathtt{w}'\}$, and for every trace $u \in \Sigma^*$, events $\mathtt{r}$, $\mathtt{f}$ and $\mathtt{c}$ are ignored, whilst events $\mathtt{g}$, $\mathtt{d}$, and $\mathtt{w}$ are respectively mapped to $\mathtt{g}'$, $\mathtt{d}'$, and $\mathtt{w}'$. In this example, q is the only input state. It is easy to see that the automaton is indeed a representation of α since it satisfies both conditions of Definition 8.

We can now define the abstraction of a DES M.

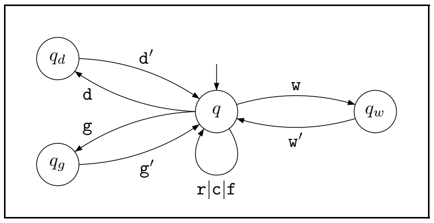

Figure 2: Example of automaton representation of an abstraction function.

Definition 9 *Let M and M' be two DES defined on the event sets Σ and Σ', and let α be an abstraction function from Σ to Σ'. Then M' is an* abstraction *of M through α.*

Note that this (very general) definition closely mirrors the definition of abstraction of a formal system given in (Giunchiglia and Walsh 1992), but deals with the *semantic* level, while Giunchiglia and Walsh' definition is at the syntactic level.

The definition ignores the languages $\mathcal{B}(M)$ and $\mathcal{B}(M')$ associated with the DES M and M', as far as such languages are respectively defined on the two sets Σ, Σ' involved by the abstraction function α. One consequence is that, given M and α there exist an infinite number of models M' that are abstractions of M through α; we will come back to the point of deriving a model M' from M and α in Section 4.

3.2 Abstraction Properties

The first property we introduce relates the observations on the original and the abstracted traces.

Definition 10 *Let α be an abstraction function from Σ to Σ' and let Σ_o, Σ'_o be the observable events of Σ and Σ'. The abstraction function is* obs-consistent *w.r.t. Σ_o, Σ'_o if $\forall u, v \in \Sigma^*$, $obs_{\Sigma_o}(u) \sqsubseteq obs_{\Sigma_o}(v) \Rightarrow obs_{\Sigma'_o}(\alpha(u)) \sqsubseteq obs_{\Sigma'_o}(\alpha(v))$.*

The obs-consistency property ensures that the abstractions of two traces that are equivalent from the point of view of the emitted observations will remain equivalent after the abstraction. An abstraction through an obs-consistent abstraction function will be also called obs-consistent.

Lemma 1 *If abstraction function α is obs-consistent w.r.t. Σ_o, Σ'_o, there exists an abstraction function α_o from Σ_o to Σ'_o such that:*

$$\forall u \in \Sigma^*, obs_{\Sigma'_o}(\alpha(u)) = \alpha_o(obs_{\Sigma_o}(u)).$$

Abstraction function α_o is called the *observable abstraction function* of α (w.r.t. Σ_o and Σ'_o).

Sketch of Proof: First of all we note that, since α is obs-consistent w.r.t. Σ_o, Σ'_o, for any $\sigma \in {\Sigma_o}^*$, for any $u \in \Sigma^*$ s.t. $obs_{\Sigma_o}(u) = \sigma$, then $obs_{\Sigma'_o}(\alpha(u)) = obs_{\Sigma'_o}(\alpha(\sigma))$ (recall that $\sigma \in \Sigma^*$, so α is defined on σ). We define α_o such that for all $\sigma \in {\Sigma_o}^*$, $\alpha_o(\sigma) = obs_{\Sigma'_o}(\alpha(\sigma))$. It is easy to show that α_o satisfies both Definition 6 (i.e., it is an abstraction function) and the additional condition in the statement of this lemma. □

The result of Lemma 1 together with Definition 10 is very important. In a diagnosis context, we observe the sequence $\sigma = obs(u)$ from an unknown trace u; but what this result says is that there exists a well-defined abstraction $\alpha_o(\sigma)$ that can be used for the diagnosis at the abstract level (see section 5).

