Preliminaries On Complexity of Diagnosis of Discrete-Event Systems

Gianfranco Lamperti and **Marina Zanella**
Department of Information Engineering, University of Brescia, Italy
email: {gianfranco.lamperti,marina.zanella}@ing.unibs.it

Abstract

This paper is an attempt to move a small step toward a complexity analysis of diagnosis of discrete-event systems (DESs). The considered conceptual model of the behavior of DESs are automata. If the DES is distributed, its behavior is implicitly described by the automaton resulting from the synchronous composition of the component automata. Solving a diagnosis problem inherent to a DES amounts to performing a search within the relevant automaton so as to find a behavioral path that starts from a given state and generates a given observation. Three dimensions are taken into account: temporal and logical uncertainty of the observation, and uncertainty about the initial state. The main outcome of this preliminary analysis is that checking whether there exists any solution to a given diagnosis problem is in PSPACE. If such a check is inherent to a non-distributed DES, it is in NP when the observation is both temporally and logically certain, while, if the DES is distributed, it is NP-hard, whichever the observation.

1 Introduction

A general definition of a DES [Cassandras and Lafortune, 2008] reads 'a discrete-state, event-driven system, that is, its state evolution depends entirely on the occurrence of asynchronous discrete events over time'. At an untimed abstraction level, a DES is described by a language. An automaton [Hopcroft *et al.*, 2006] is the most intuitive model to represent a language. This is the reason why automata were adopted as modeling primitives since the very beginning of research on Model-Based Diagnosis (MBD) of DESs [Sampath *et al.*, 1995] in the middle '90s, thus leading to what in [Grastien *et al.*, 2007] are called the 'classical' approaches.

So many years after, in spite of meaningful results on the complexity of diagnosability of DESs represented as untimed automata [Jiang *et al.*, 2001; Rintanen, 2007], a systematic analysis of the complexity of diagnosis is still missing. A DES diagnosis problem consists of a DES (automaton), the initial state of the DES and an observation inherent to the DES (assuming that some state changes of the DES are observable). Solving it means finding all the paths in the given automaton that may have produced the given observation. In [Grastien *et al.*, 2007] a diagnosis problem inherent to a DES, given an initial state that is completely certain, is first presented as a path finding problem, and then formulated as a SAT problem. This looks like a reduction of the diagnosis problem to a known NP-complete problem. Such a reduction, however, does not allow us to draw any conclusion on the complexity of diagnosis of DESs, neither when the temporal order of the observed events is certain or uncertain nor when the values of observed events are uncertain (a case which is not encompassed in [Grastien *et al.*, 2007]). Moreover, the same paper [Grastien *et al.*, 2007] states that 'diagnosis by SAT' is 'a problem harder in general than' the problem dealt with by 'the classical algorithms', which raises further questions about the complexity of diagnosis of DESs. Finding the paths (even just one of them) consistent with the given observation is at least as hard as deciding whether any such a path exists. In other words, the complexity of the diagnosis existence problem is a lower bound of the complexity of the diagnosis problem.

In this paper, before proving some theorems as to the complexity of the DES diagnosis existence problem (Section 3), the (automata based) conceptual model adopted in the analysis is described (Section 2). Conclusions are drawn in Section 4.

2 Conceptual model

A MBD *problem instance* inherent to a dynamic system modeled as a DES is a triple $(\Sigma, \Sigma_0, \mathcal{O})$, where Σ is the *DES model*, Σ_0 is the *initial state* of Σ, representing the state of the considered system at the start of a (finite) time interval of interest, and $\mathcal{O}$ is the (finite) *observation* of the behavior of the system over such a time interval.

2.1 Discrete-event system

The behavior of a DES is represented as a finite automaton (FA) Σ. In a conceptual model aimed at complexity analysis, we are interested only in the set S of *states* and the set T of state *transitions* of such an FA, where the unique identifier of each transition is accompanied by the source and target states of the transition itself. Let $\gamma : T \rightarrow E_{obs}$ be a partial *observation function*, where E_{obs} is a finite set (alphabet) of *observable events*; if γ is defined for $t \in T$, then t is *observable*. Fig. 1 displays the model of a DES consisting of 11 states (numbered from 0 to 10) and 21 transitions (whose identifiers range from $t1$ to $t21$), 10 of which are observable (each observable transition is labeled with the relevant observable event).

The initial state Σ_0 is a subset of S, that is, $\Sigma_0 \subseteq S$, which means that Σ, at the beginning of the time interval of interest, may be in any state in Σ_0 [Sohrabi *et al.*, 2010]. If Σ_0 is a singleton, then the initial state of Σ is *certain*, otherwise it is *uncertain*.

