Diagnosis of Pushdown Systems

Christophe Morvan, Sophie Pinchinat

HAL Id: inria-00337614
https://inria.hal.science/inria-00337614

Submitted on 7 Nov 2008
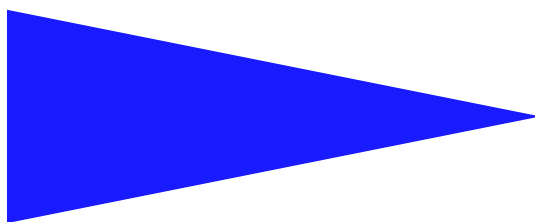
DIAGNOSIS OF PUSHDOWN SYSTEMS

CHRISTOPHE MORVAN, SOPHIE PINCHINAT

Diagnosis of Pushdown Systems

Christophe Morvan, Sophie Pinchinat

Systèmes communicants
Projets S4 et Vertecs

Abstract: Diagnosis problems of discrete-event systems consist in detecting unobservable defects during system execution. For finite-state systems, the theory is well understood and a number of effective solutions have been developed. For infinite-state systems, however, there are only few results, mostly identifying classes where the problem is undecidable.

We consider higher-order pushdown systems and investigate two basic variants of diagnosis problems: the *diagnosability*, which consists in deciding whether defects can be detected within a finite delay, and the *bounded-latency problem*, which consists in determining a bound for the delay of detecting defects.

We establish that the diagnosability problem is decidable for arbitrary sub-classes of higher-order visibly pushdown systems provided unobservable events leave the stacks unchanged. For this case, we present an effective algorithm. Otherwise, we show that diagnosability becomes undecidable already for first-order visibly pushdown automata. Furthermore, we establish that the bounded-latency problem for higher-order pushdown systems is as hard as deciding finiteness of a higher-order pushdown language. This is in contrast with the case of finite-state systems where the problem reduces to diagnosability.

Key-words: Pushdown systems, Visibly pushdown systems, Partial observation, Diagnosis.

(Résumé : tsvp)

Diagnostic des Systèmes à Piles

Résumé : Les problèmes de diagnostic des systèmes à événements discrets examinent la détection de défauts inobservables au cours de l'exécution du système. Pour les systèmes à nombre d'états fini, la théorie est déjà bien maîtrisée, et de nombreuses solutions effectives ont été développées. En revanche, le cas des systèmes à nombre d'états infini n'a fait l'objet que de peu d'études, exhibant le plus souvent des problèmes indécidables.

Nous considérons les systèmes à piles d'ordre supérieur et étudions deux problèmes élémentaires de diagnostic : la *diagnosticabilité*, problème pour lequel il s'agit de décider si les défauts seront détectés en un temps fini, et le problème de *la latence bornée*, pour lequel on souhaite déterminer si le décalage temporel entre l'occurrence d'un défaut et sa détection est borné.

Nous établissons que la diagnosticabilité est décidable pour toute sous-classe de systèmes à piles d'ordre supérieur dès lors que que tout événement inobservable ne modifie l'état des piles. Dans ce cas, nous présentons des algorithmes. Autrement, nous montrons que la diagnosticabilité est indécidable pour la sous-classe des systèmes dits "visibly pushdown", un classique de la littérature. En outre, nous établissons que le problème de la latence bornée pour les systèmes à piles d'ordre supérieur est au moins aussi difficile que celui de décider la finitude des langages à piles d'ordre supérieur. Ce dernier résultat est en opposition avec le cas des systèmes à nombre d'états fini pour lesquels le problème de la latence bornée se réduit à celui de diagnosticabilité.

Mots clés : Systèmes à piles, "Visibly pushdown systems", Observation partielle, Diagnostic.

Contents

1 Introduction

Absolute knowledge of the actual execution of a computer driven system is not practicable. The so-called *partial observation* approaches provide abstraction-based settings where the system executions analysis is more realistic. Relative to this matter, diagnosis is a fertile topic (*e.g.* [7, 9, 10, 13])

In diagnosis problems, one aims at constructing a device, the *diagnoser*, intended to observe the system executions on-line via sensors, and to detect their potential defects. Sensors are not formally described, but they are instead modeled by partial observation abilities which split information about the system execution into observable and unobservable ones; the former represent what the sensors have gathered.

Discrete-event systems are basic models of systems, for which executions consist in sequences of atomic events; events are typed either *observable* or *unobservable*. Observing an execution yields the sub-sequence of observable events, called the *observation* of this execution; two different executions may yield the same observation. The defects of executions are given by a property: the set of executions that have these defects. Defects cannot be directly observed on the execution in general,

An observation is an input of the diagnoser. The corresponding output depends on the actual knowledge emanating from this observation. Sensitive situations, called *equivocalness*, arise when the observation is justified by different executions that disagree on the defects property. Equivocalness therefore precludes a real-time response of the diagnoser to defect occurrences. As originally advocated by [13], response delays must be considered, bringing two basic variants of diagnosis issues: the *diagnosability* and the *bounded latency*.

Diagnosability is a qualitative property of the diagnoser which can ensure a finite response delay for any faulty execution under observation; it ratifies the completeness of the diagnoser. On a quantitative angle of view, the bounded-latency property can ensure a uniform bound of the response delay.

For finite-state systems, the theory is well understood and a number of effective solutions have been developed [13, 10, 9]. For infinite-state systems, however, we are not aware of any result.

In this paper, we consider pushdown systems, arising from the configuration graph of higher-order pushdown automata [11]. We establish that the diagnosability problem is decidable for arbitrary sub-classes of higher-order visibly pushdown systems [1, 8] provided unobservable events leave the stacks unchanged. Under this assumption, we present an effective algorithm. Otherwise, we show that diagnosability is undecidable already for classical visibly pushdown automata. Furthermore, we show that the bounded-latency problem for higher-order pushdown systems is as hard as deciding the finiteness of higher-order pushdown languages. This is in contrast with the case of finite-state systems where the problem reduces to diagnosability.

Technical content. Effective solutions in partial observation settings crucially relies on the closure under projection property of languages. Additionally, diagnosability is dually equivalent to the existence of an infinite path in an infinite-state model obtained as a product of structures [9]. Monadic second-order logic can be used to express this property, and its model-checking is decidable [12, 6]. Rather, lacks of closure properties under projection and intersection of the models cause computational limitations in diagnosis problems.

First-order pushdown systems are closed under projection, at the price of loosing determinism, but they are notoriously not closed under product; hence, diagnosability cannot be decided in general (Theorem 4.1). Seeking adequate sub-classes, we consider visibly pushdown systems, presentable by visibly pushdown automata whose input alphabet uniformly dictates the stack operations. Any collection of visibly pushdown automata over a fixed alphabet is closed under product [1]. Moreover, we show that visibly pushdown languages are closed under projections which erase only symbols corresponding to so-called "internal" transitions, i.e. transitions that leave the stack unchanged. This latter condition raises a class of first-order pushdown systems where diagnosability and bounded-latency problems are decidable (Theorems 5.2 and 4.7).

