Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 654059, 10 pages
http://dx.doi.org/10.1155/2013/654059

Research Article

Verification of Opacity and Diagnosability for Pushdown Systems

Koichi Kobayashi and Kunihiko Hiraishi

School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa 923-1292, Japan

Correspondence should be addressed to Koichi Kobayashi; k-kobaya@jaist.ac.jp

Received 26 February 2013; Revised 28 April 2013; Accepted 30 April 2013

Academic Editor: Guiming Luo

In control theory of discrete event systems (DESs), one of the challenging topics is the extension of theory of finite-state DESs to that of infinite-state DESs. In this paper, we discuss verification of opacity and diagnosability for infinite-state DESs modeled by pushdown automata (called here pushdown systems). First, we discuss opacity of pushdown systems and prove that opacity of pushdown systems is in general undecidable. In addition, a decidable class is clarified. Next, in diagnosability, we prove that under a certain assumption, which is different from the assumption in the existing result, diagnosability of pushdown systems is decidable. Furthermore, a necessary condition and a sufficient condition using finite-state approximations are derived. Finally, as one of the applications, we consider data integration using XML (Extensible Markup Language). The obtained result is useful for developing control theory of infinite-state DESs.

1. Introduction

To extend control theory of discrete event systems (DESs) expressed by finite automata [1] to that of infinite-state DESs is one of the challenging topics. Infinite automata [2] and unbounded Petri nets [3, 4] are common models of infinite-state DESs. In this paper, we focus on a pushdown automaton (PDA) [5, 6], which is one of the standard classes of infinite automata. A PDA is a finite automaton having a stack with infinite length, and a state transition is decided by the input symbol (event) and the stack symbol in the top of the stack. The stack can be manipulated by the input symbol in which the empty symbol is included. Furthermore, a visibly pushdown automaton (VPA) has been proposed as a special class of PDAs [7, 8]. Input symbols in a VPA are composed of three kinds of symbols, that is, push manipulations, pop manipulations, and internal manipulations (see Section 2.1 for further details). It is easier to analyze a VPA than a PDA. Some examples are given as applications of PDAs and VPAs. In the modeling of a software system, a PDA is frequently used (see, e.g., [9]). Also in analysis of a cyber-physical system (CPS), which has recently attracted much attention, a PDA is used [10]. A CPS is a system featuring a combination between computer systems and physical systems [11]. To model the complicate behavior of such a CPS, it is important to use a PDA. In addition, as an application of a PDA, intrusion detection in the field of computer security has been studied so far [12]. As an application of a VPA, analysis of XML (Extensible Markup Language), which is one of the markup languages and is widely used in several fields [8, 13], has been studied so far [8, 13]. Thus a PDA and a VPA have many applications, and, developing theory of DESs using a PDA and a VPA is important.

For DESs modeled by PDAs and VPAs (called here pushdown systems), supervisory control [9, 14] and diagnosability verification [15] have been studied so far. Diagnosability verification is the problem of testing if the state reaches a given failure state under partial observations. In diagnosability verification, it has been proven in [15] that diagnosability of pushdown systems is in general undecidable and is decidable under the assumption that stack manipulations are completely observed. This assumption is very strong and is not practical. Furthermore, it is important to approximately verify diagnosability for pushdown systems without assumptions. On the other hand, in recent years, opacity in computer security has been studied in the framework of DESs [16–22]. Opacity aims to determine if a secret behavior in the system

is kept opaque to outsiders. In [17], it has been shown that opacity verification of infinite-state systems is undecidable. Also in [19], opacity verification of infinite-state systems has been discussed using approximate models. However, to our knowledge, opacity verification of pushdown systems has not been studied so far.

In this paper, we discuss verification of opacity and diagnosability for pushdown systems based on basic results in formal language theory [5, 6]. In opacity verification, it is proven that opacity of pushdown systems is in general undecidable. Furthermore, we clarify a condition such that opacity is decidable. In diagnosability verification, first, it is proven that if a part of observations in the system can be expressed as a regular language, that is, a language accepted by some finite automaton, then diagnosability is decidable. Next, a necessary condition for the pushdown system not to be diagnosable is derived based on finite-state overapproximations. In a similar way, a sufficient condition is also derived based on finite-state underapproximations. The proposed conditions enable us to verify diagnosability of a wider class of pushdown systems. Also, the relation among opacity verification, diagnosability verification, and the sensor selection problem is discussed. Finally, as one of the applications, we consider data integration of XML documents. In the case that documents in databases are expressed by the XML, one of the important problems is data integration, which is called here XML data integration. In particular, we discuss XML data integration with security considerations and show the effectiveness of the proposed approach. The conference paper [23] is a preliminary version of this paper. In this paper, we provide improved formulations and explanations, discussion on the computational complexity, and application to XML data integration.

This paper is organized as follows. In Section 2, first, the outline of pushdown systems is explained. Next, decidable problems and undecidable problems in PDAs and VPAs are explained. Finally, we introduce a finite-state overapproximation of PDAs. In Section 3, opacity verification is discussed. In Section 4, diagnosability verification is discussed, and some diagnosability conditions are derived. In Section 5, related topics are discussed. In Section 6, we consider an application of the proposed approach to XML data integration. In Section 7, we conclude this paper.

Notation. For a finite set Σ, let Σ^* denote a set of all finite strings, which consist of elements in Σ, including the empty string ε. Let Σ^ω denote a set of all infinite strings. Let $|\Sigma|$ denote the number of elements in Σ. The set $\mathcal{N}$ is defined as $\mathcal{N} := \{0, 1, 2, \ldots\}$.

2. Preliminaries

In this section, first, we introduce pushdown automata and their subclass. Next, undecidable problems in pushdown automata are explained. Furthermore, we introduce assumptions that will enable the undecidable problems to become decidable. Finally, a finite-state overapproximation used in diagnosability verification is explained.

2.1. Pushdown Systems. Consider the following pushdown automaton (PDA):

$$G = \left(Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F\right), \tag{1}$$

where Q is the finite set of states, $\Sigma \cup \{\varepsilon\}$ is the finite set of input symbols (events), Γ are the finite set of stack symbols, $\Delta = \Delta_{\text{push}} \cup \Delta_{\text{pop}} \cup \Delta_{\text{int}}$: transition relations, $q_0 \in Q$ are the initial state, and $Z_0 \in \Gamma$ is the initial stack symbol. $F \subseteq Q$: the set of final states.

