Computation of Projections for the Abstraction-based Diagnosability Verification

Klaus Schmidt [*]

[*] *Department of Electronic and Communication Engineering, Çankaya University, Ankara, Turkey (e-mail: schmidt@cankaya.edu.tr)*

Abstract: The verification of *language-diagnosability* (LD) for discrete event systems (DES) generally requires the explicit evaluation of the overall system model which is infeasible for practical systems. In order to circumvent this problem, our previous work proposes the *abstraction-based* LD verification using natural projections that fulfill the *loop-preserving observer* (LPO) property. In this paper, we develop algorithms for the verification and computation of such natural projections. We first present a polynomial-time algorithm that allows to test if a given natural projection is a loop-preserving observer. Then, we show that, in case the LPO property is violated, finding a minimal extension of the projection alphabet such that the LPO condition holds is NP-hard. Finally, we adapt a polynomial-time heuristic algorithm by Feng and Wonham for the efficient computation of loop-preserving observers.

Keywords: Discrete event systems, failure diagnosis, language-diagnosability, abstraction.

1. INTRODUCTION

The *diagnosability verification* for discrete event systems (DES) is based on a *plant model* that captures potential faults, a representation of the *faulty system behavior* and a characterization of the possible system *observations*. Then, it is desired to determine if each fault in the system can be detected after the occurrence of a bounded number of events using the available observations.

The diagnosability verification has been investigated in different frameworks. Sampath et al. (1995) introduce the characterization of faults by *failure events*, while Qiu and Kumar (2006); Yoo and Garcia (2008) study *language-diagnosability*, whereby faulty system behavior corresponds to the violation of a *failure specification language*. In both frameworks, diagnosability is verified in different architectures such as the *centralized* architecture (Sampath et al., 1995; Yoo and Lafortune, 2002; Jiang et al., 2001; Yoo and Garcia, 2008), the *decentralized* architecture (Su and Wonham, 2005; Wang et al., 2007; Qiu and Kumar, 2006) or the *modular* architecture (Contant et al., 2006; Debouk et al., 2002).

All these methods have in common that they are based on the explicit computation of the overall system model which is infeasible for practical systems or they require restrictive conditions. To address this issue, our previous work in (Schmidt, 2010a,b) introduces the idea of *abstraction-based* language-diagnosability verification. It is shown that *natural projections* that fulfill the *loop-preserving observer* (LPO) condition are suitable to compute system abstractions that allow to verify language-diagnosability using system models on smaller state spaces.

In this paper, we provide the required algorithmic support for the verification and computation of appropriate natural projections. We develop a polynomial-time algorithm for the verification of LPO. In case LPO is violated, it is desired to find a minimal extension of the projection alphabet in order to achieve LPO. In this context, we first show that the computation of such minimal extensions is NP-hard. Then, an adaptation of the *observer extension* algorithm by Feng and Wonham (2010) yields a polynomial-time heuristic for the computation of loop-preserving observers in practical cases.

The paper is organized as follows. Section 2 summarizes the notation and definitions required in this paper, and Section 3 elaborates the abstraction-based language-diagnosability verification based on natural projections. The verification of appropriate natural projections that fulfill LPO is performed in Section 4, while Section 5 develops an algorithm for the computation of such natural projections. Conclusions are given in Section 6.

2. PRELIMINARIES

2.1 Discrete Event Systems

For a finite alphabet Σ, the set of all finite strings over Σ is denoted as Σ^*. We write $s_1 s_2 \in \Sigma^*$ for the concatenation of strings s_1, $s_2 \in \Sigma^*$ and $s_1 \leq s$ when s_1 is a *prefix* of s. The empty string is $\epsilon \in \Sigma^*$, i.e., $s\epsilon = \epsilon s = s$ for all $s \in \Sigma^*$, and $|s|$ describes the length of s. A *language* over Σ is a subset $L \subseteq \Sigma^*$ with the *prefix closure* $\overline{L} := \{s_1 \in \Sigma^* | \exists s \in L \text{ s.t. } s_1 \leq s\}$. A language L is *prefix closed* if $L = \overline{L}$.

The *natural projection* $p : \Sigma^* \to \hat{\Sigma}^*$, $\hat{\Sigma} \subseteq \Sigma$ is defined iteratively: (1) let $p(\epsilon) := \epsilon$; (2) for $s \in \Sigma^*$, $\sigma \in \Sigma$, let $p(s\sigma) := p(s)\sigma$ if $\sigma \in \hat{\Sigma}$, or $p(s\sigma) := p(s)$ otherwise. The inverse of p is $p^{-1} : \hat{\Sigma}^* \to 2^{\Sigma^*}$, $p^{-1}(t) := \{s \in \Sigma^* | p(s) = t\}$. A condition for natural projections that is relevant for prefix-closed languages in this paper is the *observer condition* (Wong and Wonham, 2004).

Definition 1. Let $L = \overline{L} \subseteq \Sigma^*$ be a prefix-closed language. The projection $p : \Sigma^* \to \hat{\Sigma}^*$ is an observer for L if for all $s \in L$, $t \in \hat{\Sigma}^*$,

$$p(s)t \in p(L) \Rightarrow \exists u \in \Sigma^* \ s.t. \ su \in L \wedge p(su) = p(s)t. \quad (1)$$

We model a DES by an *automaton* $G = (X, \Sigma, \delta, x_0)$ with the *states* X, the *alphabet* Σ, the partial *transition function* $\delta : X \times \Sigma \to X$ and the *initial state* x_0. We write $\delta(x, \sigma)!$ if δ is defined at (x, σ), and extend δ to strings in the usual way. $|X|$ and $|\delta|$ characterize the number of states and transitions of G, respectively. $L(G) := \{s \in \Sigma^* : \delta(x_0, s)!\}$ is the language generated by G. The synchronous composition $G_1||G_2$ of two automata G_1 and G_2 is defined as, e.g., in (Cassandras and Lafortune, 2006).