Following the same lines as in the first-order case, we next extend our results to higher-order pushdown systems: we concentrate on higher-order visibly pushdown sub-classes [8]. Although the sub-classes of higher-order *visibly pushdown* languages have somehow been

discarded due to their many redhibitory weaknesses (they are neither closed under concatenation, nor under iteration, and cannot be determinized), these sub-classes still offer nice features for diagnosis issues. First, they are closed under intersection. Second, similarly to first-order visibly pushdown languages, they are closed under projections which abstract only symbols corresponding to internal transitions. We take advantage of these two closure properties to describe an effective solution for diagnosability (Theorem 5.2). On the other hand, the bounded-latency problem for higher-order pushdown systems is more involved. Indeed, this problem is as hard as deciding the finiteness of higher-order pushdown languages (by an easy generalization of the first-order case of Proposition 4.4). For real-time higher-order pushdown languages, the finiteness problem is decidable (Theorem 5.4), but the question is open for the general case, and seems difficult [4].

The paper is organized as follows: In Section 2.2, we explain diagnosis problems. First-order pushdown systems are presented in Section 3 with the visibly pushdown sub-classes. Section 4 contains the main results for the diagnosis of first-order pushdown systems, and in Section 5, we investigate how previous results generalize to higher-order.

2 Diagnosis problems

2.1 Mathematical notations and definitions

For any set E, we denote by 2^E its powerset, and the complement of a subset $B \subseteq E$ is written $\overline{B}$. For any natural number n, we write $[n] := \{1, 2, 3, \ldots, n\}$. Given an alphabet Σ (a set of symbols), we denote by Σ^* and Σ^ω the sets of finite and infinite words over Σ respectively. We use the typical elements $u, u', v, \ldots$ for Σ^*, and ε for the empty word. We use $w, w_1, \ldots$ for elements of Σ^ω. For $u \in \Sigma^*$, $|u|$ denotes its length, and for $v \in \Sigma^* \cup \Sigma^\omega$, we write $u \preceq v$ to indicate that u is a prefix of v.

Definition 2.1. A *discrete-event system (*DES*)* is a structure

$$\mathcal{S} = \langle \Sigma, S, s^0, \delta, Prop, [\![.]\!] \rangle,$$

where Σ is an alphabet, S is a set of states and $s^0 \in S$ is the initial state, $\delta : S \times \Sigma \to S$ is a (partial) transition function, and $Prop$ is a set of propositions and $[\![.]\!] : Prop \to 2^S$ is an interpretation of the propositions.

An *execution* of $\mathcal{S}$ is a word $u = a_1 a_2 \ldots a_n \in \Sigma^*$ such that there exists a sequence of states $s_0, s_1, \ldots, s_n$ such that $s_0 = s^0$ and $\delta(s_{i-1}, a_i) = s_i$ for all $1 \leq i \leq n$. A execution u *reaches* a subset $S' \in S$ whenever $\delta(s^0, u) \in S'$, by extending δ to $S \times 2^{\Sigma^*}$. We naturally extend these definitions to *infinite* executions; in particular, an infinite execution $w \in \Sigma^\omega$ reaches S' if one of its prefixes reaches S'.

A proposition m *marks* the (elements of the) set $[\![m]\!]$, and an execution *reaches* m if it reaches $[\![m]\!]$.

2.2   An overview on diagnosis

Informally, diagnosis is about synthesis: one aims at constructing a device, a *diagnoser*, intended to work on-line together with the system. While the system executes, the diagnoser collects input data via sensors and outputs a verdict on the actual execution.

In classic diagnosis, the sensors are not formally described, but instead simulated in a partial observation framework: the set of events Σ is partitioned into Σ_o and $\overline{\Sigma_o}$ composed of *observables* and *unobservables* respectively; words $\theta, \theta_1, \ldots$ over Σ_o are *observations*. The canonical projection of Σ onto Σ_o is written π_{Σ_o}, or π when Σ_o is understood; it extends to Σ^* by erasing unobservables in words. An execution u *matches* an observation θ whenever $\pi(u) = \theta$. Two executions u and u' are *indistinguishable* if they match the same observation.

Observations are the inputs of the diagnoser. Regarding the outputs, *faulty* executions of particular interest (as opposed to *safe* ones) are distinguished *a priori* by means of a proposition $f \in Prop$: an execution u is *faulty* if $\delta(s^0, u) \in [\![f]\!]$. Moreover, we require that $[\![f]\!]$ is a *trap*: $\delta([\![f]\!], a) \subseteq [\![f]\!]$, for all $a \in \Sigma$. In this way, *e.g.* executions that contain a faulty event can be adequately described; we refer to [7] for a comprehensive exposition.

An instance of a diagnosis problem is a triplet composed of a DES, $\mathcal{S} = \langle \Sigma, S, s^0, \delta, Prop, [\![.]\!] \rangle$, an alphabet of observables, Σ_o, and a proposition, f. The associated diagnoser is a structure $\mathcal{D} := \langle \Sigma_o, 2^S, I^0, \hat{\delta}, diag \rangle$ whose states are *information sets*: an information set I is the smallest subset of states reached by a set of executions of the form $\pi(u) \cap \Sigma^* \Sigma_o$, for some execution u. The initial state is $I^0 := \{s^0\}$, the transition function, $\hat{\delta} : 2^S \times \Sigma \to 2^S$, is the extension of δ to sets of states in a canonical way[1], and the output function $diag$ is defined as follows. Given an information set I, three cases exist: (a) all states of I are marked by f; (b) no state is marked; and otherwise (c) where I is *equivocal*. Formally, $diag : 2^S \to \{(a), (b), (c)\}$ with

$$diag(I) := \begin{cases} (a) & \text{if } I \subseteq [\![f]\!], \\ (b) & \text{if } I \cap [\![f]\!] = \emptyset, \\ (c) & \text{otherwise.} \end{cases}$$

By extension, an observation θ is *equivocal* if $\hat{\delta}(I^0, \theta)$ is equivocal, otherwise θ is *clear*; the empty observation is clear since $I^0 = \{s^0\}$ is not equivocal. Moreover, θ is *faulty-clear* if it is clear and $\hat{\delta}(I^0, \theta)$ is in case (a).

$\mathcal{D}$ may be infinite-state in general (if $\mathcal{S}$ is infinite-state). However, its computation can be avoided by simulating it on-the-fly, storing the current information set I, and updating this object on each observable step of the system. While the synthesis of the diagnoser is not necessary, analyzing its behaviour is crucial: in particular, because equivocalness (case (c)) precludes the real-time detection of a fault, latencies to react are tolerated. *Diagnosability* is a qualitative property of the diagnoser which can ensure a finite latency for any observation of a faulty execution; it ratifies the completeness of the diagnoser. From a quantitative angle of view, the *bounded-latency* property can ensure a uniform bound on the latencies.

---

[1] $\hat{\delta}(I, a) := \bigcup_{s \in I} \delta(s, \overline{\Sigma_o}^* a)$
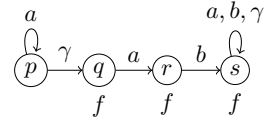
2.3 Diagnosability

In accordance with [13], we use the following definition (where the parameters Σ_o and f are understood).

