Diagnosability of Pushdown Systems

Christophe Morvan[1] **and Sophie Pinchinat**[2]

[1] Université Paris-Est,
INRIA Centre Rennes - Bretagne Atlantique
[2] IRISA, Campus de Beaulieu, 35042 Rennes, France

Abstract. Partial observation of discrete-event systems features a setting where events split into observable and unobservable ones. In this context, the diagnosis of a discrete-event system consists in detecting defects from the (partial) observation of its executions. Diagnosability is the property that any defect is eventually detected. Not surprisingly, it is a major issue in practical applications. We investigate diagnosability for classes of pushdown systems: it is undecidable in general, but we exhibit reasonably large classes of visibly pushdown systems where the problem is decidable. For these classes, we furthermore prove the decidability of a stronger property: the bounded latency, which guarantees the existence of a uniform bound on the respond delay after the defect has occurred. We also explore a generalization of the approach to higher-order pushdown systems.

1 Introduction

Absolute knowledge of the actual execution of a computer driven system is, in most settings, impossible. However, since typical systems emit information while interacting with their environment, deductions of their internal state can be performed on the basis of this *partial observation*.

From a mathematical point of view, a standard approach due to [17] uses a *discrete-event system* modeling (see, *e.g.*, [8]), provided with a partition of the event set into *observables* and *unobservables*. In this formal framework, *diagnosing* a system amounts to deducing, from its actual observation, the set I of its possible internal states, and to compare I with a distinguished subset of states P representing some property of the executions (for example the occurrence of a failure event). Diagnosing thereby brings about three different *verdicts*: the *negative* verdict when I does not meet P, the *positive* verdict when I lies in P, and the *inconclusive* verdict otherwise. The device which outputs the verdict is the *diagnoser*.

Building a diagnoser is not a difficult task, per se: it relies on classical powerset construction. For finite-state systems, it induces an unavoidable exponential blow-up [19], even for succinct representations [15]. Therefore

on-the-fly computation of the diagnoser is a key techniques for effective methods. It incidentally offers an effective solution in infinite-state settings [18,3]. Whatever method is used for the diagnoser, a central question is whether the diagnoser will eventually detect any faulty execution (execution reaching P)? This property is the *diagnosability*, expressing intrinsic features of the system (together with P). Clearly, on-the-fly methods cannot apply, since diagnosability requires an exhaustive analysis of the model. PTIME decision procedures have been developed for finite-state systems [12,11]; non-diagnosability is NLOGSPACE-complete [15]. Also, SAT-solvers can be used for symbolic systems [9].

Beyond finite-state systems, very little exists in the literature on the diagnosis of discrete-event systems: [18] considered a timed systems setting, and established the equivalence between diagnosability and non-zenoness, yielding PSPACE-completeness. Petri nets have been studied in [20], where either classical techniques apply to finite nets (i.e. with a finite-state configuration graph), or approximation methods yield only semi-algorithms. Finally, [3] considered graph transformation systems, and developed a general procedure to compute the set of executions corresponding to a given observation. Notice that this approach does not provide any algorithm for the diagnosability whose statement universally quantifies over the set of observations. Surprisingly, diagnosis issues have never been addressed for pushdown systems, although acknowledged as good abstractions for the software model-checking of recursive programs [14]. Alternation-free (branching-time) μ-calculus, hence CTL, properties can be verified in EXPTIME [21], and fixed linear-time μ-calculus properties can be checked in PTIME [4]. In addition, partial observation of pushdown systems is simple to model since the class is closed under projection[1].

In this paper, we study diagnosability of pushdown systems (of arbitrary order) represented by (higher-order) pushdown automata. Diagnosability is shown undecidable in general, via a reduction of the emptiness problem for an intersection of context-free languages. In fact diagnosability requires concomitant properties that arbitrary classes of pushdown systems do not possess in general. Recently, Alur and Madhusudan introduced *visibly pushdown automata* [1] with adapted features to handle diagnosability. As we show here, arbitrary classes of visibly pushdown systems still do not yield decidability of the diagnosability, and our contribution precisely exhibits a sufficient condition. This condition correlates

[1] ε-closure of a pushdown automaton remains a pushdown automaton [2].

the observability of the system with its recursive structure: there must exists a pushdown description of the system, where accesses to the stack are observable. In this case, we adapt the non-diagnosability algorithm for finite-state systems developed in [11], yielding a PTIME upper bound; the NLOGSPACE lower bound for finite-state systems remains valid. The results on decidability are furthermore generalized to the higher order, by considering the higher-order visibly pushdown automata of [10]. We develop a k-EXPTIME algorithm for the class of k-order pushdown systems ($k \geq 2$).

