ECAI 2012
Luc De Raedt et al. (Eds.)

doi:10.3233/978-1-61499-098-7-582

On computing correct processes and repairs using partial behavioral models

Wolfgang Mayer [1] and **Gerhard Friedrich** [2] and **Markus Stumptner** [3]

Abstract. Diagnosis and repair of failed process executions is an important task for almost any process oriented application. Because in practice complete specifications of process activities are not available, diagnosis and repair methods for partial behavior models are of great importance. We show that if the assumption of complete behavioral models is lifted, basic diagnosis and repair problems reside on the second level of the Polynomial Hierarchy.

1 INTRODUCTION

The rising adoption of orchestrated execution of complex software processes, such as Web Services, has led to increased flexibility in application deployment, but also highlighted that isolating and repairing problems in such systems remains a challenging, predominantly manual task. This is exacerbated by the distributed and dynamic nature of such systems that would profit significantly from autonomous diagnosis and repair after runtime failures.

In such open environments, the correct control flow is specified but precise models of individual services and their possible behaviors are often challenging to build and hence are unavailable, in particular if stateful and data-dependent services are considered. However, in the absence of specifications, information can be gathered from executions: the sequence of activity executions along with their individual input/output values can be captured automatically in commercial service and process platforms and exploited for diagnosis. If a failure is detected, e.g. if a service raises an exception, a repair-enabled execution engine needs the ability to execute and re-execute activities in any order for achieving a successful process execution despite the fault. Diagnosis must decide which activity executions can be assumed either as correct or faulty.

Several diagnosis and repair approaches have been proposed [4, 1, 11], with varying reasoning methods and degree of formalization. However, to date, the predominant diagnosis (and repair) approaches require detailed formal specifications of activities in a process, need a large number of observed executions to be effective, or exhibit weak discriminatory power. In this paper we investigate the properties of the diagnosis model proposed by Friedrich et al. [5] which aims to overcome these limitations. It showed that path-specific behavior must be considered, which means that purely dependency-based representations are insufficient. We investigate the computational complexity of the approach, which is of general interest for constructing model-based diagnosis and repair systems. In particular, standard model-based diagnosis systems exploit propositional SAT solvers for checking if a set of fault (or correctness) assumptions (i.e. a diagnosis candidate) is a diagnosis. Consequently, for applying such techniques it is important to investigate if diagnosis candidate checking can be reduced efficiently to propositional SAT problems. Grastien et al. [6] have shown that various diagnostic problems for discrete event systems (DES) can be translated to propositional SAT problems, but the question remained open whether this method is applicable to diagnose process trajectories in case the process behavior is only partially known. Our results demonstrate that this is not efficiently possible in general, unless the Polynomial Hierarchy collapses.

Recent work in diagnosis of process and service execution has made great strides in studying how faults, once isolated, can be *repaired*. This can be expressed as a planning problem [4], but requires a planning language with sufficient expressivity. Dealing with incomplete information requires that any such approach must incorporate means to describe actions that are only partially specified, to restrict the set of plausible worlds and plausible diagnoses.

Effective repair formalisms must also allow one to perform state classification; that is, assess if, after the execution of a sequence of repair actions, the new process state (or a part thereof) is correct, or if some repair actions have failed. We consider a state to be correct if it is equivalent to one that can be reached in some correct execution of the original process originating in the same input values.

In this paper, we show that checking whether a set of fault assumptions is indeed a diagnosis is Σ_2^P-complete if the process behavior is only partially known. Furthermore, we show that verifying if a process state is correct is Π_2^P-complete in general. These results even hold for processes which contain just a sequence of activities (although our representation captures parallelism), and for diagnosis/repair cases where the only repair action is re-execution of activities and the number of repair action executions is limited.

There are multiple consequences of this finding.

(i) Diagnosis candidate checking and state classification cannot be efficiently reduced to propositional SAT problems for the described problem domain. No encoding of the problem which is polynomially translated to propositional logic and checked by a SAT solver can provide a sound and complete characterization of the faults in arbitrary processes (unless P=NP).
(ii) There is no efficient reduction of diagnosis of process trajectories to standard diagnostic reasoning in discrete event systems if the behavior of activities is only partially known.
(iii) The result underlines the importance of expressive knowledge representation for diagnosis and repair planning, and indicates the need for further research in finding effective reasoning frameworks for such situations, for example, by further developing the basis laid down in [5].

We provide an introductory example in Section 2. In Section 3 we summarize the process model that forms the basis for our diagnosis

[1] University of South Australia, Australia, email: mayer@cs.unisa.edu.au
[2] Universität Klagenfurt, Austria, email: Gerhard.Friedrich@aau.at
[3] University of South Australia, Australia, email: mst@cs.unisa.edu.au

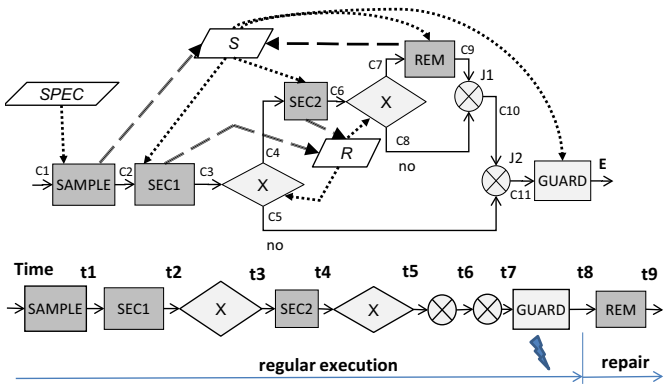

Figure 1. Example process (top) and observed execution (bottom)

model described in Section 4. The algorithmic complexity of diagnosis and state classification is investigated in Section 5.