A DES is *distributed* if it consists of several interacting *components*, each of which is a DES itself. If the identifier of a transition belongs to the T set of several components, then such a transition occurs only if and when it can occur simultaneously in all the components that share it, that is, it is an outgoing transition of all the current states of all the components that share it. Therefore, the FA relevant

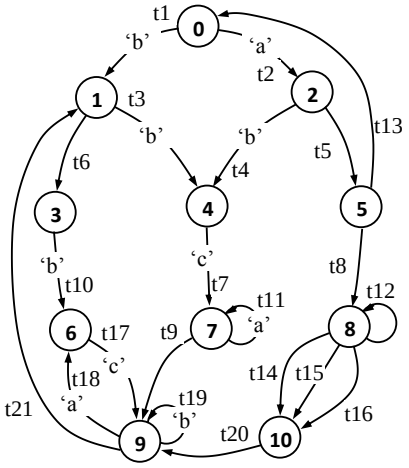

Figure 1: DES model.

to the whole system is the one (implicitly[1]) resulting from the parallel composition (often called synchronous composition [Cassandras and Lafortune, 2008]) of all the component FAs, where such a synchronization is based on shared transitions, which are in fact called *synchronous* transitions.

There are approaches in the literature [Lamperti and Zanella, 2003] where components interact 'asynchronously', by exchanging (communication) events over *links*. Including links in a DES model is a matter of expressive power, not of computational power. A link, whichever its capacity and its policy (FIFO, LIFO, etc.), could be represented as a component itself, which shares transitions with both the component(s) that feed(s) the link and the component(s) that extract(s) events from the link. Fig. 2 shows the component model of a link whose capacity is 2, whose policy is LIFO, where the events sent on the link are of two distinct types (say r and s). Each downward arrow represents a transition (shared with a component that feeds the link) that pushes an event on the stack of events in the link, while each upward arrow represents a transition (shared with a component that is fed by the link) that extracts the event on the top of the stack. State *empty* represents the case when the link is empty, while each state whose identifier begins with 1 or 2 represents the case when the link contains one or two events, respectively. The string of characters following the digit recalls the content of the stack.

Specific primitives for automata communication were introduced for several reasons: first of all, the link primitive is generic (it specifies just the capacity and the policy), that is, independent of the number and types of exchanged events, while the component model of a link depends on them; second, the size of the component model of a link includes a number of states that equals $\sum_{i=0}^{n} d^i = (d^{n+1}-1)/(d-1)$, where $d > 1$ is the cardinality of the set of communication events and n is the capacity of the link; third, the link primitive is processed more efficiently (by ad hoc algorithms) than the component model of a link. However, the class of DESs described by using links is a subset of the class of DESs described as interacting synchronously since an asynchronous communication can be modeled through the primitive for synchronous communication.

In order to base the complexity analysis on the most general class of systems, in the following we will assume to deal with DESs conceptually modeled as synchronous systems.

[1]The FA relevant to a distributed DES is implicit since no state-of-the-art approach to MBD of DESs generates it explicitly.

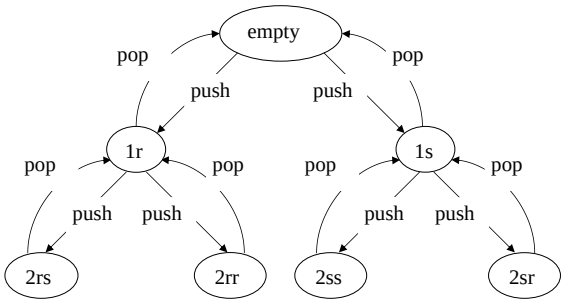

Figure 2: Component model of a link.

2.2 Observation

The observation $\mathcal{O}$ is what has externally been perceived about the system behavior over the time interval of interest. Typically the dynamic system has undergone an evolution which corresponds to a sequence of state transitions of the DES model, starting from a state in Σ_0. Such an evolution has generated a (possibly empty) sequence of observable events belonging to E_{obs}. However, in the observation the temporal order of the emission of such events may be disrupted into a partial order or even a totally unknown order (*temporal uncertainty*). This uncertain order is compatible with several sequences of observable events, among which there is the one actually emitted by the system. Moreover, also the observable events that have taken place in the DES may have been distorted, that is, a received value may range over a finite set of possible values, among which there is the right one (*logical uncertainty*). Thus, the emitted sequence of observable events may fall into a set of sequences, denoted $||\mathcal{O}||$. Set $||\mathcal{O}||$ is finite since *(i)* the number of events in $\mathcal{O}$, denoted $|\mathcal{O}|$, is finite; *(ii)* each event ranges over a finite set of values, whose cardinality has $|E_{obs}|$ as an upper bound; and *(iii)* the number of total orders compatible with the given partial order is finite, an upper bound being $|\mathcal{O}|!$.

