Diagnosability of Input Output Symbolic Transition Systems

Gauvain Bourgne[*], Philippe Dague[†], Farid Nouioua[‡] and Nicolas Rapin[§]
[*]*NII, Tokyo, Japan, Email: bourgne@nii.ac.jp*
[†]*LRI - Univ Paris 11, Orsay, France, Email: philippe.dague@lri.fr*
[‡]*LSIS - Univ Paul Cézanne, Aix-Marseille 3, France, Email: farid.nouioua@univ-cezanne.fr*
[§]*CEA LIST, Laboratory of Model driven engineering for embedded systems*
Point Courier 94, Gif-sur-Yvette, F-91191 France, Email: nicolas.rapin@cea.fr

Abstract

Diagnosability checking of discrete-event systems has been extensively studied in the framework of classical non symbolic models such as Labeled Transition Systems. It happens that in practice such models tend to need too much space to be efficiently processed. By opposition, symbolic approaches offer an expressive, easy and concise way to model systems, and checking diagnosability from such symbolic models can benefit from this reduction of space complexity. Indeed, though this will generally translate into time complexity, such a tradeoff is advantageous, as diagnosability checking is something that is usually done at design stage. This is why this paper proposes a theoretical framework to check diagnosability of Input Output Symbolic Transition Systems (IOSTS) by adapting the twin plant approach to the symbolic case and relying on the use of a symbolic model checker. This theoretical work is being currently applied to embedded functions inside a vehicle in the context of an industrial project and a simplified version of this problem will serve as a running example throughout the presentation.

Index Terms

Diagnosability checking, Input Output Symbolic Transition Systems, Symbolic Execution.

1. Introduction

Diagnosabilty checking is the problem of determining whether a faulty mode in a given system can be distinguished from normal mode through a finite number of observations. To determine this, we need some model of the system's behavior. Most classical discrete event models are based on the so-called Labeled Transition System (LTS). An LTS is a finite graph whose nodes represent states of the system, and edges are events leading from one state to another. Only some of the events are defined as observable, and diagnosability is then defined using observable traces of paths in the graph.

However, when the system is complex, this kind of models can sometimes become very large and the space complexity thus becomes problematic. One way to deal with this, is to divide the models into smaller components, and define local diagnosability, before deriving global diagnosability of the system through distributed mechanisms (cf [1], [2]). Another approach, which is presented here, is to use more expressive transition systems using variables and symbolic contents. We will thus present here diagnosability for Input Output Symbolic Transition System (IOSTS), which is a more concise and expressive way to model systems' behavior. This symbolic approach will reduce space complexity, though it will be done at the cost of higher time complexity. Moreover, IOSTSs are also especially appropriate to describe a component interacting with an environment and other components. Though we will focus here on symbolic content, future works will address distributed systems.

We will first, in Section 2 present our symbolic model of the system. Then, Section 3 will deal with faults representation and observability, before defining diagnosability for IOSTS. On this basis, we will give in Section 4 our method to automatically check the diagnosability of an IOSTS. At last, we will conclude in Section 7.

2. Input Output Symbolic Transition Systems

IOSTSs represent IOLTSs in a concise and more expressive manner by using *variables*. An IOSTS is composed of graph part and data part. The data part is given by a decidable first-order theory T of first order langage $\mathcal{L}$ both with a structure M being a model of this theory. In the sequel *variables* refers to the variables V of $\mathcal{L}$. $\mathcal{T}_A$ (resp. $\mathcal{F}_A$) denotes the set of terms (resp. formulae) of $\mathcal{L}$ containing only variables of A. A map $\nu \in M^A$ (resp. $\rho \in (\mathcal{T}_A)^A$) where $A \subseteq V$ is called an interpretation (resp. a substitution) of variables of A. It is canonically extended to terms and formulae.

2.1. Definition

To define an IOSTS, we first specify its *state variables* A and *communication channels* $C = C_o \cup C_u$, where C_o

is the set of *observable* communication channels, and C_u represents the *unobservable* ones. IOSTSs interact with their environment through actions.

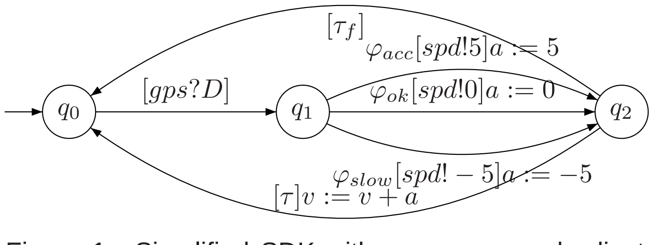
Definition 1 (Actions) *The set of actions, denoted $Act(A, C) = Input(A, C) \cup Output(A, C) \cup Internal$, where $Input(A, C) = \{c?y \mid c \in C, y \in A\}$ and $Output(A, C = \{c!t \mid c \in C, t \in \mathcal{T}_A\}$.*

Elements of $Input(A, C)$ are stimulations of the system from the environment: $c?x$ represents the reception of a value through channel c which is assigned to x where x is a state variable. $Output(A, C)$ are responses of the system to the environment: $c!t$ is the emission of the value t through the channel c. $Internal = \Sigma_o \cup \Sigma_u$ is a set of symbols τ_i used to characterize internal transitions. Σ_o and $\Sigma_u \cup \Sigma_f$ contains respectively *observable* and *unobservable actions*. Especially, $\Sigma_f \subseteq \Sigma_u$ is a subset of (unobservable) *faulty* actions.

Definition 2 (IOSTS) *An IOSTS over (A, C) is a triple $G = (Q, q_0, Trans)$ where Q is a finite a set of* locations, *$q_0 \in Q$ is the* initial location *and $Trans \subseteq Q \times \mathcal{F}_A \times Act(\Sigma) \times (\mathcal{T}_A)^A \times Q$. A transition $tr := (q, \varphi, act, \rho, q')$ of $Trans$, also denoted by $q \xrightarrow{\varphi[act]\rho} q'$, is composed of a source location q, denoted by $src(tr)$, a guard φ denoted by $grd(tr)$, an action act denoted by $act(tr)$, a substitution of variables ρ and a target location q' denoted by $tgt(tr)$. For each location $q \in Q$, there is a finite number of transitions of source location q.*

We consider IOSTSs generating a *live* language, that is, IOSTSs without sink states: for all $q \in Q$, $\bigvee_{tr \in T, src(tr)=q} grd(tr)$ is a tautology.