As explained further in the paper, diagnosability guarantees, for each execution reaching P, a finite delay to detect it. However, it does not provide a uniform bound on these delays. The *bounded-latency* problem consists in deciding whether such a bound exists, and is fully relevant for practical applications. In the literature, bounded latency has been mainly investigated in the framework of finite-state systems, although it is a direct consequence of diagnosability. [16,22] refer to *the bound*, and [11] refer to *n-diagnosability*. Unexpectedly, to our knowledge, bounded latency has not been studied for infinite-state systems, for which diagnosability does not imply bounded latency.

In this paper, we consider the bounded-latency problem for pushdown systems. We show its decidability for families of first-order pushdown systems where diagnosability is already decidable (otherwise it does not make sense). For these families, bounded latency is equivalent to the finiteness of a language accepted by a pushdown autamaton. The latter problem is in PTIME [2]. Regarding higher-order pushdown systems, we conjecture undecidability of the bounded-latency problem. As for first-order pushdown systems, checking bounded latency amounts to checking finiteness of a higher-order pushdown language. For arbitrary higher-order pushdown language, the finiteness problem is still open, to our knowledge.

The paper is organized as follows. In Section 2 we define the diagnosability and the bounded-latency problems, and recall the classic results for finite-state systems. Pushdown systems are considered in Section 3, and handled in the core Section 4 of the contribution to study their diagnosability and bounded-latency problems. In Section 5, we consider higher-order pushdown systems.

2 Diagnosability and Bounded Latency

We first introduce some mathematical notations and definitions. Assume a fixed set E. We denote by 2^E its powerset, and by $\overline{B}$ the complement of a subset $B \subseteq E$. For any $k \in \mathbb{N}$, we write $[k] := \{1, 2, 3, \ldots, k\}$. Given an alphabet (a set of symbols) Σ, we write Σ^* and Σ^ω for the sets of finite and infinite words (sequences of symbols) over Σ respectively. We use the standard notation ε for the empty finite word, and we denote by $u, u', v, \ldots$ the typical elements of Σ^*, and by $w, w_1, \ldots$ the typical elements of Σ^ω. For $u \in \Sigma^*$, $|u|$ denotes the length of the word u.

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* $1 \leq i \leq n$*. An execution* u reaches *a subset* $S' \subseteq S$ *whenever* $\delta(s^0, u) \in S'$*, by extending* δ *to* $S \times \Sigma^*$*. 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* $[\![m]\!]$*.*

We now give an overview on diagnosis. Diagnosis is about synthesis where 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 every $a \in \Sigma$. This assumption means that we focus on whether some defect (a particular event or a particular pattern of

events) has occurred in the past or not; we refer to [8] 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. For technical reasons, we need to consider *information sets*: an information set I is the set of all states reached by a set of indistinguishable executions in $\Sigma^* \Sigma_o$. We write $\mathcal{I} \subseteq 2^S$ for the set of all information sets. Notice that $\{s_0\} \in \mathcal{I}$ and is associated to the empty observation. The associated diagnoser is a structure $\mathcal{D} := \langle \Sigma_o, \mathcal{I}, I^0, \hat{\delta}, diag \rangle$ whose states are either the initial state $I^0 := \left\{s^0\right\}$ or the transition function, $\hat{\delta} : \mathcal{I} \times \Sigma_o \to \mathcal{I}$, is the extension of δ to sets of states in a canonical way, and the output function *diag* is defined as follows. Given a set $I \subseteq S$, 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,

$$\begin{aligned} diag : \mathcal{I} &\to \{(a), (b), (c)\} \\ I &\mapsto (a) \text{ if } I \subseteq [\![f]\!], (b) \text{ if } I \cap [\![f]\!] = \emptyset, \text{ and } (c) \text{ otherwise} \end{aligned}$$

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

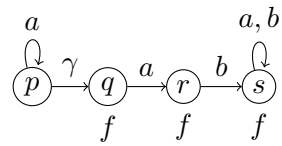
$\mathcal{D}$ may be infinite-state in general (if $\mathcal{S}$ is infinite-state). However, its computation can be avoided by simulating it on-the-fly, storing the current information set I, and updating this object on each observable step of the system. While the synthesis of the diagnoser is not necessary, analyzing its behaviour is crucial: in particular, because equivocalness (case (c)) precludes the instantaneous detection of a fault, latencies to react are tolerated.