2.2 Language-Diagnosability

Similar to (Qiu and Kumar, 2006; Yoo and Garcia, 2008; Schmidt, 2010a), we consider a partially observed DES $G = (X, \Sigma, \delta, x_0)$. The system behavior is seen through a *mask* $M : \Sigma \to \Delta \cup \{\epsilon\}$ that maps each event $\sigma \in \Sigma$ to its observation $M(\sigma) \in \Delta \cup \{\epsilon\}$ (Δ is the *set of observations*). M is recursively extended to strings by defining $M(s\sigma) = M(s)M(\sigma)$ for $s \in \Sigma^*$ and $\sigma \in \Sigma$, and $\Sigma_o := \{\sigma \in \Sigma | M(\sigma) \neq \epsilon\}$ denotes the set of *observable* events.

In the described framework, a failure corresponds to the violation of a given prefix-closed *specification language* $K = \overline{K} \subseteq L(G)$. In particular, it is desired to detect if a faulty string in $L(G) - K$ occurred by partial observation through the mask M. A formal characterization of this *language-diagnosability* is given in the following definition (Qiu and Kumar, 2006; Yoo and Garcia, 2008).

Definition 2. Let G model a DES and let $K = \overline{K} \subseteq L(G)$ be a specification language. K is language-diagnosable for G and the observation mask $M : \Sigma \to \Delta \cup \{\epsilon\}$ if

$$\begin{aligned}&(\exists n \in \mathbb{N})(\forall s \in L(G) - K)(\forall st \in L(G), |t| \geq n \text{ or } st \\ &\quad \text{deadlocks}) \Rightarrow (\forall u \in M^{-1}M(st) \cap L(G), u \notin K).\end{aligned} \quad (2)$$

If (2) holds, then every string that deviates from the correct behavior in K can be uniquely distinguished from strings in K after the occurrence of a bounded (either by n or by deadlock) number of events.

2.3 Set Theory

We present basic results from set theory as employed by Feng and Wonham (2010); Wong and Wonham (2004). We write $\mathcal{E}(Q)$ for the set of all *equivalence relations* on the set Q. For $\mu \in \mathcal{E}(Q)$, $[q]_\mu$ is the *equivalence class* containing $q \in Q$. The set of equivalence classes of μ is written as $Q/\mu := \{[q]_\mu | q \in Q\}$ and the *canonical projection* $\mathrm{cp}_\mu : Q \to Q/\mu$ maps an element $q \in Q$ to its equivalence class $[q]_\mu$. Let $f : Q \to R$ be a function. The equivalence relation $\ker f$ is the *kernel* of f and is defined as follows.

$$\text{for } q, q' \in Q, \quad q \equiv q' \mod \ker f \Leftrightarrow f(q) = f(q').$$

Given two equivalence relations η and μ on Q, $\mu \leq \eta$, i.e. μ refines η, if $q \equiv q' \mod \mu \Rightarrow q \equiv q' \mod \eta$ for all $q, q' \in Q$. In addition, we define the *meet* operation $\wedge$ for $\mathcal{E}(Q)$ as follows. For any two elements $\mu, \eta \in \mathcal{E}(Q)$, it holds for all $q, q' \in Q$ that

$$q \equiv q' \mod (\mu \wedge \eta) \Leftrightarrow q \equiv q' \mod \mu \text{ and } q \equiv q' \mod \eta.$$

Let Q and R be sets and $f : Q \to 2^R$ be a function. It is also assumed that $\varphi \in \mathcal{E}(R)$, and the canonical projection cp_φ is naturally extended to sets. The equivalence relation $\varphi \circ f$ on Q is defined for $q, q' \in Q$ by

$$q \equiv q' \mod \varphi \circ f \Leftrightarrow \mathrm{cp}_\varphi(f(q)) = \mathrm{cp}_\varphi(f(q')).$$

Now let $f_i : Q \to 2^Q$ be functions, where i ranges over an index set $\mathcal{I}$. Then $S := (Q, \{f_i | i \in \mathcal{I}\})$ is called a *dynamic system* (Wong and Wonham, 2004) and the equivalence relation $\varphi \in \mathcal{E}(Q)$ is called a *quasi-congruence* for S if

$$\varphi \leq \bigwedge_{i \in \mathcal{I}} (\varphi \circ f_i).$$

The *quotient automaton* (QA) $G_{\mu,\hat{\Sigma}} = (Y, \hat{\Sigma} \cup \{\sigma_0\}, \nu, y_0)$ of an automaton $G = (X, \Sigma, \delta, x_0)$ for an equivalence relation $\mu \in \mathcal{E}(X)$ and an alphabet $\hat{\Sigma} \subseteq \Sigma$ is introduced as in (Wong and Wonham, 2004). It holds that $Y := X/\mu$ is the *quotient set* with the associated *canonical projection* $\mathrm{cp}_\mu : X \to Y$. The initial state is $y_0 = \mathrm{cp}_\mu(x_0)$. Also $\sigma_0 \notin \Sigma$ is an additional label. The nondeterministic *induced transition function* $\nu : Y \times (\hat{\Sigma} \cup \{\sigma_0\}) \to 2^Y$ of $G_{\mu,\hat{\Sigma}}$ is defined as

$$\nu(y, \sigma) := \begin{cases} \{\mathrm{cp}_\mu(\delta(x, \sigma)) | x \in \mathrm{cp}_\mu^{-1}(y)\} & \text{if } \sigma \in \hat{\Sigma} \\ \{\mathrm{cp}_\mu(\delta(x, \gamma)) | \gamma \in (\Sigma - \hat{\Sigma}), \\ x \in \mathrm{cp}_\mu^{-1}(y)\} - \{y\} & \text{if } \sigma = \sigma_0. \end{cases}$$

We finally relate the verification of the observer condition to a quotient automaton that is computed for a particular quasi-congruence (Wong and Wonham, 2004).