Example 1 *Figure 1 represents an IOSTS modeling a simplified SDK (Smart Distance Keeping) system that enables a truck to regulate its speed according to its distance to the next vehicle, given by a GPS.*



$[\tau_f]$
$\varphi_{acc}[spd!5]a := 5$
$[gps?D]$
$\varphi_{ok}[spd!0]a := 0$
q_0
q_1
q_2
$\varphi_{slow}[spd! - 5]a := -5$
$[\tau]v := v + a$

Figure 1. Simplified SDK with error on speed adjustment. $\varphi_{ok} = D_S - 10 \leq D \leq D_S + 10$, $\varphi_{acc} = (D > D_S + 10) \wedge (V \leq V_{max})$, $\varphi_{slow} = (D < D_S - 10) \vee (V > V_{max})$.

2.2. Paths and runs of an IOSTS

A path of an IOSTS $(Q, q_0, Trans)$ is any map $p : \mathbb{N} \to Trans$ such that $src(p(0)) = q_0$ and for all $i \in \mathbb{N}$, $tgt(p(i)) = src(p(i + 1))$. Before defining a run of an IOSTS, let us first give some preliminary definitions. All are given with respect to an IOSTS $G = (Q, q_0, Trans)$ over (A, C).

$Act(M) = (C \times \{?, !\} \times M) \cup Internal$ is the set of *concrete actions*. An interpretation $\nu \in M^A$ of variables is called a *concrete state* of G. A *concrete transition* is a triple $(\nu, act_M, \nu') \in M^A \times Act(M) \times M^A$. A *concrete path* r is a sequence of concrete transitions $\mathbb{N} \to (M^A \times Act(M) \times M^A)$, such that for all i, if $r(i) = (\nu_i, a_i, \nu'_i)$ and $r(i + 1) = (\nu_{i+1}, a_{i+1}, \nu'_{i+1})$ then $\nu'_i = \nu_{i+1}$.

Now, we define a transition run, that is the interpretation of one transition, say tr, of G.

Definition 3 (Transition run) *Let $tr = q \xrightarrow{\varphi[act]\rho} q' \in Trans$. The set $Truns(tr) \subseteq M^A \times Act(M) \times M^A$ of* transition runs *of tr is such that $(\nu, act_M, \nu') \in Truns(tr)$ iff $\langle M, \nu\rangle \models \varphi$ and:*

- (i) *if $act = c!t$ (resp. $act \in Internal$) then $\nu' = \nu \circ \rho$ and $act_M = (c, !, \nu(t))$ (resp. $act_M = act$) or*
- (ii) *if act is of the form $c?y$ then there exists ν'' such that $\nu''(z) = \nu(z)$ for all $z \neq y$, $\nu' = \nu'' \circ \rho$ and $act_M = (c, ?, \nu''(y))$.*

For a transition run $r = (\nu, act_M, \nu')$, also denoted by $\nu \xrightarrow{act_M} \nu'$, $src(r)$, $act(r)$ and $tgt(r)$ denote respectively ν, act_M and ν'. A run is at last defined as:

Definition 4 (Runs of an IOSTS) *A concrete path r of an IOSTS G is a run of G if there exists a path p of G, such that $r(i) \in Truns(p(i))$ for all $i \in \mathbb{N}$. $Runs(G)$ is the set of runs of G. $Runs(G)$ is the semantics of G.*

We shall say that a concrete path r *fits* a sequence of concrete actions $s = act_0 \dots act_n \in act(M)^{\mathbb{N}}$ (denoted by $s \sqsubset r$) iff $\forall i \in \{0, \dots, n\}, act(r(i)) = act_i$. Then, $L(G) = \{s \in act(M)^{\mathbb{N}} | \exists r \in Runs(G), s \sqsubset r\}$.

Example 2 $[\nu_0 \xrightarrow{gps?70} \nu_1 \xrightarrow{spd!-5} \nu_2 \xrightarrow{\tau} \nu_3]$ *is the beginning of a run of the IOSTS of Fig. 1, with $D_S = 100$, $\nu_0(v) = 80$, $\nu_3(v) = 75$.*

In the general case, there is no restriction on the initial values of the state variables, and any interpretation M^A of the variable can a priori be the source of the first concrete transition of a run of G. In practice, however, knowing the initial values of the variables, or at least of some of them does not seems unrealistic. We thus define an *initialization domain of an IOSTS* as a (possible infinite) set of interpretations $D \subseteq M^A$. Then, a concrete path r

is a *run of G with respect to D* iff r is a run of G such that $src(r(0)) \in D$. $Runs_D(G)$ denotes the set of all runs of G wrt D. The associated language is $L_D(G) = \{s \in act(M)^{\mathbb{N}} | \exists r \in Runs_D(G), s \sqsubset r\}$.

3. Diagnosability of IOSTS

Our aim is to check the diagnosability of a component modeled with an IOSTS, that is, to determine if it is possible or not to build a local diagnoser that can detect faults in the system by observing its behavior. We suppose that the system modeled by the IOSTS is a component communicating with a non-descript environment. A local diagnoser, considered as an addition to the component, would observe its input and output on its observable communication channels as well as observable internal actions, and deliver a diagnostic regarding the state of the system, possibly leading to some automated repairs. Such a problem has been studied a lot for finite discrete-event models [3]–[5], but IOSTS add a symbolic dimension that forces some adaptation of those classical definitions of diagnosability. In [6], diagnosability is studied and defined for another compact symbolic representation: succinct transition graphs, which use boolean formulas to represent the effects of events on a set of state variables. This is close to IOSTS, and the analysis on space and time complexity provided in [7] give strong support to symbolic method. However, IOSTS offers a more procedural view which allows to easily translate problems and gives a natural way to model interacting components, which would make extension to distributed diagnosability problem easier.