2 EXAMPLE

As an example we use a simplified version of the example given in [5] depicted in Figure 1. The upper part shows the process definition, the lower part depicts the executions of activities. The process definition includes processing activities (e.g. SAMPLE) connected by a control-flow using XOR-splits (i.e. X) and OR-joins (i.e. $J1$ and $J2$) as control activities. Activities read from and output to process variables. Inputs and outputs of the process are defined by sets of process variables. In our example the input to the process is a specification of a test sample (variable $SPEC$) which is used by activity SAMPLE to generate a sample placed at S. S is inspected by security checks SEC1 and SEC2 outputting their results to decision variable R. Depending on the outcomes of SEC1 and SEC2, activity REM is executed to remove some parts of the sample. Before ending the process a guard examines the sample for a final quality control. This guard can decide that the process failed by assigning $true$ to the fault indicator E thus stopping the execution.

Assume a process execution as shown in the lower part of Figure 1. Time points mark the end of an executed activity. The completion of activity executions are observed. GUARD raises an exception by assigning $true$ to E at time $t8$. We assume that only activities SAMPLE, SEC1, SEC2, REM could be faulty. Given the flow of execution, activity executions SAMPLE_{t1} and SEC2_{t4} are the only ones that could have failed. $\langle \text{SAMPLE}_{t1}, \text{SEC2}_{t4} \rangle$ is the only minimal conflict so far. In particular, a correctness assumption of SEC1_{t2} is not needed to predict the exception raising of the guard. If we assume both SAMPLE_{t1} and SEC2_{t4} to behave correctly, S will not be changed regardless the branching of the first execution of X. However, we know that on the value assigned to S, the guard will raise an exception. Consequently, the exception is independent of the value SEC1_{t2} assigns to R.

For repairing, let us assume that a failure of SAMPLE_{t1} is unlikely, so $\{\text{SEC2}_{t4}\}$ is the only leading diagnosis. It follows that SEC2 must output to R a value such that the second occurrence of X takes the upper branch because otherwise an exception is raised. REM has to be executed to avoid the exception. If we assume that REM_{t9} works correctly then the output S will receive the correct value, no further repair actions are needed; that is, no action for faulty SEC2_{t4} is need.

The fundamental decision problem for diagnosis in this case is to decide which activity executions can be consistently assumed either to be correct or faulty. Based on a set of leading diagnoses, repairs can be performed where one of the basic repair actions is to (re-)execute activities preceded by a potential replacement. For repair the fundamental question is if after the executions of repair actions the output variables hold a correct value for all leading diagnoses. In this case the faulty process execution is assumed to be repaired.

3 PROCESS MODEL

We adopt the process model proposed by Friedrich et al. [5] for our purposes. A process is defined by discrete activities and their control- and data dependencies connecting the occurrences of activities in the process. The setting covers the middle ground between full formal models of each activity, which are not usually available, and purely dependency-based models, which suffer from imprecision [5]. Instead, the approach relies on a partial model of activities, where the behavior of each activity is gathered from execution logs and is restricted by constraints.

This process model is based on the assumption that the execution of each activity is totally observable *from the outside*. Activities are treated as "black boxes" and only little knowledge about the internal structure or behavior of an activity is utilized.

Each type of activity in our process is modeled as a finite state transition system over a set of variables. The entire process is modeled by composition of the transition systems representing the individual activities. Constraints on valid process behavior govern the overall execution of the process. For brevity, we omit the formal algorithm describing the construction of a finite state model (polynomial in the size of the process) from a process model and illustrate the idea based on the example of XOR activity X from Figure 1.

Gateway X in Fig. 1 determines if the flow of control proceeds along the *yes* or the *no* branch of the process, based on a data input R. For any given value of R, either branch can be activated, but not both. We assume that the decision outcome is a function of R, and all instances of X must behave consistently in any execution of the process, unless faulty. A process state comprises the values of data elements and activation of branches.

The execution of each activity is captured in terms of *events* that constitute transitions between system states. We assume that the pre-conditions and state changes corresponding to each event are known and events are observable. However, our model rests on the assumption (which lies at the core of debugging a software system) that the partitioning of all events into *normal* and *faulty* events is *not* known.