Another useful property of an abstraction relates this time to the faulty mode(s) attached with each trace.

Definition 11 *Let DES M' be an abstraction of DES M through α. The abstraction is* fault-consistent *if for each fault mode $l \in \Upsilon$, for each $u \in \mathcal{B}(M)$:*

$$\alpha(u) \in \mathcal{B}(M') \wedge u \in \mathcal{B}_l(M) \Rightarrow \alpha(u) \in \mathcal{B}_l(M')$$

Fault consistency ensures that if trace u is associated with fault mode l, then also its abstraction $\alpha(u)$ is associated with l[1]. In particular, if one or more behaviors $u_1, u_2, \ldots$ are abstracted to the same behavior $u' = \alpha(u_i), i = 1, 2, \ldots$ then u' must be associated with every fault mode associated with at least one of the behaviors u_i. Note that fault-consistency, contrary to obs-consistency, is not just a property of α, but it is a property of a specific abstraction M' of M through α.

Following (Giunchiglia and Walsh 1992), we now define model-increasing and model-decreasing abstractions.

Definition 12 *Let DES M' be an abstraction of DES M through α.*
The abstraction is model-increasing *(MI) if $\forall u \in \Sigma^*$, $u \in \mathcal{B}(M) \Rightarrow \alpha(u) \in \mathcal{B}(M')$.*
The abstraction is model-decreasing *(MD) if for all $u \in \Sigma^*$, $\alpha(u) \in \mathcal{B}(M') \Rightarrow u \in \mathcal{B}(M)$.*

The model-increasing property indicates that the abstracted model includes the abstractions of all behaviors in the original model. The model-decreasing property indicates that the original model includes the ground behaviors corresponding to all behaviors in the abstract model. Also MI and MD are properties of the abstraction M' of M through α, rather than just of α.

4 Building Abstract Models

Now that we have defined abstractions and several properties, we can tackle the following question: if we are given the original model M and an abstraction function α, how can we proceed to build a model M' which is an abstraction of M through α, and possibly exhibits one or more of the properties defined above? A first attempt to answer such a question is the topic of the present section.

The canonical abstraction of a model through a given abstraction function is defined as the collection of traces that can be obtained by abstracting the traces in the original model.

[1]In this paper the possible values of l are just N and F.

Definition 13 *The canonical abstraction $\widehat{M}'$ of model M through abstraction function α is the model defined by $\mathcal{B}_l(\widehat{M}') = \{u' \in \Sigma'^* \mid \exists u \in \mathcal{B}_l(M) : u' = \alpha(u)\}$.*

The definition ensures that the abstractions $u' = \alpha(u)$ of the traces u of model M are indeed traces of model $\widehat{M}'$, and that they are associated with the correct fault modes.

Lemma 2 *The canonical abstraction $\widehat{M}'$ of M is MI and fault-consistent. Moreover, the language $\mathcal{B}(\widehat{M}')$ is the minimal language with such properties w.r.t. M and abstraction function α.*

Sketch of Proof: The definitions of MI and fault-consistence are trivially satisfied by the canonical abstraction. Furthermore, it is obvious that removing any trace u from $\mathcal{B}_l$ will either make the abstraction non MI (if $u \notin \mathcal{B}_{\neg l}$) or non fault-consistent (if $u \in \mathcal{B}_{\neg l}$) where $\{l, \neg l\} = \Upsilon$, i.e., $\neg l$ represents the other fault label in the set $\{N, F\}$. □

An interesting corollary of Lemma 2 is that abstraction M' of M is MI iff $\mathcal{B}(M') \supseteq \mathcal{B}(\widehat{M}')$, which provides a lower bound for MI abstractions.

At the syntactic level, assuming abstraction function α can be represented by finite automaton $A(\alpha) = \langle Q_\alpha, \Sigma, \Sigma', T_\alpha, q_{0\alpha}\rangle$, it is possible to build the automaton of abstract model $\widehat{M}'$. To this end, we first define the extension of automaton $A(M)$.

