Diagnosis in Infinite-State Probabilistic Systems*

Nathalie Bertrand[1], Serge Haddad[2], and Engel Lefaucheux[3]

1 **Inria, France** `nathalie.bertrand@inria.fr`
2 **LSV, ENS Cachan & CNRS & Inria, France** `serge.haddad@ens-cachan.fr`
3 **Inria, France and**
LSV, ENS Cachan & CNRS & Inria, France `engel.lefaucheux@ens-cachan.fr`

Abstract

In a recent work, we introduced four variants of diagnosability (FA, IA, FF, IF) in (finite) probabilistic systems (pLTS) depending whether one considers (1) finite or infinite runs and (2) faulty or all runs. We studied their relationship and established that the corresponding decision problems are PSPACE-complete. A key ingredient of the decision procedures was a characterisation of diagnosability by the fact that a random run almost surely lies in an open set whose specification only depends on the qualitative behaviour of the pLTS. Here we investigate similar issues for infinite pLTS. We first show that this characterisation still holds for FF-diagnosability but with a G_δ set instead of an open set and also for IF- and IA-diagnosability when pLTS are finitely branching. We also prove that surprisingly FA-diagnosability cannot be characterised in this way even in the finitely branching case. Then we apply our characterisations for a partially observable probabilistic extension of visibly pushdown automata (POpVPA), yielding EXPSPACE procedures for solving diagnosability problems. In addition, we establish some computational lower bounds and show that slight extensions of POpVPA lead to undecidability.

1998 ACM Subject Classification D.2.5 Testing and Debugging; F.3.1 Specifying and Verifying and Reasoning about Programs

Keywords and phrases Diagnosis – Infinite-state systems – Partial observation – Markov chains

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2016.37

1 Introduction

Diagnosis. Monitoring (hardware and/or software) systems prone to faults involves several critical tasks: controlling the system to prevent faults as much as possible, deducing the cause of the faults, etc. Most of these tasks assume that an observer has the capability to assess the *status* of the current run based on the outputs of the system: providing information about the possible occurrence of faults. Such an observer is called a *diagnoser* and its associated task is called *diagnosis*. This framework leads to interesting decision and synthesis problems: "Does there exist a diagnoser?" and in the positive case "How to build such a diagnoser?", "Which kind of diagnoser is sufficient?", etc. The decision problem, on which we focus here, is called *diagnosability* [15].

Diagnosis of discrete event systems. In order to formally reason about diagnosability, the systems were first modelled by finite labelled transition systems (LTS). Then the specification of a diagnoser is defined by two requirements: *correctness*, meaning that the information provided by the diagnoser is accurate, and *reactivity*, ensuring that a fault will eventually

* S. Haddad has been supported by ERC project EQualIS (FP7-308087).

be detected. Within the framework of finite LTS, the decision problem was shown to be solvable in PTIME [10] and it is in fact NLOGSPACE-complete.

Diagnosis of probabilistic systems. A natural way of modelling partially observable systems consists in introducing probabilities (*e.g.* when the design is not fully known or the effects of the interaction with the environment is not predictible). Thus the notion of diagnosability was later extended to Markov chains with labels on transitions, also called probabilistic labelled transition systems (pLTS) [16]. In this context, the reactivity requirement now asks that faults will be almost surely eventually detected. Regarding correctness, two specifications have been proposed: either one sticks to the original definition and requires that the provided information is accurate, defining *A-diagnosability*; or one weakens the correctness by admitting errors in the provided information that should, however, have an arbitrary small probability defining *AA-diagnosability.* From a computational viewpoint, we recently proved that A-diagnosability is PSPACE-complete [3] and that AA-diagnosability can be solved in PTIME [4].

In case a system is not diagnosable, one may be able to control it, by forbidding some controllable actions, so that is becomes diagnosable. This property of *active diagnosability* has been studied for discrete-event systems [14, 9], and for probabilistic systems [2]. Interestingly, the diagnosability notion in the latter work slightly differs from the original one in [16]. Building on this variation, in [3] semantical issues have been investigated and four relevant notions of diagnosability (FA, IA, FF, IF) have been defined depending on (1) whether one considers finite or infinite runs and (2) faulty or all runs. In finite pLTS, it was shown that all these notions can be characterized by the fact that a random run almost surely lies in an open set, whose specification only depends on the qualitative behaviour of the pLTS.

Diagnosis of infinite-state systems. Diagnosability in infinite-state systems has been studied, on the one hand for restricted Petri nets [6], for which an accurate diagnoser can be designed, and on the other hand for visibly pushdown automata (VPA) [12], for which diagnosability can be decided via the determinisation procedure of [1]. However to the best of our knowledge diagnosis of probabilistic infinite-state systems has not yet been studied.

Contributions. The characterisations of diagnosability established in [3] strongly relied on the finiteness of the models. Our first aim is thus to establish characterisations in the infinite-state case. FF-diagnosability (the original notion of diagnosability) states that almost surely a faulty run will be detected in finite time. We establish that FF-diagnosability can be characterised by the fact that a random run almost surely lies in a G_δ set, only depending on the qualitative behaviour of the system. This characterisation also applies to IF-diagnosability for finitely-branching systems, since then the two notions coincide. An *ambiguous* infinite correct (resp. faulty) run is a run indistinguishable from a faulty (resp. correct) run. IA-diagnosability states that almost surely a run is unambiguous. The set of ambiguous runs is an analytic set (so a priori not known to be a Borel set). However in the finitely-branching case, we establish that the set of unambiguous runs is a G_δ set, yielding a characterisation of IA-diagnosability. FA-diagnosability states that the probability that a finite run is unambiguous goes to 1 when its length goes to infinity. Surprisingly, despite the fact that IA-diagnosability and FA-diagnosability are very close, we prove that FA-diagnosability cannot be characterised by the fact that a random run almost surely lies in a G_δ set. Furthermore we strenghten this result by another inexpressivess result also related to FA-diagnosability.

Partially observable probabilistic visibly pushdown automata (POpVPA) are models generating infinite-state probabilistic systems. We show how to exploit the above characterisations to design a decision procedure for diagnosability in POpVPA. More precisely we show that we can "encode" our characterisations in an enlarged probabilistic VPA and then exploit the decision procedures of [8] leading to an EXPSPACE algorithm. Since our characterisations are not regular, this requires some tricky machinery. Finally we complete this work by exhibiting an EXPTIME lower-bound and showing that slight extensions of POpVPA lead to undecidability of the diagnosability problem.

Organisation. In Section 2, we introduce probabilistic infinite-state systems, equip them with partial observation and faults, and define diagnosability notions. In Section 3, we establish characterisations of the diagnosability notions and inexpressiveness results. We exploit the characterisations to design decision procedures for POpVPA in Section 4, also proving hardness and undecidability results. We conclude and give some perspectives in Section 5. More details and all the proofs can be found in the associate research report [5].