Example: Let $e_{ACT_{i,o}}$ represent the event of executing activity ACT which takes value i as input and outputs value o. In the example we assume all domains of variables to be binary. Four different events may be observed from the execution of X in Fig. 1: $e_{X_{t,y}}$, $e_{X_{t,n}}$, $e_{X_{f,y}}$, $e_{X_{f,n}}$. Event $e_{X_{t,y}}$ activates the *yes* branch if R is true. The other events are defined analogously. As exactly one of the alternative branches must be selected, events $e_{X_{t,y}}$ and $e_{X_{t,n}}$ (nor $e_{X_{f,y}}$ and $e_{X_{f,n}}$) may not occur simultaneously in any execution. From the execution in Fig. 1 we obtain an observation that event $e_{X_{t,y}}$ has occurred. While we know with certainty that the event has occurred, it is not known if this event indeed describes a valid decision for the entire process. We must assess if $e_{X_{t,y}}$ can steer the process towards a faulty state, and if changing the decision to $e_{X_{t,n}}$ can avoid all such states while maintaining behavioral consistency of all executions.

A salient feature of this model is that the events specifying the possible executions for each activity are usually not known in full. Therefore, any diagnostic reasoner cannot assume that a unique behavior model is given for each activity. Instead, a suitable behavior model must be derived by exploring different assumptions about which events may be included or excluded in the description of a correct

behaving activity. However, not every activity behavior is possible w.r.t. the domain and therefore the assumptions may be constrained. For example, although the branch choice of an XOR with respect to a specific data input value is unknown, either of the control outputs must be true whereas the other one must be false. Furthermore, some activities may produce different results for the same input upon repeated execution. Such non-deterministic behaviors occur frequently in processes where human input is involved.

This assumption is particularly suited to software-driven processes, where recording data is usually easily accomplished but detailed models of the executed software processes are not usually available. Events that occur throughout one or more executions will be collected in a set called the "observations". We consider acyclic processes where loops have been expanded into sequential iterations. This assumption is common in this domain, where executions are usually short.

4 SYSTEM MODEL

We base our process model on the formal framework of Discrete-Event System (DES), where the system states are characterized by the values of a finite set of variables with finite domains. Events govern which transitions between system states are permitted. We adapt the formal model of Grastien et al. [6] for our purposes. For simplicity, we describe the system model for the boolean domain where 1 and 0 represent truth and falsity, respectively. However, our model extends to other finite domains. Let $\mathcal{L}$ denote the language of propositional boolean expressions over alphabet A.

Definition 1 (Generic System Model) *The system model is a tuple $SD = \langle A, \Sigma, \delta, S_0, \Omega, BC\rangle$ where A is a finite set of propositional variables; Σ is a finite set of events; $\delta : \Sigma \mapsto \mathcal{L} \times 2^{\mathcal{L}}$ assigns to each event a pair $\langle \phi, c\rangle$; S_0 is the set of initial states and Ω is the set of faulty states. $BC \subseteq 2^{\Sigma}$ is a behavior constraint, which may be specified as a logic formula over symbols in Σ.*

Each event e represents a transition from a system state s to a successor state s'. Each event is specified by a pair $\delta(e) = \langle \phi_e, c_e\rangle$. Expression $\phi_e \in \mathcal{L}$ specifies the activation condition for e, and c_e expresses the effects of the event on the system state in terms of added and deleted propositions in $\mathcal{L}$.

An event e may happen in a state s only if $s \models \phi_e$. The successor state s' is obtained from s by applying the effects in c_e. Let c_e be a consistent subset of literals in $\mathcal{L}$ such that for all $a \in A$ at most one of a and $\neg a$ are in c_e. Let $s'(v) = 1$ if $v \in c_e$, $s'(v) = 0$ if $\neg v \in c_e$, and $s'(v) = s(v)$ for all remaining variables. We write $succ(s, c_e)$ for the successor obtained from s by applying c_e.

In this model, multiple events may occur simultaneously. A set of events $E = \{e_1, \ldots, e_n\}$ may happen in state s if (i) for all $e \in E$, e may happen in s, and (ii) no two $e_1, e_2 \in E$ interfere with each other. Events e_i and e_j are said to *interfere* if complementary literals are present in $c_i \cup c_j$. We define the successor state s' of s under a set of events E as $succ(s, \bigcup_{e\in E} c_e)$.

The dynamic evolution of the system proceeds as follows. Initially the transition system is in a state $s_0 \in S_0$. As events happen, the system exercises a trajectory along a sequence of states. A *trajectory* is a sequence $s_0, E_1, s_1, \ldots, E_n, s_n$ where the E_i are sets of non-interfering events, all $e \in E_i$ may happen in s_{i-1}, and $s_i = succ(s_{i-1}, \bigcup_{e\in E_i} c_e)$. In the remaining paper we restrict our attention to non-interfering trajectories.

Not all possible combinations of assumed-faulty events may be admissible, as restrictions on the behavior of system components may prohibit certain system trajectories. For example, a component representing a choice between two alternatives based on the current system state could be modeled as two transitions, each representing an alternate outcome. Whereas each individual transition reflects an alternative system behavior, no valid system model may include both transitions unless their preconditions are mutually exclusive.

We introduce a constraint BC that describes the allowed sets of events in a DES. Satisfying this constraint ensures that each diagnosis admits only system evolutions consistent with all necessary conditions imposed by the system components and process structure. It is assumed that $BC \neq \emptyset$; otherwise no diagnosis exists.

Although the normal behavior of X in Fig. 1 is not known precisely, necessary conditions for its behavior can be stated. For example, BC for a deterministic decision node excludes all sets that include either $\{e_{X_{t,y}}, e_{X_{t,n}}\}$ or $\{e_{X_{f,y}}, e_{X_{f,n}}\}$.