Definition 14 *Given model M and abstraction function α from Σ to Σ', the* extension *of automaton $A(M)$ with respect to $A(\alpha)$ is the automaton $A(M_x) = \langle Q_x, \Sigma_x, T_x, \Sigma_{ox}, q_{0x}, \mathcal{Q}_x\rangle$ defined by:*

- *$Q_x = Q \times Q_\alpha$ and $q_{0x} = \langle q_0, q_{0\alpha}\rangle$,*
- *$\Sigma_x = \Sigma \cup \Sigma'$ and $\Sigma_{ox} = \Sigma_o$,*
- *$T_x = T_{xM} \cup T_{xM'}$, where*
  - *$T_{xM} = \{\langle\langle q, q_\alpha\rangle, \mathtt{e}, \langle q', q'_\alpha\rangle\rangle \mid \mathtt{e} \in \Sigma \wedge \langle q_\alpha, \mathtt{e}, q'_\alpha\rangle \in T_\alpha \wedge \langle q, \mathtt{e}, q'\rangle \in T\}$, and*
  - *$T_{xM'} = \{\langle\langle q, q_\alpha\rangle, \mathtt{e}, \langle q', q'_\alpha\rangle\rangle \mid \mathtt{e} \in \Sigma' \wedge \langle q_\alpha, \mathtt{e}, q'_\alpha\rangle \in T_\alpha \wedge\ q = q'\}$,*
- *$\mathcal{Q}_x(\langle q, q_\alpha\rangle) = \mathcal{Q}(q)$.*

A Σ-path *on the extended automaton $A(M_x)$ is a double sequence of states and transitions $q_x^0 \xrightarrow{\mathtt{e_1}} \ldots \xrightarrow{\mathtt{e_k}} q_x^k$ such that all events $\mathtt{e_i}$ are in Σ. When there exists a Σ-path from q_x^0 to q_x^k, we write $q_x^0 \xrightarrow{\Sigma^*} q_x^k$.*

The extended automaton follows two traces on $A(M)$ and on $A(\alpha)$ together: it follows a path on $A(M)$ as long as it stays in an input state of $A(\alpha)$; when it reaches an output state of $A(\alpha)$, it takes all the transitions on Σ' events before continuing the path on $A(M)$.

We can now define the canonical automaton as a projection of the extended automaton on the abstract events.

Definition 15 *The* canonical abstraction automaton *of model M, whose extension is $A(M_x) = \langle Q_x, \Sigma_x, T_x, \Sigma_{ox}, q_{0x}, \mathcal{Q}_x\rangle$, with respect to α is the automaton $A(\widehat{M}') = \langle Q', \Sigma', T', \Sigma'_o, q'_0, \mathcal{Q}'\rangle$ defined by:*