2 Diagnosis specifications for infinite-state probabilistic systems

2.1 Probabilistic labelled transition systems

Probabilistic labelled transition systems (pLTS) are labelled transition systems equipped with probability distributions on transitions outgoing from a state.

▸ **Definition 1.** A pLTS is a tuple $\mathcal{M} = \langle Q, q_0, \Sigma, T, \mathbf{P}\rangle$ where:
- Q is a finite or countable set of states with $q_0 \in Q$ the initial state;
- Σ is a finite set of events;
- $T \subseteq Q \times \Sigma \times Q$ is a set of transitions;
- $\mathbf{P} : T \to \mathbb{Q}_{>0}$ is the transition probability fulfilling: $\forall q \in Q,\ \sum_{(q,a,q')\in T} \mathbf{P}[q,a,q'] = 1$.

Given a pLTS $\mathcal{M}$, the transition relation of the *underlying LTS* $\mathcal{L}$ is defined by $q \xrightarrow{a} q'$ for $(q,a,q') \in T$; this transition is then said to be *enabled* in q. In order to emphasise the relation between the pLTS and the LTS, we sometimes write $\mathcal{M} = (\mathcal{L}, \mathbf{P})$. Note that since we assume the state space to be at most countable, a pLTS is by definition at most countably branching: from every state q, there are at most countably many transitions enabled in q.

▸ **Example 2.** The pLTS of Figure 1 represents a server that accepts jobs (event `in`) until it randomly decides to serve the jobs (event `serve`). When a job is done the result is delivered (event `out`). When all jobs are done, the server waits for a new batch of jobs. However randomly, the server may trigger a fault (event **f**) and then abort all remaining jobs (event `abort`). Afterwards, the server is reset (event `reset`). In the figure, the label of a transition (q,a,q') is depicted as $\mathbf{P}[q,a,q'] \cdot a$.

Let us now introduce some important notions and notations that will be used throughout the paper. A *run* ρ of a pLTS $\mathcal{M}$ is a (finite or infinite) sequence $\rho = q_0a_0q_1\ldots$ such that for all i, $q_i \in Q$, $a_i \in \Sigma$ and when q_{i+1} is defined, $q_i \xrightarrow{a_i} q_{i+1}$. The notion of run can be generalised, starting from an arbitrary state q. We write Ω for the set of all infinite runs of $\mathcal{M}$ starting from q_0, assuming the pLTS is clear from context. When it is finite, ρ ends in a state q and its *length*, denoted $|\rho|$, is the number of events occurring in it. Given a finite run $\rho = q_0a_0q_1\ldots q_n$ and a (finite or infinite) run $\rho' = q_na_nq_{n+1}\ldots$, the concatenation of ρ and ρ', written $\rho\rho'$, is the run $q_0a_0q_1\ldots q_na_nq_{n+1}\ldots$; the run ρ is then a *prefix* of $\rho\rho'$,

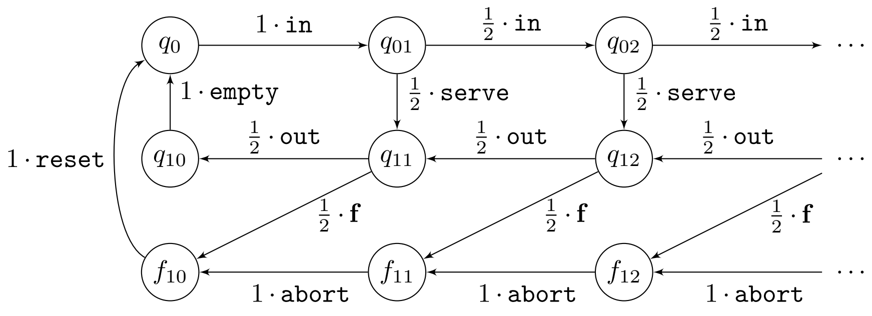

q_0
$1 \cdot \texttt{in}$
q_{01}
$\frac{1}{2} \cdot \texttt{in}$
q_{02}
$\frac{1}{2} \cdot \texttt{in}$
$\dots$
$1 \cdot \texttt{empty}$
$\frac{1}{2} \cdot \texttt{serve}$
$\frac{1}{2} \cdot \texttt{serve}$
$1 \cdot \texttt{reset}$
q_{10}
$\frac{1}{2} \cdot \texttt{out}$
q_{11}
$\frac{1}{2} \cdot \texttt{out}$
q_{12}
$\frac{1}{2} \cdot \texttt{out}$
$\dots$
$\frac{1}{2} \cdot \mathbf{f}$
$\frac{1}{2} \cdot \mathbf{f}$
$\frac{1}{2} \cdot \mathbf{f}$
f_{10}
f_{11}
f_{12}
$\dots$
$1 \cdot \texttt{abort}$
$1 \cdot \texttt{abort}$
$1 \cdot \texttt{abort}$

Figure 1 An infinite-state pLTS.

which we denote $\rho \preceq \rho\rho'$. The *cylinder* defined by a finite run ρ is the set of all infinite runs that extend ρ: $C(\rho) = \{\rho' \in \Omega \mid \rho \preceq \rho'\}$. Cylinders form a basis of open sets for the standard topology on the set of runs (which can be viewed as an infinite tree). One equips a pLTS with a probability measure on Ω with σ-algebra being $\mathcal{B}$, the set of Borel sets, and which is uniquely defined by Caratheodory's extension theorem from the probabilities of the cylinders:

$$\mathbb{P}(C(q_0 a_0 q_1 \dots q_n)) = \mathbf{P}[q_0, a_1, q_1] \cdots \mathbf{P}[q_{n-1}, a_{n-1}, q_n] \ .$$

We will sometimes omit the C and write $\mathbb{P}(\rho)$ for $\mathbb{P}(C(\rho))$. It is well-known that once the measure is fixed, one can enlarge the set of of measurable sets by considering the smallest σ-algebra containing $\mathcal{B}$ and the "null" sets: $\{A \mid \exists B \in \mathcal{B}\ A \subseteq B \wedge \mathbb{P}(B) = 0\}$ and then extend the original measure to a (complete) measure on this enlarged σ-algebra. We consider this measure in the sequel.

The sequence associated with $\rho = q a_0 q_1 \dots$ is the word $\sigma_\rho = a_0 a_1 \dots$, and we write either $q \xrightarrow{\rho}{}^{*}$ or $q \xrightarrow{\sigma_\rho}{}^{*}$ (resp. $q \xrightarrow{\rho}{}^{*} q'$ or $q \xrightarrow{\sigma_\rho}{}^{*} q'$) for an infinite (resp. finite) run ρ. A state q is *reachable* (from q_0) if there exists a run such that $q_0 \xrightarrow{\rho}{}^{*} q$, which we alternatively write $q_0 \rightarrow^{*} q$. The (infinite) language of pLTS $\mathcal{M}$ consists of all infinite words that label runs of $\mathcal{M}$ and is formally defined as $\mathsf{L}^\omega(\mathcal{M}) = \{\sigma \in \Sigma^\omega \mid q_0 \xrightarrow{\sigma}{}^{*}\}$.