Definition 2.2. A discrete-event system is *diagnosable* if every infinite observation of an infinite faulty execution has a clear finite prefix.

Remark that diagnosability considers only infinite executions that do not *diverge*, where an infinite executions diverges if it as an unobservable infinite suffix. In other words, we are only interested in fair behaviours of the system w.r.t. observability.

Notice that in a diagnosable system, safe executions may yield arbitrarily long equivocal observations. To illustrate this, consider the finite-state system below with alphabet $\{a, b, \gamma\}$, the initial state p, and the unobservable faulty event γ. The infinite observation a^ω, obtained from the unique safe execution a^ω, loops infinitely in the equivocal information set $\{p, q, r\}$. Yet, this system is diagnosable since a faulty execution yields an infinite observation with the clear prefix $a^n b$.

Theorem 2.3. **[13]** *Diagnosability of finite-state discrete-event systems is decidable.*

2.4 Bounded latency

The *latency* of a diagnosable system is the minimal number of additional observation steps that is needed to detect a faulty execution.

Definition 2.4. Let $\mathcal{S} = \langle \Sigma, S, s^0, \delta, Prop, [\![.]\!] \rangle$ be a DES, Σ_o be an alphabet of observables, and $f \in Prop$ such that $[\![f]\!]$ is a trap. A *just-faulty-clear* observation is a faulty-clear observation whose strict prefixes are equivocal.

Given a equivocal observation θ, the *latency* for θ is

$$\ell(\theta) := \max \left\{ |\vartheta| \mid \theta\vartheta \text{ is just-faulty-clear} \right\}$$

$\mathcal{S}$ is *bounded-latency* if there exists $N \in \mathbb{N}$ such that $\ell(\theta) \leq N$, for every equivocal observation θ. The least such N is the *bounded-latency value*.

The bounded-latency value of the example above is 2: any equivocal observation is of the form a^n (with $n \in \mathbb{N}$), and its just-faulty-clear extensions are $a^n b$ and $a^n ab$, yielding $\ell(a^n) = 2$.

Bounded-latency systems are diagnosable, but the converse does not hold in general, unless the systems are finite-state [9].

Remark 2.5. Note that two DES $\mathcal{S}_1$ and $\mathcal{S}_2$ with the same set of executions and such that any execution u reaches $[\![f]\!]$ in $\mathcal{S}_1$ if, and only if, it reaches $[\![f]\!]$ in $\mathcal{S}_2$, have the same diagnosability and bounded-latency properties (with the same bounded-latency value, if any).

3 Pushdown systems

Pushdown automata are finite-state machines that use a stack as an auxiliary data structure (see for example [2]). Pushdown systems are configuration graphs of pushdown automata; they are infinite-state in general.

3.1 Ordinary pushdown automata

Definition 3.1. A *pushdown automaton (*PDA*)* is a structure

$$\mathcal{A} = (\Sigma, \Gamma, Q, q_0, F, \Delta) \text{ where}$$

Σ and Γ are finite alphabets of respectively *input* and *stack* symbols, Q is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is a set of final states, and $\Delta \subseteq Q \times (\Gamma \cup \{\varepsilon\}) \times (\Sigma \cup \{\varepsilon\}) \times Q \times \Gamma^*$ is the set of transition. We use $p, q, \ldots$ (resp. $X, Y, \ldots$, and $U, V, W, \ldots$) for typical elements of Q (resp. Γ, and Γ^*).

Without loss of generality, we assume Δ in *normal form*: (1) *pop* transitions of the form $(p, X, a, q, \varepsilon)$ pop the top symbol of the stack, (2) *push* transitions of the form $(p, \varepsilon, a, q, X)$ push a symbol on top of the stack, and (3) *internal* transitions of the form $(p, \varepsilon, a, q, \varepsilon)$ leave the stack unchanged.

The PDA $\mathcal{A} = (\Sigma, \Gamma, Q, q_0, F, \Delta)$ is *deterministic* if: (1) $\forall (p, X, a) \in Q \times \Gamma \times \Sigma \cup \{\varepsilon\}$, $|\Delta(p, X, a)| \leq 1$, and (2) $\forall (p, X, a) \in Q \times \Gamma \times \Sigma$, $\Delta(p, X, \varepsilon) \neq \emptyset$ implies $\Delta(p, X, a) = \emptyset$. $\mathcal{A}$ is *real-time* if $\Delta \subseteq Q \times (\Gamma \cup \{\varepsilon\}) \times \Sigma \times Q \times \Gamma^*$. A *configuration* of $\mathcal{A}$ is a word $qU \in Q\Gamma^*$; the *initial configuration* is $q_0\varepsilon$, and a configuration qU is *final* if $q \in F$. Transitions (between configurations) are elements of $Q\Gamma^* \times \Sigma \cup \{\varepsilon\} \times Q\Gamma^*$: there is a transition $(qU, a, q'U')$ whenever there exists $(p, X, a, q, V) \in \Delta$ with $U = WX$ and $U' = WV$. A finite *run* of $\mathcal{A}$ is a finite sequence $r = q_0 U_0 a_1 q_1 U_1 a_2 \ldots a_n q_n U_n$ such that $U_0 = \varepsilon$ is initial, and $(q_i U_i, a_i, q_{i+1} U_{i+1})$ is a transition, for all $0 \leq i < n$. We say that $a_1 a_2 \ldots a_n$ is *the word of* r, or that r is a *run on* $a_1 a_2 \ldots a_n$. The run is *accepting* if $q_n U_n$ is final. The *language accepted* by $\mathcal{A}$ is $L(\mathcal{A}) \subseteq \Sigma^*$, the set of words $u \in \Sigma^*$ such that there is an accepting run on u.

Proposition 3.2. [2] *Any* PDA *is equivalent to a real-time* PDA. *The construction is effective.*

PDA accept *context-free languages (*CF *languages)*, while *deterministic* PDA yield a proper subclass of *deterministic* CF languages which contains all regular languages. Moreover, CF languages are closed under union, concatenation, and iteration, and their emptiness is decidable. Closure under intersection is more involved.

Proposition 3.3. [2] *The emptiness problem of an intersection of deterministic* CF *languages is undecidable.*

A *pushdown system (*PD *system)* [14][2] $\mathcal{S}$ is the configuration graph of a real-time PDA $\mathcal{A}$; $\mathcal{S}$ is then *presented* by $\mathcal{A}$. If $\mathcal{A}$ is deterministic, $\mathcal{S}$ is a DES. Notice that if in $\mathcal{S}$, each

---

[2] We here use "system" instead of "process" as to identify the objects with DES.

proposition interpretation is a *regular* set of configurations – in the usual sense of "regular", when configurations are seen as words –, then $\mathcal{S}$ can be presented by a PDA, and the monadic second-order logic, MSO, properties can be evaluated.