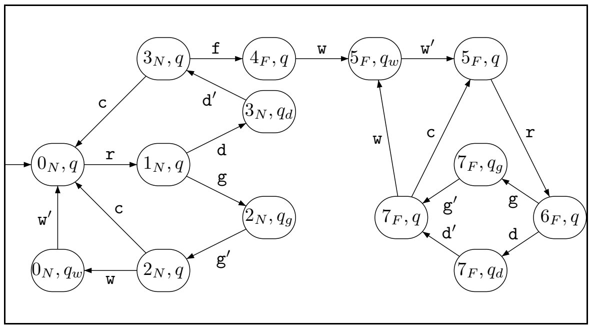

Figure 3: Extended automaton

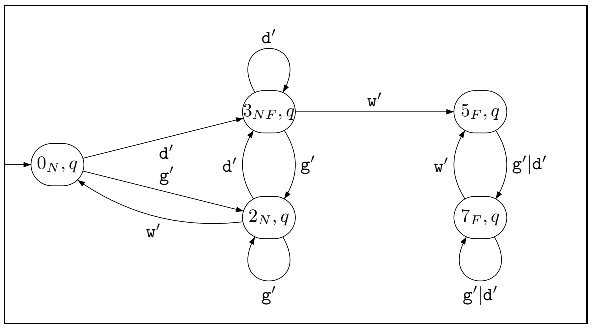

Figure 4: Canonical automaton

- *$Q' = Q_x$ and $q'_0 = q_{0x}$,*
- *$\Sigma'_o = \emptyset$,*
- *$T' = T_{xM'} \cup T_{jump}$ where*
  - *$T_{xM'}$ is defined as above,*
  - *$T_{jump} = \{\langle q, \mathtt{e}, q'\rangle \mid \exists q'' \in Q_x : (q \xrightarrow{\Sigma^*} q'') \wedge (\langle q'', \mathtt{e}, q'\rangle \in T_{xM'})\}$,*
- *$\mathcal{Q}'(\langle q, q_\alpha\rangle) = \bigcup_{q' \in Q, q \xrightarrow{\Sigma^*} q'} \mathcal{Q}(q')$.*

The extended automaton and the canonical automaton for the model of Figure 1 and the abstraction function represented in Figure 2 are given in Figure 3 and Figure 4.

We note that state $\langle 3, q\rangle$ in the canonical automaton is labeled with N, F, i.e., the union of $\mathcal{Q}(\langle 3, q\rangle)$ and $\mathcal{Q}(\langle 4, q\rangle)$ since $\langle 3, q\rangle$ and $\langle 4, q\rangle$ are connected by the Σ-path $\langle 3, q\rangle \xrightarrow{\mathtt{f}} \langle 4, q\rangle$ in the extended automaton.

Theorem 1 *The canonical automaton presented in Definition 15 is a syntactic representation of the canonical model presented in Definition 13.*

Sketch of Proof: On the one hand, it can be shown that any trace u_x on the extended automaton corresponds to the intertwinning of a trace u and (possibly a prefix of) its abstraction $\alpha(u)$, and that it contains all such u_x. On the other hand, it is trivial to see that the canonical automaton is the projection of the traces of the extended automaton on the events of

Σ'. Therefore, the canonical automaton contains exactly the traces that are abstractions of traces of the original model. Furthermore, it is trivial to see that each trace u' of the canonical automaton is associated with the fault modes associated with the trace u such that $\alpha(u) = u'$. $\square$

Given an abstraction function α, we can therefore generate the canonical abstraction automaton $A(\widehat{M'})$ from the original model $A(M)$, and such abstracion is MI, fault-consistent and (trivially) obs-consistent. While these properties are fundamental for diagnosis, as we shall see in the next section, there is a priori no guarantee that the abstract automaton will be smaller than the original one. Since the simplification of the model is one of the main goals of abstraction, additional syntactic transformations based on the characteristics of α will in general be performed on $A(\widehat{M'})$ in order to reduce the size of the automaton without losing its properties; see (Pencolé, Kamenetsky, and Schumann 2006; Kan John, Grastien, and Pencolé 2010) for two examples of syntactic transformations which can yield significant size reductions.

5 Abstract Diagnosis

We now discuss how diagnosis can be computed with an abstract model M'. We assume that M' is an abstraction of the original model M through an obs-consistent abstraction function α, and that α_o is the observable abstraction function derived from α.

The system produces a trace $u \in \Sigma^*$ and generates observation $\sigma = obs(u)$. The diagnosis problem could be defined at the ground level by $P = \langle M, \sigma\rangle$; however we want to solve the problem using the abstract model M'. Since observation σ is given in terms of Σ_o events instead of the abstract observable events Σ'_o, we need to abstract σ using α_o.