Proposition 1. Let $G = (X, \Sigma, \delta, x_0)$ be an automaton and $p : \Sigma^* \to \hat{\Sigma}^*$ a natural projection with $\hat{\Sigma} \subseteq \Sigma$. Let $S = (X, \{\Delta_\sigma | \sigma \in \hat{\Sigma}\})$ be a dynamic system with

$$\Delta_\sigma : X \to 2^X : x \to \{\delta(x, u\sigma u') | uu' \in (\Sigma - \hat{\Sigma})^*\}.$$

Then, the coarsest quasi-congruence

$$\mu^* := \sup\{\mu \in \mathcal{E}(X) | \mu \leq \bigwedge_{\sigma \in \hat{\Sigma}} (\mu \circ \Delta_\sigma)\}$$

on S exists and can be computed with a complexity of $\mathcal{O}(|X|^3 \cdot |\delta|)$. Furthermore, $p : \Sigma^* \to \hat{\Sigma}^*$ is an observer for $L(G)$ if and only if the quotient automaton $G_{\mu^*,\hat{\Sigma}}$ is deterministic and does not contain any σ_0-transitions. □

Fig. 1 illustrates Proposition 1. Considering the automaton G and the alphabet $\hat{\Sigma} = \{\alpha, \beta, \gamma\}$, the dynamic system S is shown in Fig. 1 (b). Here, the shaded boxes indicate the equivalence classes of the coarsest quasi-congruence μ^* on S. Accordingly, Fig. 1 (c) shows the obtained quotient automaton $G_{\mu^*,\hat{\Sigma}}$. Since $G_{\mu^*,\hat{\Sigma}}$ contains a σ_0-transition, $p : \Sigma^* \to \hat{\Sigma}^*$ is not an observer for $L(G)$.

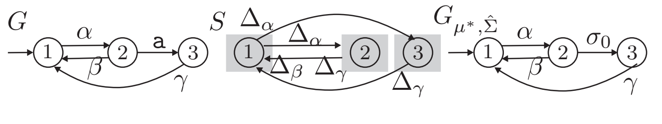

Fig. 1. Observer verification: (a) automaton G; (b) dynamic system S; (c) quotient automaton $G_{\mu^*,\hat{\Sigma}}$.

3. ABSTRACTION-BASED DIAGNOSABILITY

3.1 General Setup

In this section, we outline the concept of abstraction-based language diagnosability introduced in (Schmidt, 2010a). As discussed in Section 2.2, we use a model G and an observation mask M. Moreover, we assume that the specification $K \subseteq \Sigma^*$ can be formulated based on a reduced specification $K' = \overline{K}' \subseteq \Sigma'^*$ with $\Sigma' \subseteq \Sigma$ s.t.

$$K = K' || L(G) \subseteq L(G).$$

Then, (Schmidt, 2010a) proposes to use an *abstracted model* $\hat{G}$ over an *abstraction alphabet* $\hat{\Sigma}$ with $\Sigma' \subseteq \hat{\Sigma} \subseteq \Sigma$ for the verification of language-diagnosability. $\hat{G}$ is computed by applying the natural projection $p : \Sigma^* \to \hat{\Sigma}^*$, and the *abstracted specification* is $\hat{K} \subseteq \hat{\Sigma}^*$ such that

$$\begin{aligned} L(\hat{G}) :=& p(L(G)), \\ \hat{K} :=& K' || L(\hat{G}) = p(K). \end{aligned}$$

The *abstracted observation mask* is $\hat{M} : \hat{\Sigma} \to \hat{\Delta} \cup \{\epsilon\}$, whereby $\hat{\Delta} = \{M(\sigma) | \sigma \in \hat{\Sigma}\}$ contains the observations of events in $\hat{\Sigma}$, and for all $\sigma \in \hat{\Sigma}$, $\hat{M}(\sigma) = M(\sigma)$.

That is, the idea of abstraction-based diagnosability verification is to use the abstracted (and supposedly smaller) entities $\hat{G}$, $\hat{K}$ and $\hat{M}$ to perform the classical diagnosability test, whereby certain sufficient conditions have to be met in order to ensure equivalence to the original diagnosability test with G, K and M.

3.2 Sufficient Conditions

Based on the abstraction described in the previous section, Schmidt (2010a) derives sufficient conditions such that language-diagnosability for an automaton G, an observation mask M and a reduced specification K' can be verified using the abstractions $\hat{G}$, $\hat{M}$ and $\hat{K}$. First, M has to be *consistent* with the natural projection p, i.e., all events with the same non-empty observation must either be retained or removed by the abstraction.

Definition 3. The observation mask $M : \Sigma \to \Delta \cup \{\epsilon\}$ with the set of observable events $\Sigma_o := \{\sigma \in \Sigma | M(\sigma) \neq \epsilon\}$ is consistent with the natural projection $p : \Sigma^* \to \hat{\Sigma}^*$ if

$$\sigma \in \hat{\Sigma} \cap \Sigma_o \Rightarrow M^{-1}M(\sigma) \subseteq \hat{\Sigma}.$$

Second, p has to be a *loop-preserving observer* (LPO), i.e., an observer for $L(G)$ that additionally ensures that any arbitrarily long strings in $L(G)$ also appear as such strings in the abstraction $p(L(G))$.

Definition 4. Let $p : \Sigma^* \to \hat{\Sigma}^*$ be an observer for $L := L(G)$ and let s, t be as in Definition 1. p is an LPO if there exists an $N \in \mathbb{N}$ such that for all $u \in \Sigma^*$ s.t. $su \in L$ and $p(su) = p(s)t$ it holds that $|u| < N(|t| + 1)$.