Projection and trace. The diagnosability property of an IOSTS is defined according to the corresponding LTS. Intuitively, an IOSTS is said to be diagnosable if and only if the occurrence of any fault is detectable after a finite number of observations of the concrete unfolding of an LTS corresponding to a possible instantiation of the IOSTS. This unfolding consists in a trace of observable elements.

Thus we define a projection function P associating to each sequence $s = act_0 \ldots act_n \in L(G)$ (where $act_i \in Act(M)$ is a concrete action and $L(G) \subseteq Act(M)^*$) an observable trace $w = P(s)$ defined as follows:

- $P(\epsilon) = \epsilon$, where ϵ is the empty trace.
- $P(act) = \epsilon$ if $act \in \Sigma_f \cup \Sigma_u \cup Input(A, C_u) \cup Output(A, C_u)$.
- $P(act) = act$ if $act \in \Sigma_o$.
- if $act = c\ Mod\ v$, where $c \in C_o$, $Mod \in \{?, !\}$ and v is a concrete value, then $P(c\ Mod\ v) = c\ Mod\ v$.
- $P(\alpha W) = P(\alpha)P(W)$, where $\alpha \in Act(M)$ is an arbitrary action and W is an arbitrary trace.

The inverse projection provides for each observable trace w of an IOSTS, the set of traces whose projection is w. It is defined for any $w \in P(L(G))$ by $P^{-1}(w) = \{s \in L(G) | P(s) = w\}$. However, this inverse projection might sometimes be too general. Indeed, to one sequence of actions $s = act_1 \ldots act_n \in L(G)$ might correspond several concrete runs, depending on the initial interpretation ν_0. Indeed, one may have some clue about the initialization of the system. If $\nu_0 \in M^A$ is an interpretation of all state variables, we define the inverse projection of $w \in P(L(G))$ with respect to initialization ν_0 by $P^{-1}_{\nu_0}(w) = \{s \in L(G) | P(s) = w \wedge (\exists r \in Runs(G), s \sqsubset r \wedge src(r(0)) = \nu_0)\}$

Definition of diagnosability. We denote by $\Psi(\tau_f)$ the set of traces of $L(G)$ ending in a fault action $\tau_f \in \Sigma_f$: $\Psi(\tau_f) = \{s \in L(G) \mid s = s_0 \ldots s_{n-1}\tau_f\}$. The postlanguage of $L(G)$ after s is denoted by $L(G)/s = \{t \in Act(M)^* | st \in L(G)\}$. An IOSTS G will be said to be *traceable* if it generates a *live* language and there is no runs in $Runs(G)$ containing an infinite sequence of concrete transitions whose action is not observable. Then, following [3], we define *full diagnosability* of a system by:

Definition 5 *Let $G = (Q, q_0, Trans)$ be a traceable IOSTS over (A, C). L(G) is* fully diagnosable *iff:*

$$\forall \tau_f \in \Sigma_f, \exists n_i \in N, \forall s \in \Psi(\tau_f), \forall t \in L(G)/s$$
$$\|t\| \geq n \Rightarrow [w \in P^{-1}[P(st)] \Rightarrow \tau_f \in w]$$

This definition means that for any fault mode, it is possible to detect the occurrence of a fault of this kind after at most a bounded number of actions, by only knowing the observable trace since the beginning, regardless of the initial values of the state variables. Since knowing the initialization domains of a system does not seem unrealistic, we provide a less demanding definition of diagnosability with respect to an initialization domain D.

Definition 6 *Let $G = (Q, q_0, Trans)$ be a traceable IOSTS over (A, C), and $D \subseteq M^A$ an initialization domain. L(G) is* diagnosable for D *iff:*

$$\forall \tau_f \in \Sigma_f, \exists n_i \in \mathbb{N}, \forall s \in \Psi(\tau_f), \forall t \in L(G)/s$$
$$[(\|t\| \geq n \wedge st \in L_D(G)) \Rightarrow (w \in P^{-1}_D(P(st)) \Rightarrow \tau_f \in w)]$$

It means that a diagnoser can be built for diagnosing the system if it is initialized by any interpretation in D. Note that if $L(G)$ is diagnosable for D_1 and $L(G)$ is diagnosable for D_2, we do not necessarily have $L(G)$ diagnosable for $D_1 \cup D_2$. Indeed, if two diagnosers can be build for D_1 and D_2, one diagnosing a fault for an observable trace w_1 when the other does not,the system would not be diagnosable for $D_1 \cup D_2$ since w_1 leads to an ambiguity. On the other hand, if a system is diagnosable for the domain D, it is diagnosable for any subset of D. Full diagnosability is equivalent to diagnosability for M^A, and implies diagnosability for any domain.

4. Checking diagnosability

We propose here an adaptation of the twin-plant method [4], where a synchronized product is built in order to check diagnosability. Then, we use a symbolic model checker to check some property on this product equivalent to diagnosability of the initial graph. Translation of a diagnosability problem into a model checking one has also been done in other works such as [5] and [8], which also uses a twin plant before model checking. In this last reference, however, only zero-delay diagnosability is studied.

4.1. The IOSTS based twin plant

Let $G = (Q, q_0, Trans)$ be a traceable IOSTS over the signature $\Sigma = (A, C)$ and $Act(\Sigma)$ be the set of actions ($Act(\Sigma) = Input(\Sigma) \cup Output(\Sigma) \cup \{\tau, \tau_f\}$). For the sake of simplicity and without loss of generality, we will focus our study on the assumption that only one kind of fault (denoted hereafter by τ_f) can occur in the system. Indeed, to prove that a system is diagnosable for all faults, it suffices to prove that it is diagnosable for each kind of fault taken alone.

We shall define the IOSTS G_d, which results from the symbolic synchronization of two identical copies of G on their concrete observables, beginning by specifying its variables and actions.

4.1.1. State variables and actions of G_d. The state variables of G_d is given by $A_d = A^1 \cup A^2 \cup \{R, F^1, F^2, Amb\}$ which is a finite set of variables such that to each variable $a \in A$ correspond two variables in A_d: $a^1 \in A^1$ and $a^2 \in A^2$. R is a new state variable that will be used to receive and dispatch values to variables of A^1 and A^2. F^1 and F^2 are specific boolean state variables which indicate if a fault action τ_f has been executed before reaching a given state. Amb is a boolean variable indicating whether a valuation over A is ambiguous or not. Intuitively, a valuation over A is said to be ambiguous if it is formed of two valuations over A^1 and A^2 with different values of the instances F^1 and F^2 indicating the occurrences of fault actions : $Amb = F^1 \oplus F^2$. As for communications channels, G_d uses $C_d = C_o \cup C_u^1 \cup C_u^2$, $c^p \in C_u^p$ represents the unobservable channel c of the copy p.