In theory of DESs, the empty string ε is regarded as an unobservable event. In the PDA to be studied here, transition relations are composed of Δ_{push}, Δ_{pop}, and Δ_{int}, which are called here a "push" manipulation, a "pop" manipulation, and an "internal" manipulation, respectively. In the push manipulation, a particular symbol is pushed to the top of the stack. In the pop manipulation, the top of the stack is popped off. In the internal manipulation, the stack is not changed, and only the state is changed. More precisely, three manipulations are given as

$$\begin{aligned} \Delta_{\text{push}} &\subseteq (Q \times \Sigma \times \Gamma) \times Q, \\ \Delta_{\text{pop}} &\subseteq (Q \times \Sigma) \times \left(Q \times \left(\Gamma - \{Z_0\}\right)\right), \\ \Delta_{\text{int}} &\subseteq (Q \times \Sigma) \times Q, \end{aligned} \tag{2}$$

respectively. Functional forms of $\delta = (q, \sigma, \gamma, q') \in \Delta_{\text{push}}$, $\delta = (q, \sigma, q', \gamma) \in \Delta_{\text{pop}}$, and $\delta = (q, \sigma, q') \in \Delta_{\text{int}}$ are given as $\delta : (q, \sigma, \gamma) \to q'$, $\delta : (q, \sigma) \to (q', \gamma)$, and $\delta : (q, \sigma) \to q'$, respectively. If the state and the symbol sequence in the stack are uniquely determined for given initial state and input symbol, a given PDA is said to be deterministic.

In addition, if the state reaches some element of a subset $Q' \subseteq Q$ by an execution $u \in \Sigma^*$, then it is said that an execution u reaches Q'. Furthermore, we can naturally extend these definitions to an infinite execution. If the state reaches some element of Q' by one of the prefixes of an infinite execution $w \in \Sigma^\omega$, then it is said that the infinite execution w reaches Q'.

Let $\Sigma_o \subseteq \Sigma$ denote the set of observable events, and the set $\Sigma_{uo} := \overline{\Sigma}_o$ is the set of unobservable events. Let $L(G)$ denote the set of languages accepted by G. Then let $P(L(G))$ denote the set of sequences of observable events; that is, the following two relations hold:

$$\begin{aligned} P(\sigma) &= \sigma \quad \text{if } \sigma \in \Sigma_o, \\ P(\sigma) &= \varepsilon \quad \text{if } \sigma \in \Sigma_{uo}. \end{aligned} \tag{3}$$

Since $P(\sigma_1) = P(\sigma_2)$ is satisfied for the two unobservable events $\sigma_1, \sigma_2 \in \Sigma_{uo}$, two events σ_1, σ_2 are indistinguishable.

Next, we introduce visibly PDAs [7, 8] as a subclass of PDAs.

Definition 1. A PDA (1) is said to be a visibly PDA (VPA) if Σ comprises $\Sigma = \Sigma_{\text{push}} \cup \Sigma_{\text{pop}} \cup \Sigma_{\text{int}}$, where Σ_{push}, Σ_{pop}, and Σ_{int} are input symbols corresponding to Δ_{push}, Δ_{pop}, and Δ_{int}, respectively.

In VPAs, manipulations to the stack can be visualized from Σ.

We illustrate this with a simple example.

Example 2. Consider the VPA G in Figure 1, where

$$\begin{aligned} Q &= \{q_0, q_1, q_2\}, \\ \Sigma &= \{a, b, c, d\}, \\ \Gamma &= \{Z_0, X, Y\}, \\ \Delta_{\text{push}} &= \{(q_0, a, X, q_0), (q_0, a, Y, q_0), (q_0, a, X, q_1)\}, \\ \Delta_{\text{pop}} &= \{(q_1, b, q_1, X), (q_2, d, q_2, Y)\}, \\ \Delta_{\text{int}} &= \{(q_0, c, q_2)\}, \\ F &= \{q_1, q_2\} \subset Q. \end{aligned} \tag{4}$$

In Figure 1, $a/+X$ and $a/+Y$ express push manipulations, $b/-X$ and $d/-Y$ express pop manipulations, and c expresses an internal manipulation. Since $\Sigma = \{a, b, c, d\}$ comprises

$$\Sigma_{\text{push}} = \{a\}, \quad \Sigma_{\text{pop}} = \{b, d\}, \quad \Sigma_{\text{int}} = \{c\}, \tag{5}$$

we see that this automaton is a VPA. In addition, we have

$$L(G) = \{a^n b^n \mid n \geq 1\} \cup \{a^m c d^m \mid m \geq 0\}, \tag{6}$$

where $a^n b^n$ corresponds to the set of event sequences such that the state reaches q_1. $a^n c d^n$ corresponds to the set of event sequences such that the state reaches q_2.

Hereafter, a discrete event system expressed by a PDA and a VPA is called a DES-PDA and a DES-VPA, respectively. If it is not necessary to distinguish a DES-PDA and a DES-VPA, then these discrete event systems are called here pushdown systems (PDSs).

2.2. Decidable Problems and Undecidable Problems in Pushdown Automata. In this subsection, we explain decidable problems and undecidable problems in a language $L(G)$ accepted by PDA or VPA G, that is, a context-free language (CFL). See [5, 6] for further details.

Suppose that the two PDAs G_1 and G_2 are given. Then $L(G_1)$ and $L(G_2)$ are CFLs, and the following result is known.

Lemma 3. *The emptiness problem of an intersection of $L(G_1)$ and $L(G_2)$, that is, the problem of deciding if $L(G_1) \cap L(G_2) = \emptyset$ holds, is undecidable.*

This lemma follows from the fact that $L(G_1) \cap L(G_2)$ is not a CFL in general. The emptiness problem of an intersection of two CFLs is decidable under a given assumption.