It is shown in (Schmidt, 2010a) that the conditions in Definition 3 and 4 are sufficient for the abstraction-based language-diagnosability verification.

Theorem 2. Let G be a model automaton, $K' \subseteq \Sigma'^*$ be a reduced specification and $M : \Sigma \to \Delta \cup \{\epsilon\}$ be an observation mask. We also define $\hat{G}$, $\hat{K}$ and $\hat{M}$ as above for the abstraction alphabet $\hat{\Sigma}$ with $\Sigma' \subseteq \hat{\Sigma} \subseteq \Sigma$. If $p : \Sigma^* \to \hat{\Sigma}^*$ is an LPO and M is consistent with p, then language-diagnosability of $\hat{K}$ for $\hat{G}$ and $\hat{M}$ implies language-diagnosability of $K := K' || L(G)$ for G and M, while $\hat{G}$ has a smaller state space than G. If additionally $\Sigma_o \subseteq \hat{\Sigma}$, the reverse implication also holds, i.e., $\hat{K}$ is language-diagnosable for $\hat{G}$ and $\hat{M}$ if K is language-diagnosable for G and M.

Hence, language-diagnosability can be confirmed using only the abstracted model and the violation of language-diagnosability can be determined if $\Sigma_o \subseteq \hat{\Sigma}$. Moreover, since the abstracted models are usually considerably smaller than the original models, computational savings are achieved by the abstraction-based verification. An example of such computation is given in (Schmidt, 2010a).

Theorem 2 is illustrated in Fig. 2. Here, we assume an observation mask M with $M(\alpha) = \alpha$, $M(\beta) = \beta$ and $M(\sigma) = \epsilon$ for all remaining events. Furthermore, it is specified that the event $\mathtt{b}$ should not occur, i.e., $K' = \{\epsilon\}$ for $\Sigma' = \{\mathtt{b}\}$. Choosing the abstraction alphabet $\hat{\Sigma} = \{\alpha, \beta, \mathtt{a}, \mathtt{b}\}$, it holds that $\hat{\Sigma}$ is consistent with M. Finally, we use the LPO $p : \Sigma^* \to \hat{\Sigma}^*$ to compute $\hat{G}$ and a specification automaton $\hat{C}$ with $L(\hat{C}) = \hat{K} = K' || L(\hat{G})$ as shown in Fig. 2. Then, language-diagnosability can be verified for the abstracted system since the occurrence of $\mathtt{b}$ is detected if the substrings $\beta\alpha$ or $\beta\beta$ are observed. Hence, the original system is also language-diagnosable for the specification $K = K' || L(G)$.

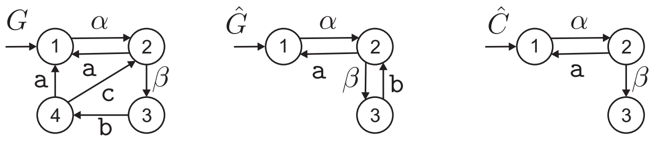

Fig. 2. Abstraction-based language-diagnosability.

It has to be noted that Schmidt (2010a) focuses on finding sufficient conditions for the abstraction-based language-diagnosability verification without providing algorithms for the verification of the sufficient conditions or the computation of appropriate natural projections. Solutions for both issues are derived in the following sections and constitute the main contribution of this paper.

4. VERIFICATION

4.1 Consistency

First, the consistency of M with $p : \Sigma^* \to \hat{\Sigma}^*$ has to be determined. For each observable event in the abstraction alphabet $\sigma \in \Sigma_o \cap \hat{\Sigma}$, it has to be confirmed that there is no event $\hat{\sigma} \notin \hat{\Sigma}$ with the same observation $M(\hat{\sigma}) = M(\sigma)$. That is, consistency can be verified in polynomial time that is bounded by $\mathcal{O}(|\Sigma|^2)$. Also note that, if M is not consistent with p, then it is sufficient to add all events that cause the consistency violation to $\hat{\Sigma}$. Hence, in the following, we assume that M is consistent with p.

4.2 Loop-preserving Observer

Second, it has to be verified if $p : \Sigma^* \to \hat{\Sigma}^*$ is an LPO. Intuitively, this requires to check if the observer condition in (1) holds and G has no loops with events in $\Sigma - \hat{\Sigma}$. Formally, the verification of the LPO condition is achieved by Proposition 3. We first recall the notion of a *strongly connected component* (Hopcroft and Ullman, 1975).

Definition 5. Let $G = (X, \Sigma, \delta, x_0)$ be an automaton. A subautomaton of G with the states $\mathcal{G} \subseteq X$ is called a strongly connected component (SCC) of G if there is $u, u' \in \Sigma^*$ for all state pairs $x, x' \in \mathcal{G}$ s.t. $\delta(x, u) = x'$ and $\delta(x', u') = x$, and for all $\mathcal{G}' \supset \mathcal{G}$, $\mathcal{G}'$ is not an SCC of G. An SCC is *trivial* if it has a single state without any selfloops.

Proposition 3. Let $G = (X, \Sigma, \delta, x_0)$ be an automaton and $p : \Sigma^* \to \hat{\Sigma}^*$ the natural projection. Furthermore, define the automaton $\overline{G} = (X, \Sigma - \hat{\Sigma}, \overline{\delta}, x_0)$ s.t. $\overline{\delta}$ is derived from δ by simply deleting all transitions with events in $\hat{\Sigma}$. Then, p is a loop-preserving observer iff (i) p is an observer for $L(G)$ and (ii) $\overline{G}$ contains only trivial SCCs.