Definition 16 *An* abstract MBD problem *is a tuple $AP = \langle M', \alpha_o, \sigma\rangle$ where M' is a model, α_o is an abstraction function from Σ_o to Σ'_o, and $\sigma \in {\Sigma_o}^*$ is an observable trace.*

The abstract MBD $\Delta(AP)$ *is the MBD of problem $P' = \langle M', \alpha_o(\sigma)\rangle$.*

We notice how important the obs-consistency property is. If it does not hold, then it means that α_o may not exist; therefore, it would be impossible to translate σ to a sequence of observations in the abstract language.

The question is however whether the result $\Delta(P')$ of the abstract MBD problem will be equivalent to the result $\Delta(P)$ of the original problem. There are many reasons why these results might be different:

- the fault modes associated with a trace and its abstraction might differ: this could affect both correctness and precision.
- M' may include traces that correspond to no trace in M and conversely: this could affect respectively precision and correctness.
- two distinguishable traces in $\mathcal{B}(M)$ may be indistinguishable in $\mathcal{B}(M')$ (i.e. $obs(u) \neq obs(u') \wedge obs'(\alpha(u)) = obs'(\alpha(u')))$: this could affect precision.

In the following theorem we identify a set of sufficient conditions such that correctness is preserved.

Theorem 2 *Let $P = \langle M, \sigma\rangle$ be an MBD problem and let α be an abstraction function such that M' is an abstraction of M through α.*

If M' is a model-increasing, obs-consistent, and fault-consistent abstraction of M through α then the following holds: if diagnosis $\Delta(P)$ is correct, then diagnosis $\Delta(AP)$ of abstract diagnosis problem $AP = \langle M', \alpha_o, \sigma\rangle$ is correct.

Proof: Let $\delta \in \Upsilon$ denote the actual diagnosis mode of the system.

If $\Delta(P)$ is correct, then $\delta \in \Delta(P)$; therefore, there exists $u \in \mathcal{B}_\delta(M)$ such that $obs(u) = \sigma$.

Let us denote $\alpha(u)$ by u'. Because the abstraction is model-increasing, $u' \in \mathcal{B}(M')$; furthermore, because the abstraction is fault-consistent, $u' \in \mathcal{B}_\delta(M')$. Finally, because the abstraction is obs-consistent, $obs'(u') = \alpha_o(obs(u)) = \alpha_o(\sigma)$. Therefore, $\delta \in \Delta(P')$. $\square$

This result is quite important, and from now on we shall assume abstractions that are model-increasing, obs-consistent, and fault-consistent (we will denote such abstractions as MI-OF abstractions); these properties are quite natural, and relate to separate elements of the model (list of traces, observations, fault modes) which can be checked independently.

Unfortunately, it is not possible to state a similar result about the precision of diagnosis. However, it should be noted that for dynamic systems such as DES, a fault usually requires time to propagate and become identifiable; rather than precision, the property the system is usually required to satisfy is diagnosability. Therefore, instead of tackling the problem of preserving the precision of diagnosis across abstraction, we will discuss how to preserve diagnosability across abstraction. This is the focus of next section.

6 Abstract Diagnosability

Diagnosability is the problem of determining whether the diagnostic algorithm will be able to diagnose a fault if it occurs. Assume that the model given in Figure 1 models exactly the possible behaviors of the system and consider the abstraction given in Figure 5 where abstraction function α is the one represented in Figure 2. Assume also that events $\mathtt{g}'$, $\mathtt{d}'$ and $\mathtt{w}'$ are observable.

Looking only at Figure 5, it seems the system is not diagnosable since it might follow a trace $w' = \mathtt{d}'^{\omega}$ which is both nominal and faulty, generating the observation $\sigma' = obs'(w') = \mathtt{d}'^{\omega}$. However, looking at the model in Figure 1, the system cannot generate a faulty trace w_f such that $\alpha(w_f) = w'$. In actual facts, the model in Figure 5 is diagnosable because a faulty behavior will always generate the observable abstract sub trace $\mathtt{d}' \cdot \mathtt{w}'$ which can only be emitted by faulty abstract traces.