4.1 Diagnosis Model

Our diagnosis model departs significantly from "conventional" discrete-event models for diagnosis, such as that of Grastien et al. [6], which assume that all events are either known correct or known faulty, and that the faulty and some of the correct events may not be observable precisely. The difficulty of diagnosis in such a model usually stems from the fact that the presence of events and their ordering in a trajectory consistent with an observed event sequence may not be known in full.

In contrast, our model rests on the assumption that the presence and absence of events in a set of traces is known precisely, yet the partitioning of all observed events in *normal* and *faulty* is not known. Therefore, calculating a diagnosis in our model amounts to choosing a label (either *normal* or *faulty*) for each (observed and unobserved) transition such that the successful completion of all possible process executions is guaranteed. In contrast, "conventional" diagnosis models seek to infer the presence or absence of any of a set of unobservable (fault) events in a trajectory. Instead of their preassigned "fault" states, we need to infer the normal or faulty nature of events from the process execution (a faulty event eventually leads to a faulty state).

Definition 2 (Diagnosis Instance) *A Diagnosis Instance is a tuple $\langle SD, O\rangle$ comprised of system description $SD = \langle A, \Sigma, \delta, S_o, \Omega, BC\rangle$, and a set of observed events $O \subseteq \Sigma$.*

We assume that we can unambiguously observe events. The observations are a subset of events that occur throughout at least one observed trajectory of the system. We record only the presence of individual events in the trace, and ignore the relative order of events and timing information.

We consider only finite evolutions of our system. A trajectory $T = s_0, E_1, \ldots, E_n, s_n$ is said to be *faulty* if it contains a state $s_i \in \Omega$, and *normal* otherwise.

Computing a diagnosis for a given set of trajectories amounts to separating the observed events into those that reflect the intended behavior of the system from those that reflect the abnormal behavior of the corresponding process element. Any such partition is a diagnosis if the assignment of normal and faulty labels is consistent with the behavior constraints BC and each trajectory admitted by the resulting transition system completes successfully.

Definition 3 (Diagnosis) *Let $DI = \langle SD, O\rangle$ be a diagnosis instance. A set $\Delta \subseteq O$ is a diagnosis for DI iff there exists a set $\Sigma' \subseteq \Sigma$ such that*

(i) $O \setminus \Delta \subseteq \Sigma'$ and $\Sigma' \cap \Delta = \emptyset$;
(ii) $\Sigma' \in BC$; and
(iii) no trajectory $s_0, E_1, \ldots, E_k, s_k$ with $s_0 \in S_0$ and all $E_i \subseteq \Sigma'$ ($i \in \{1, \ldots, k\}$) includes a state $s_i \in \Omega$.

Diagnosis Δ is minimal if there is no $\Delta' \subset \Delta$ that is a diagnosis.

The first condition ensures that all observed events that are assumed normal are included and none of the assumed-faulty events are included in the transition system implied by the diagnosis. The second condition ensures that the selection of normal and faulty transitions obeys all behavioral constraints. The third condition ensures that each possible normal evolution of the system cannot reach a faulty state. That is, each trajectory comprises only normal events and avoids all states Ω.

Example: For the example process, Ω includes all states where the fault indicator E is true. For the observation set $O = \{e_{SAMPLE_{t,t}}, e_{SEC1_{t,t}}, e_{X_{t,y}}, e_{SEC2_{t,f}}, e_{X_{f,n}}, e_{GUARD_{t,t}}\}$, hypothesis $\Delta = \emptyset$ is not a diagnosis, since all corresponding system models comprise O and therefore admit a trajectory ending in a faulty state. Likewise, $\Delta = \{e_{SEC1_{t,t}}\}$ is not a diagnosis since all system models include $O \setminus \{e_{SEC1_{t,t}}\}$ and therefore SAMPLE with value t is not changed by REM in all possible models and the guard raises an exception. Diagnosis $\Delta = \{e_{SEC2_{t,f}}\}$ is indeed valid, as there exists a system model where the behavior of $SEC2$ may activate REM on $S = t$ which in turn may avoid the failure signaling of $GUARD$ by changing the value of S.

Without loss of generality, we consider only acyclic trajectories in condition (iii) in Def. 3. This does not result in the loss of diagnosis candidates: any cyclic trajectory that falsifies condition (iii) of Def. 3 has an acyclic sub-trajectory that also falsifies it. Conversely, if an acyclic trajectory falsifies condition (iii), returning to one of its states cannot result in a trajectory without faulty states.

Furthermore, it is sufficient to restrict our attention to trajectories of length n, where n is the maximum number of events in any acyclic trajectory of SD. Grastien et al. [6] showed that such an over-approximation yields correct results. Intuitively this is the case because shorter trajectories can be extended to the desired number of time steps by appending empty event sets that preserve the state.