The model of the observation proposed in [Lamperti and Zanella, 2002] is a directed acyclic graph, where each node represents an event perceived by the observer and each arc represents a temporal precedence relationship according to the emission order. The relative emission order of any pair of nodes such that one is not reachable from the other, as the shaded ones in the observation in the center of Fig. 3, is unknown. In case the observation is logically certain, every node contains a single event; if, instead, the observation is logically uncertain, then one node at least contains several events, just one of which was actually emitted by the system. In case the observed event represented by a node of the observation graph may be just pure noise, then the node is bound to include several observable events one of which is the null event (ϵ). All the three observations in Fig. 3 are logically uncertain, and they all include a node containing the null event. Considering the observation on the left of the figure, set $||\mathcal{O}||$ includes 32 sequences, among which the following: $\langle a, b, a, a, b, c, a\rangle$, $\langle b, b, a, a, b, c, a\rangle$, $\langle a, b, c, a, b, c, a\rangle$, $\langle a, b, c, c, c, a\rangle$, $\langle b, b, c, c, c, a\rangle$.

2.3 Diagnostic output

The solution of a problem instance $(\Sigma, \Sigma_0, \mathcal{O})$ is a (possibly empty) set of *candidate diagnoses*, each of which explains what has been observed. The most concrete notion of a candidate diagnosis is that providing the information for following a path in the FA representation of the behavior of the

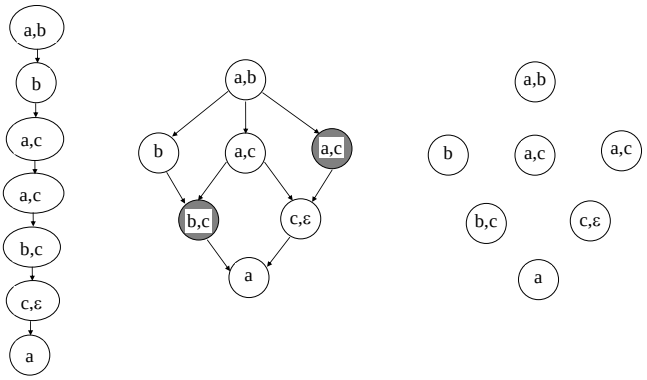

Figure 3: A totally temporally ordered observation (left), a partially temporally ordered observation (center), and an observation with unknown temporal order (right).

whole system, where such a path starts from a state in Σ_0 and generates a sequence of observable events that belongs to $||\mathcal{O}||$. This means that there exists a sequence of all the nodes in $\mathcal{O}$, where the total order of such a sequence is compliant with the (possibly partial or unknown) temporal order represented in the observation graph, such that, by picking up an event from each node of the sequence of nodes, the obtained sequence of observable events, once all the occurrences of the null event have been removed, equals the sequence of observable events generated by the path. For instance, path $\wp = \langle t2, t4, t7, t11, t9, t19, t18\rangle$ in the DES model of Fig. 1 generates the sequence of observable events $\langle a, b, c, a, b, a\rangle$: such a sequence is compliant with all the three observations in Fig. 3. A candidate diagnosis can be represented as a finite sequence of (component) transitions (where each transition uniquely singles out both the source and target component state(s) of the transition itself). A path starting from a state in Σ_0 is usually called a *history*. Thus, if state 0 belongs to Σ_0, $\wp$ is a history.

If the system model includes unobservable cycles, a set of candidate diagnoses that produce the same sequence of observable events and differ just for the number of times each unobservable cycle is followed, can be expressed at a higher abstraction level as a *route*. Syntactically, a route can be represented as a sequence of transitions (starting from a state $\sigma_0 \in \Sigma_0$) that possibly includes pairs of (possibly nested) parentheses, where each pair contains a sequence of transitions identifying an unobservable cycle.

For instance, $r = \langle t2, t5,\ t8, (t12),\ t15, t20, t19,\ t18,\ t17, t19, t18\rangle$ is a route of the DES model in Fig. 1: the pair of parentheses identifies the unobservable cycle consisting just of transition $t12$. Route r cumulatively represents an unbound number of histories, which differ from each other for the (possibly null) number of times transition $t12$ is performed, each of which, however, generates the sequence of observable events $\langle a, b, a, c, b, a\rangle$, that belongs to $||\mathcal{O}||$ for all the three observations in Fig. 3.