Proof. "$\Rightarrow$": Assume that p is an LPO. Then, by Definition 4, p is an observer for $L(G)$. It remains to show (ii). Assume the contrary, i.e., $\overline{G}$ contains a loop that starts from a state $x \in X$ s.t. $\overline{\delta}(x, u) = x$ for some string $u \in (\Sigma - \hat{\Sigma})^* - \{\epsilon\}$. Now let $x = \delta(x_0, s)$ for some $s \in L(G)$. Then, observing that $\delta(x, u) = \overline{\delta}(x, u) = x$, it holds that $su^n \in L(G)$ for any $n \in \mathbb{N}$. In particular, considering the abstracted string $t := p(s)$, for any $N \in \mathbb{N}$ there is an $n \in \mathbb{N}$ s.t. $N \cdot (|t| + 1) < u := sv^n$. Hence, p cannot be an LPO which leads to contradiction.

"$\Leftarrow$": Assume that $\overline{G}$ only contains trivial SCCs and p is an observer for $L(G)$. To verify that p is an LPO, it has to be shown that for all u in (1), $|u| < N(|t| + 1)$ for some $N \in \mathbb{N}$. We prove that this statement is true for $N = |X|$ by induction. Initially, let $t_0 = \epsilon$ and $s_0 = \epsilon$. Since $\overline{G}$ only contains trivial SCCs, the longest string $u_0 \in (\Sigma - \hat{\Sigma})^*$ with $p(u_0) = \epsilon$ can have at most $|X| - 1$ events. Hence, $|u_0| < |X|(|t| + 1) = |X|$. Now assume that $|u_0\sigma_1 u_1 \cdots \sigma_k u_k| < |X|(|\sigma_1 \cdots \sigma_k| + 1)$ for some $k \in \mathbb{N}$ and $u_i \in (\Sigma - \hat{\Sigma})^*$, $\sigma_i \in \hat{\Sigma}$, and let $\sigma_1 \cdots \sigma_k \sigma_{k+1} \in p(L(G))$ for $\sigma_{k+1} \in \hat{\Sigma}$. Then an analogous argument as above shows that any $u_{k+1} \in (\Sigma - \hat{\Sigma})^*$ s.t. $u_0\sigma_1 \cdots u_k \sigma_{k+1} u_{k+1} \in L(G)$ cannot have more than $|X| - 1$ events. Hence, $|u_0\sigma_1 u_1 \cdots \sigma_k u_k \sigma_{k+1} u_{k+1}| < |X|(|\sigma_1 \cdots \sigma_k \sigma_{k+1}| + 1)$ $\square$

With the result in Proposition 3, the algorithmic verification of the LPO condition is performed as follows.

Algorithm 1. Input: automaton G, abstraction alphabet $\hat{\Sigma}$

```
if $p : \Sigma^* \to \hat{\Sigma}^*$ is not an observer for $L(G)$     1
    return false                                                 2
Compute $\overline{G}$                                               3
if $\overline{G}$ only contains trivial SCCs                         4
    return true                                                  5
else                                                             6
    return false                                                 7
```

Here, line 1 is accomplished by the observer verification in Proposition 1 that runs with a complexity $\mathcal{O}(|X|^3 \cdot |\delta|)$. Furthermore, line 3 takes a complexity of $\mathcal{O}(|\delta|)$, while line 4 is performed by Tarjan's algorithm (Hopcroft and Ullman, 1975) that is bounded by $\mathcal{O}(|\delta| + |X|)$. Together, the loop-preserving observer verification can be done in polynomial time with a complexity of $\mathcal{O}(|X|^3 \cdot |\delta|)$.

4.3 Verification Example

Fig. 3 illustrates the LPO verification in Algorithm 1 for the example automaton G and the abstraction alphabet $\hat{\Sigma}_1 = \{\alpha, \mathtt{a}\}$. The observer verification in line 1 yields a positive result. However, $\bar{G}_1$ that is computed in line 3 has a non-trivial SCC (states 2,3,4). Hence, the LPO verification terminates with a negative result in line 7.

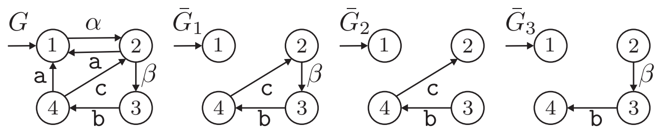

Fig. 3. Illustration of the LPO verification.

In contrast, choosing the abstraction alphabet $\hat{\Sigma}_2 = \{\alpha, \mathtt{a}, \beta\}$ (automaton G_2) or $\hat{\Sigma}_3 = \{\alpha, \mathtt{a}, \mathtt{c}\}$ (automaton G_3) leads to a projection p that fulfills LPO.

5. COMPUTATION OF NATURAL PROJECTIONS

A projection $p : \Sigma^* \to \hat{\Sigma}^*$ can be used for the abstraction-based language-diagnosability verification if Algorithm 1 returns a positive result. However, in case LPO fails, an appropriate extension of the abstraction alphabet has to be found in order to apply Theorem 2. In this section, we develop an algorithm that extends a given alphabet such that the loop-preserving observer condition is fulfilled.

In principle, it is desired to add as few events as possible to the abstraction alphabet in order to obtain small abstracted models. In (Feng and Wonham, 2010), an analogous problem is studied for the observer condition in (1). It is stated that there is no unique minimal extension and the problem of finding such minimal extension is NP-hard. In this paper, we point out that there is also no unique minimal extension to achieve the LPO property. Both extensions $\hat{\Sigma}_2$ and $\hat{\Sigma}_3$ for the example in Fig. 3 constitute minimal extensions of $\hat{\Sigma}_1$. Furthermore, we show that the problem of finding a minimal extension, denoted as the $\mathrm{MX_{LPO}}$ problem, is NP-hard.