4.2 Logical Model

Each diagnosis instance DI can be encoded into logical form as follows. We construct a formula Φ whose satisfying assignments represent the feasible trajectories of the system model $SD = \langle A, \Sigma, \delta, S_0, \Omega, BC\rangle$. Let $T = s_0, E_1, \ldots, E_n, s_n$ be an arbitrary sequence of states and sets of events of SD such that the indexes denote time steps. For each state $s_i \in T$ we introduce variables a_i for all $a \in A$, and for each event $e \in \Sigma$ we introduce variables e_i that indicate whether e is in E_i. Given a formula ψ, we denote by ψ_i the result of replacing each variable a in ψ by its indexed variant a_i. For state s, let $[s]$ denote the conjunction $\left(\bigwedge_{s(a)=1} a \wedge \bigwedge_{s(a)=0} \neg a\right)$.

The models of T describe a trajectories of SD if the following conditions are true for all $i \in \{1, \ldots, n\}$:

(1) Each event $e \in \Sigma$ is applicable in state s_{i-1} only if its precondition ϕ_e is true in s_{i-1}: $e_i \rightarrow \phi_{e_{i-1}}$.
(2) Each event may affect its successor state: $e_i \rightarrow \bigwedge_{l \in c_e} l_i$
(3) State variables change their valuations only as a consequence of events:
$$\neg a_{i-1} \wedge a_i \rightarrow \bigvee_{e \in Eff_a} e_i \quad \text{and} \quad a_{i-1} \wedge \neg a_i \rightarrow \bigvee_{e \in Eff_{\neg a}} e_i$$
where Eff_l denotes the set of events that include literal l in their effect set.
(4) Two events cannot occur simultaneously if they interfere: $\neg(e_{1i} \wedge e_{2i})$ for all pairs e_1, e_2 of mutually interfering events.
(5) State s_0 must be an initial state: $\bigvee_{s \in S_0} [s]_0$

Note that these formulas are direct counterparts to the components of Grastien et al.'s definition [6], except that we do not follow their distinction between events and event instances (which may have different pre- and postconditions even if they correspond to the same event). Without restriction of generality we simply assume that different event instances are encoded as different events.

The diagnosis criteria in Def. 3 can be related to this definition as follows: We use a set of hypothesis variables H to associate diagnostic hypotheses about events with transitions in the system model. Each particular choice of values for variables in H defines a particular specification of acceptable processes. H is disjoint from the set of variables in Φ. For each event $e \in \Sigma$ we introduce a proposition h_e in H that reflects whether a transition for e is enabled. For observed events $e \in O$, h_e expresses a fault assumption (normal or faulty). If e is assumed to be normal, h_e is true, and if e is assumed to be faulty, $\neg h_e$ is true. For events that have not been observed, h_e is true iff event e may take part in a possible unseen system evolution. We partition H into variables that correspond to observed events $H_o = \{h_e \in H | e \in O\}$ and those that correspond to unobserved events $H_u = \{h_e \in H | e \notin O\}$. Any truth assignment $\bar{H}_o$ to H_o corresponds to a diagnosis candidate: $\Delta = \{e \in \Sigma | h_e \in H_o, \bar{H}_o \models \neg h_e\}$. We will use $\Delta_l = \bigwedge_{\{h_e | e \in \Delta\}} \neg h_e \wedge \bigwedge_{\{h_e | e \in (O \setminus \Delta)\}} h_e$ to refer to the logical encoding. By construction, any set Σ' induced by an assignment to Δ and H_u satisfies Def. 3(i).

The system behavior must satisfy its behavioral constraints (Def. 3(ii)). We construct a *behavior compliance* formula ρ over variables in H such that ρ is satisfiable for a truth assignment $\bar{H}$ to H iff $\{e | h_e \in H, \bar{H} \models h_e\} \in BC$.

An event is applicable only if its corresponding hypothesis variable is true. We enforce this adding the following requirement to the construction of Φ above:

(6) For each event $e \in \Sigma$, $e_i \rightarrow h_e$

This encodes the $E_i \subseteq \Sigma'$ part of Def. 3(iii).

The logical system model Φ representing all trajectories of length up to n time steps is given by the conjunction of the formulas (1)–(6).

Def. 3(iii) quantifies over all trajectories and therefore cannot be expressed concisely as a pure satisfiability problem. Instead, we express the diagnosis criterion as an all-quantified boolean formula.

Intuitively, a set Δ is a diagnosis for a diagnosis instance DI iff the events in Δ are blocked in the system model, and a set of unobserved transitions H_u can be selected in addition to the remaining observed events $O \setminus \Delta$ such that all trajectories generated by this revised system model avoid all faulty states in Ω. We express the selection of transitions using existential quantification, whereas the path exploration over all trajectories requires universal quantification.

Proposition 1 *Let DI be diagnosis instance with system model SD and logical encoding Φ using variables V. A set $\Delta \subseteq O$ is a diagnosis for DI iff there exists an assignment $\bar{H}_u$ to variables in H_u such that*

(1) $\Delta_l \wedge \bar{H}_u \wedge \rho$ is satisfiable, and
(2) $\Delta_l \wedge \bar{H}_u \wedge (\forall V : \Phi \rightarrow \neg \bigvee_{i=1\ldots n, s \in \Omega} [s]_i)$ is consistent

Remember that ρ is the behavior compliance formula that states that the assignments in H must satisfy the behavior constraint BC.

5 COMPLEXITY

For each time step, the construction of the logical model is polynomial in the size of the given diagnosis instance. Overall, the size of Φ and the diagnosis criteria depends on the length n of traces that can be

represented. Assuming a polynomial bound on the length n, the size of the entire logical model is polynomial in size of the underlying diagnosis instance.