Diagnosability is a qualitative property of the diagnoser which ensures a finite latency for any observation of a faulty execution; it corroborates the completeness of the diagnoser. From a quantitative point of view, the *bounded-latency* property ensure a uniform bound on the latencies. We develop these two notions.

In accordance with [17], 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 clearly-faulty finite prefix.*

Safe executions of diagnosable systems may have arbitrarily long equivocal observations as illustrated here with a system whose initial state p, and with the unobservable γ which leading to the marked state q. Since any faulty execution only yields an infinite observation with the clear prefix $a^n b$, the system is diagnosable, but the infinite observation a^ω of the the unique safe execution loops in the equivocal information set $\{p, q\}$.



Lemma 2.3 **[11]** *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.*

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

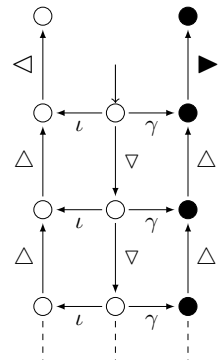
We now consider the *latency* of a diagnosable system as the minimal number of additional observation steps that is needed to detect a faulty execution.

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. The* latency *of an execution u is defined by: $\ell(u) := \max\{|\vartheta|, \pi(u)\vartheta$ is not clearly-faulty$\}$ if u reaches f, and 0 otherwise.*
$\mathcal{S}$ is bounded-latency *if there exists $N \in \mathbb{N}$ such that $\ell(u) \leq N$, for every execution u; the least such N is the* bounded-latency value.

The bounded-latency value of the system above is 1: indeed, fix an observed execution u that reaches f and whose observation is not clearly-faulty (hence equivocal). This execution necessarily ends either in state q or in state r. If in q, the only sequence of observations ϑ such that $\pi(u)\vartheta$ is not clearly-faulty is $\vartheta = a$; therefore $\ell(u) = 1$. If in r, we have $\ell(u) = 0$.

Remark that a system is diagnosable if, and only if, $\ell(u)$ is a finite value, for every execution u, but not necessarily bounded. Therefore any bounded-latency system is diagnosable, but the converse does not hold in general.

The system depicted here is diagnosable when 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 $\blacktriangleright$ if, and only if, γ has occurred. However, the system is not bounded-latency since arbitrarily many $\triangle$'s can occur between γ and $\blacktriangleright$.



Since, diagnosability and bounded latency only depend on the set of executions of the system, one is allowed to decide these problems over a transformed system as long as executions are preserved.

For finite-state systems, it is easy to prove that diagnosability and bounded-latency properties coincide.

Theorem 2.5 [17,12,15]
For finite-state systems:
(i) Diagnosability is decidable in PTIME.
(ii) Non-diagnosability is NLOGSPACE-*complete.*

3 Pushdown systems

We now investigate the case of pushdown systems where the picture is more involved. We recall that pushdown automata are finite-state machines that use a stack as an auxiliary data structure (see for example [2]); pushdown systems are derived as configuration graphs of pushdown automata and are infinite-state in general.

Definition 3.1 *A* pushdown automaton (PDA) *is a structure* $\mathcal{A} = (\Sigma, \Gamma, Q, q_0, F, \Delta)$ *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 transitions.*

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 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) $\forall (p, X, a) \in Q \times \Gamma \times \Sigma \cup \{\varepsilon\}$, there is at most one pair (q, V) such that $(p, X, a, q, V) \in \Delta$, and (2) $\forall (p, X) \in Q \times \Gamma$, if there exists (q, V) such that $(p, X, \varepsilon, q, V) \in \Delta$, then there is no triple $(q', a, V') \in Q \times \Sigma \times \Gamma^*$ such that $(p, X, a, q', V') \in \Delta$. An automaton $\mathcal{A}$ 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 $qU \in Q\Gamma^*$; q is *the state* of the configuration. 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 $(q, X, a, q', V) \in \Delta$ with $U = WX$ and $U' = WV$. A finite *run* of $\mathcal{A}$ is a finite sequence $r = q_0U_0a_1q_1U_1a_2 \ldots a_nq_nU_n$ such that $U_0 = \varepsilon$ is initial, and $(q_iU_i, a_i, q_{i+1}U_{i+1})$ is a transition, for all $0 \leq i < n$. We say that $a_1a_2 \ldots a_n$ is *the word of* r, or that r is a *run on* $a_1a_2 \ldots a_n$. The run is *accepting* if q_nU_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 the proper subclass of *deterministic* CF languages, containing all regular languages. Moreover, CF languages are closed under union, concatenation, and iteration, but not under intersection, and their emptiness is decidable.