Lemma 4. *Assume that either $L(G_1)$ or $L(G_2)$ is given as a regular language; that is, either G_1 or G_2 is given as a finite automaton. Then the emptiness problem of an intersection of $L(G_1)$ and $L(G_2)$ is decidable. In addition, this problem can be solved in PTIME.*

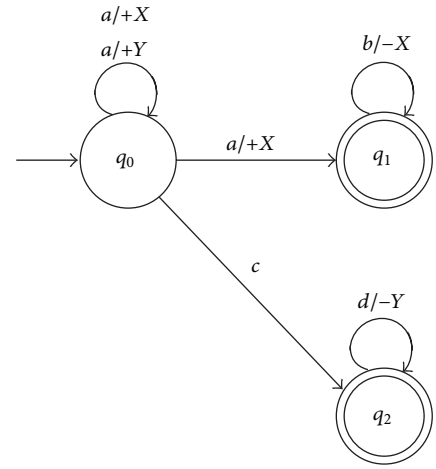

Figure 1: Example of VDPDAs G.

Under the assumption in Lemma 4, $L(G_1) \cap L(G_2)$ is a CFL, and the emptiness problem is decidable. This result in the emptiness problem of an intersection is used in diagnosability verification in Section 4.

Next, the following result on the inclusion problem has been obtained.

Lemma 5. *The inclusion problem of two PDAs, G_1 and G_2, that is, the problem of deciding if $L(G_1) \subseteq L(G_2)$ holds, is undecidable.*

For VPAs, the following result has been obtained [7].

Lemma 6. *The inclusion problem of two VPAs G_1 and G_2, that is, the problem of deciding if $L(G_1) \subseteq L(G_2)$ holds is decidable. In addition, this problem can be solved in EXPTIME.*

On the other hand, a class of deterministic PDAs such that the inclusion problem is decidable is called superdeterministic PDAs [24]. A deterministic VPA is a subclass of the superdeterministic PDAs. This fact is clear from the definitions in [7, 24]. As a result, the following result can be obtained.

Lemma 7. *Assume that G_1 and G_2 are given as a PDA and a deterministic VPA, respectively. Then the problem of deciding if $L(G_1) \subseteq L(G_2)$ is decidable. In addition, this problem can be solved in 2-EXPTIME.*

These results of the inclusion problem are used in opacity verification in Section 3.

2.3. Finite-State Overapproximation of Pushdown Automata. In this section, we explain the finite-state overapproximations of PDAs. Several approximations have been proposed so far (see, e.g., [25] for further details). In this paper, the following finite-state overapproximation proposed in [9] is used.

Definition 8. For a given PDA G of (1), the finite-state overapproximation of G is defined by

$$G_o = \left(Q, \Sigma_{.Q}, \Delta_o, q_0, F\right), \tag{7}$$

where $\Sigma_{.Q}$ and Δ_o are defined as follows:

(i) $\Sigma_{.Q} = \{\sigma \cdot q \mid \sigma \in \Sigma, q \in Q\}$.

(ii) Δ_o is given as $\Delta_o \subseteq (Q \times \Sigma_{.Q}) \times Q$. $\delta = (q, \sigma \cdot q', q') \in \Delta_o$ holds if and only if one of the following three conditions holds:

(1) $\delta = (q, \sigma, \gamma, q') \in \Delta_{\text{push}}$ holds for some $\gamma \in \Gamma$;

(2) $\delta = (q, \sigma, q', \gamma) \in \Delta_{\text{pop}}$ holds for some $\gamma \in \Gamma$;

(3) $\delta = (q, \sigma, q') \in \Delta_{\text{int}}$ holds.

Let G_s denote the PDA obtained by replacing $\sigma \in \Sigma$ with $\sigma.q \in \Sigma_{.Q}$. In the transitions of G_o, the conditions on the stack are ignored. So $L(G_s) \subseteq L(G_o)$ holds.

A finite-state overapproximation will be used in the topic on diagnosability (see Section 4.4 for further details).

3. Opacity Verification

This section looks at opacity verification of PDSs. First, the notion of opacity is defined. Next, after a condition for a PDS to be opaque is derived, a decidable class of PDSs is clarified.

3.1. Definition of Opacity. First, secret states are defined. Let $Q_s \subset F$ denote the set of secret states. Then we define the notion of opacity based on the definition in [20].

Definition 9. For the PDS G of (1), suppose that the set of secret states Q_s and the set of observable events Σ_o are given. Then the PDS is said to be opaque if for all $t \in L(G)$ there exists $s \in L(G) - \{t\}$ such that if t reaches Q_s, then s does not reach Q_s and $P(s) = P(t)$.

Several definitions of opacity have been proposed so far (see, e.g., [17, 20–22]). In this paper, the simplest definition in [20] is used.

3.2. Decidability of Opacity. Let V_1 denote the set of finite event sequences such that the state reaches Q_s. Let V_2 denote the set of finite event sequences such that the state reaches $Q - Q_s$. Then the opacity condition of PDSs can be derived as follows.

Theorem 10. *A PDS is opaque if and only if $P(V_1) \subseteq P(V_2)$ holds.*

Proof. If a PDS is opaque, then $P(V_1)$ is included in $P(V_2)$. Conversely, if $P(V_1) \subseteq P(V_2)$ holds, then for any $v_1 \in P(V_1)$, there exists $v_2 \in P(V_2)$ such that $v_1 = v_2$. So the PDS is opaque. □

From Lemma 5 and Theorem 10, we can obtain the following result immediately.

Theorem 11. *Opacity of a DES-PDA is in general undecidable.*

In addition, from Lemma 5 and Theorem 10, we can obtain the following result.

Theorem 12. *Opacity of a DES-VPA is in general undecidable.*

Proof. Even if V_1 and V_2 are languages accepted by VPAs, then $P(V_1)$ and $P(V_2)$ are in general given as languages accepted by PDAs [15]. Then, from Lemma 5 and Theorem 10, opacity of a DES-VPA is undecidable. □

From Theorems 11 and 12, we see that opacity of a DES-PDA and a DES-VPA are in general undecidable. On the other hand, from Lemma 6 and Theorem 10, we see that if observations of a DES-PDA is expressed by some VPA, then opacity of a DES-PDA may be decidable. Then we can obtain the following result.

Theorem 13. *Assume that $P(V_1)$ and $P(V_2)$ are languages accepted by VPAs, respectively. Then the problem of deciding if $P(V_1) \subseteq P(V_2)$ holds is decidable; that is, opacity of a PDS is decidable. In addition, this problem can be solved in EXPTIME.*

In addition, from Lemma 7 and Theorem 10, we can obtain the following result.