5.1 Complexity of the LPO Extension

In order to prove that $\mathrm{MX_{LPO}}$ is NP-hard, we first investigate the *non-trivial extension problem* ($\mathrm{NTX_{LPO}}$) similar to (Feng and Wonham, 2010) and show that (1) $\mathrm{NTX_{LPO}}$ is polynomial-time reducible to $\mathrm{MX_{LPO}}$ and (2) $\mathrm{NTX_{LPO}}$ is NP-complete. Given an automaton G over the alphabet Σ and an abstraction alphabet $\hat{\Sigma} \subset \Sigma$, we say that $(G, \hat{\Sigma})$ has a non-trivial extension if there is a Σ' with $\hat{\Sigma} \subseteq \Sigma' \subset \Sigma$ such that $p' : \Sigma^* \to \Sigma'^*$ is an LPO. The goal is to determine for each pair $(G, \hat{\Sigma})$ if such extension exists. Then, we obtain the following result.

Lemma 4. $\mathrm{NTX_{LPO}}$ is polynomial-time reducible to $\mathrm{MX_{LPO}}$.

Proof. Let the algorithm EXT solve $\mathrm{MX_{LPO}}$ with the minimal extension $\Sigma' = EXT(G, \hat{\Sigma})$ for $(G, \hat{\Sigma})$. Then, the following polynomial-time algorithm decides $\mathrm{NTX_{LPO}}$:

```
Compute $\Sigma' = EXT(G, \hat{\Sigma})$                                1
if $\Sigma' = \Sigma$, then return false, else return true               2
```

In the next step, we show that $\mathrm{NTX_{LPO}}$ is NP-complete. On the one hand, $\mathrm{NTX_{LPO}}$ belongs to the set NP of nondeterministic polynomial-time problems (Sipser, 2006) since it can be verified in polynomial time if an alphabet Σ' is a non-trivial extension for a given pair $(G, \hat{\Sigma})$ using Algorithm 1. On the other hand, we develop a polynomial time reduction of the known NP-complete problem *monotone 3-in-1 SAT* (Schaefer, 1978) to $\mathrm{NTX_{LPO}}$.

We consider a boolean formula that is given in 3 conjunctive normal form (3cnf) over m variables $x_1, \ldots, x_m$:

$$\Phi(x_1, \ldots, x_m) = \bigwedge_{r=1}^{n} (x_{i_r} \vee x_{j_r} \vee x_{k_r}), \tag{3}$$

where each *clause* consists of three non-negated *literals* x_{i_r}, x_{j_r} and x_{k_r}, $i_r, j_r, k_r \in \{1, \ldots, m\}$. Then, Φ fulfills monotone 3-in-1 SAT if there is an assignment $(b_1, \ldots, b_m)$ such that $\Phi(b_1, \ldots, b_m)$ is true and exactly one literal in each clause is true.

Proposition 5. Monotone 3-in-1 SAT is polynomial-time reducible to $\mathrm{NTX_{LPO}}$, i.e., $\mathrm{NTX_{LPO}}$ is NP-complete.

Proof. We convert each formula Φ to an automaton G over an alphabet $\Sigma = \{\alpha, x_1, \ldots, x_m, c_1, \ldots, c_n\}$ such that the pair $(G, \hat{\Sigma})$ with $\hat{\Sigma} = \{c_1, \ldots, c_n\}$ has a non-trivial extension if and only if Φ fulfills monotone 3-in-1 SAT. To this end, we first construct an automaton $G_r = (X_r, \Sigma_r, \delta_r, x_{0,r})$ over the alphabet $\Sigma_r = \{\alpha, x_{i_r}, x_{j_r}, x_{k_r}, c_r\}$ for each clause $x_{i_r} \vee x_{j_r} \vee x_{k_r}$ such that the pair $(G_r, \hat{\Sigma}_r := \{c_r\})$ has the non-trivial extensions $\{c_r, x_{i_r}\}$, $\{c_r, x_{j_r}\}$ and $\{c_r, x_{k_r}\}$, while all other extensions are either not loop-preserving observers or trivial. Furthermore, we ascertain that an event $x_{i_r}, x_{j_r}, x_{k_r}$ belongs to the abstraction alphabet if its value is 1. Hence, a non-trivial extension for G_r exists if and only if $x_{i_r} \vee x_{j_r} \vee x_{k_r}$ is monotone 3-in-1 SAT. The appropriate automaton G_r is shown in Fig. 4 (a). It can be verified that only the extensions $\{c_r, x_{i_r}\}$, $\{c_r, x_{j_r}\}$, $\{c_r, x_{k_r}\}$ and $\{c_r, x_{i_r}, x_{j_r}, x_{k_r}, \alpha\}$ lead to an LPO, whereby the last extension is trivial. Furthermore, the case where two literals are equivalent is captured by the automaton in Fig. 4 (b). W.l.o.g. it is assumed that $x_{j_r} = x_{k_r}$ in this construction. If a clause contains only one literal, then that literal is simply 1.

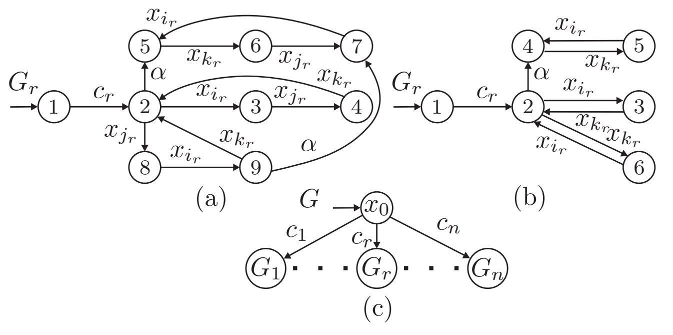

Fig. 4. (a) and (b) automaton G_r; (c) automaton G.