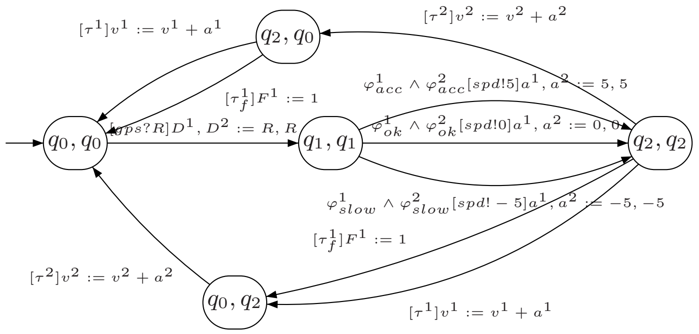
The *set of actions* in the synchronized IOSTS is $Act_d(A_d, C) = Input(A_d, C) \cup Output(A_d, C_d) \cup Internal_d$, where $Input(A_d, C_d) = \{c?R \mid c \in C_d\}$, $Output(A_d, C_d) = \{c!t \mid c \in C_d, t \in \mathcal{T}_{A^1} \cup \mathcal{T}_{A^2}\}$, $Internal_d = \Sigma_o \cup \{\tau^1\} \cup \{\tau^2\} \cup \{\tau_f^1\}$. An element $c?R$ of $Input(A_d, C_o)$ represents a synchronization of two receptions of a same value v through the communication channel c. This value is then assigned to either $y_i^1 \in A^1$ or $y_j^2 \in A^2$ in the substitution of variables, simulating the two synchronized actions $c?y_i^1$ and $c?y_j^2$ that both give rise to the same observation $c?v$. Likewise, an element $c!t$ of $Output(A_d, C_o)$ represents a synchronization of the emission of a term t^1 by the first copy and of another term t'^2 by the second copy through the same channel c, both terms having the same value v as ensured by $(t^1 = t'^2)$ in the guard. Both actions have the same trace $c!v$. The symbol τ^p (resp. τ_f^p) is used to characterize unobservable internal transitions (resp. fault transitions) of one of the two synchronized copies of the system (the first if $p = 1$ or the second if $p = 2$). Likewise $c^p?R$ or $c^p!t$ with $c \in C_u^p$ represents unobservable communications of one of the two copies. All these transitions have the same trace ϵ. At last, elements of Σ_o represents the synchronization of the same observable internal transition $o \in \Sigma_o$, made by each one of the two synchronized copies of the system, the trace of both transitions being o.

4.1.2. Synchronized product G_d. The synchronized IOSTS over Σ_d is the triple $G_d = (Q_d, (q_0, q_0), Trans_d)$ where $Q_d = Q \times Q$ is the set of *locations*, (q_0, q_0) is the *initial location* and $Trans_d \subseteq Q_d \times (\mathcal{F}_{A^1} \cup \mathcal{F}_{A^2}) \times Act_d(\Sigma_d) \times ((\mathcal{T}_{A^1})^{A^1} \cup (\mathcal{T}_{A^2})^{A^2}) \times Q_d$. Let (q_i, q_j) and (q'_i, q'_j) be two states of G_d. There will three kinds of transition in G_d:

- *Synchronized transitions* corresponding to two transitions $q_i \xrightarrow{\varphi_i^1[act_i]\rho_i^1} q'_i$ in the first copy and $q_j \xrightarrow{\varphi_j^2[act_j]\rho_j^2} q'_j$ in the second copy with the same observable trace. We have in G_d the transition $(q_i, q_j) \xrightarrow{\varphi_d[act_d]\rho_d} (q'_i, q'_j)$ where:
  - if $act_i = act_j \in \Sigma_o$, $act_d = act_i$, $\varphi_d = \varphi_T = \varphi_i^1 \wedge \varphi_j^2$ and $\rho_d = \rho_T$ defined as:
  $$\rho_T(x) = \begin{cases} \rho_i^1(x) & \text{if} \quad x \in A^1 \\ \rho_j^2(x) & \text{if} \quad x \in A^2 \\ x & \text{if} \quad x \in \{F^1, F^2\} \\ \rho_T(F^1) \oplus \rho_T(F^2) & \text{if} \quad x = Amb \end{cases}$$
  - if $act_i = c?x_i^1$ and $act_j = c?x_j^2$ with $c \in C_o$, then $act_d = c?R$, $\varphi_d = \varphi_T$ and $\rho_d = \rho_T \circ (x_i^1, x_j^2 := R, R)$ (φ_T, ρ_T defined above)
  - if $act_i = c!t_i^1$ and $act_j = c!t_j^2$ with $c \in C_o$, then $act_d = c!t_i^1$, $\varphi_d = \varphi_T \wedge (t_i^1 = t_j^2)$ and $\rho_d = \rho_T$ (φ_T, ρ_T defined above)
- *Non-synchronized transitions* of the first copy corresponding to a transition $q_i \xrightarrow{\varphi_i^1[act_i]\rho_i^1} q'_i$ in the first copy with an unobservable action. Then, if (q_i, q_j) is a state of G_d, we have in G_d the transition $(q_i, q_j) \xrightarrow{\varphi_i^1[act_U]\rho_U} (q'_i, q_j)$ where
  - if $act_i \in \Sigma_u \cup Output(A, C_u)$, then $act_U = \tau^1$ or $c^1!T_i^1$ and ρ_U is defined as $\rho_U = \rho_1 \circ \rho_{Amb}$ where $\rho_1(x) = \rho_i^1$ if $x \in A^1$, x otherwise, and $\rho_{Amb} = \rho_U(F^1) \oplus \rho_U(F^2)$ if $x = Amb$, x otherwise.
  - if $act_i = c?x_i^1$ with$c \in C_u$, then $act_U = c^1?R$ and ρ_U is defined as $\rho_U = \rho_1 \circ (x_i^1 := R) \circ \rho_{Amb}$ where ρ_1, ρ_{Amb} are defined above.