2.2 Partial observation and faults

The observation of a pLTS is given by a mask function. This function projects every event to its observation. This observation is partial as an event can have no observation or shares its observation with another event, but it is deterministic.

▶ **Definition 3.** A *partially observable pLTS* (POpLTS) is a tuple $\mathcal{N} = \langle \mathcal{M}, \Sigma_o, \mathcal{P} \rangle$ consisting of a pLTS $\mathcal{M}$ equipped with a mapping $\mathcal{P} : \Sigma \to \Sigma_o \cup \{\varepsilon\}$ where Σ_o is the set of observations.

Note that our setting generalises most existing frameworks of fault diagnosis by considering a mask function $\mathcal{P}$ onto a possibly different alphabet rather than a partition of the event alphabet into observable and unobservable events. An event $a \in \Sigma$ is said *unobservable* if $\mathcal{P}(a) = \varepsilon$, otherwise, it is *observable* and we distinguish a being *fully observable* if $\mathcal{P}(a) \neq \varepsilon$ and $\mathcal{P}^{-1}(\{\mathcal{P}(a)\}) = \{a\}$ or *partially observable* if $\mathcal{P}(a) \neq \varepsilon$ and $|\mathcal{P}^{-1}(\{\mathcal{P}(a)\})| > 1$. The set of unobservable events is denoted Σ_u.

Let $\sigma \in \Sigma^*$ be a finite word; its length is denoted $|\sigma|$. The mapping $\mathcal{P}$ is extended to finite words inductively: $\mathcal{P}(\varepsilon) = \varepsilon$ and $\mathcal{P}(\sigma a) = \mathcal{P}(\sigma)\mathcal{P}(a)$. We say that $\mathcal{P}(\sigma)$ is the *mask* of σ. Write $|\sigma|_o$ for $|\mathcal{P}(\sigma)|$. When σ is an infinite word, its mask is the limit of the masks of its finite prefixes. This mask function is applicable to runs via their associated sequence; it can be either finite or infinite. As usual the mask function is extended to languages. With respect to $\mathcal{P}$, a POpLTS $\mathcal{N}$ is *convergent* if there is no infinite sequence of unobservable events from any reachable state: $\mathsf{L}^\omega(\mathcal{M}) \cap \Sigma^*\Sigma_u^\omega = \emptyset$. When $\mathcal{N}$ is convergent, for every $\sigma \in \mathsf{L}^\omega(\mathcal{M})$, $\mathcal{P}(\sigma) \in \Sigma_o^\omega$. In the rest of the paper we assume that POpLTS are convergent. $\mathcal{P}$ can also be be viewed as a mapping from runs to Σ_o^ω by defining $\mathcal{P}(q_0 a_0 q_1 a_1 \ldots) = \mathcal{P}(a_0 a_1 \ldots)$. Remark that this mapping is continuous. We will refer to a *sequence* for a finite or infinite word over Σ, and an *observed sequence* for a finite or infinite sequence over Σ_o. Clearly, the application of the mask function onto Σ_o of a sequence yields an observed sequence.

The *observable length* of a run ρ denoted $|\rho|_o \in \mathbb{N} \cup \{\infty\}$, is the number of observable events that occur in it: $|\rho|_o = |\sigma_\rho|_o$. A *signalling* run is a finite run whose last event is observable. Signalling runs are precisely the relevant runs w.r.t. partial observation issues since each observable event provides additional information about the execution to an external observer. Given states q, q' and an observed sequence $\sigma \in \Sigma_o^+$, we write $q \overset{\sigma}{\Longrightarrow} q'$ if there is a signalling run from q to q' with observed sequence σ.

In the sequel starting from the initial state q_0, SR denotes the set of signalling runs, and SR_n the set of signalling runs of observable length n. Since we assume that the POpLTS are convergent, for all $n > 0$, SR_n is equipped with a probability distribution defined by assigning measure $\mathbb{P}(\rho)$ to each $\rho \in \mathsf{SR}_n$. Given ρ a finite or infinite run, and $n \leq |\rho|_o$, $\rho_{\downarrow n}$ denotes the signalling subrun of ρ of observable length n. For convenience, we consider the empty run q_0 to be the single signalling run, of null length.

2.3 Fault diagnosis for POpLTS

To model the problem of fault diagnosis in POpLTS, we assume the event alphabet Σ contains a special event $\mathbf{f} \in \Sigma$ called the *fault*. A run ρ is then said to be *faulty* if its associated sequence of events contains a fault, *i.e.* $\sigma_\rho \in \Sigma^*\mathbf{f}\Sigma^\omega$; otherwise it is *correct*. The set of faulty (resp. correct) runs is denoted F (resp. C). For $n \in \mathbb{N}$, we write F_n for the set of infinite runs ρ such that $\rho_{\downarrow n}$ is faulty and C_n for the set of infinite runs ρ such that $\rho_{\downarrow n}$ is correct. By definition, for all n, $\Omega = \mathsf{F}_n \uplus \mathsf{C}_n$, moreover, $\mathsf{F} = \bigcup_{n\in\mathbb{N}} \mathsf{F}_n$ and $\mathsf{C} = \bigcap_{n\in\mathbb{N}} \mathsf{C}_n$.

In order to reason about faults we partition sequences of observations into three subsets: an observed sequence $\sigma \in \Sigma_o^\omega$ is *surely correct* if $\mathcal{P}^{-1}(\sigma) \cap \mathsf{L}^\omega(\mathcal{M}) \subseteq (\Sigma \setminus \mathbf{f})^\omega$; it is *surely faulty* if $\mathcal{P}^{-1}(\sigma) \cap \mathsf{L}^\omega(\mathcal{M}) \subseteq \Sigma^*\mathbf{f}\Sigma^\omega$; otherwise, it is *ambiguous*. For finite sequences, we need to rely on signalling runs: a finite observed sequence $\sigma \in \Sigma_o^*$ is *surely faulty* (resp. *surely correct*) if for every signalling run ρ with $\mathcal{P}(\sigma_\rho) = \sigma$, ρ is faulty (resp. correct); otherwise it is ambiguous. A (finite signalling or infinite) run ρ is *surely faulty* (resp. *surely correct*, *ambiguous*) if $\mathcal{P}(\rho)$ is surely faulty (resp. surely correct, ambiguous).