Theorem 3.4. [12] *Model-checking a* PD *system w.r.t.* MSO-*properties is decidable.*

By Proposition 3.3, PD systems are not closed under product[3], which causes limitations in effective methods for their analysis, and in particular regarding diagnosis (see Section 4). We consider more friendly sub-classes of PDA: the *visibly pushdown automata* [1].

3.2 Visibly pushdown automata

Visibly pushdown automata are PDA with restricted transition rules: whether a transition is push, pop, or internal depends only on its input letter.

Definition 3.5. A *visibly pushdown automaton (*VPA*)* is a pushdown automaton $\mathcal{A} = (\Sigma, \Gamma, Q, q_0, F, \Delta)$, where $\perp \in \Gamma$ is a special bottom-stack symbol, and whose input alphabet and transition relation are partitioned into $\Sigma := \Sigma_{push} \cup \Sigma_{pop} \cup \Sigma_{int}$, and $\Delta := \Delta_{push} \cup \Delta_{pop} \cup \Delta_{int}$ respectively, with the constraints that

$$\left\{ \begin{array}{llllllllll} \Delta_{push} & \subseteq & Q & \times & \{\varepsilon\} & \times & \Sigma_{push} & \times & Q & \times & (\Gamma \setminus \{\perp\}) \\ \Delta_{pop} & \subseteq & Q & \times & \Gamma & \times & \Sigma_{pop} & \times & Q & \times & \{\varepsilon\} \\ \Delta_{int} & \subseteq & Q & \times & \{\varepsilon\} & \times & \Sigma_{int} & \times & Q & \times & \{\varepsilon\} \end{array} \right.$$

The transitions in Δ_{int} hence leave the stack unchanged. The sub-alphabet Σ_{int} is the *internal alphabet.*

Theorem 3.6. [1] *Any* VPA *is equivalent to a deterministic* VPA *over the same alphabet. The construction is effective.*

Regarding the terminology, VPA present *visibly pushdown systems (*VP *systems)*, and accept *visibly pushdown languages (*VP *languages)*. A language L is a $[\Sigma_{int}]$-VP *language* if it is accepted by some VPA whose internal alphabet is Σ_{int}.

According to [1], any family of VP languages with a fixed partition $\Sigma_{push}, \Sigma_{pop}, \Sigma_{int}$ of the input alphabet is a Boolean algebra. In particular the synchronous product $\mathcal{A}_1 \times \mathcal{A}_2$ of VPA is well-defined, and accepts the intersection of the respective languages.

3.3 Projection of visibly pushdown languages

Projections of languages on a sub-alphabet are central operations for partial observation issues; we recall that the class of CF languages is projection-closed. We examine properties of the projection over the class of VP languages.

---

[3]usual synchronous product.

Proposition 3.7. *Any* CF *language is the projection of a* [∅]*-*VP *language.*

Proof. Let L be accepted by a PDA $\mathcal{A} = (\Sigma, \Gamma, Q, q_0, F, \Delta)$. Without loss of generality, we can assume that $\mathcal{A}$ is real-time, and that Δ is in normal form (see Definition 3.1).

Define the VPA $\mathcal{A}' := ([\Sigma, \{\tau_{pop}\}, \emptyset], \Gamma \cup \{Z\}, Q', q_0, F, \Delta')$, where

- Z and τ_{pop} are fresh symbols,

- $Q' := Q \cup \{q_{pop}\} \cup \{q_X \mid X \in \Gamma\}$, and

- Δ' is defined by:

$$\begin{aligned} (p, \varepsilon, a, q, X) \in \Delta'_{push} \quad &\text{if } (p, \varepsilon, a, q, X) \in \Delta, \\ (p, \varepsilon, a, q_{pop}, Z) \in \Delta'_{push} \quad &\text{if } (p, \varepsilon, a, q, \varepsilon) \in \Delta, \\ \left.\begin{array}{l}(q_X, \varepsilon, a, q_{pop}, Z) \in \Delta'_{push} \\ (p, X, \tau_{pop}, q_X, \varepsilon) \in \Delta'_{pop}\end{array}\right\} \quad &\text{if } (p, X, a, q, \varepsilon) \in \Delta. \end{aligned}$$

Furthermore, for each $q \in Q$, $(q_{pop}, Z, \tau_{pop}, q, \varepsilon) \in \Delta'_{pop}$.

It is easy to verify that $\mathcal{A}'$ accepts the interleavings of words of L and words of $\{\tau_{pop}\}^*$, which concludes the proof. □

Because there exist non-deterministic CF languages and since every VP language is deterministic (Theorem 3.6), VP languages are not closed under projections.

Proposition 3.8. *The projection of a* [Σ_{int}]*-*VP *language onto* Σ'^*, *with* $\overline{\Sigma'} \subseteq \Sigma_{int}$, *is a* [$\Sigma_{int}$]*-*VP *language. The construction is effective.*

Proof. Let L be a [Σ_{int}]VP language, with alphabet $\Sigma := \Sigma_{push} \cup \Sigma_{pop} \cup \Sigma_{int}$, and let $\mathcal{A} = (\Sigma, \Gamma, Q, q_0, F, \Delta)$ be a VPA which accepts L.

We use a standard "ε-closure" of the transitions of $\mathcal{A}$. Let us write $p \Rightarrow p'$ whenever there exists $(p, \varepsilon, a, p', \varepsilon) \in \Delta_{int}$ with $a \in \overline{\Sigma'}$.

We defined the VPA $\pi(\mathcal{A})$ by

$$\pi(\mathcal{A}) := (\Sigma_{push} \cup \Sigma_{pop} \cup (\Sigma_{int} \setminus \overline{\Sigma'}), \Gamma, Q, q_0, F', \Delta')$$

where

$$\begin{cases} (p, a, X, q) \in \Delta'_{push} \text{ if } (p', a, X, q) \in \Delta_{push} \text{ and } p \Rightarrow^* p', \text{ for some } p' \in Q, \\ \Delta'_{pop} \text{ and } \Delta'_{int} \text{ are defined similarly, and} \\ p \in F' \text{ if } p \Rightarrow^* p' \text{ and } p' \in F. \end{cases}$$

It is easy to check that $L(\pi(\mathcal{A})) = \pi(L)$.   □

4   Diagnosis problems of pushdown systems

We show that diagnosability of arbitrary deterministic PD systems is undecidable. Next, we focus on VP systems whose diagnosability is also undecidable in general, unless unobservable transitions leave the stack unchanged.