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

We finally need to recall the following theorem.

Theorem 3.4 **[4]** *The emptiness problem of a Büchi pushdown automaton is in* PTIME*.*

A *pushdown system (*PD *system)* $\mathcal{S}$ is the configuration graph of a real-time PDA $\mathcal{A} = (\Sigma, \Gamma, Q, q_0, F, \Delta)$, which *represented* $\mathcal{S}$. Notice that the set F of $\mathcal{A}$ is irrelevant for $\mathcal{S}$. However, using standard techniques, the statement that $[\![f]\!]$ is a regular set of configurations in $\mathcal{S}$ can be transformed into $[\![f]\!] = F\Gamma^*$ (see appendix for details).

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

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* $\Sigma := \Sigma_{push} \cup \Sigma_{pop} \cup \Sigma_{int}$*, where* Σ_{int} *is the* internal alphabet*, and* $\Delta := \Delta_{push} \cup \Delta_{pop} \cup \Delta_{int}$ *respectively, with the constraints that* $\Delta_{push} \subseteq Q \times \{\varepsilon\} \times \Sigma_{push} \times Q \times (\Gamma \setminus \{\perp\})$*,* $\Delta_{pop} \subseteq Q \times \Gamma \times \Sigma_{pop} \times Q \times \{\varepsilon\}$*, and* $\Delta_{int} \subseteq Q \times \{\varepsilon\} \times \Sigma_{int} \times Q \times \{\varepsilon\}$*.*

A $[\Sigma_{int}]$-VP language *is a language accepted by some* VPA *whose internal alphabet is* Σ_{int}*.*

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

We now turn to the projection operation on languages with respect to a sub-alphabet as a central operation for partial observation issues; we recall that the class of CF languages is projection-closed, whereas VP languages are not; more precisely,

Proposition 3.7
(i) Any CF *language is the projection of a* $[\emptyset]$-VP *language, and this is effective.*
(ii) The projection of a $[\Sigma_{int}]$-VP *language onto* Σ'^**, with* $\overline{\Sigma'} \subseteq \Sigma_{int}$*, is a* $[\Sigma_{int}]$-VP *language (with* $\Sigma_{push}, \Sigma_{pop}$ *fixed). The construction is effective.*

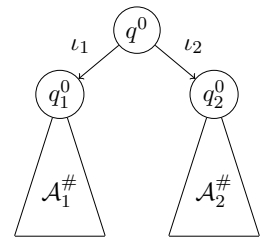
4 Diagnosability and Bounded Latency of PD systems

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.

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

This theorem is a corollary of Proposition 3.3 and the following construction together with Lemma 4.2. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two deterministic PDA over Σ_1 and Σ_2 respectively, and let $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \{\iota_1, \iota_2, \#\}$, with fresh symbols $\#$, ι_1 and ι_2.

For $i = 1, 2$, let $\mathcal{A}_i^{\#}$ be a deterministic PDA which accepts $L(\mathcal{A}_i)\#\Sigma^*$, the set of words $u\#v$ where $u \in L(\mathcal{A}_i)$. Let $\mathcal{A}_1^{\#} \oplus \mathcal{A}_2^{\#}$ be 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* $\mathcal{S}$ *represented by* $\mathcal{A}_1^{\#} \oplus \mathcal{A}_2^{\#}$ *is diagnosable w.r.t.* $\Sigma \setminus \{\iota_1, \iota_2\}$ *and* f *if, and only if,* $L(\mathcal{A}_1) \cap L(\mathcal{A}_2) = \emptyset$.