Theorem 14. *Assume that $P(V_1)$ and $P(V_2)$ are languages accepted by a PDA and a deterministic VPA, respectively. Then the problem of deciding if $P(V_1) \subseteq P(V_2)$ holds is decidable; that is, opacity of a PDS is decidable. In addition, this problem can be solved in 2-EXPTIME.*

From Theorem 14, we see that if $P(V_2)$ is given as a language accepted by a deterministic VPA, then the decidable condition is relaxed. Comparing Theorem 13 with Theorem 14, the inclusion problem in Theorem 14 is more difficult than that in Theorem 13. This is because in Theorem 14, $P(V_1)$ may be given as a language accepted by a PDA. If an overapproximation of a PDA can be derived as a VPA, then we can obtain a sufficient condition of a PDS to be opaque. In future work we plan to consider how to approximate a PDA by a VPA. On the other hand, in recent years, VPAs have been applied to several applications (see, e.g., [7–9]). Therefore, Theorem 13 and Theorem 14 will be useful.

Finally, we show a simple example.

Example 15. Consider the DES-VPA in Figure 1. Assume $d = b$. Suppose that $\Sigma_o = \{a, b\}$, $\Sigma_{uo} = \{c\}$, and $Q_s = \{q_2\}$ are given. Then we obtain

$$P\left(V_1\right) = P\left(V_2\right) = a^n b^n. \tag{8}$$

$P(V_1)$ and $P(V_2)$ are languages accepted by VPAs, respectively, and $P(V_1) = P(V_2)$ holds. Therefore, this system is opaque.

4. Diagnosability Verification

In this section, first, diagnosability for the PDS (1) is defined according to the definition in [15, 26, 27]. Next, after the existing results in [15] are explained, two kinds of diagnosability conditions are proposed.

4.1. Definition of Diagnosability. First, failure states are defined. Let $Q_f \subset Q$ denote the set of failure states. For Q_f, the following two assumptions are made: (i) $Q_f \subset F$ and (ii) if the state reaches the set Q_f, then the state stays within Q_f. Since in this paper we focus on whether some failure has occurred in the past or not, these assumptions are imposed. These assumptions also imply that any failure is not restored automatically. Let $L_f \subset L(G)$ denote the set of executions such that the state reaches some element in Q_s.

Next, the notion of diagnosability for the PDS (1) is defined according to [15, 26, 27].

Definition 16. For the PDS (1), suppose that the set of failure states Q_f and the set of observable events Σ_o are given. Then the PDS (1) is diagnosable if the following condition holds:

$$(\exists n \in \mathcal{N} - \{\infty\})\left(\forall s \in L_f\right)\left(\forall t \in \frac{L(G)}{s}\right) \\ |t| \geq n \Longrightarrow P^{-1}P(st) \cap L(G) \subseteq L_f, \tag{9}$$

where $L(G)/s := \{t \in \Sigma^* \mid st \in L(G)\}$, and $P^{-1}P(st)$ is the set of executions such that an observation is $P(st)$.

In this definition, $P^{-1}P(st) \cap L(G) \subseteq L_f$ implies that all executions such that an observation is $P(st)$ are included in L_f. So a failure can be detected from a finite observation. In other words, if there exists an unobservable infinite suffix, then the system is not diagnosable. Furthermore, deriving n in this definition is important, but this topic is not discussed in this paper. In [15], under some assumptions, a method for deriving n has been already discussed.

Hereafter, a construction method of a diagnoser is not focused on (see [15] for construction of a diagnoser), and a method to test diagnosability is considered.

4.2. Existing Results. First, we introduce a necessary and sufficient condition of the PDS (1) not to be diagnosable [15, 28].

Lemma 17. *For the PDS* (1)*, suppose that the set of failure states Q_f and the set of observable events Σ_o are given. Then the PDS is not diagnosable if and only if there exist two indistinguishable infinite executions w_1 and w_2 such that w_1 reaches Q_f while w_2 does not.*

Next, the existing results [15] on diagnosability verification are introduced.

Lemma 18. *Diagnosability of a DES-PDA is in general undecidable.*

Lemma 19. *Diagnosability of a DES-VPA is in general undecidable.*

These results can be obtained immediately, because the emptiness problem for an intersection of observations of infinite executions w_1 and w_2 such that w_1 reaches Q_f while w_2 does not is undecidable from Lemma 3.

In [15], the following result has been derived.

Lemma 20. *Assume $\Sigma_{uo} \subseteq \Sigma_{\text{int}}$. Then diagnosability of a DES-VPA is decidable.*

In Lemma 20, it is assumed that events corresponding to push and pop manipulations of the stack are observable. In [15], the diagnosability verification problem is reduced to the emptiness problem of Büchi automata under $\Sigma_{uo} \subseteq \Sigma_{\text{int}}$. So the property of pushdown automata is not considered. In this paper, as another approach, we consider diagnosability verification under other assumptions. Furthermore, approximate methods for verifying diagnosability of PDSs without assumptions are proposed.

4.3. Proposed Diagnosability Condition. First, based on Lemma 4, we propose a diagnosability condition of PDSs.

Let W_1 denote the set of finite event sequences such that the state reaches $Q - Q_f$. Let W_2 denote the set of finite event sequences such that the state reaches Q_f.

Then we obtain the following result.

Theorem 21. *Assume that either $P(W_1)$ or $P(W_2)$ is given as a language accepted by some finite automaton, that is, a regular language. Then diagnosability of a PDS is decidable. Furthermore, a PDS is not diagnosable if and only if $P(W_1) \cap P(W_2) \neq \emptyset$ holds. In addition, this problem can be solved in PTIME.*

Proof. From Lemma 4 and Lemma 17, this theorem is obtained straightforwardly. □

We show a simple example.