Then, the automaton $G = (X, \Sigma, \delta, x_0)$ is constructed as follows. We introduce a new state $x_0 \notin \bigcup_{r=1}^{n} X_r$ and define $X := \{x_0\} \cup \bigcup_{r=1}^{n} X_r$. Noting that $\Sigma = \bigcup_{r=1}^{n} \Sigma_r$, we define δ such that for all $r = 1, \ldots, n$, $\delta(x_0, c_r) = x_{0,r}$. Otherwise, δ follows the transition relations δ_r, $r = 1, \ldots, n$. That is, G is assembled from the automata G_r, where the initial state of each G_r is reached from the initial state of G by the event c_r as illustrated in Fig. 4 (c). Then, it is readily observed that a projection $p' : \Sigma^* \to \Sigma'^*$ with $\{c_1, \ldots, c_n\} \subseteq \Sigma'$ is an LPO for $L(G)$ if and only if each projection $p'_r : \Sigma_r^* \to (\Sigma_r \cap \Sigma')^*$ is an LPO for $L(G_r)$ for each $r = 1, \ldots, n$. But then, the construction of G_r from the clause $x_{i_r} \vee x_{j_r} \vee x_{k_r}$ implies that the alphabet Σ' is a non-trivial extension of $\hat{\Sigma} = \bigcup_{r=1}^{n} \hat{\Sigma}_r = \{c_1, \ldots, c_n\}$ for the pair $(G, \hat{\Sigma})$ if and only if each clause $x_{i_r} \vee x_{j_r} \vee x_{k_r}$, $r = 1, \ldots, n$, is monotone 1-in-3 SAT which is equivalent to Φ being monotone 1-in-3 SAT. Since the construction of G from Φ is polynomial in the number of variables and clauses, $\mathrm{NTX_{LPO}}$ is NP-complete. □

Finally, it follows that $\mathrm{MX_{LPO}}$ is NP-hard.

Theorem 6. $\mathrm{MX_{LPO}}$ is NP-hard.

Proof. A problem is NP-hard if an NP-complete problem is polynomial-time reducible to it (Sipser, 2006). Since $\mathrm{NTX_{LPO}}$ is NP-complete with Proposition 5 and $\mathrm{MX_{LPO}}$ is polynomial time reducible to $\mathrm{MX_{LPO}}$ according to Lemma 4, the statement follows.

5.2 Extension Algorithm

Considering Theorem 6, it is not expected to find a polynomial-time algorithm that computes a minimal extension for a given pair $(G, \hat{\Sigma})$. Hence, we propose to use a heuristic algorithm that extends the work by Feng and Wonham (2010) in order to determine practical alphabet extensions that meet the LPO property.

We first point out the three possible causes for a violation of the loop-preserving observer condition using the example automaton G in Fig. 5 (a). The quasi-congruence obtained for the abstraction alphabet $\hat{\Sigma} = \{\alpha, \beta, \gamma\}$ is indicated by the shaded boxes that also represent the equivalence classes of the corresponding quotient automaton. Then, it can be seen that the LPO condition is violated because of (i) a *silent transition* between different equivalence classes with the event c, (ii) a nondeterministic transition to two different equivalence classes with the event β and (iii) a non-trivial SCC with events in $\Sigma - \hat{\Sigma}$ (states 6, 7, 8). Here, the first two causes result in a violation of the observer condition and were already identified by Feng and Wonham (2010). The third cause is a direct implication of the loop-preserving observer condition. We now develop appropriate measures in order to extend the abstraction alphabet $\hat{\Sigma}$ such that the loop-preserving observer condition is fulfilled.

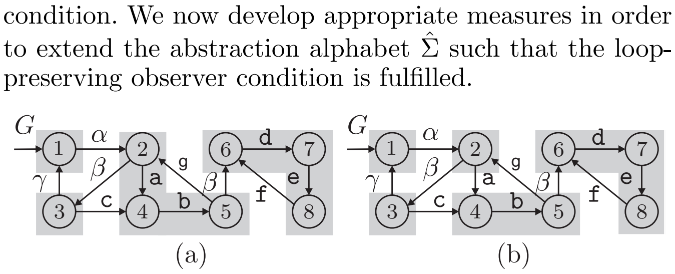

Fig. 5. (a) Violation of LPO; (b) LPO is fulfilled.

A solution for item (i) and (ii) is presented in (Feng and Wonham, 2010). All events that label silent transitions have to be added to $\hat{\Sigma}$. Furthermore, we use the function split : $Y \times 2^{\Sigma} \to \{\text{true}, \text{false}\}$ such that $\text{split}(y, \hat{\Sigma})$

verifies if the choice of the abstraction alphabet $\hat{\Sigma}$ leads to nondeterminism in state $y \in Y$.[1] Let $\mathcal{D}$ denote the set of deterministic automata. To address item (iii), we introduce the function acyclic : $\mathcal{D} \to \{\text{true}, \text{false}\}$ such that acyclic(G) determines if a given automaton $G \in \mathcal{D}$ only has trivial SCCs. Then, we adapt the algorithm in (Feng and Wonham, 2010) for the LPO computation.