In order to specify various requirements for diagnosability we need to refine the notion of ambiguity. Let $\mathcal{N}$ be a POpLTS and $n \in \mathbb{N}$ with $n \geq 1$. Then:

- FAmb_∞ (resp. CAmb_∞) is the set of infinite faulty (resp. correct) ambiguous runs of $\mathcal{N}$;
- FAmb_n (resp. CAmb_n) is the set of infinite runs of $\mathcal{N}$ whose signalling subrun of observable length n is faulty (resp. correct) and ambiguous;

At this point it is interesting to look at the status of the different subsets of runs we have introduced with respect to the Borel hierarchy. The complementary sets F_n and C_n are unions of cylinders; so they are open (and by complementation) closed sets. The set of faulty (resp. correct) runs F (resp. C) is an open (resp. closed) set as a union (resp. intersection) of open (resp. closed) sets. The sets FAmb_n and CAmb_n are unions of cylinders; so they are open. The sets FAmb_∞ and CAmb_∞ may be defined as follows. Consider $(\Sigma_o^2)^\omega$ and Ω^2 both equipped with the product topology. $\mathsf{SameObs} = \{(\rho, \rho') \mid \mathcal{P}(\rho) = \mathcal{P}(\rho')\}$ is the inverse image by a continuous mapping of the closed set $\{(\sigma, \sigma) \mid \sigma \in \Sigma_o^\omega\}$. Therefore $\mathsf{SameObs}$ is closed.

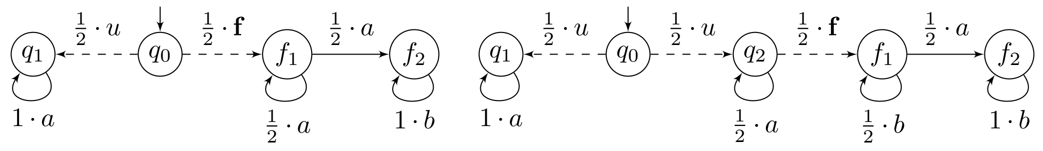


Figure 2 Left: a POpLTS that is IF-diagnosable but not IA-diagnosable. Right: a POpLTS that is IA-diagnosable but not FA-diagnosable.

Thus $\mathsf{C} \times \mathsf{F} \cap \mathsf{SameObs}$ is a Borel set. The first and second projections are exactly CAmb_∞ and FAmb_∞ which establishes that these sets are analytic sets (*i.e.* continuous images of Borel sets). The set of analytic sets is a strict superset of Borel sets but every analytic set is still measurable w.r.t. the complete measure [13, 2H8 p.83].

In the context of finite POpLTS, we introduced four possible specifications of diagnosability [3]. There are two discriminating criteria: whether the non ambiguity requirement holds for faulty runs only or for all runs, and whether ambiguity is defined at the infinite run level or for longer and longer finite signalling subruns.

▸ **Definition 4.** Let $\mathcal{N}$ be a POpLTS. Then:

- $\mathcal{N}$ is IF-*diagnosable* if $\mathbb{P}(\mathsf{FAmb}_\infty) = 0$.
- $\mathcal{N}$ is IA-*diagnosable* if $\mathbb{P}(\mathsf{FAmb}_\infty \uplus \mathsf{CAmb}_\infty) = 0$.
- $\mathcal{N}$ is FF-*diagnosable* if $\limsup_{n\to\infty} \mathbb{P}(\mathsf{FAmb}_n) = 0$.
- $\mathcal{N}$ is FA-*diagnosable* if $\limsup_{n\to\infty} \mathbb{P}(\mathsf{FAmb}_n \uplus \mathsf{CAmb}_n) = 0$.

We recall in the next theorem all the implications that hold between these definitions. Missing implications do not hold, already for finite-state POpLTS.

▸ **Theorem 5** ([3]). *Let $\mathcal{N}$ be a POpLTS. Then*

- *$\mathcal{N}$ FA-diagnosable $\Rightarrow$ $\mathcal{N}$ IA-diagnosable and FF-diagnosable;*
- *$\mathcal{N}$ IA-diagnosable or FF-diagnosable $\Rightarrow$ $\mathcal{N}$ IF-diagnosable;*
- *If $\mathcal{N}$ is finitely branching, then $\mathcal{N}$ is IF-diagnosable iff $\mathcal{N}$ is FF-diagnosable.*

In order to illustrate the different kinds of diagnosability, we describe below some discriminating examples, already presented in [3].

Consider the POpLTS $\mathcal{N}$ on the left of Figure 2 where $\{u, \mathbf{f}\}$ is the set of unobservable events (represented by dashed arrows) and $\mathcal{P}$ is the identity over the other events. A faulty run will almost surely produce a b-event that cannot be mimicked by the single correct run. Thus this POpLTS is IF-diagnosable. The unique correct run $\rho = q_0uq_1aq_1\dots$ has probability $\frac{1}{2}$ and its corresponding observed sequence a^ω is ambiguous. Thus the POpLTS is not IA-diagnosable. This simple example shows that, already for finite-state POpLTS, IF-diagnosability does not imply IA-diagnosability.

Similarly, let us look at the POpLTS on the right of Figure 2 where $\{u, \mathbf{f}\}$ is the set of unobservable events and $\mathcal{P}$ is the identity over the other events. Any infinite faulty run will contain a b-event, and cannot be mimicked by a correct run, therefore $\mathsf{FAmb}_\infty = \varnothing$. The two infinite correct runs have a^ω as observed sequence, and cannot be mimicked by a faulty run, thus $\mathsf{CAmb}_\infty = \varnothing$. As a consequence, this POpLTS is IA-diagnosable. Consider now the infinite correct run $\rho = q_0uq_1aq_1\dots$. It has probability $\frac{1}{2}$, and all its finite signalling subruns are ambiguous since their observed sequence is a^n, for some $n \in \mathbb{N}$. Thus for all $n \geq 1$, $\mathbb{P}(\mathsf{CAmb}_n) \geq \frac{1}{2}$, so that this POpLTS is not FA-diagnosable.

3 Characterisation of diagnosability

The aim of this section is to establish "simple" characterisations of the diagnosability notions for a POpLTS $\mathcal{N} = ((\mathcal{L}, \mathbf{P}), \Sigma_o, \mathcal{P})$ and more precisely to study whether one can express it as a Borel set $B \in \mathcal{B}$ only depending on the underlying LTS $\mathcal{L}$ and the mask function $\mathcal{P}$, such that almost surely a random run belongs to B if and only if $\mathcal{N}$ is diagnosable. Furthermore if possible, one looks for a set B belonging to a low level of the Borel hierarchy. Observe that for all notions, this requires some machinery since the finite runs-based notions FF and FA are expressed by a family of Borel sets and the infinite runs-based notions IF and IA are expressed by a set which is not *a priori* a Borel set.

Pursuing this goal, we introduce a language pathL for specifying Borel sets of runs. It is based on *path formulae.* A path formula α is a predicate over finite prefixes of runs. The (pseudo-)syntax of a formula of pathL is:

$$\phi ::= \alpha \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \Diamond\phi$$

where α is a path formula. In the sequel we use the standard shortcut $\Box\phi \equiv \neg\Diamond\neg\phi$.