Some approaches [Pencolé and Cordier, 2005; Grastien *et al.*, 2007] represent (and compute) a candidate diagnosis as a sequence of sets of non interfering transitions, where two distinct transitions belonging to two distinct components do not interfere if they may occur concurrently. A set of non interfering transitions, called *trace* in [Pencolé and Cordier, 2005], is a concise representation of a 'diamond' in the system model, i.e. a portion of the FA that starts in a state, ends in another, and such that each permutation of the transitions in the set is a path from the former to the latter. This is another concise way to represent a set of histories. Actually, the advantages of both routes and traces could be combined in order to represent a till larger set of histories. However, the concepts of route and trace (and their combination) do not alter the fact that a diagnosis problem has a solution if and only if there exists a history of Σ that generates a sequence of observable events belonging to $||\mathcal{O}||$. Given a (finite, possibly empty) observation $\mathcal{O}$, a history that is consistent with it may be infinite, owing to unobservable cycles, therefore no diagnostic algorithm can produce it. The route corresponding to an infinite history, instead, is finite, therefore we will perform a decidability analysis by appealing to the notion of a route as a diagnostic output.

3 Decidability and complexity

Let D be the set of all DES diagnosis problem instances, as described in Section 2, each of which may be affected by three orthogonal forms of uncertainty: (i) temporal uncertainty in the observation, (ii) logical uncertainty in the observation, and (iii) uncertainty about the initial state. Domain D can be partitioned into D_{nd} and D_d, these including all the problem instances inherent to non-distributed and distributed DESs, respectively. Let DIAGNOSIS-EXISTENCE(D) be the set (i.e. the language) of all problem instances $P \in D$ such that $P = (\Sigma, \Sigma_0, \mathcal{O})$ is solvable, that is, there exists a route of Σ, starting from a state in Σ_0, that generates a sequence o of observable events s.t. $o \in ||\mathcal{O}||$. Without loss of generality, we assume that the last transition in a route that solves the problem is observable.

Theorem 1. *DIAGNOSIS-EXISTENCE(D) is decidable.*

Proof. (*Sketch*) The model of Σ consists of either one or several FAs, one for each component. Whichever the chosen initial states of all components, the synchronous composition of all such FAs is an FA, which includes a finite number of unobservable cycles (if any). For whichever observation, the number of nodes is finite, and each node contains a finite set of observable events. Thus, set $||\mathcal{O}||$, which contains completely certain (finite) sequences of observable events, is finite. So it is possible, first of all, to generate all the sequences in $||\mathcal{O}||$. Then, for each state $\sigma_0 \in \Sigma_0$, the synchronous composition of all the FAs starting from σ_0 can be carried out, thus obtaining an FA, say A_{σ_0}. Finally, for each sequence $o \in ||\mathcal{O}||$, a brute force search within A_{σ_0} can be performed to find out whether a route starting from σ_0 and generating *o* exists. Such a search requires to detect possible unobservable cycles occurring along a path, so as not to visit any of them an unbound number of times: since the number and length of such cycles is finite, the brute force algorithm can detect them. As soon as a route that starts from σ_0 and generates o is found, 'yes' is returned. If no such route is found, 'no' is returned. □

Let us call *minimal history* a (necessarily finite) history drawn from a route by removing all unobservable cycles.

Theorem 2. *A DES diagnosis problem $P = (\Sigma, \Sigma_0, \mathcal{O})$, $P \in D$, is solvable iff there exists a minimal history that solves it.*

Proof. By definition, P is solvable iff there exists a route that solves it. If such a route does not include any unobservable cycles, then it is a minimal history. If the route includes some unobservable cycles, then a minimal history can be drawn from it. Such a history solves the problem as well since all its transitions are executable in the given order and produce the same observation as the route. □

Theorem 3. *Every minimal history h is such that, for each pair (o_i, o_{i+1}) of consecutive observable events produced by h, the subsequence of unobservable transitions of h between the two observable transitions that produce o_i and o_{i+1}, respectively, cannot include more than $L_{max} - 1$ transitions, where L_{max} is the number of states of Σ.*

Proof. Let us assume that h produces a sequence o of observable events. Each state reached by Σ according to h, here called *history-state*, is univocally identified by a pair consisting in the state of Σ and an *observation index* $i \in [0, \ldots, |o|]$ that specifies the (sub)sequence of the observable events in o that have already taken place. Thus, value 0 of index i denotes that no observable events have taken place, while value $|o|$ denotes that all observable events in o have taken place. Given a history-state characterized by value i of the observation index, a history-state characterized by value $i + 1$ is reachable from it iff one observable transition is performed, where such a transition produces observable event o_{i+1}. However, before reaching the new history-state, several unobservable transitions may be performed, these leading from the current state to other history-states characterized by the same value i of the observation index. The maximum number of history-states characterized by value i and distinct from the current one[2] is $L_{max} - 1$. □