4.1 Undecidability of diagnosability

Theorem 4.1. *Diagnosability of deterministic* PD *systems is undecidable.*

This theorem is a corollary of Proposition 3.3 and the following construction (see also Lemma 4.2). Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two deterministic PDA over Σ_1 and Σ_2 respectively, and let $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \{i_1, i_2, \#\}$, with fresh symbols $\#$, i_1 and i_2. For $i = 1, 2$, let $\mathcal{A}_i^{\#}$ be a deterministic PDA which accepts $L(\mathcal{A}_i)\#\Sigma^*$, the set of words $i_i u \# v$ where $u \in L(\mathcal{A}_i)$. Note $\mathcal{A}_1^{\#} \oplus \mathcal{A}_2^{\#}$ the PDA depicted on the right. Mark all configurations of $\mathcal{A}_1^{\#} \oplus \mathcal{A}_2^{\#}$ whose state is in $\mathcal{A}_1^{\#}$ by f; $[\![f]\!]$ is a regular set and a trap, by construction. Notice that $\mathcal{A}_1^{\#} \oplus \mathcal{A}_2^{\#}$ is deterministic.
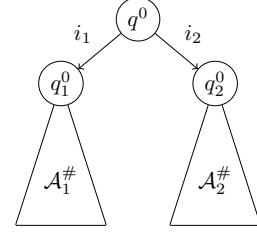
Lemma 4.2. *The* PD *system presented by* $\mathcal{A}_1^{\#} \oplus \mathcal{A}_2^{\#}$ *is diagnosable w.r.t.* $\Sigma \setminus \{i_1, i_2\}$ *and* f *if, and only if,* $L(\mathcal{A}_1) \cap L(\mathcal{A}_2) = \emptyset$.

Proof. Let $\mathcal{S}$ be presented by $\mathcal{A}_1^{\#} \oplus \mathcal{A}_2^{\#}$. We use the following alternative characterization of diagnosability; its equivalence with Definition 2.2 is immediate.

Lemma 4.3. *A* DES *is not diagnosable w.r.t. the set of observables* Σ_o *and the proposition* f *if, and only if, there exist two indistinguishable infinite executions* w_1 *and* w_2 *such that* w_1 *reaches* f *while* w_2 *does not.*

Consider $w_1 := i_1 u \#^{\omega}$ indistinguishable from $w_2 := i_2 u \#^{\omega}$ with $u \in L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$. Thus w_1 reaches f but w_2 does not. Apply Lemma 4.3 to conclude. Reciprocally, if $\mathcal{S}$ is not diagnosable, then by Lemma 4.3, there exist indistinguishable infinite executions w_1 and w_2 such that only w_1 reaches f; necessarily, $w_1 i_1 u \# w$ and $w_2 = i_2 u \# w$ for some u, entailing $u \in L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$, which concludes the proof. □

4.2 The bounded-latency problem

The VP system of Figure 4.1, is diagnosable for ι and γ unobservable and f (black) marking executions that contain the faulty event γ. Indeed, every maximal execution is finite, and its last event is ▲ if, and only if, γ has occurred. However, the system is not bounded latency since ▲ can occur after arbitrarily long executions.

We explain here the intrinsic complexity of the bounded-latency problem which is shown to somehow contain the finiteness problem for languages, as stated by Proposition 4.4. Notice however that the statement becomes trivial for CF languages, as their finiteness is decidable [2]. This Proposition however will be very useful in Section 5.

Proposition 4.4. *Let* $\mathcal{L}$ *be a class of languages which contains finite languages, and which is closed under concatenation and union. If the language finiteness problem is undecidable on* $\mathcal{L}$, *so is the bounded-latency problem on the class of systems whose set of executions is in* $\mathcal{L}$.
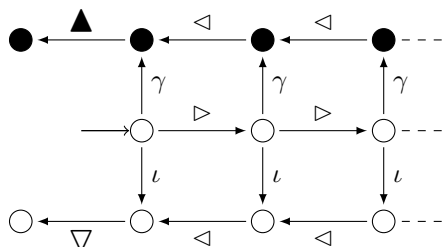
Figure 4.1: Diagnosable but not bounded latency.

Proof. Let $L \in \mathcal{L}$ (say with alphabet Σ). We build $\mathcal{S}_L$ such that $\mathcal{S}_L$ is bounded-latency if, and only if, L is finite. The event set of $\mathcal{S}_L$ is $\Sigma \cup \{i_1, i_2, \#, \$\}$, with fresh symbols i_1, i_2, $\#$, and $\$$. $\mathcal{S}_L$ has two components $L\#$ and $L\$$. By construction, $\mathcal{S}_L \in \widetilde{\mathcal{L}}$.

With the unobservables i_1 and i_2, and f which marks the configurations of the $L\$$ component, $\mathcal{S}_L$ is diagnosable because events $\#$ or $\$$ always eventually occur along any execution. It is easy to see that $\mathcal{S}_L$ is bounded-latency if, and only if, L finite.
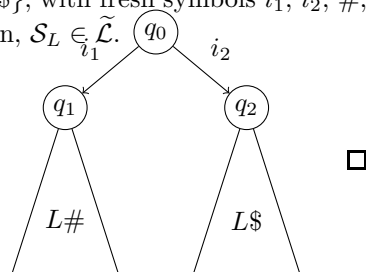
4.3   The case of visibly pushdown systems

In this section we address the diagnosability and the bounded-latency problems for VP systems. We show that their diagnosability is undecidable in general, unless the unobservables belong to the internal alphabet (Theorem 4.5). Under the latter hypothesis, bounded-latency can also be decided (Theorem 4.7); for both problems we draw an algorithm.

We use the terminology $[\Sigma_{int}]$-VPA to put the emphasis on the internal alphabet Σ_{int} of the automata.

Theorem 4.5. *(a) Diagnosability of* VP *systems is undecidable.*
(b) Diagnosability w.r.t. a set of observables Σ_o and a proposition f is decidable over any class of $[\Sigma_{int}]$-VP systems whenever $\overline{\Sigma_o} \subseteq \Sigma_{int}$ and f marks a regular set of configurations.

Proof. Point (a) is an immediate corollary of Theorem 4.1 and Proposition 3.7: if the diagnosability of VP systems was decidable, so would it be for ordinary PD systems; this contradicts the undecidability result of Theorem 4.1. Indeed, diagnosability w.r.t. Σ_o and f of a PD system presented by a PDA $\mathcal{A}$ is equivalent to the diagnosability w.r.t. Σ_o and f of the VP system presented by a VPA whose Σ_o-projection is $\mathcal{A}$ and whose internal alphabet is empty (Proposition 3.7).

Regarding Point (b), let $\mathcal{S}$ be a VP system, and consider an alphabet of observables Σ_o such that $\overline{\Sigma_o} \subseteq \Sigma_{int}$, and a proposition f which marks a regular set of configurations of $\mathcal{A}$. We sketch an algorithm to decide the diagnosability of $\mathcal{S}$ w.r.t. Σ_o and f; the proposed method extends the solution of [10] for finite-state systems. Let the VPA $\mathcal{A} = (\Sigma, \Gamma, Q, q_0, F, \Delta)$ represent $\mathcal{S}$. We can assume $\mathcal{A}$ deterministic[4], and, using standard techniques, we can assume that $[\![f]\!] = F \times \Gamma^*$. We consider the (non-deterministic) $[\Sigma_{int} \setminus \overline{\Sigma_o}]$-VPA $\pi(\mathcal{A}) \times \pi(\mathcal{A})$ (over the stack alphabet $\Gamma \times \Gamma$), obtained from the Σ_o-projection explained in the proof of Proposition 3.7, and the standard product of VPA [1]. From now on, we write $\overline{F}$ for $Q \setminus F$.