A formula is evaluated at some position k of a run $\rho = q_0 a_0 q_1 \ldots$. The prefix $\rho[0,k]$ of ρ is defined by $\rho[0,k] = q_0 a_0 q_1 \ldots q_k$. The semantics of pathL is inductively defined by:

- $\rho, k \vDash \alpha$ if and only if $\alpha(\rho[0,k])$;
- $\rho, k \vDash \neg\phi$ if and only if $\rho, k \nvDash \phi$;
- $\rho, k \vDash \phi_1 \wedge \phi_2$ if and only if $\rho, k \vDash \phi_1$ and $\rho, k \vDash \phi_2$;
- $\rho, k \vDash \Diamond\phi$ if and only if there exists $k' \geq k$ such that $\rho, k' \vDash \phi$.

Finally $\rho \vDash \phi$ if and only if $\rho, 0 \vDash \phi$. Due to the presence of path formulae (with no restriction) this language subsumes LTL and more generally any ω-regular specification language. In order to reason about the probabilistic behaviour of a POpLTS, we introduce qualitative probabilistic formulae $\mathbb{P}^{\bowtie p}(\phi)$ with $\bowtie \in \{<, >, =\}$, $p \in \{0, 1\}$ and $\phi \in$ pathL. The semantics is obvious: $\mathcal{N} \vDash \mathbb{P}^{\bowtie p}(\phi)$ if and only if $\mathbb{P}_{\mathcal{N}}(\{\rho \in \Omega \mid \rho \vDash \phi\}) \bowtie p$. Since pathL is closed by complementation the probabilistic formulae can be restricted to $\mathbb{P}^{=0}(\phi)$ and $\mathbb{P}^{>0}(\phi)$.

Let us give some examples of path formulae. Given a finite run $\rho = q_0 a_0 q_1 \ldots q_k$, let $\mathfrak{f}$ be defined by $\mathfrak{f}(\rho) = \mathsf{true}$ if $a_i = \mathbf{f}$ for some index i. This path formula characterises the faulty finite runs. Let $\mathfrak{U}$ be defined by $\mathfrak{U}(\rho) = \mathsf{true}$ if there exists a correct signalling run ρ' with $\mathcal{P}(\rho) = \mathcal{P}(\rho')$. Using the path formulae $\mathfrak{f}$ and $\mathfrak{U}$, we exhibit a formula of pathL that characterises FF-diagnosability.

▸ **Proposition 6.** *Let $\mathcal{N}$ be a POpLTS. Then $\mathcal{N}$ is* FF*-diagnosable iff $\mathcal{N} \vDash \mathbb{P}^{=0}(\Diamond\Box(\mathfrak{f} \wedge \mathfrak{U}))$.*

Due to Theorem 5, in finitely-branching POpLTS the above characterisation also holds for IF-diagnosability. We also need the finitely-branching assumption in order to characterise IA-diagnosability. To this goal, let us introduce a more intricate path formula. For $\sigma \in \Sigma_o^*$, we define $\mathsf{firstf}(\sigma)$ by $\mathsf{firstf}(\sigma) = \min\{k \mid \exists\rho \text{ signalling run } \mathcal{P}(\rho) = \sigma \wedge \rho_{\downarrow k} \text{ is faulty}\}$ with the convention that $\min(\varnothing) = \infty$. Then the path formula $\mathfrak{W}$ is defined by: $\mathfrak{W}(\varepsilon) = \mathsf{false}$ and $\mathfrak{W}(q_0 a_0 \ldots q_{n+1}) = \mathsf{true}$ if $\mathsf{firstf}(\mathcal{P}(q_0 a_0 \ldots q_{n+1})) = \mathsf{firstf}(\mathcal{P}(q_0 a_0 \ldots q_n)) < \infty$.

▸ **Proposition 7.** *Let $\mathcal{N}$ be a finitely branching POpLTS. Then $\mathcal{N}$ is* IA*-diagnosable iff $\mathcal{N} \vDash \mathbb{P}^{=0}(\Diamond\Box(\mathfrak{U} \wedge \mathfrak{W}))$.*

The POpLTS of Figure 3 illustrates the need for the finitely-branching assumption in Proposition 7. The set of unobservable events is $\{u, \mathbf{f}\}$ and $\mathcal{P}$ is the identity over the other events. Observation b occurs in every infinite correct run, while the observed sequence of the single infinite faulty run is a^ω. This POpLTS is thus IA-diagnosable. However, it does not satisfy $\mathbb{P}^{=0}(\Diamond\Box(\mathfrak{U} \wedge \mathfrak{W}))$ since the unique infinite faulty run has probability $\frac{1}{2}$ and satisfies at

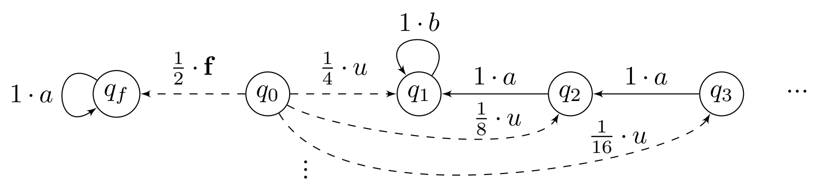


Figure 3 An infinitely-branching IA-diagnosable POpLTS.

the same time $\Box\mathfrak{W}$, by unicity, and $\Box\mathfrak{U}$. Indeed for every $n \in \mathbb{N}$, there is a correct signalling run with observed sequence a^n.

Observe that the sets of runs specified by the characterisations of FF-diagnosability ($\Diamond \Box (\mathfrak{f} \wedge \mathfrak{U})$) and IA-diagnosability ($\Diamond \Box (\mathfrak{U} \wedge \mathfrak{W})$) are F_σ sets, *i.e.* countable unions of closed sets. Surprisingly, we show that such a characterisation is impossible for FA-diagnosability: there is no F_σ set E such that a POpLTS $\mathcal{N}$ is FA-diagnosable if and only if $\mathcal{N} \models \mathbb{P}^{=0}(E)$.

▸ **Proposition 8.** *There exists a finitely-branching LTS $\mathcal{L}$ and a mask function $\mathcal{P}$ such that for every F_σ set E of runs, there exists a POpLTS $\mathcal{N} = ((\mathcal{L}, \mathbf{P}), \Sigma_o, \mathcal{P})$ such that:*

- *either $\mathcal{N}$ is FA-diagnosable and $\mathbb{P}_{\mathcal{N}}(E) > 0$;*
- *or $\mathcal{N}$ is not FA-diagnosable and $\mathbb{P}_{\mathcal{N}}(E) = 0$.*

We conjecture that the impossibility also holds for arbitrary Borel sets. The next proposition shows that a positive probability characterization cannot exist whatever the Borel set.