Corollary 1. *The length of a minimal history solving a diagnosis problem $P = (\Sigma, \Sigma_0, \mathcal{O})$, $P \in D$, is $O(|\mathcal{O}|L_{max})$, where $|\mathcal{O}|$ is the number of nodes in $\mathcal{O}$ and L_{max} is the number of states of Σ.*

Proof. Since the considered history solves the problem, it produces a sequence $o \in ||\mathcal{O}||$, whose maximum length is $|\mathcal{O}|$. As stated by Theorem 3, in a minimal history the number of unobservable transitions in between two observable ones cannot exceed $L_{max} - 1$. Therefore the length of a minimal history solving the problem is $O(|\mathcal{O}|L_{max})$. □

Theorem 4. *DIAGNOSIS-EXISTENCE(D_{nd}) is in NP if the observation is both logically and temporally certain.*

Proof. Given $P \in D_{nd}$, let the certificate be a minimal history h. Based on Corollary 1, $|h|$ is polynomial in the size of P. The pseudo-code of a verification algorithm which is polynomial in time is here below.

```
1.  function Verify(P, h)
2.    where P = (Σ, Σ0, O): a problem instance in D_nd,
3.    Σ0: a set of initial states,
4.    O = ⟨e1, ..., e|O|⟩: a sequence of observable events,
5.    and h = ⟨t1, ..., t|h|⟩: a minimal history of Σ.

6.    for each σ0 ∈ Σ0 do
7.      s = σ0
8.      j = 1
9.      for each i ∈ [1 .. |h|] do
10.       if t_i does not exit from s then
11.         go to the next iteration of cycle at line 6
12.       if t_i is observable then
13.         e = γ(t_i) // observable event generated by t_i
14.         if j ≤ |O| and e == e_j then j++ else
15.           go to the next iteration of cycle at line 6
16.       s = state reachable from s by applying t_i
17.     end-for
18.     if j == |O| + 1 then return true
19.   end-for
20.   return false
21. end {Verify}
```

For each possible initial state in Σ_0, *Verify* simply takes into account each transition t_i in h, according to the order transitions appear in h. It checks whether t_i exits for the current state, and, in case t_i is observable, whether it produces the observable event expected according to $\mathcal{O}$. If, after all transitions in h have been encompassed, all (and only) the observable events in $\mathcal{O}$ have been produced, then true is returned, meaning that h actually solves P. □

[2]Only history-states distinct from the current one have to be considered as h is a minimal history and, as such, it does not include any unobservable cycle. A sequence of transitions that comes back to the current state without any observable event having occurred would instead follow an unobservable cycle.

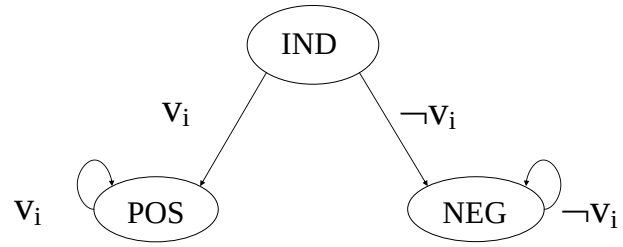

Figure 4: Automaton corresponding to a Boolean variable.

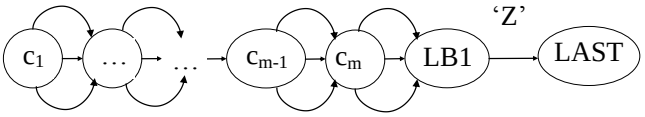

Figure 5: Automaton corresponding to a 3-CNF.

Theorem 5. *DIAGNOSIS-EXISTENCE(D_d) is NP-hard.*

Proof. NP-hardness is proven by means of a reduction from 3-SAT. Let ϕ be a Boolean formula in 3-CNF with variables $V = \{v_1, ..., v_n\}$ and clauses $C = \{c_1, ..., c_m\}$. We will construct an instance of DIAGNOSIS-EXISTENCE(D_d) relevant to a diagnosis problem $P_\phi = (\Sigma_\phi, \Sigma_{\phi 0}, \mathcal{O}_\phi)$ such that P_ϕ is solvable iff ϕ is satisfiable. Since a completely certain observation is a special case of a (temporally and/or logically) uncertain observation, and a completely certain initial state is a special case of an uncertain one, we will propose a reduction where P_ϕ is not affected by any uncertainty, since the same reduction can be reused in all the other settings.