Indeed, consider $w_1 := \iota_1 u \#^\omega$ indistinguishable from $w_2 := \iota_2 u \#^\omega$ with $u \in L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$. Thus w_1 reaches f but w_2 does not. Apply Lemma 2.3 to conclude. Reciprocally, if $\mathcal{S}$ is not diagnosable, then by Lemma 2.3, there exist indistinguishable infinite executions w_1 and w_2 such that only w_1 reaches f; necessarily, $w_1 = \iota_1 u \# w$ and $w_2 = \iota_2 u \# w$ for some u, entailing $u \in L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$, which concludes the proof. □

Theorem 4.3 *(a) Diagnosability of* VP *systems is undecidable.*
(b) Diagnosability w.r.t. a set of observables Σ_o *and a proposition* f *is decidable in* PTIME *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 the undecidability of diagnosability for PD systems (Theorem 4.1) and the fact that any CF language is the projection of some VP language (Proposition 3.7 Point (i)). For Point (b) of Theorem 4.3, let $\mathcal{S}$ be a VP system represented by a deterministic $[\Sigma_{int}]$-VPA $\mathcal{A} = (\Sigma, \Gamma, Q, q_0, F, \Delta)$, 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 [11] for finite-state systems.

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 by Σ_o-projecting $\mathcal{A}$ and by building the standard product of VPA [1].

Lemma 4.4 *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.*

Note that for $\pi(\mathcal{A}) \times \pi(\mathcal{A})$, an infinite run remaining in the set of configurations $(F \times \overline{F})(\Gamma \times \Gamma)^*$ denotes an infinite observation which has no clear prefix. By Lemma 2.3, this equivalently rephrases as "the system is not diagnosable". Now, the existence of such a run is equivalent to check the non emptiness of the Büchi automaton whose structure is $\pi(\mathcal{A}) \times \pi(\mathcal{A})$ and whose accepting states are all elements of $(F \times \overline{F})$ (use the fact that $[\![f]\!]$ is a trap). By Theorem 3.4, this can be decided in NLOGSPACE. □

We now establish that for the classes of PD systems that yield effective methods to answer diagnosability problems, bounded latency is also decidable.

Theorem 4.5 *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 in* PTIME *whether* $\mathcal{S}$ *is bounded latency or not. Furthermore, the bound can be effectively computed.*

Proof. Without loss of generality, we can assume $\mathcal{S}$ diagnosable (which is decidable by the hypothesis and Theorem 4.3), otherwise it is not bounded-latency.

Let the deterministic $[\Sigma_{int}]$-VPA $\mathcal{A}$ represent $\mathcal{S}$. Derive from the VPA $\pi(\mathcal{A}) \times \pi(\mathcal{A})$ the (non-deterministic) PDA $\mathcal{A}'$ as follows: re-label with ε all transitions leaving states in $\overline{F} \times \overline{F}$, remove all transitions leaving states in $F \times F$, let (q_0, q_0) be the initial state, and let $F \times F$ be the final states. As such, $\mathcal{A}'$ accepts the words ϑa $(a \in \Sigma_o)$ where for some execution u that reaches f, $\pi(u)\vartheta a$ is clearly-faulty but $\pi(u)\vartheta$ is not. By Definition 2.4, $L(\mathcal{A}')$ is finite (which is decidable in PTIME [2]) if, and only if, $\mathcal{S}$ is bounded-latency; if finite, the value is $\max\{|\vartheta| \mid \vartheta \in L(\mathcal{A}')\Sigma^{-1}\}$[2]. □

5 Extension to Higher-order pushdown systems

Higher-order pushdown automata [13] extend PDA and reach context-sensitive languages. We only sketch their definition, following [7].

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

[2] We use the standard notation $U\Sigma^{-1}$ to denote the set of words v such that $v.a \in U$ for some $a \in \Sigma$.

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 $\overline{\mathrm{copy}}_k$, and act on $(k+1)$-stacks as follows ($S_1, \ldots, S_n$ are k-stacks).

$$\mathrm{copy}_k([S_1, \ldots, S_n]_{k+1}) := [S_1, \ldots, S_n, S_n]_{k+1}$$
$$\overline{\mathrm{copy}}_k([S_1, \ldots, S_n, S_n]_{k+1}) := [S_1, S_2, \ldots, S_n]_{k+1}$$

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 affect operations on the k-stack of the automaton. We refer to [7] 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 form 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 [10].