▸ **Proposition 9.** *There exists a finitely-branching LTS $\mathcal{L}$ and a mask function $\mathcal{P}$ such that for every Borel set E of runs, there exists a POpLTS $\mathcal{N} = ((\mathcal{L}, \mathbf{P}), \Sigma_o, \mathcal{P})$ such that:*

- *either $\mathcal{N}$ is FA-diagnosable and $\mathbb{P}_{\mathcal{N}}(E) = 0$;*
- *or $\mathcal{N}$ is not FA-diagnosable and $\mathbb{P}_{\mathcal{N}}(E) > 0$.*

4 Diagnosis for probabilistic pushdown automata

We now turn to a concrete model for infinite-state POpLTS, namely the ones generated by probabilistic pushdown automata, and more specifically by probabilistic visibly pushdown automata. Our goal is to use the characterisations from the previous section to decide the diagnosability of POpLTS generated by partially observable probabilistic visibly pushdown automata (POpVPA). To do so, we face the difficulty that the Borel sets that characterise IF-, IA- and IF-diagnosability are not *a priori* regular, even in the finite branching case. Yet, for POpVPA, we circumvent this problem, and manage to specify these sets by pLTL formulae on a determinisation of the model, tagged with the needed atomic propositions. The decidability of the qualitative model checking for recursive probabilistic systems [8] then yields the decidability of the above three diagnosability notions for POpVPA.

4.1 Probabilistic visibly pushdown automata

Among probabilistic infinite-state systems the ones generated by probabilistic pushdown automata [11, 8] support relevant decision procedures. Already in the non-probabilistic case, the subclass of visibly pushdown automata (VPA) [1] is more tractable than the general model. In VPA, the type of events determines whether the operation on the stack is a push, a pop, or possibly changes the top stack symbol, so that the languages defined by VPA enjoy most of the desirable properties regular languages have.

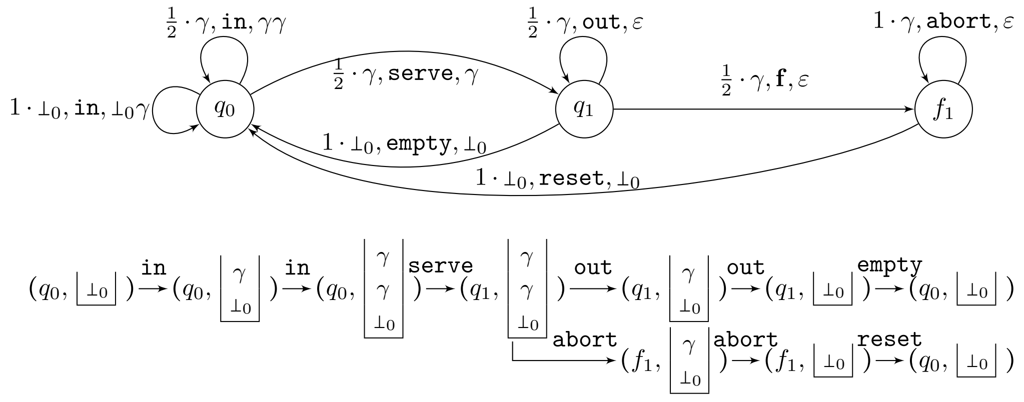

$\frac{1}{2}\cdot\gamma,$ in$,\gamma\gamma$
$\frac{1}{2}\cdot\gamma,$ out$,\varepsilon$
$1\cdot\gamma,$ abort$,\varepsilon$
$1\cdot\bot_0,$ in$,\bot_0\gamma$
$\frac{1}{2}\cdot\gamma,$ serve$,\gamma$
$\frac{1}{2}\cdot\gamma,$ f$,\varepsilon$
$1\cdot\bot_0,$ empty$,\bot_0$
$1\cdot\bot_0,$ reset$,\bot_0$
q_0
q_1
f_1
in
serve
out
empty
abort
reset

Figure 4 A pVPA generating the pLTS from Figure 1 with two finite runs.

▶ **Definition 10.** A *probabilistic visibly pushdown automaton* (pVPA) is a tuple $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, \mathbf{P})$ where:

- Q is a finite set of control states with q_0 the initial state;
- Σ is a finite alphabet of events, partitionned into local, push and pop events $\Sigma = \Sigma_\natural \uplus \Sigma_\sharp \uplus \Sigma_\flat$.
- Γ is a finite alphabet of stack symbols including a set of bottom stack symbols $\Gamma_\bot$ with initial symbol $\bot_0 \in \Gamma_\bot$;
- $\delta \subseteq Q \times \Gamma \times \Sigma \times Q \times \Gamma^*$ is the set of transitions such that for every $(q, \gamma, a, q', w) \in \delta$, $|w| \leq 2$, $\gamma \in \Gamma_\bot$ implies $w \in \Gamma_\bot(\Gamma \setminus \Gamma_\bot)^*$ and $\gamma \notin \Gamma_\bot$ implies $w \in (\Gamma \setminus \Gamma_\bot)^*$;
- $\mathbf{P}$ is the transition probability function fulfilling for every $q \in Q$ and $\gamma \in \Gamma$: $\sum_{(q,\gamma,a,q',w)\in\delta} \mathbf{P}[(q, \gamma, a, q', w)] = 1$.

A transition $t = (q, \gamma, a, q', w) \in \delta$ is said to be a *local* (resp. *push*, *pop*) transition if $|w| = 1$ (resp. $|w| = 2$, $|w| = 0$). We require that for every transition $t = (q, \gamma, a, q', w) \in \delta$, t is a local (resp. push, pop) transition iff a is a local (resp. push, pop) event.

The semantics of a pVPA is an infinite-state pLTS whose states are pairs (q, z) consisting of a control state and a stack contents.

▶ **Definition 11.** A pVPA $\mathcal{V} = (Q, \Sigma, \Gamma, \delta, \mathbf{P})$ defines a pLTS $\mathcal{M}_\mathcal{V} = (Q_\mathcal{V}, (q_0, \bot_0), \Sigma, T_\mathcal{V}, \mathbf{P}_\mathcal{V})$ where:

- $Q_\mathcal{V} = \{(q, z) \mid q \in Q \wedge z \in \Gamma_\bot(\Gamma \setminus \Gamma_\bot)^*\}$;
- $T_\mathcal{V} = \{((q, z\gamma), a, (q', zw)) \mid z\gamma \in \Gamma_\bot(\Gamma \setminus \Gamma_\bot)^* \wedge (q, \gamma, a, q', w) \in \delta\}$;
- For every $((q, z\gamma), a, (q', zw)) \in T_\mathcal{V}$, $\mathbf{P}_\mathcal{V}[((q, z\gamma), a, (q', zw))] = \mathbf{P}[(q, \gamma, a, q', w)]$.