Example 22. Consider the DES-VPA in Figure 1 again. Suppose that $\Sigma_o = \{a, b, c\}$, $\Sigma_{uo} = \{d\}$, and $Q_f = \{q_2\}$ are given. Then we obtain

$$W_1 = a^n b^n, \quad W_2 = a^m c d^m, \quad n \geq 1,\ m \geq 0, \\ P(W_1) = a^n b^n, \quad P(W_2) = a^m c, \quad n \geq 1,\ m \geq 0. \tag{10}$$

Since $P(W_2)$ is a regular language, diagnosability of this DES-VPA is decidable. In fact, we see that $P(W_1) \cap P(W_2) \neq \emptyset$ does not hold, and the defect can be detected by observing the event c. This example is very simple. However, we note that diagnosability of this system is undecidable in the case using the existing result in [15]. This is because this system does not satisfy the assumption in Lemma 20; that is, the unobservable event d is not an internal manipulation and is a pop manipulation.

From this example, we can indicate that even if W_1, W_2 are CFLs, then $P(W_1)$, $P(W_2)$ are generally neither CFLs nor regular languages. In the above example, W_2 is a CFL, but $P(W_2)$ is a regular language. So we may consider to directly approximate $P(W_1)$, $P(W_2)$ by a regular language. From this viewpoint, we propose diagnosability conditions using the finite-state over/under-approximations.

4.4. Proposed Diagnosability Conditions Using Finite-State Approximations. Now, we show a diagnosability condition using a finite-state overapproximation in Definition 8.

Theorem 23. *Suppose that either $P(W_1)$ or $P(W_2)$ is approximated by a finite-state overapproximation in Definition 8. Let $\overline{P}_1$ and $\overline{P}_2$ denote approximated $P(W_1)$ and $P(W_2)$, respectively. Then a necessary condition for a PDS to be not diagnosable is that either $\overline{P}_1 \cap P(W_2) \neq \emptyset$ or $P(W_1) \cap \overline{P}_2 \neq \emptyset$ holds. In addition, this condition can be solved in PTIME.*

Proof. From Theorem 21, $P(W_1) \subseteq \overline{P}_1$, and $P(W_2) \subseteq \overline{P}_2$, we obtain the theorem immediately. □

Theorem 23 provides a necessary condition.

On the other hand, it is also important to obtain a sufficient condition. Then a finite-state underapproximation must be considered. A simple method to derive a finite-state underapproximation is that the length of the stack in a PDS is limited to a given finite length. See Example 25 below for further details. So we assume that a finite-state under-approximation of either $P(W_1)$ or $P(W_2)$ is given.

Then we obtain the following result.

Theorem 24. *Suppose that either $P(W_1)$ or $P(W_2)$ is approximated by a finite-state under-approximation. Let $\underline{P}_1$ and $\underline{P}_2$ denote approximated $P(W_1)$ and $P(W_2)$, respectively. Then a sufficient condition for a PDS to be not diagnosable is that either $\underline{P}_1 \cap P(W_2) \neq \emptyset$ or $P(W_1) \cap \underline{P}_2 \neq \emptyset$ holds. In addition, this condition can be solved in PTIME.*

Proof. From Theorem 21, $\underline{P}_1 \subseteq P(W_1)$, and $\underline{P}_2 \subseteq P(W_2)$, we obtain the theorem immediately. □

From Theorems 23 and 24, we see that diagnosability of a PDS can be approximately verified in PTIME. So Theorems 23 and 24 are simple but can be applied to several systems such as software systems.

Example 25. Suppose that $P(W_1)$ and $P(W_2)$ are given as

$$\begin{aligned} P(W_1) &= \left\{a^n b^n c^i \mid n \geq 1,\ i \geq 1\right\}, \\ P(W_2) &= \left\{a^i b^n c^n \mid n \geq 1,\ i \geq 1\right\}, \end{aligned} \tag{11}$$

respectively. Then $P(W_1) \cap P(W_2) = \{a^n b^n c^n \mid n \geq 1\} \neq \emptyset$ is obtained, and we see that this system is not diagnosable. However, $P(W_1) \cap P(W_2)$ is not a CFL (see [6] for further details).

Next, we verify diagnosability using Theorem 23. In this example, we approximate $P(W_1)$ by a regular language. The PDA G_1 accepting $P(W_1)$ is given by

$$\begin{aligned} G_1 &= (Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F), \\ Q &= \{q_0, q_1, q_2\}, \\ \Sigma &= \{a, b, c\}, \\ \Gamma &= \{Z_0, X\}, \\ \Delta &= \Delta_{\text{push}} \cup \Delta_{\text{pop}} \cup \Delta_{\text{int}}, \\ \Delta_{\text{push}} &= \{(q_0, a, X, q_0), (q_0, a, X, q_1)\}, \\ \Delta_{\text{pop}} &= \{(q_1, b, q_1, X)\}, \\ \Delta_{\text{int}} &= \{(q_1, c, q_2), (q_2, c, q_2)\}, \\ F &= \{q_2\} \subset Q. \end{aligned} \tag{12}$$

See also Figure 2. Consider to derive the finite-state overapproximation in Definition 8. Then $\Sigma_{\cdot Q}$ is given as

$$\begin{aligned} \Sigma_{\cdot Q} = \{&a \cdot q_0,\ a \cdot q_1,\ a \cdot q_2,\ b \cdot q_0,\ b \cdot q_1,\ b \cdot q_2, \\ &c \cdot q_0,\ c \cdot q_1,\ c \cdot q_2\}. \end{aligned} \tag{13}$$

Next, we derive Δ_o. From Δ_{push}, we obtain

$$\delta_1 = (q_0,\ a \cdot q_0,\ q_0), \quad \delta_2 = (q_0,\ a \cdot q_1,\ q_1). \tag{14}$$

From Δ_{pop}, we obtain

$$\delta_3 = (q_1,\ b.q_1,\ q_1). \tag{15}$$

From Δ_{int}, we obtain

$$\delta_4 = (q_1,\ c.q_2,\ q_2), \quad \delta_5 = (q_2,\ c.q_2,\ q_2). \tag{16}$$

So $\Delta_o = \{\delta_1, \delta_2, \delta_3, \delta_4, \delta_5\}$ is obtained. Thus we can obtain the finite-state overapproximation $\overline{G}_1$ of the PDA G_1 (see Figure 3). Using $\Sigma = \{a, b, c\}$, the language accepted by $\overline{G}_1$ is obtained by

$$L(\overline{G}_1) = \{a^* a b^* c c^*\}. \tag{17}$$

From the obtained $L(\overline{G}_1)$, we see that $P(W_1) \subseteq L(\overline{G}_1)$ holds. Furthermore, since $L(\overline{G}_1) \cap P(W_2) \neq \emptyset$ holds, the necessary condition in Theorem 23 is satisfied.