(*Reduction*) For each $v_i \in V$ we build an FA resembling that in Fig. 4, where each transition is marked by a tag for synchronization. Then we build an FA corresponding to C, depicted in Fig. 5, this way: it includes $m + 2$ states, where each of the first m states corresponds to a distinct clause c_i. The generic state corresponding to clause c_i has three exiting transitions, each having a synchronization tag, corresponding to a distinct literal of c_i. Informally, we can move from the state corresponding to clause c_i to that corresponding to clause c_{i+1} only if c_i is true, which holds only iff at least one literal of its is true. This means that, when the current state of the FA in Fig. 5 is c_{i+1}, all clauses $c_1, \ldots, c_i$ are true. State $LB1$ has an exiting transition, generating observable event Z and leading to state *LAST*. Informally, state $LB1$ is reached only if all clauses are true. This is the only observable transition of Σ_ϕ. Altogether, the $n + 1$ automata obtained this way constitute Σ_ϕ. The initial state $\Sigma_{\phi 0}$ includes the state of the FA in Fig. 5 corresponding to clause c_1 and the state *IND* of the remaining n automata. Observation $\mathcal{O}_\phi$ consists of just one node, containing event Z. It it easy to see that the construction of P_ϕ is polynomial in the size of ϕ.

(*Direct implication*) Let us assume that ϕ is satisfiable, that is, there exists a variable assignment $\bar{V}$ of V such that ϕ is true. Let us build a sequence h of transitions of Σ_ϕ of length $m + 1$ this way: for each clause c_i, in the order of i from 1 to m, we pick up one of its literals, and consider the assignment of the variable such a literal refers to: if the literal is v_j and the assignment inherent to v_j in $\bar{V}$ is 1, or

the literal is $\neg v_j$ and the assignment inherent to v_j in $\bar{V}$ is 0, then we add a transition marked by the tag corresponding to such a literal (that is, tag v_j for literal v_j, and tag $\neg v_j$ for literal $\neg v_j$), which corresponds to following a (synchronous) transition marked by such a tag both in the FA corresponding to variable v_j and the FA corresponding to C; otherwise, we try the next literal in c_i, until the condition is fulfilled (the condition is bound to be fulfilled by a literal of c_i since ϕ is satisfiable, henceforth all its disjunctive clauses are true). After these m transitions, we add the last transition of the sequence, that leading from state *LB1* to *LAST* in the FA corresponding to C. The obtained sequence h is a history of Σ_ϕ. In fact, starting from the initial state of each FA corresponding to a variable v_j, in a mutually exclusive way we follow either the transition marked by tag v_j (if the assignment in $\bar{V}$ is $v_j = 1$) or the transition marked by tag $\neg v_j$ (if the assignment in $\bar{V}$ is $v_j = 0$); any subsequent transition in h taken from the same FA, is marked by the same tag as the previous one (as there is a unique assignment to v_j), which is compliant with the FA itself. Moreover, such a history is compliant with $\mathcal{O}_\phi$ since its last transition produces event Z. The history is minimal since the FA in Fig. 5 forces a state change every time a transition is triggered.

(*Reverse implication*) Let h_ϕ be a history of Σ_ϕ consistent with $\mathcal{O}_\phi$, and let s_f be the state reachable by applying h_ϕ to $\Sigma_{\phi 0}$. Since the only observable transition of Σ_ϕ is that leading from state *LB1* to *LAST* in the FA in Fig. 5, such a transition has necessarily to belong to h_ϕ and that it has to be preceded by m (unobservable synchronous) transitions that lead from the initial state c_1 to state $LB1$. Since every FA corresponding to a variable v_i (Fig. 4) contains just transitions that are synchronous with some transitions in the FA corresponding to C (Fig. 5), in h_ϕ there cannot be more that m transitions preceding the only observable one. There cannot be any transitions following the observable one since there are no transitions exiting from state *LAST* in the FA in Fig. 5. Moreover, by construction (see Fig. 4), h_ϕ cannot include any pair of transitions that belong to the same FA corresponding to a variable in V and are marked by distinct events. Let us consider an assignment of V obtained this way: for each variable $v_j \in V$, if the state of the FA relevant to v_j in s_f is *POS*, then the assignment is $v_j = 1$; if the state of the FA relevant to v_j in s_f is *NEG*, then the assignment is $v_j = 0$; finally, if the state of the FA relevant to v_j in s_f is *IND*, then the assignment is indifferently either $v_j = 1$ or $v_j = 0$. This assignment satisfies ϕ, that is, each clause c_i in C is true, since there exists a transition exiting from the i-th state of the FA corresponding to C, where such a transition corresponds to a literal in clause c_i, which is the same as that of the i-th transition in h_ϕ. □

No upper bound has been given for the complexity of DIAGNOSIS-EXISTENCE. The next theorem, which is inherent to both distributed and non-distributed DESs, is all we know so far.