Lemma 4.6. *The* VPA $\pi(\mathcal{A}) \times \pi(\mathcal{A})$ *with initial state* (q_0, q_0) *and final states* $F \times \overline{F}$ *accepts the equivocal observations.*

Proof. Consider an equivocal observation θ. By definition, there exists two indistinguishable executions u_1 and u_2, such that u_1 and u_2 match θ, but only u_1 reaches f (i.e. is in $L(\mathcal{A})$). By determinism, the unique runs of $\mathcal{A}$ on u_1 and u_2 respectively synchronize in a run of $\pi(\mathcal{A}) \times \pi(\mathcal{A})$ on θ, which necessarily reaches a configuration in $(F \times \overline{F}) \times (\Gamma \times \Gamma)^*$.

Reciprocally, assume an accepting run r of $\pi(\mathcal{A}) \times \pi(\mathcal{A})$ on some $\theta \in \Sigma_o^*$. Because r can be tracked as a run of $\mathcal{A}$, θ is an observation. Moreover, r decomposes into a pair of runs r_1 and r_2 in $\pi(\mathcal{A})$ (on θ) which end in F and in $\overline{F}$ respectively. Now, r_1 and r_2 yield two runs in $\mathcal{A}$ respectively accepting and rejecting, and whose respective words match θ; θ is therefore equivocal. $\square$

In $\pi(\mathcal{A}) \times \pi(\mathcal{A})$, an infinite run remaining in the set of configurations $(F \times \overline{F}) \times (\Gamma \times \Gamma)^*$ denotes an infinite observation which has no clear prefix. By Lemma 4.3, this equivalently rephrases as "the system is not diagnosable". Since $(F \times \overline{F}) \times (\Gamma \times \Gamma)^*$ is regular the existence of such a run is expressible in MSO, and can be decided (Theorem 3.4).

More precisely, we have to check that the graph satisfies the formula: $\exists X \mathrm{Path}_\infty(X)$. This formula express the existence of a set X of vertices that forms a infinite path (whether it is a loop or a straight path may also be checked). The sub-formulas are built from the base formulas as follows:

$$\mathrm{Path}_\infty(E) = \mathrm{Path}(E) \wedge (\forall x \in E, \exists y \in E, (x \to y))$$

Here are the expression for $\mathrm{Path}(E)$:

$$
\begin{aligned}
\mathrm{Path}(E) \quad = \quad & (\forall x, y, z \in E, (x \to y \wedge z \to y) \Rightarrow (x = z)) \wedge \\
& (\forall x, y, z \in E, (x \to y \wedge x \to z) \Rightarrow (y = z)) \wedge \\
& (\forall x, y \in E, (x \to^* y \vee y \to^* x))
\end{aligned}
$$

---

[4]by Theorem 3.6, $\mathcal{A}$ can be transformed into a deterministic VPA $\mathcal{B}$, and by Remark 2.5, we equivalently decide the diagnosability of the system presented by $\mathcal{B}$.

The last sub-formula is $\rightarrow^*$:

$$
\begin{aligned}
x \rightarrow^* y &= \forall X((x \in X \wedge \text{Closed}(X)) \Rightarrow y \in X) \\
\text{Closed}(X) &= \forall x, y((x \in X \wedge x \rightarrow y) \Rightarrow y \in X)
\end{aligned}
$$

$\square$

Theorem 4.7. *Given a $[\Sigma_{int}]$-VP system $\mathcal{S}$, an observation alphabet Σ_o with $\overline{\Sigma_o} \subseteq \Sigma_{int}$, and a proposition f which marks a regular set of configurations, it is decidable whether $\mathcal{S}$ is bounded-latency or not. Furthermore, the bound can be effectively computed.*

Proof. Without loss of generality, let $\mathcal{A}$ be a deterministic $[\Sigma_{int}]$-VPA which presents $\mathcal{S}^5$. Also, we can assume $\mathcal{S}$ diagnosable, otherwise it is not bounded-latency and, by the hypothesis on Σ_o and f, diagnosability is decidable (Theorem 4.5). We derive the (non-deterministic) PDA $\mathcal{A}'$ from the VPA $\pi(\mathcal{A}) \times \pi(\mathcal{A})$ as follows. We re-label with ε all transitions outgoing the states in $\overline{F} \times \overline{F}$, we remove all transitions outgoing the states in $F \times F$, and we let (q_0, q_0) be the initial state and $F \times F$ be the final states. As such, $\mathcal{A}'$ accepts the words ϑ such that $\theta\vartheta$ is a just-faulty-clear observation, for some equivocal observation θ. By Definition 2.4, $L(\mathcal{A}')$ is finite if, and only if, $\mathcal{S}$ is bounded-latency, and it is well established that the finiteness of CF languages is decidable. Moreover, when $L(\mathcal{A}')$ is finite, the value is $\max\{|\vartheta| \mid \vartheta \in L(\mathcal{A}')\}$.

$\square$

5 Higher-order pushdown systems

Higher-order pushdown automata [11] extend PDA and reach context-sensitive languages. We only sketch their definition, following [5].

Let Γ be a stack alphabet. For any integer $k \geq 1$, k *level* stacks, or shortly k-*stacks*, (over Γ) are defined by induction: A 1-stack is of the form $[U]_1$, where $U \in \Gamma^*$, and the empty stack is written $[]_1$; 1-stacks coincide with stacks of PDA. For $k > 1$, a k-stack is a finite sequence of $(k-1)$-stacks; the empty k-stack is written $[]_k$. An *operation of level k* acts on the topmost k-stack of a $(k+1)$-stack; operations over stacks (of any level) preserve their level. Operations of level 1 are the classical $push_X$ and pop_X, for all $X \in \Gamma$: $push_X([U]_1) = [UX]_1$ and $pop_X([UX]_1) = [U]_1$. Operations of level $k > 1$ are copy_k and $\overline{\text{copy}}_k$, and act on $(k+1)$-stacks as follows ($S_1, \ldots, S_n$ are k-stacks).

$$
\begin{aligned}
\text{copy}_k([S_1, \ldots, S_n]_{k+1}) &:= [S_1, \ldots, S_n, S_n]_{k+1} \\
\overline{\text{copy}}_k([S_1, \ldots, S_n, S_n]_{k+1}) &:= [S_1, S_2, \ldots, S_n]_{k+1}
\end{aligned}
$$

Any operation ρ of level k extends to arbitrary higher level stacks according to: $\rho([S_1, \ldots, S_n]_\ell) = [S_1, \ldots, \rho(S_n)]_\ell$, for $\ell > k+1$.

A *higher-order pushdown automaton (*HPDA*) of order k* is a structure $\mathcal{A} = (\Sigma, \Gamma, Q, q_0, F, \Delta)$ like a PDA, but where Δ specifies transitions which effects operations on the k-stack of the

---

[5] We use Remark 2.5 to make this assumption valid.

automaton. We refer to [5] for a comprehensive contribution on the analysis of HPDA; following this contribution, a set of configurations is *regular* whenever the sequences of operations that are used to reach the set forms a regular language, in the usual sense. *Higher-order pushdown systems (*HPDS*)* are configuration graphs of HPDA. By Theorem 4.1, their diagnosability is undecidable. However, similarly to first-order PD systems, *higher-order* VPA *(*HVPA*)* can be considered [8].