Next, we verify diagnosability using Theorem 24. By limiting the length of the stack in $P(W_1)$ to $n = 2$, the finite-state under-approximation $\underline{P}_1$ can be obtained as

$$\underline{P}_1 = \{aabbcc^*\} \subseteq P(W_1). \tag{18}$$

$\underline{P}_1$ is accepted by the finite automaton in Figure 4. Since $\underline{P}_1 \cap P(W_2) \neq \emptyset$ holds, the sufficient condition in Theorem 24 is satisfied.

5. Discussions

5.1. Relationship between Opacity and Diagnosability Verifications. Opacity and diagnosability are closely related concepts. We can interpret that opacity is the converse notion of diagnosability. However, decidability conditions are different.

In opacity verification, if $P(V_1)$ and $P(V_2)$ are languages accepted by some VPA, respectively, then opacity is decidable (see also Theorem 13). That is, opacity can be discussed only in the framework of VPAs.

On the other hand, in diagnosability verification, if either $P(W_1)$ or $P(W_2)$ is a language accepted by some finite automaton, then diagnosability is decidable (see Section 4.3). Thus approximations such as finite-state approximations described in Section 2.3 are required.

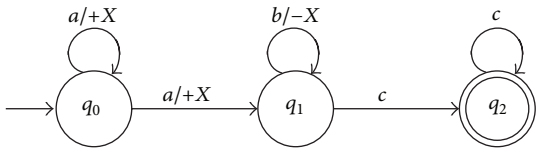

Figure 2: PDA G_1.

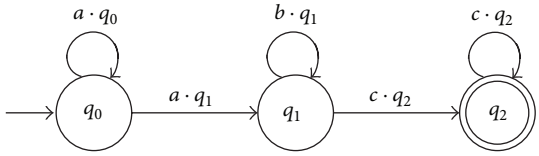

Figure 3: Finite-state overapproximation $\overline{G}_1$ of G_1.

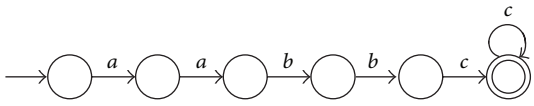

Figure 4: Finite automaton accepting $\underline{P}_1$.

From these results, the difference between opacity verification and diagnosability verification is clarified from the theoretical viewpoint. However, it is known that the computational complexity of the inclusion problem of VPAs is EXPTIME-complete and that of finite automata is PSPACE-complete [7]. Thus, from the computational viewpoint, it may be desirable to use finite-state approximations in opacity verification.

5.2. Optimal Sensor Selection. The optimal sensor selection problem, that is, the problem of minimizing the number of sensors which observe events, such that the system is diagnosable has been discussed in [29, 30]. Further, in opacity, the sensor selection problem has been discussed in [18]. One of the trivial solutions to the sensor selection problem in opacity is that the number of sensors should be zero, but this is not practical. So it is important to consider the optimal sensor selection problem such that the system is opaque and diagnosable simultaneously under $Q_s \neq Q_f$. Then we can derive an optimal solution of this problem according to the results in Sections 4 and 3. This is one of the challenges to be undertaken in the future, that is, to develop an efficient algorithm.

5.3. Extension to Higher-Order Pushdown Systems. Higher-order pushdown automata (HPAs) [31, 32] have been proposed as one of the extensions of PDAs. HPAs are defined by using higher-order stacks, that is, a nested "stack of stacks" structures. HPAs are closely related to infinite graph theory and higher-order logic programming. Decidable problems and undecidable problems in HPAs are basically similar to those in PDAs. Thus our proposed approach will be applied to HPAs. In fact, the conditions that diagnosability of discrete event systems modeled by HPAs is decidable have been obtained in [15]. Since one of the conditions is $\Sigma_{uo} \subseteq \Sigma_{\text{int}}$, the conditions obtained can be regarded as a natural extension of Lemma 20.

6. Application to XML Data Integration

In this section, we consider XML data integration as an application. The XML (Extensible Markup Language) is a markup language that defines a set of rules for encoding data and documents and is widely used in several fields such as data and documents on the Web and medical databases. In cases where documents in databases are expressed by the XML, one of the important problems is data integration, which is called here XML data integration. In XML data integration, first, XML documents are extracted from multiple databases. Next, the extracted XML documents are integrated as one XML document. Here, we assume the existence of a database where the integrated XML document is indistinguishable from its extracted components. The XML document in such a database is called here the target XML document. Then we consider the following problems:

Problem 1. Suppose that an integrated XML document and a target XML document are given. Then

(i) can the integrated XML document be indistinguishable from the target XML document?

(ii) by masking particular data in the integrated XML document, can the integrated XML document be indistinguishable from the target XML document?

These problems are important from the viewpoint of security. In addition, this problem is closely related to opacity verification of PDSs in Section 3. This is because in recent years, a method for expressing XML documents as a VPA has been studied in, for example, [8, 13].

Hereafter, the relation between VPAs and XML documents will be explained. Next, by using a simple example, we explain how to solve the above problems.

6.1. Relation between Pushdown Systems and XML Documents. First, an example of XML documents expressing information about books is shown as follows:

```
<book>
  <title>
    Introduction to Discrete Event Systems
  </title>
  <author> C. G. Cassandras </author>
  <author> S. Lafortune </author>
  <ISBN> 978-0-387-33332-8 </ISBN>
</book>
```

The XML scheme in the above example is based on the example in [13]. In addition, the above example implies information about the book of [1]. <book>, <title>, <author>, and

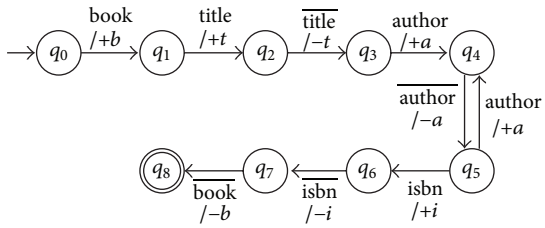

Figure 5: VPA S_1 expressing the XML document (information about books).