  - if $act_i = \tau_f$ then $act_U = \tau_f^1$ and ρ_U is defined as $\rho_U = \rho_1 \circ (F^1 := 1) \circ \rho_{Amb}$ where ρ_1, ρ_{Amb} are defined above.
- Non-synchronized transitions of the second copy are defined in a symmetric manner, though we can omit faulty transitions to benefit from the symmetry.

Example 3 *Figure 2 shows the synchronized product of two copies of the IOSTS depicted in figure 1. We did not represent Amb in the substitution of variables as it is always $F^1 \oplus F^2$.*


$[\tau^1]v^1 := v^1 + a^1$
q_2, q_0
$[\tau^2]v^2 := v^2 + a^2$
$\varphi^1_{acc} \wedge \varphi^2_{acc}[spd!5]a^1, a^2 := 5, 5$
$[\tau^1_f]F^1 := 1$
$[gps?R]D^1, D^2 := R, R$
q_0, q_0
q_1, q_1
$\varphi^1_{ok} \wedge \varphi^2_{ok}[spd!0]a^1, a^2 := 0, 0$
q_2, q_2
$\varphi^1_{slow} \wedge \varphi^2_{slow}[spd!-5]a^1, a^2 := -5, -5$
$[\tau^1_f]F^1 := 1$
$[\tau^2]v^2 := v^2 + a^2$
q_0, q_2
$[\tau^1]v^1 := v^1 + a^1$
Figure 2. Synchronized product.

In this example product, the path

$$p = (q_0, q_0) \xrightarrow{[gps?R]D^1, D^2 := R, R} (q_1, q_1) \xrightarrow{\varphi^1_{ok} \wedge \varphi^2_{ok}[spd!0]a^1, a^2 := 0, 0}$$
$$(q_2, q_2) \xrightarrow{[\tau^1_f]F^1 := 1} (q_0, q_2) \xrightarrow{[\tau^2]v^2 := v^2 + a^2} (q_0, q_0)$$

leading back to the state (q_0, q_0) corresponds to the observable behavior $[gps?d][spd!0]$. At the end of its execution, Amb equals 1 and keeps this value as long as it is possible to continue without any occurrence of a fault in the second copy. This path corresponds to the synchronisation, in the original IOSTS, of the two paths

$$p_1 = q_0 \xrightarrow{[gps?D]} q_1 \xrightarrow{\varphi_{ok}[spd!0]a := 0} q_2 \xrightarrow{[\tau_f]} q_0$$
$$p_2 = q_0 \xrightarrow{[gps?D]} q_1 \xrightarrow{\varphi_{ok}[spd!0]a := 0} q_2 \xrightarrow{[\tau]v := v + a} q_0$$

having the same observable trace $[gps?d][spd!0]$ but where the fault event τ_f occurs only in the path p_1 but not in p_2. If we prove (using a model checker) that the path p can be infinite (without introducing a fault in p_2), then we deduce that the system is not diagnosable.

5. Symbolic execution and model checking

As seen above, an IOSTS G is not diagnosable if there exists pairs of different infinite runs sharing the same observable events, one being affected by a considered fault and not the other. It is equivalent to say that there exists an unfolding of the twin plant where the Amb variable remains true forever after a given step. This latter formulation can be easily formalized into an LTL expression, which is: $FG(Amb)$ (where F states for Finally and G for Globally). Indeed this is how we check diagnosability of an IOSTS. We use an LTL model checker dedicated to IOSTS models which as been developed by the CEA LIST in France [9]. The whole technique is explained below.

5.1. Symbolic Execution

Symbolic execution, which has been first defined for programs [10], allows to explore executions of a program without enumerating all possible values of all variables. Symbolic execution produces a concise representation of executions like, in set theory, comprehensive definitions are concise for defining huge sets. The main idea of this technique is to use a new fresh symbol of variable instead of a value, each time that a reception (including initialization) occurs. The role of this new fresh variable is to represent any value of the input. This technique can naturally be adapted to the framework of IOSTS. In the sequel, the set of *symbolic inputs* is the countable set F of new fresh variables ($F \cap A = \emptyset$). Consequently state variables register terms and the guard of a transition specifies a condition on those terms for the transition to be executable. Along a path, the conjunction of those conditions is the necessary condition (over symbolic inputs) under which a symbolic state can be reached from the initial state: it is called a *path-condition*. Using these, a *symbolic state* is defined by:

Definition 7 (Symbolic state) *A* symbolic state over F *of G is a triple $\eta = (q, \pi, \sigma)$ where $q \in Q$, $\pi \in \mathcal{F}_F$ is called a* path-condition *and $\sigma \in \mathcal{T}_F^A$ is called a* symbolic assignment. *$\eta = (q, \pi, \sigma)$ is said to be consistent if π is satisfiable.*[1]

The following definition shows the construction of one step of a symbolic execution, that is the symbolic transition, or transition between symbolic states, associated with a transition of G. Note that if the communication action of the transition is an input message affecting a variable, then a new fresh symbol is introduced.

Definition 8 (Symbolic transition) *A* symbolic transition *over F is a triple (η, sa, η'), where η, η' are symbolic states, and sa is an action over (F, Ch). Let $tr = q \xrightarrow{\varphi[act]\rho} q'$ be a transition of G. Let $\eta = (q, \pi, \sigma)$ be a symbolic state over F of G. Let z be a variable in F such that z is used neither in π nor in $\sigma(v)$ for all variable $v \in A$ (z is a fresh variable not used in η). Then the* symbolic transition associated with tr and η *is (η, sa, η'), where η' and sa are defined by:*

if $act = c!t$, then $sa = c!\sigma(t)$ and $\eta' = (q', \pi \wedge \sigma(\varphi), \sigma \circ \rho)$,

if $act = c?x$ with x in A then $sa = c?z$, and $\eta' = (q', \pi \wedge \sigma(\varphi), \sigma \circ [z/x] \circ \rho)$,

1. Let us recall that here, π is *satisfiable* if and only if there exists $\mu \in M^F$ such that $\langle M, \mu \rangle \models \pi$ since variables of π are in F by construction.

if $act = \tau$ then $sa = \tau$, and $\eta' = (q', \pi \wedge \sigma(\varphi), \sigma \circ \rho)$.