A *k-order* VPA has $(2k+1)$ sub-alphabets Σ_{push}, Σ_{pop}, Σ_{int}, Σ_{copy_r}, and $\Sigma_{\overline{\mathrm{copy}_r}}$, where $r \in [k]$, each of which determines the nature (*e.g. push*, *pop*, *internal*, copy_r, $\overline{\mathrm{copy}_r}$) of the transitions on its symbols. Transitions on elements of Σ_{int} leave the stacks of any level unchanged. According to [10], 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 3.7 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 $L(\pi(\mathcal{A})) = \pi(L)$. $\square$

Theorem 5.2 *For any class of k-order* VP *systems with the sub-alphabets Σ_{push}, Σ_{pop}, Σ_{int}, Σ_{copy_r}, and $\Sigma_{\overline{\mathrm{copy}_r}}$ ($r \in [k]$), diagnosability w.r.t. the set of observables Σ_o and the proposition f is decidable in k-*EXPTIME*, 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 represented by $\mathcal{A} = (\Sigma, \Gamma, Q, q_0, F, \Delta)$. By Proposition 5.1, $\pi(\mathcal{A})$ is a k-order VPA, and Lemma 4.4 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.*

Assuming the VPA $\mathcal{A}$ is a k-order pushdown automaton, so is the VPA $\pi(\mathcal{A}) \times \pi(\mathcal{A})$. As in the proof of Theorem 4.3, checking diagnosability amounts to decide the non emptiness of the language accepted by the Büchi k-order pushdown automaton $\pi(\mathcal{A}) \times \pi(\mathcal{A})$ with accepting states in $F \times \overline{F}$. According to [5], this is decidable in k-EXPTIME, but the lower bound is still an open question. □

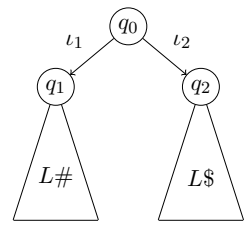
Regarding the bounded-latency problem, Theorem 4.5 does not easily extend to HVP systems. Indeed, in the proof of this theorem, deciding the finiteness of a CF language (namely $L(\mathcal{A}')$ page 11) is a key point, and fortunately this is decidable: the standard decision procedure makes the assumption that the automaton to represent the language is real-time, which is always possible for CF languages using an effective method. If we were able to restrict to real-time HPDA, we would have a similar result since one can show the following.

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

Nevertheless, it is an open question whether arbitrary higher-order pushdown languages are real-time or not; in fact, [6] conjectures they are not. At the moment, deciding the finiteness of an arbitrary HPD language is a difficult question, and so is the bounded-latency property of a higher-order pushdown system, as resolving the latter problem solves the former.

Proposition 5.5 *Let $\mathcal{L}$ be a class of higher-order pushdown languages which is closed under concatenation and union. For each $L \in \mathcal{L}$, there exists a* DES *$\mathcal{S}_L$ such that $\mathcal{S}_L$ is bounded latency if, and only if, L is finite.*

Proof. Assume $L \in \mathcal{L}$ with alphabet Σ. The set of events of $\mathcal{S}_L$ is $\Sigma \cup \{\iota_1, \iota_2, \#, \$\}$, with fresh symbols ι_1, ι_2, $\#$, and $\$$. $\mathcal{S}_L$ has two components $L\#$ and $L\$$ (see figure next page). By construction, the set of executions of $\mathcal{S}_L$ is in $\mathcal{L}$. By letting ι_1 and ι_2 be unobservable, and f mark the configurations of the $L\$$ component, $\mathcal{S}_L$ is diagnosable.
Indeed, event $\#$ or event $\$$ always eventually occur along any execution, revealing the actual running component of the system. It is easy to verify that $\mathcal{S}_L$ is bounded-latency if, and only if, L finite.



□

As a consequence, Proposition 5.5 considerably lessens hopes to decide the bounded latency problem for arbitrary HVP systems. We nevertheless exhibit cases where the problem can sometimes be answered.

Consider a HVP system represented by the HVPA $\mathcal{A} = (\Sigma, \Gamma, Q, q_0, F, \Delta)$. By Lemma 5.3, the real-time HPDA $\pi(\mathcal{A}) \times \pi(\mathcal{A})$ (with initial state (q_0, q_0) and final states $F \times \overline{F}$) accepts the set Υ of equivocal observations, whose finiteness is decidable by Theorem 5.4. We consider the possible cases:

If Υ is finite, then the system is bounded-latency and the bound is $\max\{|\vartheta| \mid \exists \theta \in \Upsilon, \theta\vartheta \text{ is not clearly-faulty}\}$.