<ISBN> are open tags. </book>, </title>, </author>, and </ISBN> are close tags. Actual data such as "978-0-387-33332-8" is called a local symbol. Since in this paper we focus on XML schemas, we omit local symbols. The above XML document can then be expressed as the VPA $S_1 = (Q, \Sigma, \Gamma, \Delta_{\text{push}} \cup \Delta_{\text{pop}} \cup \Delta_{\text{int}}, q_0, Z_0, F)$, where

$$\begin{aligned} Q &= \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}, \\ \Sigma &= \left\{\text{book}, \overline{\text{book}}, \text{title}, \overline{\text{title}}, \text{author}, \overline{\text{author}}, \text{isbn}, \overline{\text{isbn}}\right\}, \\ \Gamma &= \{Z_0, a, b, i, t\}, \\ \Delta_{\text{push}} &= \{(q_0, \text{book}, b, q_1), (q_1, \text{title}, t, q_2), \\ &\quad (q_3, \text{author}, a, q_4), (q_5, \text{author}, a, q_4), \\ &\quad (q_5, \text{isbn}, i, q_6)\}, \\ \Delta_{\text{pop}} &= \left\{\left(q_2, \overline{\text{title}}, q_3, t\right), \left(q_4, \overline{\text{author}}, q_5, a\right), \right. \\ &\quad \left.\left(q_6, \overline{\text{isbn}}, q_7, i\right), \left(q_7, \overline{\text{isbn}}, q_8, b\right)\right\}, \\ \Delta_{\text{int}} &= \emptyset, \\ F &= \{q_8\} \subset Q. \end{aligned} \tag{19}$$

Close tags are expressed by $\overline{\cdot}$. See also Figure 5. Thus a given XML document can be expressed by a VPA.

6.2. XML Data Integration with Security Considerations. Next, suppose that a VPA expressing information about publishers is given as S_2 in Figure 6, where pub, pname, and addr imply "publisher," "publisher name," and "address," respectively. Consider how to integrate the VPA S_1 with the VPA S_2. One of the simple methods is to connect S_1 with S_2 in series. Then we can obtain the VPA S_3 in Figure 7.

Here, suppose that a target XML document is given as the VPA in Figure 8, where "#" implies a wild-card event or a wild-card stack symbol. We consider Problem 1 (i). That is, we must check if $L(S_3) \subseteq L(T)$ is satisfied. This is the same as opacity verification. In this example, $L(S_3) \subseteq L(T)$ is not satisfied, because by observing "country," the VPA S_3 is distinguishable from the VPA T. Therefore, a solution of Problem 1 (i) is "no." In other words, the integrated XML document is not opaque. From Theorem 13, it is assumed that this problem is decidable in general cases. We remark that in this case, $\Sigma = \Sigma_o$ holds.

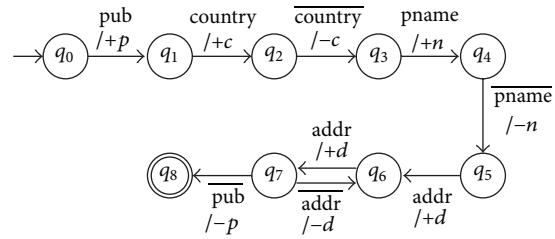

Figure 6: VPA S_2 expressing information about publishers.

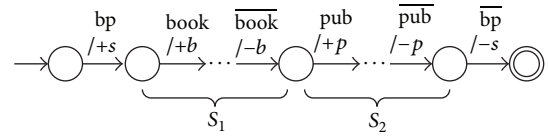

Figure 7: VPA S_3 obtained by integrating S_1 with S_2.

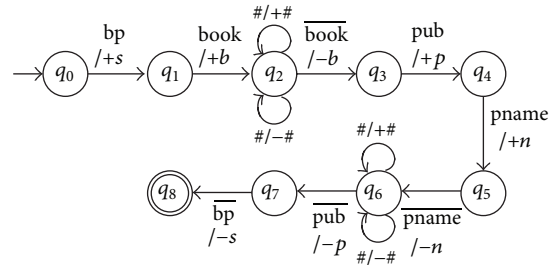

Figure 8: VPA T expressing a target XML document.

Next, consider Problem 1 (ii). Suppose that the set of unobservable events Σ_{uo} is given as $\Sigma_{uo} = \{\text{country}\}$. Then the VPA S_3 is indistinguishable from the VPA T. Therefore, a solution of Problem 1 (i) is "yes." In other words, we can say that the integrated XML document becomes opaque by masking "country." Also in general cases, this problem is decidable. On the other hand, it is one part of the future work, that is, to develop an efficient algorithm for finding masked events. Since this problem is the converse notion of the optimal sensor selection problem for diagnosability, there is a possibility that the existing result on sensor selection can be applied.

7. Conclusion and Future Work

In this paper, opacity and diagnosability of discrete event systems expressed by pushdown automata (called here pushdown systems, PDSs) have been discussed based on formal language theory. In opacity verification, we have not only proven that opacity of a PDS is in general undecidable, but also characterized the condition such that opacity is decidable. We have also clarified that depending on a class of observations, the computational complexity is different. In diagnosability verification, we have proposed a new diagnosability condition, based on the fact that an intersection of a context-free language and a regular language is derived as a context-free language. In addition, we have derived diagnosability conditions using finite-state over/under-approximations of observations. Also we have

clarified that the proposed diagnosability conditions can be solved in PTIME. In particular, since diagnosability conditions using approximations can be applied to a general class of PDSs and can be efficiently solved, several applications will be able to be considered. Finally, as an application, we have considered XML data integration. The obtained result is valuable as the basis of verification of PDSs and will be the first step toward development of control theory for infinite-state discrete event systems.

In future work, there are many open problems, for example, implementation using a model checker and decentralized diagnosis [33]. In addition, the result on diagnosability analysis of unbounded Petri nets has been obtained in [4]. Also, for recursive tile systems, which are a class of infinite discrete event systems, the result on opacity and diagnosability has been obtained in [34]. It is important to clarify the relation between our result and these results. Finally, in order to show the effectiveness of our proposed method, it is important to consider several applications.

Acknowledgment

This work was partially supported by Grant-in-Aid for Young Scientists (B) 23760387.

References

[1] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, Springer, 2nd edition, 2008.

[2] W. Thomas, "A short introduction to infinite automata," in *Developments in Language Theory*, vol. 2295 of *Lecture Notes in Computer Science*, pp. 130–144, 2002.