A symbolic transition $sp = (\eta, sa, \eta')$ is denoted by $\eta \xrightarrow{sa} \eta'$; $source(sp) = \eta$ and $target(sp) = \eta'$.

Definition 9 (Symbolic path) *A symbolic path is a countable infinite sequence of symbolic transitions $[st_0, \ldots, st_n, \ldots]$ associated with an IOSTS such that for all $i \in \mathbb{N}$, $target(st_i) = source(st_{i+1})$. It is said* consistent *if all of its symbolic states are consistent.*

The following definition shows how a symbolic path can be interpreted as a concrete path of the IOSTS.

Definition 10 (Interpretation of a symbolic path) *Let $sp = [(q_0, \pi_0, \sigma_0) \xrightarrow{sa_0} \cdots (q_n, \pi_n, \sigma_n) \xrightarrow{sa_n} \cdots]$ be a symbolic path. If $\mu \in M^F$ is an interpretation of variables of F such that $\langle M, \mu \rangle \vDash \pi_i$ for all $i \in \mathbb{N}$, then the interpretation of sp by μ is $r = [\nu_0 \xrightarrow{act_0} \cdots \nu_n \xrightarrow{act_n} \cdots]$ such that for all $i \in \mathbb{N}$:*

- *$\nu_i(x) = \mu(\sigma_i(x))$*
- *$act_i = c@\mu(t)$ if $sa_i = c@t$ else $act_i = \tau$.*

Then, as we defined runs from concrete path, we want to define symbolic execution path as symbolic path whose interpretations will be runs of G.

Definition 11 (Symbolic execution path) *A symbolic path $[\eta_0 \xrightarrow{sa_0} \ldots \eta_{n-1} \xrightarrow{sa_{n-1}} \ldots]$ is called a* symbolic execution path *of G iff $\eta_0 = (q_0, \pi_0, \sigma_0)$, where q_0 is the initial location and σ_0 is an injective substitution in F^A, $\pi_0 = \bigwedge_{a \in T} \sigma_0(a) = f_{def}(a)$ is the initial path condition ensuring that every tool variable is initialized with its default value and each symbolic transition is associated with a transition of G.*

Symbolic execution is correct and complete i.e. the union of all interpretations of all symbolic paths of an IOSTS G is exactly $Runs(G)$.

Then, at last, a symbolic execution path with respect to an initialization domain D would be symbolic path whose interpretations are runs of G wrt D. An interpretation of a symbolic execution path sp by μ would be a run of G with respect to an initialization domain D if $\mu o \sigma_0 \in D$. Since μ is such that $\langle M, \mu \rangle \vDash \pi_i$ for all $i \in \mathbb{N}$, we can integrate this initialization condition in the symbolic path itself by defining π_0 as $\pi_D = \bigvee_{\nu_0 \in D}(\bigwedge_{x \in A}(\sigma_0(x) = \nu_0(x)))$, that is $\exists \nu_0 \in D, \forall x \in A, \sigma_0(x) = \nu_0(x)$. Thus a *symbolic execution path with respect to $D \subseteq M^A$* is a symbolic execution path whose first symbolic state η_0 has a path-condition π_D instead of the path-condition $\bigwedge_{a \in T} \sigma_0(a) = f_{def}(a)$.

5.2. Unfolding rules

Our model-checking algorithm, as usual, unfold the model in order to prove that there exists at least a run satisfying the negation ($\neg g$) of the expected property g to be checked. In our case, we have $\neg g = FG(Amb)$. We describe here unfolding rules, used to compute symbolic paths of the IOSTS being such that any of their numerical interpretation satisfies the formula. Those rules are inspired from tableau unfolding of LTL formulas whose principle consists in the decomposition of a formula into (i) atomic formulas to be verified in the current state and (ii) formulas to be verified in the next state. They are linked to symbolic execution by the fact that a rule transforms a *context* which is a couple whose first component is a symbolic state η and whose second component is a tuple of three sets of LTL formulas (Φ, Γ, Υ). The set Φ, called *Finally Set*, contains finally formulas of the form $\mathbf{F}\psi$, such that ψ has to be checked in the future; Γ, called *Current Set*, is the set of formulas to be checked in the current state; Υ, called *Next Set* is the set of formulas to be checked in the next state.

Let G be an IOSTS whose initial location is q_0 and whose set of state variables is A. The *initial symbolic state* is $init = (q_0, \pi_0, \sigma_0)$ where σ_0 is an injective substitution in F^A (where F is the set of fresh variables, see Section 5.1) and π_0 is either $\bigwedge_{a \in T} \sigma_0(a) = f_{def}(a)$ if checking full diagnosability, or $\pi_{Twin(D)}$ if checking diagnosability with respect to initialization domain D. If f is the temporal formula given as an input to our set of rules, the first context is $[init, (\Phi_{init}, \{f\}, \emptyset)]$ where Φ_{init} is the set containing all finally sub-formulas of f (*i.e* of the form $\mathbf{F}\psi$).

5.2.1. Rules related to Current Set.

First, the rules related to Current Set Γ are applied until $\Gamma = \emptyset$. A fraction style rule denotes a substitution: the upper context vanishes and is replaced by the lower context of the fraction bar. If the rule $[Finally_1]$ can be applied, it is also the case of the rule $[Finally_2]$: then the algorithm forks in two avatars; on each avatar is applied one of the two rules.