Theorem 6. *DIAGNOSIS-EXISTENCE(D) is in PSPACE.*

Proof. We show an algorithm, called *Existence*, that solves DIAGNOSIS-EXISTENCE(D) in polynomial space. The notions of history-state and observation index, defined in the proof of Theorem 3, are exploited here.

```
1.  function Existence(P)
2.    where P = (Σ, Σ_0, O): a problem instance.

3.    for each σ_0 ∈ Σ_0 do
4.      S_in = (σ_0, 0)
5.      Enumerate all sequences o ∈ ||O|| (stripped of any ε)
6.      for each o do
7.        if Check(S_in, o, 1)
8.          then return true
9.    return false
10. end {Existence}

1.  function Check(S_0, o, i)
2.    where S_0: a history-state of system Σ,
3.    o: a sequence of observable events,
4.    i: an integer value in [1 .. |o| + 1] which
        is the successor of the observation index of S_0.

5.    if i == |o| + 1
6.      then return true
7.    Enumerate all history-states S_i whose observation
      index is i
8.    for each S_i do
9.      if Path(S_0, S_i, L_max)
10.       then if Check(S_i, o, i + 1)
11.         then return true
12.   return false
13. end {Check}

1.  function Path(S_a, S_b, L)
2.    where S_a, S_b: history-states of Σ,
3.    L: an integer value in [1 .. L_max].

4.    i_a = the observation index of S_a
5.    i_b = the observation index of S_b
6.    if (i_a == i_b) ∧ there is an unobservable transition
      from S_a to S_b
7.      then return true
8.      else if (i_b > i_a) ∧ there is an observable transition
        from S_a to S_b ∧ such a transition generates o[i_b]
9.      then return true
10.     else if L > 1
11.       Enumerate all history-states S' having the same
          observation index as S_a
12.       for each S' ≠ S_a do
13.         if Path(S_a, S', ⌈L/2⌉)
14.           then if Path(S', S_b, ⌈L/2⌉)
15.             then return true
16.   return false
17. end {Path}
```

Function *Existence* has to find out whether there exists a minimal history that solves problem P. Intuitively, it tries all the sequences of transitions that satisfy Theorem 3. For each state $\sigma_o \in \Sigma_o$, it creates the relevant history-state S_{in} (line 4). Then it enumerates, one by one, by using, for instance, a counter, all the sequences (of observable events) belonging to $||\mathcal{O}||$. For each of such sequences o, by invoking function *Check*, it checks whether there exists a behavioral path starting from the initial history-state S_{in} and compliant with $o[1 .. |o|]$. This means that the call of function *Check* at line 7 returns true if there exists a whole history of Σ, starting from S_{in} and producing o. In such a case, function *Existence* signals that there exists a solution of problem P by returning true. If, instead, there exists no $o \in ||\mathcal{O}||$ that can be produced by any history of Σ starting from some state in Σ_o, then *Existence* returns false.

Function *Check* is for checking whether there exists a behavioral path of Σ, starting from history-state S_0 and producing the observation $o[i .. |o|]$. If i equals $|o| + 1$, then true is returned since there certainly exists an (empty) path that does not generate any observable event. Otherwise, the search is split in two parts: one for a path that generates event $o[i]$ only (line 9), and the other for a path that generates the remaining of o, that is, $o[i + 1 .. |o|]$ (line 10). Since a path that generates event $o[i]$ ends in a history-state characterized by value i of the observation index, all the history-states S_i of Σ whose observation index value is i are enumerated, and, for each of them, by invoking func-

tion *Path* (line 9), it is tried whether there exists a path from S_0 to S_i, whose length is L_{max} at most, in accordance with Theorem 3. If such a path exists, then it is checked, by a recursive call of *Check* (line 10), whether there exists a behavioral path starting from S_i that generates the remaining of o. If this is the case, then we can conclude that there exists a behavioral path of Σ, starting from S_0 and producing the observation $o[i\,..\,|o|]$, thus *Check* returns true. If there is no path like that for any S_i, then *Check* returns false.

Function *Path* has to find out whether there exists a path, of length L at most, originating from the given history-state S_a of Σ and ending in the given history-state S_b. If there exists a transition of Σ leading from S_a to S_b, then true is returned. Otherwise, if L equals 1, false is returned. If, instead, L is greater than 1, the problem is split in two subproblems, which are conquered by recursive calls of *Path* (lines 13 and 14). Each subproblem is inherent to a path whose length is $\lceil L/2 \rceil$ at most. The first path has to start from S_a and end in a state S', which is distinct from S_a but has the same observation index as S_a. This is justified by considering that two cases can occur: either (i) *Path* is invoked by *Check*, or (ii) the call of *Path* is recursive. In the former case, the observation indexes of S_a and S_b have two consecutive values, thus the last transition in the path from S_a to S_b has to produce an observable event, while all the previous ones have to be unobservable. Hence, the first path has to lead to a state that share the same observation index as S_a. In the latter case, that is, every time *Path* is recursively called at line 13, S_a and S' have the same observation index. Every time *Path* is recursively called at line 14, either the observation indexes of S' and S'_b are the same or the latter is the successor of the former. In both cases, at any call, all the history-states having the same observation index as the first parameter of *Path* are enumerated at line 11. For each of them, if the first path exists, the existence of a path from S' to S_b is investigated. If both paths exist, true is returned. If, for any S', no such pair of paths exists, then false is returned.