We show that deciding whether Δ is a diagnosis for a diagnosis instance is equivalent to deciding the validity of a specific class of Quantified Boolean Formula (QBF). We further prove that deciding if an output is correct can be phrased as the complement of deciding if a Δ is a diagnosis.

Definition 4 (Quantified Boolean Formula, QBF) *A QBF is a sentence of the form $Q_1 x_1 \dots Q_r x_r \gamma(x_1, \dots, x_r)$, $r \geq 0$ where $\gamma(x_1, \dots, x_r)$ is a propositional formula whose propositional variables are $x_1, \dots, x_r$ and where each $Q_i, 1 \leq i \leq r$ is one of the quantifiers $\forall, \exists$ ranging over $\{true, false\}$.*

Such a sentence has quantifier alterations for each $s > 1$ such that $Q_s \neq Q_{s+1}$ and for Q_1 itself. Then the set of $\text{QBF}_{k,\exists}$ of true sentences with k quantifier alternations and $Q_1 = \exists$ is a language in Σ_k^P. That is, $\text{QBF}_{1,\exists}$ corresponds to propositional SAT-problems. If $Q_1 = \forall$ then this is a language in Π_k^P. Σ_2^P corresponds to the languages recognizable by a non-deterministic Turing machine exploiting an NP oracle (NP^{NP}). Furthermore, NP $= \Sigma_1^P$, co-NP $= \Pi_1^P$, $\Delta_2^P = \text{P}^{\text{NP}}$, and co-$\Sigma_2^P = \Pi_2^P$.

Theorem 1 *Deciding whether Δ is a diagnosis for a diagnosis instance DI is Σ_2^P-hard. The decision problem is complete for this class if only trajectories of length polynomial in the size of DI are considered.*

Proof: Membership: By Prop. 1, deciding whether Δ is a diagnosis for DI can be done by establishing satisfiability of a propositional formula (Prop. 1 (1)), which is in class NP, and deciding the validity of a $\text{QBF}_{2,\exists}$ (Prop. 1 (2)), which is Σ_2^P-complete. Therefore, deciding the diagnosis problem clearly is a member of Σ_2^P.

Hardness: We show that deciding the truth of a $\text{QBF}_{2,\exists}$ can be reduced (using a polynomial reduction) to deciding whether $\emptyset$ is a diagnosis for a diagnosis instance.

Let $\psi = \exists x_1, \dots, x_m \forall u_1, \dots, u_n \gamma$ be a QBF where γ is a propositional formula containing only variables from $X = \{x_1, \dots, x_m\}$ and $U = \{u_1, \dots, u_n\}$. Let $DI = \langle SD, O \rangle$ be a diagnosis instance constructed from ψ and let $SD = \langle A, \Sigma, \delta, S_0, \Omega, BC \rangle$ be its system model depicted in Fig. 2. Let $A = X \cup U \cup \{q_0, q_1, q_2\}$, $\Sigma = \{e_x, e_u, e_\gamma | x \in X, u \in U\}$, $S_o = \{s_0\}$ such that $s_0(q_0) = 1$ and $s_0(a) = 0$ for $a \in A, a \neq q_0$, and $\Omega = \{s_3\}$ where $s_3(a) = 0$ for $a \in A$. Let BC be unconstrained. Events are defined as follows:

$$\delta(e_x) = \langle q_0, \{\neg q_0, q_1, x\} \rangle \text{ for all } x \in X$$
$$\delta(e_u) = \langle q_1, \{\neg q_1, q_2, u\} \rangle \text{ for all } u \in U$$
$$\delta(e_\gamma) = \langle q_2 \wedge \neg\gamma, \{\neg a | a \in A\} \rangle$$

We show that ψ is true if $\Delta = \emptyset$ is a diagnosis for $O = \{e_u, e_\gamma | u \in U\}$. By construction of SD, it is sufficient to consider trajectories over 3 time steps. Formula $\exists \bar{H}_u (\bar{H}_u \wedge \Delta_l \wedge \forall A \Phi \rightarrow [s_3]_3)$ must be true by Prop. 1. Here, H_u denotes the set of hypothesis variables for events that have not been observed: $H_u = \{h_{e_x} | x \in X\}$. As before, we write $\bar{H}_u$ for an assignment to variables in H_u. By construction of the events, this implies that there is an assignment to variables $X_1 = \{x_1 | x \in X\}$ determined by $\bar{H}_u$ such that for all assignments to the remaining variables, Φ implies $\neg[s_3]_3$. Since $\Delta = \emptyset$, all events in O and hence in $E_u = \{e_u | u \in U\}$ must be considered. By construction of the events, all combinations of events in E_u are explored, and therefore all possible assignments to variables in $U_2 = \{u_2 | u \in U\}$ are explored. By the definition of diagnosis, each such state must prohibit the transition to the faulty state, which is enabled only if $\neg\gamma$ is satisfied. Therefore, $\neg\gamma$ must be false for each assignment to U given the chosen assignment to variables in X, and ψ must be true.