$[Atom]$ $p \in \mathcal{F}_A$

$$\frac{(q, \pi, \sigma), (\Phi, \{p\} \cup \Gamma, \Upsilon)}{(q, \pi \wedge \sigma(p), \sigma), (\Phi, \Gamma, \Upsilon)}$$

$[Finally_1]$

$$\frac{\eta, (\Phi, \{\mathbf{F}g\} \cup \Gamma, \Upsilon)}{\eta, (\Phi, \{g\} \cup \Gamma, \Upsilon)}$$

$[Finally_2]$

$$\frac{\eta, (\Phi, \{\mathbf{F}g\} \cup \Gamma, \Upsilon)}{\eta, (\Phi, \{f\} \cup \Gamma, \Upsilon \cup \{\mathbf{F}g\})}$$

$[Globally]$

$$\frac{\eta, (\Phi, \{\mathbf{G}g\} \cup \Gamma, \Theta, \Upsilon)}{\eta, (\Phi, \{g\} \cup \Gamma, \Theta, \Upsilon \cup \{\mathbf{G}g\})}$$

By applying these rules, the formulas in Current set are decomposed in atomic formulas and next formulas of the form Xg that are put in Next set Υ. When an atomic formula in $\mathcal{F}_A$ is reached, it is added to the path condition after substitution of variables by the terms defined by the symbolic assignment (rule $[Atom]$): its consistency will be checked when the Transition rule will be applied. Rules

$[Finally]$ are explained by the equivalence between $\mathbf{F}g$ and $g \vee (\mathbf{X}(\mathbf{F}g))$. Rule $[Globally]$ is explained by the equivalence between $\mathbf{G}g$ and $g \wedge \mathbf{X}(\mathbf{G}g)$.

5.2.2. Transition rules. Once Γ is empty, the next step is to construct symbolic transitions using the transition rules. Let $\eta \xrightarrow{sa} \eta'$ be a symbolic transition associated with G (see definition 8), such that η and η' are consistent. Then the transition rule is the following:

- If the last context of the path is $[\eta, (\Phi, \emptyset, \Upsilon)]$ and $\Phi \neq \emptyset$ then construct
$$\eta, (\Phi, \emptyset, \Upsilon) \xrightarrow{sa} \eta', (\Phi \cap \Upsilon, \Upsilon, \emptyset)$$
- When Finally set is empty (*i.e* all finally formulas have been checked), the new Finally set is Φ_{init}, the set of all finally sub-formulas. So the rule is:
$$\eta, (\emptyset, \emptyset, \Upsilon) \xrightarrow{sa} \eta', (\Phi_{Init} \cap \Upsilon, \Upsilon, \emptyset)$$

$[(\eta, C) \xrightarrow{sa} (\eta', C')]$ is a symbolic transition extended to contexts. The formulas in Next Set Υ have to be checked in the new state, so they are put in Current Set Γ. Moreover the new Finally set is the intersection of the old Finally (or Φ_{init} if it was empty) set with this new current set: so Finally set will contain Finally formulas that still have to be checked. After Transition rule has been applied, the context at the source of the transition will remain unchanged by the following rules. Then, the other rules (related to Current Set) can be applied in the new state.

5.2.3. f-unfoldings. An *f-unfolding of an IOSTS G* is a finite or infinite sequence of transitions $[(\eta_0, C_0) \xrightarrow{sa_0} (\eta_1, C_1) \xrightarrow{sa_1} (\eta_2, C_2) \ldots]$ resulting from the application of rules defined above (*i.e* for all i, $(\eta_i, C_i) \xrightarrow{sa_i} (\eta_{i+1}, C_{i+1})$ can be obtained by unfolding rules), starting at $[init, (\Phi_{init}, \{f\}, \emptyset, \emptyset)]$. A *matching f-unfolding* of an IOSTS G is then an infinite f-unfolding such that the Finally set is empty infinitely often. The *symbolic projection* of an unfolding is the sequence $[\eta_0 \xrightarrow{sa_0} \eta_1 \xrightarrow{sa_1} \eta_2 \ldots]$ obtained by ignoring the second coordinate of contexts. Rule unfolding is *correct* and *complete*: the union of all interpretations of symbolic projections of all matching f-unfoldings of G is exactly the set of all runs of G satisfying f.

5.3. Termination criteria

There exist an ambiguous run of G (satisfying $f = FG(Amb)$) iff there exists a matching f-unfolding of G. Since such unfolding are infinite, we need some criterion to stop the unfolding when we know that current finite unfolding can be used to build one, or that it cannot evolve into one. These criteria will be based on the notion of *omega sets*.

5.3.1. Omega sets. Intuitively the omega set of a symbolic state η relatively to $\delta \subseteq F$, denoted by Ω^{δ}_{η}, characterizes the relation existing between possible concrete assignations of the system variables A and interpretations of a given set δ of the symbolic inputs they depend on. For example with $M = (\mathbb{Z}, +, <)$, $A = \{x\}$ and $\eta = (q, a > 0, x \mapsto a + 1)$ the omega set of η relatively to $\{a\}$, noted $\Omega^{\{a\}}_{\eta}$, is $\{(x \mapsto 2, a \mapsto 1), (x \mapsto 3, a \mapsto 2), (x \mapsto 4, a \mapsto 3) \ldots\}$. The couple $(x \mapsto 2, a \mapsto 1)$ expresses the fact that x is assigned by 2 when the interpretation of a is 1.

In the following definition $SI(\pi)$ denotes the set (included in F) of symbols of variables having at least an occurrence in the path-condition π. Moreover $SI(Ran(\sigma))$, where σ is a symbolic assignment, denotes the union of the sets of symbols of variables having at least an occurrence in a term $\sigma(x)$ for $x \in A$.

Definition 12 (Omega set of a symbolic state) *Let $\eta = (q, \pi, \sigma)$ be a symbolic state over F. Let us note δ a finite subset of symbols of variables of F and $\gamma = [SI(\pi) \cup SI(Ran(\sigma))] \setminus \delta$. The omega set of η relatively to δ, noted Ω^{δ}_{η} is $\{(\nu, \mu) \in M^A \times M^{\delta} / \exists \beta \in M^{\gamma}, \langle M, \nu, \mu, \beta \rangle \models (\pi \wedge \bigwedge_{x \in A} (x = \sigma(x)))\}$*

An omega set of $\eta = (q, \pi, \sigma)$ is thus a couple of interpretation ν, μ of A and δ such that there exists an interpretation β of the other relevant variables of F ensuring that η is consistent (that is π is satisfied) and that variables of F and A are indeed linked by the assignment σ.