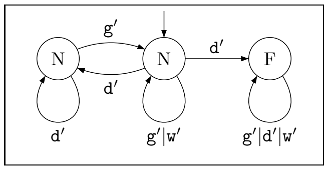

Figure 5: Example of incorrect diagnosability analysis.

We formalize this result by including the exact system model in the definition of diagnosability; therefore, diagnosability involves two models of the system while classical definitions involve only one model. Because the exact model of the system is usually not available, we make the common assumption that the available ground model exactly captures the system behavior.

6.1 Diagnosability with an Abstract Model

We mentioned in a former section that correctness is preserved if the abstraction is MI-OF. Therefore, our new definition of diagnosability is meant to apply particularly to the ground model M and an MI-OF abstraction M' of such model.

Note that, since it was proved in (Cordier, Travé-Massuyès, and Pucel 2006) that diagnosability for multiple faults reduces to diagnosability for every fault, our simplifying assumption that there is only one fault mode does not impact our discussion on diagnosability.

Definition 17 *System Γ is* diagnosable *by model M' iff*

1. *there exists a finite bound n such that for any faulty trace u of system Γ, if u generated n or more events after reaching the faulty mode, then the diagnosis of problem $P'(M', \alpha_o(obs(u)))$ is $\{F\}$, and*
2. *for any trace u of system Γ, the diagnosis of problem $P'(M', \alpha_o(obs(u)))$ returns $\{F\}$ only if trace u is faulty.*

Diagnosability definitions usually only include the first item as it is assumed the model and diagnostic algorithm are correct, which implies the second item of Definition 17. Since we assume that model M' is correct (because it is an MI-OF abstraction), we also concentrate on the first item, which can be expressed as the following equation (inspired by the classical definition of (Sampath et al. 1995)).

$$\begin{array}{c}\exists n \in \mathbf{N} : \forall u, u_1, u_2 \in {\Sigma_G}^*, \\ (u = u_1.u_2 \in \mathcal{B}(M)) \ \wedge \ (u_1 \in \mathcal{B}_F(M)) \ \wedge \\ (|u_2| \geq n) \ \Rightarrow D\end{array} \qquad (1)$$

where D is the diagnosability property

$$\Delta(\langle M', obs_{\Sigma'_o}(\alpha(u))\rangle) = \{F\}. \qquad (2)$$

As opposed to the classical definition where it is assumed $M = M'$, Equation 1 involves two models: the ground model M and the model M' used for the diagnosis. The only reference to this distinction we are aware of is (Grastien 2009).

6.2 Diagnosability by Twin Plant

Diagnosability is usually performed using the twin plant technique (Jiang et al. 2001). Two models are used: the first model is used to generate a faulty trace (u in Equation 1) and the second model represents the diagnoser which tries to find a nominal trace that generates the same observation as the faulty trace. Both models are synchronized so that a trace on the twin plant represents two traces from each model generating the same observation. Following the role of each model, we call *simulation model* M_S the first model and *diagnoser model* M_D the second model.

In the standard twin plant technique, models M_S and M_D are instances of the *same* system model. Following Equation 1, we need to redefine the twin plant approach when M_D is an MI-OF abstraction of M_S. The intuition is given in Figure 6. Because M_S and M_D are defined on different sets of events, they cannot be directly synchronized. Instead, they are transformed in such a way that they can be synchronized. Consider a faulty trace u on simulation model M_S; this trace generates the observations $obs(u)$. A transformation that makes the observations $obs(u)$ meaningful for diagnosis model M_D is to translate these observations by $\alpha_o(obs(u))$. The transformation Tr is performed by synchronizing M_S with the automaton representation of α_o.