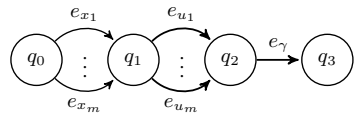

Figure 2. System model for $\exists \bar{X} \forall \bar{U} \gamma$

Conversely, assume that $\Delta = \emptyset$ is not a diagnosis. Since BC is unconstrained, the first part of Prop. 1 is trivially satisfied, and the second part must be false. This implies that there exists an assignment to hypothesis variables in H_u such that there is a trajectory from q_0 that reaches q_3. Reaching q_3 is possible only via e_γ, which implies that there is an assignment to variables in U_2 that satisfies $\neg\gamma$. Therefore, the assignments to X_1 and U_2 witness unsatisfiability of ψ.

Theorem 1 has a number of interesting implications. First, diagnosis of parallel systems is no harder than sequential systems:

Corollary 1 *Deciding whether Δ is a diagnosis for a diagnosis instance DI encoding a purely sequential trajectory model without parallelism is Σ_2^P-complete.*

Each trajectory in the proof of Th. 1 can be expanded into a sequence of at most polynomially many individual transitions and states. Therefore the result holds for the sequential instance.

Corollary 2 *Deciding whether Δ is a diagnosis for a diagnosis instance DI without external input is Σ_2^P-complete.*

External input can be simulated by non-deterministic activities. The same holds for restricting the model to a single initial state.

Corollary 3 *Deciding whether Δ is a diagnosis for a diagnosis instance DI is in* NP *if for all assignments $\bar{H}_o, \bar{H}_u$ in each state at most one event is applicable, i.e. the process behaves deterministically.*

For each initial state there is single trajectory, hence the second condition in Prop. 1 reduces to satisfiability.

Once diagnosis has been performed and a likely root cause has been identified, it is desirable to "repair" the execution. Building a repair plan to bring the execution into a state that corresponds to a fault-free execution of a sub-process is a common strategy. While selecting appropriate root causes and devising a repair plan is application-specific and not covered here, we investigate the problem of deciding if a state obtained after a repair plan has been executed is correct.

Definition 5 (Correct Output Assumption) *Let Δ be a diagnosis for a diagnosis instance $DI = \langle SD, O \rangle$ with $SD = \langle A, \Sigma, \delta, S_0, \Omega, BC \rangle$. Let $s_0 \in S_0$ be a distinct initial state and let ψ be a formula over variables in A. Formula ψ is a* correct output assumption *iff for all Σ' that satisfy conditions Def. 3(i)–(iii), it holds that there exists a trajectory $s_0, E_1, \dots, E_n, s_n$ such that $[s_n] \models \psi$.*

For example, for diagnosis $\Delta = \{e_{SEC2_{t,f}}\}$, $\neg S \wedge R$ is a correct output assumption.

Theorem 2 *Deciding if ψ is a correct output assumption for a given DI, Δ, and s_0 is Π_2^P-complete.*

Proof: Membership: Deciding the property of Def. 5 corresponds to verifying that for all $\Sigma' \subseteq \Sigma$ condition (i) or (ii) of Def. 3 are not satisfied or there exists a trajectory T of DI with events in Σ' which includes a faulty state (condition (iii) of Def. 3 is not satisfied) or ψ is satisfied in the final state of T. This can be decided by a $\text{QBF}_{2,\forall}$ thus proving membership in Π_2^P.

Hardness: We show that deciding whether a Δ is a diagnosis can be formulated as the complement of an output classification instance.

By Theorem 1, solving the diagnosis question is Σ_2^P-complete; this implies that deciding output correctness is Π_2^P-hard.

We construct from DI an output classification instance $OS = \langle SD', O\rangle$ where $SD' = \langle A \cup \{a\}, \Sigma \cup \{e_\psi\}, \delta', S_0, \Omega, BC\rangle$, $s(a) = 0$ for all $s \in S_0$, $\delta'(e) = \delta(e)$ for $e \in \Sigma$, and $\delta'(e_\psi) = \left\langle \bigvee_{s\in\Omega}[s] \vee a, \{a\}\right\rangle$. Let $\psi = a$ and let s_0 be chosen arbitrarily from S_0. We show that Δ is a diagnosis for DI iff ψ is *not* a correct output assumption for DI',Δ,s_0.

Assume that Δ is a diagnosis for DI. By Def. 3, any Σ' such that no state in Ω is reached is a suitable witness for Δ. In any such Σ', e_ψ cannot apply and $s \models \neg a$ for each state reachable from s_0 under Σ'. Therefore, ψ is *not* a correct output assumption for OS.

Conversely, assume that ψ is not a correct output assumption for OS. By Def. 5, there exists a Σ' that satisfies the conditions of a diagnosis (Def. 3) yet no state satisfying $s(a) = 1$ is reachable. By construction of δ', this implies that no state in Ω is reachable, and therefore Σ' is a witness for Δ being a diagnosis.

6 RELATED WORK

Dependency tracking techniques have been applied to diagnosis of programs [10] and Web Services [1, 11]. States and communication events of services are expressed as automata, and diagnosis is conducted by automaton composition and tracing dependencies. While such dependency tracking approaches provide efficiency, precision often suffers compared to methods exploiting partial knowledge [5].