[3] M. P. Cabasino, A. Giua, S. Lafortune, and C. Seatzu, "Diagnosability analysis of unbounded Petri nets," in *Proceedings of the Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, pp. 1267–1272, 2009.

[4] M. P. Cabasino, A. Giua, S. Lafortune, and C. Seatzu, "A new approach for diagnosability analysis of Petri nets using verifier nets," *IEEE Transactions on Automatic Control*, vol. 57, no. 12, pp. 3104–3117, 2012.

[5] J.-M. Autebert, J. Berstel, and L. Boasson, "Context-free languages and pushdown automata," in *Handbook of Formal Languages*, vol. 1, pp. 111–174, Springer, 1997.

[6] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 3rd edition, 1979.

[7] R. Alur and P. Madhusudan, "Visibly pushdown languages," in *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pp. 202–211, ACM, 2004.

[8] V. Kumar, P. Madhusudan, and M. Viswanathan, "Visibly pushdown automata for streaming XML," in *Proceedings of the 16th International Conference on World Wide Web*, pp. 1053–1062, 2007.

[9] K. Hiraishi and P. Kucera, "Applications of DES theory to verification of software components," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 92, no. 2, pp. 604–610, 2009.

[10] N. Saeedloei and G. Gupta, "Verifying complex continuous real-time systems with coinductive CLP(R)," in *Proceedings of the 4th International Conference on Language and Automata Theory and Applications*, vol. 6031 of *Lecture Notes in Computer Science*, pp. 536–548, 2010.

[11] E. Lee, "Cyber physical systems: design challenges," in *Proceedings of the 11th International Symposium on Object Oriented Real-Time Distributed Computing*, pp. 363–369, 2008.

[12] D. Wagner and D. Dean, "Intrusion detection via static analysis," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 156–168, 2001.

[13] A. Thomo and S. Venkatesh, "Rewriting of visibly pushdown languages for XML data integration," *Theoretical Computer Science*, vol. 412, no. 39, pp. 5285–5297, 2011.

[14] C. Griffin, "A note on the properties of the supremal controllable sublanguage in pushdown systems," *IEEE Transactions on Automatic Control*, vol. 53, no. 3, pp. 826–829, 2008.

[15] C. Morvan and S. Pinchinat, "Diagnosability of pushdown systems," in *Proceedings of the Haifa Verification Conference 2009*, vol. 6405 of *Lecture Notes in Computer Science*, pp. 21–33, 2011.

[16] E. Badouel, M. Bednarczyk, A. Borzyszkowski, B. Caillaud, and P. Darondeau, "Concurrent secrets," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 17, no. 4, pp. 425–446, 2007.

[17] J. W. Bryans, M. Koutny, L. Mazare, and P. Y. A. Ryan, "Opacity generalised to transition systems," *International Journal of Information Security*, vol. 7, no. 6, pp. 421–435, 2008.

[18] F. Cassez, J. Dubreil, and H. Marchand, "Synthesis of opaque systems with static and dynamic masks," *Formal Methods in System Design*, vol. 40, no. 1, pp. 88–115, 2012.

[19] J. Dubreil, "Opacity and abstractions," in *Proceedings of the 1st International Workshop on Abstractions for Petri Nets and Other Models of Concurrency*, 2009.

[20] A. Saboori and C. N. Hadjicostis, "Notions of security and opacity in discrete event systems," in *Proceedings of the 46th IEEE Conference on Decision and Control*, pp. 5056–5061, 2007.

[21] A. Saboori and C. N. Hadjicostis, "Verification of infinite-step opacity and complexity considerations," *IEEE Transactions on Automatic Control*, vol. 57, no. 5, pp. 1265–1269, 2012.

[22] A. Saboori and C. N. Hadjicostis, "Verification of K-step opacity and analysis of its complexity," *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 3, pp. 549–559, 2011.

[23] K. Kobayashi and K. Hiraishi, "On opacity and diagnosability in discrete event systems modeled by pushdown automata," in *Proceedings of the 8th IEEE International Conference on Automation Science and Engineering*, pp. 658–663, 2012.

[24] S. A. Greibach and E. P. Friedman, "Superdeterministic PDAs: a subcase with a decidable inclusion problem," *Journal of the Association for Computing Machinery*, vol. 27, no. 4, pp. 675–700, 1980.

[25] M. Mohri and M. J. Nederhof, "Regular approximation of context-free grammars through transformation," in *Robustness in Language and Speech Technology*, J.-C. Junqua and G. van Noord, Eds., chapter 6, pp. 153–163, 2000.

[26] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. C. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 40, no. 9, pp. 1555–1575, 1995.

[27] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. C. Teneketzis, "Failure diagnosis using discrete-event models," *IEEE Transactions on Control Systems Technology*, vol. 40, no. 2, pp. 105–124, 1996.

[28] T. Jeron, H. Marchand, S. Pinchinat, and M.-O. Cordier, "Supervision patterns in discrete event systems diagnosis," in *Proceedings of the 8th Workshop on Discrete Event Systems*, 2006.

[29] S. Jiang, R. Kumar, and H. E. Garcia, "Optimal sensor selection for discrete-event systems with partial observation," *IEEE Transactions on Automatic Control*, vol. 48, no. 3, pp. 369–381, 2003.

[30] T.-S. Yoo and S. Lafortune, "NP-completeness of sensor selection problems arising in partially observed discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 47, no. 9, pp. 1495–1499, 2002.

[31] A. Carayol and S. Wöhrle, "The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata," in *Proceedings of the Foundations of Software Technology and Theoretical Computer Science*, vol. 2914 of *Lecture Notes in Computer Science*, pp. 112–123, 2003.

[32] T. Knapik, D. Niwinski, and P. Urzyczyn, "Higher-order pushdown trees are easy," in *Proceedings of the 5th International Conference on Foundations of Software Science and Computation Structures*, vol. 2303 of *Lecture Notes in Computer Science*, pp. 205–222, 2002.

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

[34] S. Chédor, C. Morvan, S. Pinchinat, and H. Marchand, "Analysis of partially observed recursive tile systems," in *Proceedings of the 11th International Workshop on Discrete Event Systems*, pp. 265–271, 2012.