▶ **Example 12.** Figure 4 gives an example of a pVPA. The event alphabet is composed of local events $\{$`serve`$,$ `empty`$,$ `reset`$\}$, a push event `in` and pop events $\{$`out`$,$ **f**$,$ `abort`$\}$. A transition $t = (q, \gamma, a, q', w)$ is represented by an edge from state q to state q' and labelled by $\mathbf{P}[t] \cdot \gamma, a, w$. The semantics of this pVPA is precisely the pLTS from Figure 1. Indeed, the stack alphabet consists of two letters $\Gamma = \{\gamma, \bot_0\}$ where the set of bottom stack symbol is $\Gamma_\bot = \{\bot_0\}$. Thus one can encode the stack using a counter that gives the number of γ in the stack. For instance, in the pLTS from Figure 1 the configuration $(q_1, \bot_0\gamma^n)$ of the pVPA corresponds to the state q_{1n}.

To define partially observable pVPA, we equip a pVPA with a mask function and require that only local events may be unobservable, and that pushes and pops can still be distinguished. This restriction is crucial since it ensures that the observed sequence of a signalling run of a POpVPA still provides the information about the height of the stack.

▸ **Definition 13.** A *partially observable pVPA* (POpVPA) is a tuple $\langle \mathcal{V}, \Sigma_o, \mathcal{P} \rangle$ consisting of a pVPA $\mathcal{V}$ equipped with a mapping $\mathcal{P} : \Sigma \to \Sigma_o \cup \{\varepsilon\}$ such that:

- $\Sigma_o = \Sigma_{o,\natural} \uplus \Sigma_{o,\sharp} \uplus \Sigma_{o,\flat}$ is the set of observations;
- $\mathcal{P}(\Sigma_\natural) \subseteq \Sigma_{o,\natural} \cup \{\varepsilon\}$, $\mathcal{P}(\Sigma_\sharp) \subseteq \Sigma_{o,\sharp}$ and $\mathcal{P}(\Sigma_\flat) \subseteq \Sigma_{o,\flat}$.

In the sequel, we may identify a POpVPA with the POpLTS it generates. In particular, the various concepts of diagnosability are lifted from POpLTS to POpVPA.

4.2 Diagnosability for POpVPA

To obtain an algorithm for the diagnosability of POpVPA, we follow the finite-state case approach [3]. First, we determinise POpVPA $\mathcal{V}$ into $\mathcal{A}(\mathcal{V})$, with the diagnosis objective in mind, building on the deterministic automaton recognising unambiguous sequences from [9]. We therefore introduce tags that reflect the category of runs (faulty or correct) given an observed sequence with a distinction between "old" and "young" faulty runs. It then suffices to check whether the characterisations hold on the synchronised product $\widehat{\mathcal{V}} \times \mathcal{A}(\mathcal{V})$ where $\widehat{\mathcal{V}}$ enlarges $\mathcal{V}$ by keeping track of a fault occurrence. To reduce to a decidable model checking question, we specify the Borel sets from Section 3 by LTL formulae.

4.2.1 Diagnosis-oriented determinisation

The determinisation of $\mathcal{V}$ (where probabilities are irrelevant for this transformation) into $\mathcal{A}(\mathcal{V})$ exploits some ideas of the original determinisation by Alur and Madhusudan [1], yet, it is customised to diagnosis. In particular, it uses tags that were first defined to construct a deterministic Büchi automaton recognising the unambiguous sequences of a finite LTS [9]. The complete definition of the estimate VPA $\mathcal{A}(\mathcal{V})$ associated with a POpVPA $\mathcal{V}$ is technical and detailed in [5]. We emphasise here some aspects of the construction and illustrate them on an example. Figure 5 represents the deterministic VPA associated with our example POpVPA. For readability, we use shortcuts on the transitions in this figure, namely symbols a_0^{X}, a_1^{X}, etc. denote stack symbols of $\mathcal{A}(\mathcal{V})$.

Figure 6 displays two finite runs of the deterministic VPA $\mathcal{A}(\mathcal{V})$ from Figure 5 sharing most transitions to the exception of the last one.

States and stack symbols. The VPA $\mathcal{A}(\mathcal{V})$ tracks all runs with the same observation in parallel memorising their status w.r.t. faults. More precisely to the current set of runs corresponds the symbol on the top of the stack which is a set of tuples where each tuple is written as a fraction $\frac{\gamma,\mathsf{X},q}{\gamma^-,\mathsf{X}^-,q^-}$. Let us describe the meaning of this tuple:

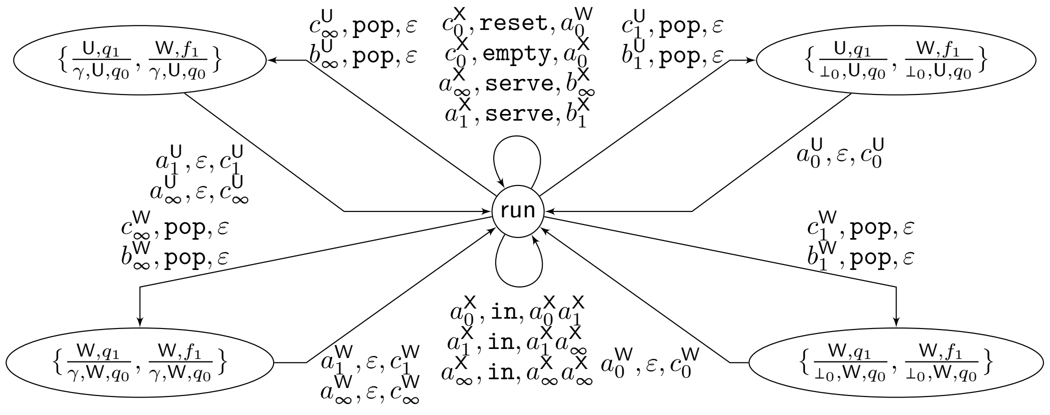
- q is the current state of the run and γ is the symbol on the top of its stack;
- $\mathsf{X} \in \mathsf{Tg} = \{\mathsf{U}, \mathsf{V}, \mathsf{W}\}$ is the status of the run: U for a correct run, V for a young faulty run and W for an old faulty run;
- The denominator $(\gamma^-, \mathsf{X}^-, q^-)$, is related to the configuration just after the last push event of the run: γ^- is the stack symbol under the top symbol, while X^- is the status of the run reaching this configuration and q^- the state of this configuration.

A priori, a single state run would be enough. However the simulation of a pop event in the original VPA is performed in two steps requiring some additional states that we explain later.