Function *Path* needs to manipulate history-states, each of which includes an (integer) observation index and a state of Σ, which in turn includes the states of all components of Σ. The activation record of *Path* is $O(|\mathcal{C}|))$, where $\mathcal{C}$ is the set of all components. The length of the recursive chain of *Path* is $O(log(L))$, thus the space required by a run of *Path* is $O(|\mathcal{C}|log(L))$. The maximum value of parameter L is that passed by $Check$, that is, L_{max}, this being the product of the number of states of all components. Therefore, the space required by the run of *Path* is polynomial.

The activation record of *Check* is $O(|\mathcal{C}|))$. The length of the recursive chain of *Check* is $O(|o|)$, whose upper bound is the number of observation nodes $|\mathcal{O}|$. *Check* invokes *Path*, therefore it requires a space $O(|\mathcal{O}||\mathcal{C}|)$ + $O(|\mathcal{C}|log(L_{max}))$, which is polynomial.

Function *Existence* is not recursive and requires both the space for running *Check*, the space for variable o (which is $O(|\mathcal{O}|)$, and the space to save problem instance P. Thus, altogether, the algorithm requires a polynomial space. □

4 Conclusion

The complexity analysis faced in this paper is inherent to a decision problem (is there a candidate that solves a DES diagnosis problem?), whose complexity is a lower bound of that of the corresponding function problem (find the candidates of a DES diagnosis problem).

In [Portinale *et al.*, 2004] the decision problem consisting in determining whether an instance of a static diagnostic problem has a solution is proven to be NP-complete. Intuitively, the complexity of the same problem when dynamic systems are considered cannot be lower than that of static ones. However, the preliminary analysis performed by this paper is not enough to draw such a conclusion.

According to Theorem 4, checking the existence of a candidate is in NP if the DES is non-distributed and the observation is certain. This result is obtained since the length of a minimal history that solves the problem is polynomial in the number of states of the non-distributed DES, that is, for the effort to synchronize the behavior of system components is not included in the analysis. According to Theorem 5, checking the existence of a candidate is NP-hard if the DES is distributed. The main outcome (Theorem 6) is that deciding whether a DES diagnosis problem is solvable is in PSPACE. Future research will concentrate on finding a more precise upper bound for the complexity of such a problem, as well as on providing results inherent to non-distributed DESs in settings where the observation is uncertain.

References

[Cassandras and Lafortune, 2008] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer Science+Business Media, LLC, New York, NY, second edition, 2008.

[Grastien *et al.*, 2007] A. Grastien, Anbulagan, J. Rintanen, and E. Kelareva. Modeling and solving diagnosis of discrete-event systems via satisfiability. In *Eighteenth International Workshop on Principles of Diagnosis – DX'07*, pages 114–121, Nashville, TN, 2007.

[Hopcroft *et al.*, 2006] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA, third edition, 2006.

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

[Lamperti and Zanella, 2002] G. Lamperti and M. Zanella. Diagnosis of discrete-event systems from uncertain temporal observations. *Artificial Intelligence*, 137(1–2):91–163, 2002.

[Lamperti and Zanella, 2003] G. Lamperti and M. Zanella. *Diagnosis of Active Systems – Principles and Techniques*, volume 741 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publisher, Dordrecht, NL, 2003.

[Pencolé and Cordier, 2005] Y. Pencolé and M.O. Cordier. A formal framework for the decentralized diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence*, 164:121–170, 2005.

[Portinale *et al.*, 2004] L. Portinale, D. Magro, and P. Torasso. Multi-modal diagnosis combining case-based and model-based reasoning: a formal and experimental analysis. *Artificial Intelligence*, 158(1):109–153, 2004.

[Rintanen, 2007] J. Rintanen. Diagnosers and diagnosability of succinct transition systems. In *20th International Joint Conference on Artificial Intelligence – IJCAI'07*, pages 538–544, Hyderabad, India, 2007.

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

[Sohrabi *et al.*, 2010] S. Sohrabi, J.A. Baier, and S. McIlraith. Diagnosis as planning revisited. In *Twelfth International Conference on Knowledge Representation and Reasoning – KR 2010*, pages 26–36, Toronto, Canada, 2010. Association for the Advancement of Artificial Intelligence.