A *k-order* VPA has $(2k+1)$ sub-alphabets Σ_{push}, Σ_{pop}, Σ_{int}, Σ_{copy_r}, and $\Sigma_{\overline{\text{copy}_r}}$, where $r \in [k]$, each of which determines the nature (*e.g. push, pop, internal*, copy$_r$, $\overline{\text{copy}_r}$) of the transitions on its symbols. Transitions on elements of Σ_{int} leave the stacks of any level unchanged. According to [8], HVPA are neither closed under concatenation, nor under iteration, and cannot be determinized; they are however closed under intersection.

Proposition 5.1. *The projection onto Σ'^* of a k-order* VP *language with internal alphabet Σ_{int} is a k-order* VP *language, provided $\overline{\Sigma'} \subseteq \Sigma_{int}$.*

Proof. The proof of Proposition 3.8 easily adapts here. Let L be a k-order VP language accepted by the k-order HVPA $\mathcal{A} = (\Sigma, \Gamma, Q, q_0, F, \Delta)$. We again write $p \Rightarrow p'$ whenever there exists $(p, \varepsilon, a, p', \varepsilon) \in \Delta_{int}$ with $a \in \overline{\Sigma_o}$.

The HVPA $\pi(\mathcal{A})$ which accepts $\pi(L)$ is obtained by adding new transitions, and by letting $p \in F'$ if $p \Rightarrow^* p'$, for some $p' \in F$. The transitions in Δ' are obtained by replacing, in a transition of Δ, the origin state p by the state r, provided $r \Rightarrow p$ in $\mathcal{A}$. Notice that $\Delta \subseteq \Delta'$.

This construction is correct in the sense that $L(\pi(\mathcal{A})) = \pi(L)$. □

Theorem 5.2. *For any class of k-order* VP *system with the sub-alphabets Σ_{push}, Σ_{pop}, Σ_{int}, Σ_{copy_r}, and $\Sigma_{\overline{\text{copy}_r}}$ ($r \in [k]$), diagnosability w.r.t. the set of observables Σ_o and the proposition f is decidable, whenever $\overline{\Sigma_o} \subseteq \Sigma_{int}$ (the internal alphabet) and f marks a regular set of configurations.*

Proof. Let $\mathcal{S}$ be a k-order VP system presented by $\mathcal{A} = (\Sigma, \Gamma, Q, q_0, F, \Delta)$. By Proposition 5.1, $\pi(\mathcal{A})$ is a k-order VPA, and Lemma 4.6 for first-order VP system can be easily adapted.

Lemma 5.3. *The non-deterministic k-order* VPA $\pi(\mathcal{A}) \times \pi(\mathcal{A})$ *with initial state (q_0, q_0) and final states $F \times \overline{F}$ accepts the equivocal observations.*

As in the proof of Theorem 4.5, checking diagnosability amounts to decide an MSO-property (the existence of an infinite run remaining in configurations whose states are in $(F \times \overline{F})$). Since Theorem 3.4 for MSO-properties model-checking extends to HPDA [6], we are done. □

The bounded-latency problem. We are unable to generalize Theorem 4.7 to HVP systems for the following reasons. Deciding the finiteness of $L(\mathcal{A}')$ in the proof of Theorem 4.7

is a key point. Fortunately, the finiteness of CF languages is decidable, thanks somehow to its decidability for real-time PDA[6]. For HPDA, we have a similar result.

Theorem 5.4. *The finiteness of a real-time* HPD *language is decidable.*

Proof. Let L be a real-time HPD language presented by the real-time HPDA $\mathcal{A} = (\Sigma, \Gamma, Q, q_0, F, \Delta)$. Because $\mathcal{A}$ is real-time, the finiteness of L is equivalent to the finiteness of the set of accepting runs of $\mathcal{A}$.

By [5], the following sets are regular:

- $\mathcal{R}$, the set of configurations reachable from the initial configuration;

- $\mathcal{F}$, the set of final configurations;

- $co\mathcal{R}(C)$, the set of configurations that reach a fixed configuration C;

If the regular set $\mathcal{R} \cap \mathcal{F}$ is infinite, then we can build infinitely many accepting runs, L is hence infinite. Otherwise, $\mathcal{R} \cap \mathcal{F} = \{C_1, \ldots, C_m\}$. Consider the regular set $\mathcal{D} := \mathcal{R} \cap (co\mathcal{R}(C_1) \cup \ldots co\mathcal{R}(C_m))$; it is the set of configurations which belong to some accepting run of $\mathcal{A}$.

If $\mathcal{D}$ is infinite, we can build arbitrarily long accepting runs; therefore, L is infinite. Otherwise, we consider the finite-state automaton obtained by restricting the configuration graph of $\mathcal{A}$ to the finite set $\mathcal{D}$. The language of this automaton is finite if, and only if, L is finite. □

Unfortunately, the Theorem 5.4 requires to assume a finite set of initial configurations, which is too restrictive in general as illustrated by the example of Figure 5.1: the proposed system is a higher-order
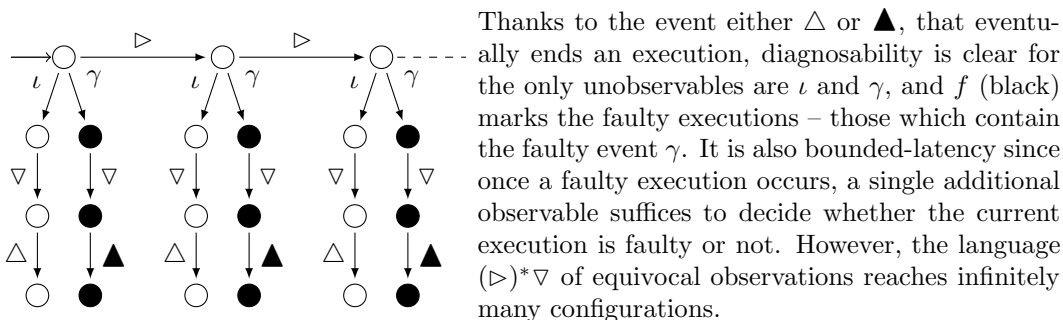

Thanks to the event either $\triangle$ or $\blacktriangle$, that eventually ends an execution, diagnosability is clear for the only unobservables are ι and γ, and f (black) marks the faulty executions – those which contain the faulty event γ. It is also bounded-latency since once a faulty execution occurs, a single additional observable suffices to decide whether the current execution is faulty or not. However, the language $(\triangleright)^* \triangledown$ of equivocal observations reaches infinitely many configurations.

Figure 5.1: Bounded delay, but infinitely many equivocal configurations

However, the PDA $\mathcal{A}'$ is not real-time in general, and deciding whether an arbitrary HPDA is equivalent to a real-time one or not is an open question, conjectured negative by

---

[6]and Proposition 3.2.