Otherwise, we inspect the set $\mathcal{C}$ of configurations reached by Υ, which by [7] is a regular set (that can be effectively computed). Now, decide whether $\mathcal{C}$ is finite or not.

If $\mathcal{C}$ is finite, for each configuration $C \in \mathcal{C}$, build the real-time HPDA $\mathcal{A}_C$ as follows: (1) cut in the automaton $\pi(\mathcal{A}) \times \pi(\mathcal{A})$ every transitions that leaves $F \times F$, (2) set $\mathcal{C}$ as the initial configuration, and (3) $F \times F$ as the final states. Since $L(\mathcal{A}_C)$ is a real-time HPDA its finiteness is decidable. If every $L(\mathcal{A}_C)$ is finite (which can be check since $\mathcal{C}$ is finite), then the system is bounded-latency and the bound is $\max\{|\vartheta| \mid \vartheta \in (\cup_{C \in \mathcal{C}} L(\mathcal{A}_C)).\Sigma^{-1}\}$. If $\mathcal{C}$ is infinite, nothing can be inferred.

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. P. Baldan, T. Chatain, S. Haar, and B. König, *Unfolding-based diagnosis of systems with an evolving topology*, in CONCUR, F. van Breugel and M. Chechik, eds., vol. 5201 of Lecture Notes in Computer Science, Springer, 2008, pp. 203–217.

4. Bouajjani, Esparza, and Maler, *Reachability analysis of pushdown automata: Application to model-checking*, in CONCUR: 8th International Conference on Concurrency Theory, LNCS, Springer-Verlag, 1997, pp. 135–150.
5. T. Cachat and I. Walukiewicz, *The complexity of games on higher order pushdown automata*, CoRR, abs/0705.0262 (2007).
6. A. Carayol, *Notions of determinism for rational graphs*. Private communication, 2001.
7. ———, *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.
8. C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, 1999.
9. A. Grastien, Anbulagan, J. Rintanen, and E. Kelareva, *Diagnosis of discrete-event systems using satisfiability algorithms*, in AAAI, AAAI Press, 2007, pp. 305–310.
10. S. Illias, *Higher order visibly pushdown languages*, Master's thesis, Indian Institute of Technology, Kanpur, 2005.
11. 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.
12. 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.
13. A. Maslov, *Multilevel stack automata.*, Problems of Information Transmission, 12 (1976), pp. 38–43.
14. C.-H. L. Ong, *Hierarchies of infinite structures generated by pushdown automata and recursion schemes*, in MFCS, L. Kucera and A. Kucera, eds., vol. 4708 of Lecture Notes in Computer Science, Springer, 2007, pp. 15–21.
15. J. Rintanen, *Diagnosers and diagnosability of succinct transition systems*, in IJCAI, M. M. Veloso, ed., 2007, pp. 538–544.
16. M. Sampath, R. Sengupta, S. Lafortune, K. Sinaamohideen, and D. Teneketzis, *Diagnosability of discrete event systems*, IEEE Transactions on Automatic Control, 40 (1995), pp. 1555–1575.
17. ———, *Failure diagnosis using discrete event models*, IEEE Transactions on Control Systems Technology, 4 (1996), pp. 105–124.
18. S. Tripakis, *Fault diagnosis for timed automata*, in FTRTFT, W. Damm and E.-R. Olderog, eds., vol. 2469 of Lecture Notes in Computer Science, Springer, 2002, pp. 205–224.
19. J. N. Tsitsiklis, *On the control of discrete event dynamical systems*, Mathematics of Control Signals and Systems, 2 (1989), pp. 95–107.
20. T. Ushio, I. Onishi, and K. Okuda, *Fault detection based on Petri net models with faulty behaviors*, IEEE International Conference on Systems, Man, and Cybernetics., 1 (1998), pp. 113–118 vol.1.
21. 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.
22. T.-S. . Yoo and S. Lafortune, *Polynomial-time verification of diagnosability of partially-observed discreteevent systems*, IEEE Transactions on Automatic Control, 47 (2002), pp. 1491–1495.