5.3.2. Theorems. Let $ct = [\eta, C]$ and $ct' = [\eta', C']$ be two contexts associated to the same IOSTS, such that $\eta = (q, \pi, \sigma)$, $\eta' = (q', \pi', \sigma')$, $C = (\Phi, \emptyset, \Upsilon)$ and $C' = (\Phi', \emptyset, \Upsilon')$.

- ct' is said to be *strongly related to* ct if $C = C'$, $q = q'$ and $\Omega^{SI(Ran(\sigma))}_{\eta} \cap \Omega^{SI(Ran(\sigma))}_{\eta'} \neq \emptyset$.
- ct' is said to be *included in* ct if $C = C'$, $q = q'$ and $\Omega^{\emptyset}_{\eta'} \subseteq \Omega^{\emptyset}_{\eta}$.

Let us now consider an IOSTS G and a f-unfolding $P = [ct_0 \xrightarrow{sa_0} ct_1 \xrightarrow{sa_1} \cdots ct_n]$ of G.

1) If for some $i \in \{0, \ldots, n-1\}$, ct_n is strongly related to ct_i, then P is said to verify the *lasso criterion*.
2) If for some $i \in \{0, \ldots, n-1\}$, ct_n is included in ct_i, then P is said to verify the *inclusion criterion*.
3) If P satisfies (1) or (2) and is such that there is an empty Finally set among $\Phi_i, \ldots, \Phi_n$ of the contexts $ct_i, \ldots, ct_n$, then P is said to verify the*Finally criterion*.

Theorem 1 *If a f-unfolding of G verifies the lasso criterion and the Finally criterion, then there is a run in the semantics of G verifying f.*

Theorem 2 *If all f-unfoldings of G are finite or verify the inclusion criterion but not the finally criterion, then there is no run in $Runs(G)$ verifying f.*

These two theorems give sufficient conditions for respectively non-diagnosability and diagnosability (with $f = FG(Amb)$).

6. Experimental results and implementation issues

The diagnosability checking technique described in the previous sections has been implemented as an extension of the AGATHA tool [11]–[13] which provides a symbolic execution engine for the IOSTS formalism and supports Presburger arithmetics for the data part (thanks to the Omega Library [14]). In practice the twin-plant is not computed in advance but on-the-fly by AGATHA which provides facilities to deal with systems described by several components and some interaction rules. This aspect contributes to tackle some space complexity difficulties (in some favorable cases non-diagnosability is proved while not all transitions of the twin-plant have been built so far). This work was supported by the french projet DIAFORE [15]. The first experiment of the technique were conducted during the DIAFORE project on a full industrial version of the SDK which was provided by Renault Trucks (the industrial partner of this project). This full version combines a classical speed regulation function with an anti-collision function which adjust the distant and the speed relatively to a vehicle running front. A significant aspect of the whole technique is that path conditions associated with witnesses bring some good intuitions to understand and explain the highlighted cases of non-diagnosability. This is very usefull to decide whether or not non-diagnosibility should be solved and if so, how to solve it. This was particularly appreciated by the end user, Renault Trucks.

7. Conclusion

We presented here IOSTS, a symbolic way to represent a system's behavior, and defined full diagnosability and diagnosability over an initialization domain for such a representation. An adaptation of the twin plant method for checking diagnosability, combined with a LTL model-checking, was also detailed. Future works include studying more complex industrial case to propose a full methodology, and generalizing this framework for distributed systems.

References

[1] W. Qiu and R. Kumar, "Decentralized failure diagnosis of discrete-event systems," *IEEE Trans. on Systems, Man and Cybernetics - Part A*, vol. 36, no. 2, pp. 384–395, 2006.

[2] A. Schumann and Y. Pencolé, "Scalable diagnosability checking of event-driven systems." in *Proc. of IJCAI07*, 2007, pp. 575–580. [Online]. Available: http://dblp.uni-trier.de/db/conf/ijcai/ijcai2007.html/SchumannP07

[3] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Trans. on Aut. Contr.*, vol. 40, no. 9, pp. 1555–1575, 1995. [Online]. Available: http://dx.doi.org/10.1109/9.412626

[4] S. Jiang, Z. Huang, V. Chandra, and R. Kumar, "A polynomial algorithm for testing diagnosability of discrete event systems," *IEEE Trans. on Aut. Contr.*, vol. 46, pp. 1318–1321, 2001.

[5] S. Jiang and R. Kumar, "Failure diagnosis of discrete event systems with linear-time temporal logic fault specifications," *IEEE Trans. on Aut. Contr.*, vol. 49, no. 8, pp. 934–945, 2004.

[6] J. Rintanen and A. Grastien, "Diagnosability testing with satisfiability algorithms," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, M. Veloso, Ed. AAAI Press, 2007.

[7] J. Rintanen, "Diagnosers and diagnosability of succinct transition systems," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*. AAAI Press, 2007.

[8] A. Cimatti, C. Pecheur, and R. Cavada, "Formal verification of diagnosability via symbolic model checking," in *Proc. of IJCAI03)*, 2003, pp. 363–369.

[9] N. Rapin, "Symbolic execution based model checking of open systems with unbounded variables," in *TAP, International Conference on Tests And Proofs*, Zurich, Switzerland, 2009.

[10] J. C. King, "A new approach to program testing," in *Proc. of the international conference on Reliable software*. New York, NY, USA: ACM Press, 1975, pp. 228–233.

[11] N. Rapin, C. Gaston, A. Lapitre, and J.-P. Gallois, "Behavioural unfolding of formal specifications based on communicating automata," in *Proceedings of the first Workshop on Automated technology for verification and analysis*, Taiwan, 2003.

[12] C. Gaston, P. LeGall, N. Rapin, and A. Touil, "Symbolic execution techniques for test purpose definition," in *TestCom*, New York, USA, 2006.

[13] P. LeGall, N. Rapin, and A. Touil, "Symbolic execution techniques for refinement testing," in *TAP, International Conference on Tests And Proofs*, 2007, pp. 131–148.

[14] W. Kelly, V. Maslov, W. Pugh, E. Rosser, T. Shpeisman, and D. Wonnacott, "The omega library interface guide," College Park, MD, USA, Tech. Rep., 1995.

[15] "DIAFORE project (DIAgnostic de FOnctions REparties), SYSTEM@TIC, ANR."