[4]. The finiteness of an arbitrary HPD language is therefore a difficult question. Moreover, provided the argument for Proposition 4.4 easily extends to HVP language, hopes to decide the bounded-latency problem for HVP systems considerably lessen.

Nevertheless, this problem can sometimes be answered positively.

Consider a HVP system $\mathcal{S}$, presented by the HVPA $\mathcal{A} = (\Sigma, \Gamma, Q, q_0, F, \Delta)$. By Lemma 5.3, the language Υ of the real-time HPDA $\pi(\mathcal{A}) \times \pi(\mathcal{A})$ (with initial state (q_0, q_0) and final states $F \times \overline{F}$) is the set of equivocal observations; its finiteness is hence decidable (Theorem 5.4).

- If Υ is finite, the bounded-latency value is

$$\max\{|\vartheta| \mid \exists \theta \in \Upsilon, \theta\vartheta \text{ is just-faulty-clear}\}$$

- Otherwise, we inspect the set $\mathcal{C}$ of configurations reached by Υ; it is regular by [5], and can be effectively computed.

  - If $\mathcal{C}$ is finite, for each $C \in \mathcal{C}$, define the real-time HPDA $\mathcal{A}_C$ as $\pi(\mathcal{A}) \times \pi(\mathcal{A})$ where transitions outgoing states in $F \times F$ have been cut, with initial configuration C, and with the final states $F \times F$. Decide the finiteness of $L(\mathcal{A}_C)$ (Theorem 5.4). If every $L(\mathcal{A}_C)$ is finite, then $\mathcal{S}$ is bounded-latency with the value

$$\max\{|\vartheta| \mid \vartheta \in \cup_{C \in \mathcal{C}} L(\mathcal{A}_C)\}$$

  - If $\mathcal{C}$ is infinite, nothing can be inferred.

6 Conclusion

Formal methods for the diagnosis of discrete-event systems have been widely investigated since the seminal paper of [13]. Sticking to so-called *centralized diagnosis problems*, that is diagnosis problems with a single system and a single diagnoser, particular attention has been paid to the diagnosability property, as a cornerstone to establish the completeness of the diagnoser. Deciding the diagnosability of finite-state systems is easy, with polynomial time solutions [9, 10, 13]; notice that moreover, the bounded-latency property is an immediate corollary of diagnosability. On the other hand, to our knowledge, the diagnosability and the bounded-latency problems for infinite-state discrete-event systems were unexplored so far.

The present work brings to light distinctive elements to decide diagnosability, and circumscribes relevant classes of instances of diagnosis problems. The obtained results extend in two directions. On the model side, we have considered the rich collection of pushdown systems, whose set of executions may be context-free languages, and even context-sensitive languages for higher-order pushdown systems. On the specification side, it is important to remark that the class of properties we have considered, although necessarily restricted to topologically open sets of words [3], strictly extends regular sets [9]: allowing properties characterized by a regular set of configurations (in the sense of [5]) yields a set of executions

that is not regular, i.e. that is not characterized by a finite-state automaton (indeed it can be seen as the product of the pushdown automaton with a finite state system).

An essential outcome of this contribution is the need to consider classes of models where the projection over the alphabet of observables has good computational properties, and where the product can also be effectively handled. Our way to enforce this computability gives only a sufficient condition based on syntactic criteria between the alphabets, and strongly depends on the presentation used to describe the system's executions. Since presentations are not unique in general, we may wish to exhibit conditions for decidability at a more semantic level: to illustrate this, consider a system whose set of executions is $(ab)^n(cd)^n$. It is easy to design a (first-order) visibly pushdown automaton over the alphabet partition $[\{a,b\}, \{c,d\}, \emptyset]$ that describes this system. For such a presentation, our criterion (Theorem 4.5.(b)), does not allow any unobservable event (because each event operates on the stack) and yet, the projection of $(ab)^n(cd)^n$ onto the alphabet $\{b,c,d\}$ is $b^n(cd)^n$, a visibly pushdown language. In fact, the presentation chosen above to describe $(ab)^n(cd)^n$ was unfortunate, whereas another visibly pushdown automaton over the alphabet e.g. $(\{b\}, \{d\}, \{a,c\})$ exists and is adequate for diagnosis problem instances where a is unobservable. We believe that a structural analysis of a fixed presentation can shed light on the alphabets of observables that yield computable projections with good properties, hence entailing more general methods for diagnosis problems, or other connected topics of partial observation.

References

[1] R. Alur and P. Madhusudan, *Visibly pushdown languages*, in STOC 04, ACM, 2004, pp. 202–211.

[2] J.-M. Autebert, J. Berstel, and L. Boasson, *Context-free languages and pushdown automata*, in Handbook of formal languages, Vol. 1, Springer-Verlag, 1997, pp. 111–174.

[3] A. Bauer and S. Pinchinat, *A topological perspective on diagnosis*, in 9th edition of Workshop on Discrete Event Systems, Göteborg, Sweden, May 2008.

[4] A. Carayol, *Private communication.* Private communication.

[5] ——, *Regular sets of higher-order pushdown stacks*, in MFCS, J. Jedrzejowicz and A. Szepietowski, eds., vol. 3618 of Lecture Notes in Computer Science, 2005, pp. 168–179.

[6] A. Carayol and S. Woerhle, *The caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata*, in FSTTCS 03, P. K. Pandya and J. Radhakrishnan, eds., vol. 2914 of LNCS, 2003, pp. 112–123.

[7] C. G. CASSANDRAS AND S. LAFORTUNE, *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, 1999.

[8] S. ILLIAS, *Higher order visibly pushdown languages*, Master's thesis, Indian Institute of Technology, Kanpur, 2005.

[9] T. JÉRON, H. MARCHAND, S. PINCHINAT, AND M.-O. CORDIER, *Supervision patterns in discrete event systems diagnosis*, in 8th Workshop on Discrete Event Systems, Ann Arbor, Michigan, USA, July 2006.

[10] S. JIANG, Z. HUANG, V. CHANDRA, AND R. KUMAR, *A polynomial time algorithm for diagnosability of discrete event systems*, IEEE Transactions on Automatic Control, 46 (2001), pp. 1318–1321.

[11] A. MASLOV, *Multilevel stack automata.*, Problems of Information Transmission, 12 (1976), pp. 38–43.

[12] D. MULLER AND P. SCHUPP, *The theory of ends, pushdown automata, and second-order logic*, Theoretical Computer Science, 37 (1985), pp. 51–75.

[13] M. SAMPATH, R. SENGUPTA, S. LAFORTUNE, K. SINAAMOHIDEEN, AND D. TENEKETZIS, *Failure diagnosis using discrete event models*, IEEE Transactions on Control Systems Technology, 4 (1996), pp. 105–124.

[14] I. WALUKIEWICZ, *Model checking ctl properties of pushdown systems*, in FSTTCS, S. Kapoor and S. Prasad, eds., vol. 1974 of Lecture Notes in Computer Science, Springer, 2000, pp. 127–138.