Algorithm 2. Input: G, $\hat{\Sigma}$, set $\hat{\Sigma}' := \hat{\Sigma}$

```
while 1                                                                   1
 Compute $\mu^*$ for $(G, \hat{\Sigma}')$ and the quotient $G_{\mu^*,\hat{\Sigma}'}$                 2
 Compute $\Sigma_C := \{\sigma \in (\Sigma - \hat{\Sigma}') | \exists x \in X, u, u' \in$
 $(\Sigma - \hat{\Sigma}')^*$ s.t. $x = \delta(x, u\sigma u')\}$ (events in local loops)   3
 if $G_{\mu^*,\hat{\Sigma}'}$ is deterministic, no $\sigma_0$-transitions; $\Sigma_C = \emptyset$      4
   return $\hat{\Sigma}$                                                           5
 else
   Denote $\Sigma_s$ as all events that label silent transitions          6
   Set $\hat{\Sigma}' := \hat{\Sigma}' \cup \Sigma_s$                                                 7
   Compute $N := \{y \in Y | \exists \sigma \in \hat{\Sigma}'$ s.t. $|\eta(y, \sigma)| > 1\}$
   (states in $G_{\mu^*,\hat{\Sigma}'}$ with nondeterministic transitions)         8
   Compute $\Sigma_N := \bigcup_{x \in \mathrm{cp}^{-1}(N)} \{\sigma \in \Sigma - \hat{\Sigma}' | \delta(x, \sigma)!\}$
   (all events in cosets with potential nondeterminism)   9
   Set $\hat{\Sigma}' := \hat{\Sigma}' \cup \Sigma_N \cup \Sigma_C$                                       10
   for each $\sigma \in \Sigma_N \cup \Sigma_C - \Sigma_s$                                  11
     Compute $\bar{G}$ as in Proposition 3 for $\hat{\Sigma}' - \{\sigma\}$               12
     if $(\forall y \in N)$split$(y, \hat{\Sigma}' - \{\sigma\}) \wedge$ acyclic$(\bar{G})$        13
       Set $\hat{\Sigma}' := \hat{\Sigma}' - \{\sigma\}$                                         14
```

It is readily verified that the projection $p : \Sigma^* \to \hat{\Sigma}'^*$ is an LPO for $L(G)$ if Algorithm 2 terminates since all causes for an LPO violation are resolved. It is guaranteed that the algorithm terminates in at most $|\Sigma - \hat{\Sigma}|$ steps since at most $|\Sigma - \hat{\Sigma}|$ events can be added to the initial abstraction alphabet until the LPO condition is fulfilled. However, it is not ensured that a minimal abstraction is found, whereas reasonable abstractions can be computed in practice.

We finally apply Algorithm 2 to the example in Fig. 5 (a) with $\hat{\Sigma} = \{\alpha, \beta, \gamma\}$. Here, $\mathtt{c}$ is added to $\hat{\Sigma}'$ due to the silent transition (line 6). Furthermore, $\Sigma_N = \{\mathtt{a},\mathtt{b}\}$ and $\Sigma_C = \{\mathtt{d},\mathtt{e},\mathtt{f}\}$ (line 8 and 9). Then, successively, events from $\Sigma_N \cup \Sigma_C - \Sigma_s = \{\mathtt{a},\mathtt{b},\mathtt{d},\mathtt{e},\mathtt{f}\}$ are removed until the nondeterminism due to the event β and the loop with the states 6, 7, 8 are erased. Depending on the order of removal, a possible result is $\hat{\Sigma}' = \{\alpha, \beta, \gamma, \mathtt{a}, \mathtt{c}, \mathtt{f}\}$ (see Fig. 5 (b)).

6. CONCLUSION

In this paper, we studied natural projections with the loop-preserving observer property that are suitable for the abstraction-based language-diagnosability verification. We first showed that this property can be verified in polynomial time for a given natural projection. Then, we considered the case, where it is desired to extend the projection alphabet of a given natural projection in order to achieve the loop-preserving observer property. We proved that finding a minimal alphabet extension is NP-hard, and developed an algorithm that finds practical alphabet extensions in polynomial time. All algorithms presented in this paper are available in the open-source library `libFAUDES` for discrete event systems (lib, 2009).

[1] Please consult (Feng and Wonham, 2010) for a detailed description.

REFERENCES

(2009). libFAUDES – software library for discrete event systems. URL `http://www.rt.eei.uni-erlangen.de/FGdes/faudes/index.html`.

Cassandras, C.G. and Lafortune, S. (2006). *Introduction to Discrete Event Systems.* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Contant, O., Lafortune, S., and Teneketzis, D. (2006). Diagnosability of discrete event systems with modular structure. *Discrete Event Dynamic Systems: Theory and Applications*, 16, 9–17.

Debouk, R., Malik, R., and Brandin, B. (2002). A modular architecture for diagnosis of discrete event systems. In *Decision and Control, IEEE Conference on*, 417–422.

Feng, L. and Wonham, W. (2010). On the computation of natural observers in discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 20(1), 63–102.

Hopcroft, J.E. and Ullman, J.D. (1975). *The design and analysis of computer algorithms.* Addison-Wesley.

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

Qiu, W. and Kumar, R. (2006). Decentralized failure diagnosis of discrete event systems. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 36(2), 384–395.

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

Schaefer, T.J. (1978). The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, 216–226.

Schmidt, K. (2010a). Abstraction-based failure diagnosis for discrete event systems. *System & Control Letters*, 59, 42–47.

Schmidt, K. (2010b). Abstraction-based verification of codiagnosability for discrete event systems. *to appear in Automatica.* doi:10.1016/j.automatica.2010.06.010.

Sipser, M. (2006). *Introduction to the Theory of Computation.* 2nd edition, Course Technology.

Su, R. and Wonham, W.M. (2005). Global and local consistencies in distributed fault diagnosis for discrete-event systems. *Automatic Control, IEEE Transactions on*, 50(12), 1923–1935.

Wang, Y., Yoo, T.S., and Lafortune, S. (2007). Diagnosis of discrete event systems using decentralized architectures. *Discrete Event Dynamic Systems*, 17(2), 233–263.

Wong, K.C. and Wonham, W.M. (2004). On the computation of observers in discrete-event systems. *Discrete Event Dynamic Systems*, 14(1), 55–107.

Yoo, T.S. and Garcia, H.E. (2008). Diagnosis of behaviors of interest in partially-observed discrete-event systems. *System & Control Letters*, 57(12), 1023–1029.

Yoo, T.S. and Lafortune, S. (2002). Polynomial time verification of diagnosability of partially observed discrete-event systems. *Automatic Control, IEEE Transactions on*, 47(9), 1491–1495.