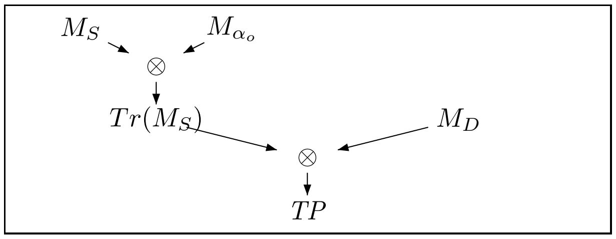

Figure 6: Twin-Plant with an Abstract Model

Definition 18 *The* twin plant $TP(M_S, M_D, M_{\alpha_o})$ *is the Büchi automaton $\langle Q, \Sigma, T, q_0, A\rangle$ defined as follows:*

- *$Q = Q_S \times Q_D \times Q_{\alpha_o}$ and $q_0 = \langle q_{S0}, q_{D0}, q_{\alpha_o 0}\rangle$,*
- *$\Sigma = \Sigma_S \cup \Sigma_D$,*
- *$T = \{\langle\langle q_S, q_D, q_{\alpha_o}\rangle, \mathsf{e}, \langle q'_S, q'_D, q'_{\alpha_o}\rangle\rangle\}$ such that*
  1 **a**– $\mathsf{e} \in \Sigma_{\alpha_o} \wedge \langle q_{\alpha_o}, \mathsf{e}, q'_{\alpha_o}\rangle \in T_{\alpha_o}$ *or*
  b– $\mathsf{e} \notin \Sigma_{\alpha_o} \wedge (q_{\alpha_o} = q'_{\alpha_o})$,
  and
  2 **a**– $\mathsf{e} \in \Sigma_S \wedge \langle q_S, \mathsf{e}, q'_S\rangle \in T_S \wedge (q_D = q'_D)$ *or*
  b– $\mathsf{e} \in \Sigma_D \wedge (q_S = q'_S) \wedge \langle q_D, \mathsf{e}, q'_D\rangle \in T_D$.
  and
- *$A = \{q \in Q_S \mid F \in \mathcal{Q}_S(q)\} \times \{q \in Q_D \mid N \in \mathcal{Q}_D(q)\} \times Q_{\alpha_o}$.*

Twin plant TP is defined on the Cartesian product of the state space of the models and the automaton $A(M_{\alpha_o})$. Each transition on twin plant TP corresponds to a transition on either simulation model M_S or diagnoser model M_D; in case the associated event is observable, it also triggers a state change on automaton $A(M_{\alpha_o})$ which forces the equivalence between the observations of both models. Set A represents the ambiguous states, i.e., states where the actual trace $u_S \in M_S$ is faulty whilst a nominal diagnosed trace $u_D \in M_D$ exists that is observationally equivalent.

Theorem 3 *Let M be the ground model of system Γ, let M' be an MI-OF abstraction of M through abstraction function α. Then, system Γ is diagnosable through model M' iff twin plant $TP(M, M', M_{\alpha_o})$ contains no infinite cycle on ambiguous states reachable from its initial state.*

Sketch of Proof: Because the abstraction is MI-OF, we only need to prove Equation 1.

1) Assume $TP(M, M', M_{\alpha_o})$ contains an infinite cycle on ambiguous states reachable from its initial state. Then, there exists a trace $w_{TP} \in \mathcal{L}_A(TP)$ (where the index A means that the trace reaches and stays in the set of ambiguous states). Then because of the way the twin plant was defined the projections of w_{TP} on Σ_G and Σ' represent an infinite faulty trace w_G of M as well as a nominal trace w' of M'. Furthermore, those traces are such that $\alpha_o(obs_M(w_G)) = obs_{M'}(w')$. Therefore, the system is not diagnosable.

2) Assume the system is not diagnosable. Then there exist an infinite faulty trace $w_G \in \mathcal{L}_F(M)$ and a nominal trace $w' \in \mathcal{L}_N(M') \cup \mathcal{B}_N(M')$ such that for all $u_G \sqsubset w_G$, $\exists u' \sqsubseteq w'$ where $obs_{\Sigma'}(u') = \alpha(obs_{\Sigma_o}(u_G))$. Because the state spaces of M, M' and M_{α_o} are finite, it is possible to choose w' and w_G such that they loop, i.e., such that $w_G = v_1 \cdot (v_2)^\omega$, $w' = v'_1 \cdot (v'_2)^\omega$. It is then easy to show that there exists $w_{TP} \in \mathcal{L}_A(TP)$ that corresponds to the intertwining of these traces. □

The test complexity is quadratic in the size of twin plant TP (Jiang et al. 2001).