Zhao et al. [12] account for incomplete discrete-event systems (DES) models by allowing transitions which are not justified by the system model. The fraction of trajectories that may not be fully justified by the nominal model is controlled by a numeric parameter. Variations of this parameter have influence on the number of spurious diagnoses. For any given parameter value, the approach corresponds to traditional diagnosis employing DES and therefore can be reduced to SAT. Consequently, no additional expressivity is added.

Kwong et al. [8] extend DES to incomplete models, with problem solving based on abduction. However, it is still assumed that an almost correct model of the system is available where only few transitions are missing. Multiple faults must be encoded as single states which may exponentially increase the number of states.

Planning for diagnosis and repair [9] has been concerned with analyzing sequences of actions and events that lead to a particular set of observations. This approach is predominantly concerned with devising an explanatory process trace while assuming that the operations (but not states) are fully known. In contrast, we assume that the process structure is given but the definitions of each activity are not entirely known.

Eiter et al. [3] showed that various problems of logic-based abduction reside on the second level of the Polynomial Hierarchy. Their results apply to diagnosis frameworks which assume complete fault models. One of the first studies on the incompleteness of models in diagnosis was given in [2], where the anonymous cause was introduced to cope with incomplete models. Kuhn et al. [7] developed an approach to reason about incomplete models due to hidden structural faults in circuits. While causes (fault modes) can be classified into correct and faulty in this (and indeed in most) classical model-based diagnosis frameworks, partial knowledge approaches dispense with that fixed assumption and infer this information from observations.

7 CONCLUSION

In this paper we introduced a formal model for the analysis of diagnosing process executions in situations where the existence of a pre-specified complete model cannot be assumed. Such circumstances can be found in many diagnosis problems related to software, for example, embedded systems or orchestration scenarios in Service-Oriented Architectures [5]. We showed that checking whether a set of assumptions is a diagnosis is Σ_2^P-complete in general if the possible behaviors of individual activities are only partially known.

This result is particularly significant because current diagnosis frameworks have usually assumed that diagnosis candidate checking can be reduced to propositional satisfiability checking (or constraint satisfaction). As we have shown that this does not apply in partial knowledge circumstances, traditional diagnostic modeling and reasoning approaches cannot produce sound and complete results in important emerging application domains for diagnosis systems, such as the service or embedded systems domains.

Moreover, deciding if the values in a process state could be obtained by a correct execution is Π_2^P-complete under the same assumptions. This questions the common use of simple satisfiability-based methods to predict the correctness of output values obtained from the execution of repair activities. Our results underline the practical necessity of further work in the area of expressive reasoning frameworks, such as Answer Set Programming, for diagnosis and repair. Our results indicate that there is a need both for sufficiently powerful representation mechanisms for diagnosis and repair problems and for improved reasoning mechanisms that can operate effectively in environments where complete models cannot be attained, as well as for further studies into which types of systems, for examples with limited number of inputs or outputs might result in problem subclasses of lesser complexity, or which additional assumptions (such as restrictions on observability) would make the problem harder.

REFERENCES

[1] L. Ardissono, L. Console, A. Goy, G. Petrone, C. Picardi, M. Segnan, and D. Theseider Dupré, 'Enhancing web services with diagnostic capabilities', in *European Conference on Web Services*, (2005).

[2] L. Console, D. Theseider Dupré, and P. Torasso, 'A theory of diagnosis for incomplete causal models', in *IJCAI*, pp. 1311–1317, (1989).

[3] T. Eiter and G. Gottlob, 'The complexity of logic-based abduction', *J. ACM*, **42**(1), 3–42, (1995).

[4] G. Friedrich, M. Fugini, E. Mussi, B. Pernici, and G. Tagni, 'Exception handling for repair in service-based processes', *IEEE TSE*, (2010).

[5] G. Friedrich, W. Mayer, and M. Stumptner, 'Diagnosing process trajectories under partially known behavior', in *ECAI*, pp. 111–116. IOS Press, (2010).

[6] A. Grastien, Anbulagan, J. Rintanen, and E. Kelareva, 'Diagnosis of discrete-event systems using satisfiability algorithms', in *AAAI*, pp. 305–310. AAAI Press, (2007).

[7] L. Kuhn and J. de Kleer, 'Diagnosis with incomplete models: Diagnosing hidden interaction faults', in *Proc. DX-10 Workshop*, pp. 225–232, (2010).

[8] Raymond H. Kwong and David L. Yonge-Mallo, 'Fault diagnosis in discrete-event systems: Incomplete models and learning', *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, **41**(1), 118–130, (2011).

[9] S. A. McIlraith, 'Explanatory diagnosis: Conjecturing actions to explain observations', in *KR*, pp. 167–179, (1998).

[10] F. Wotawa, 'On the relationship between model-based debugging and program slicing', *Artif. Intell.*, **135**(1-2), 125–143, (2002).

[11] Y. Yan, P. Dague, Y. Pencolé, and M.-O. Cordier, 'A model-based approach for diagnosing fault in web service processes', *Int. J. Web Service Res.*, **6**(1), (2009).

[12] X. Zhao and D. Ouyang, 'Model-based diagnosis of discrete event systems with an incomplete system model', in *ECAI*, volume 178, pp. 189–193. IOS Press, (2008).