7 Conclusions

This paper presented a theory of abstraction for diagnosis of DES. In contrast to the general theory of reformulation defined in (Choueiry, Iwasaki, and McIlraith 2005), we are only interested in the abstraction of DES system models, and in how the abstract model compares with respect to the original model forthe specific task of diagnosis.

Abstraction has already been exploited for the diagnosis of DES. Pencolé et al. (2006) defined the model as a set of synchronous automata each one modeling a component in the system; the abstraction operated on the model eliminates some component automata that are identified as unnecessary for diagnosing a specific fault. This idea is further extended by Kan John et al. (2010), where some connections among components are ignored; the isolated components are automatically eliminated as a side-effect. These works define abstractions that are simple, intuitive and useful, but they do not provide a general formal framework for abstraction in DES diagnosis, which to the best of our knowledge is still missing from the literature.

We identified the obs-consistency property as essential to allow diagnosis with an abstract model, and the model-increasing (MI) and fault-consistency properties as essential for abstract diagnosis correctness. In order to check the diagnostic precision of an abstract DES model, we also introduced a diagnosability test which involves both the abstract and the ground model. The proposed theory is formulated at the semantic level as a transformation of the set of possible behaviors of the DES; this choice made it possible to express the fundamental properties mentioned above (most notably, MI) in a natural and intuitive way. However, throughout the paper, we have linked the semantic notions to the corresponding syntactic notions, based on the representation of the DES model as a finite automaton; this will allow us to study different types of abstraction and, in particular, possible abstraction operations to be applied to the canonical abstraction automaton.

References

Cassandras, Chr., and Lafortune, St. 1999. *Introduction to discrete event systems.* Kluwer Academic Publishers.

Choueiry, B.; Iwasaki, Yu.; and McIlraith, Sh. 2005. Towards a practical theory of reformulation for reasoning about physical systems. *Artificial Intelligence (AIJ)* 162:145–204.

Cordier, M.-O.; Travé-Massuyès, L.; and Pucel, X. 2006. Comparing diagnosability in continuous and discrete-event systems. In *Seventeenth International Workshop on Principles of Diagnosis (DX-06)*, 55–60.

Giunchiglia, F., and Walsh, T. 1992. A theory of abstraction. *Artificial Intelligence (AIJ)* 56(2–3):323–390.

Grastien, Al. 2009. Symbolic testing of diagnosability. In *20th International Workshop on Principles of Diagnosis (DX-09)*, 131–138.

Jiang, Sh.; Huang, Zh.; Chandra, V.; and Kumar, R. 2001. A polynomial algorithm for diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control (TAC)* 46(8):1318–1321.

Kan John, Pr.; Grastien, Al.; and Pencolé, Y. 2010. Synthesis of a distributed and accurate diagnoser. In *21st International Workshop on Principles of Diagnosis (DX-10)*, 209–216.

Krysander, M., and Nyberg, M. 2008. Statistical properties and design criterions for fault isolation in noisy systems. In *Nineteenth International Workshop on Principles of Diagnosis (DX-08)*, 101–108.

Nayak, P., and Levy, A. 1995. A semantic theory of abstraction. In *Proc. IJCAI*, 196–202.

Pencolé, Y.; Kamenetsky, D.; and Schumann, An. 2006. Towards low-cost fault diagnosis in large component-based systems. In *Sixth IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (SafeProcess-06).*

Sampath, M.; Sengupta, R.; Lafortune, St.; Sinnamohideen, K.; and Teneketzis, D. 1995. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control (TAC)* 40(9):1555–1575.

Zanella, M., and Lamperti, Gi. 2003. *Diagnosis of active systems.* Kluwer Academic Publishers.