Illustration. The initial configuration of the VPA $\mathcal{A}(\mathcal{V})$ of Figure 5 $(\mathsf{run}, \left|\{\frac{\perp_0,\mathsf{U},q_0}{\perp_0,\mathsf{U},q_0}\}\right|)$ corresponds to the empty run represented by a singleton. The denominator of bottom stack symbols is by convention $(\perp_0, \mathsf{U}, q_0)$ and is irrelevant for specifying the transitions of $\mathcal{A}(\mathcal{V})$.

$$a_0^{\mathsf{X}} = \{\tfrac{\bot_0,\mathsf{X},q_0}{\bot_0,\mathsf{X},q_0}\}, a_1^{\mathsf{X}} = \{\tfrac{\gamma,\mathsf{X},q_0}{\bot_0,\mathsf{X},q_0}\}, a_\infty^{\mathsf{X}} = \{\tfrac{\gamma,\mathsf{X},q_0}{\gamma,\mathsf{X},q_0}\}, b_1^{\mathsf{X}} = \{\tfrac{\gamma,\mathsf{X},q_1}{\bot_0,\mathsf{X},q_0}\}, b_\infty^{\mathsf{X}} = \{\tfrac{\gamma,\mathsf{X},q_1}{\gamma,\mathsf{X},q_0}\}$$

$$c_0^{\mathsf{X}} = \{\tfrac{\bot_0,\mathsf{X},q_1}{\bot_0,\mathsf{U},q_0}, \tfrac{\bot_0,\mathsf{X},f_1}{\bot_0,\mathsf{U},q_0}\}, c_1^{\mathsf{X}} = \{\tfrac{\gamma,\mathsf{X},q_1}{\bot_0,\mathsf{X},q_0}, \tfrac{\gamma,\mathsf{X},f_1}{\bot_0,\mathsf{X},q_0}\}, c_\infty^{\mathsf{X}} = \{\tfrac{\gamma,\mathsf{X},q_1}{\gamma,\mathsf{U},q_0}, \tfrac{\gamma,\mathsf{X},f_1}{\gamma,\mathsf{U},q_0}\}, \qquad \mathsf{X} \in \{\mathsf{U},\mathsf{W}\}$$



run
c_∞^{U}, pop, ε
b_∞^{U}, pop, ε
c_0^{X}, reset, a_0^{W}
c_0^{X}, empty, a_0^{X}
a_∞^{X}, serve, b_∞^{X}
a_1^{X}, serve, b_1^{X}
c_1^{U}, pop, ε
b_1^{U}, pop, ε
$\{\frac{\mathsf{U},q_1}{\gamma,\mathsf{U},q_0}, \frac{\mathsf{W},f_1}{\gamma,\mathsf{U},q_0}\}$
$\{\frac{\mathsf{U},q_1}{\bot_0,\mathsf{U},q_0}, \frac{\mathsf{W},f_1}{\bot_0,\mathsf{U},q_0}\}$
$a_1^{\mathsf{U}}, \varepsilon, c_1^{\mathsf{U}}$
$a_\infty^{\mathsf{U}}, \varepsilon, c_\infty^{\mathsf{U}}$
c_∞^{W}, pop, ε
b_∞^{W}, pop, ε
$a_0^{\mathsf{U}}, \varepsilon, c_0^{\mathsf{U}}$
c_1^{W}, pop, ε
b_1^{W}, pop, ε
a_0^{X}, in, $a_0^{\mathsf{X}}a_1^{\mathsf{X}}$
a_1^{X}, in, $a_1^{\mathsf{X}}a_\infty^{\mathsf{X}}$
a_∞^{X}, in, $a_\infty^{\mathsf{X}}a_\infty^{\mathsf{X}}$
$\{\frac{\mathsf{W},q_1}{\gamma,\mathsf{W},q_0}, \frac{\mathsf{W},f_1}{\gamma,\mathsf{W},q_0}\}$
$a_1^{\mathsf{W}}, \varepsilon, c_1^{\mathsf{W}}$
$a_\infty^{\mathsf{W}}, \varepsilon, c_\infty^{\mathsf{W}}$
$a_0^{\mathsf{W}}, \varepsilon, c_0^{\mathsf{W}}$
$\{\frac{\mathsf{W},q_1}{\bot_0,\mathsf{W},q_0}, \frac{\mathsf{W},f_1}{\bot_0,\mathsf{W},q_0}\}$

Figure 5 The VPA $\mathcal{A}(\mathcal{V})$ associated with the POpVPA $\mathcal{V}$ of Figure 4.

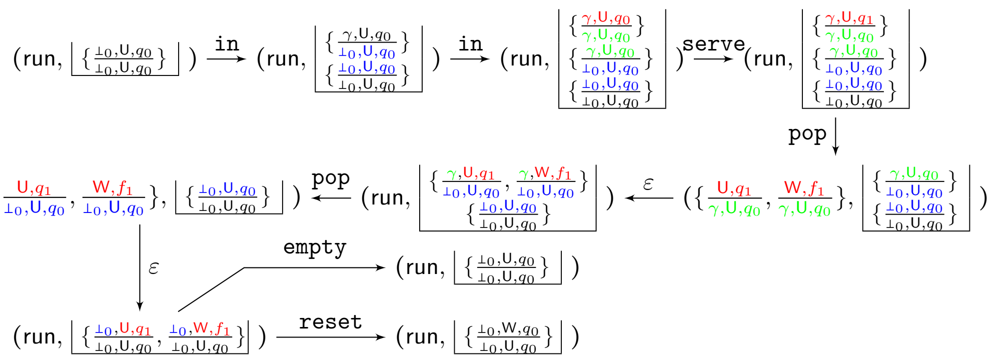


Figure 6 Two runs of the VPA from Figure 5.

Tag updates. Let us explain how the tag X of an item $\frac{\gamma,\mathsf{X},q}{\gamma^-,\mathsf{X}^-,q^-}$ of the current stack symbol is determined. If this item corresponds to a correct run then $\mathsf{X} = \mathsf{U}$. When, in a current state, after a transition of $\mathcal{A}(\mathcal{V})$ a (tracked) correct run becomes faulty in the next state, there are two cases. Either there was no tag W in (the numerators of items of) the top stack symbol of the current state then the run is tagged by W. Otherwise it is tagged by V meaning that it is a young faulty run. The tag V (young) becomes W (old) when, in the previous state, there was no tag W in the top stack symbol. A tag W is unchanged along the run.

Push transitions. Given an observed push event $o \in \Sigma_{o,\sharp}$, from the control state run with top stack symbol bel, there is a looping push transition $(\mathsf{run}, bel, o, \mathsf{run}, bel'bel'')$ in $\mathcal{A}(\mathcal{V})$ that encodes the possible signalling runs with observation o in $\mathcal{V}$. More precisely for every transition sequence $(q,\alpha) \overset{o}{\Longrightarrow} (r,\beta^-\beta)$ in $\mathcal{V}$ (*i.e.* a sequence of unobservable local events ending by an event e with $\mathcal{P}(e) = o$) and $\frac{\alpha,\mathsf{X},q}{\alpha^-,\mathsf{X}^-,q^-} \in bel$ one inserts $\frac{\beta^-,\mathsf{Y},r}{\alpha^-,\mathsf{X}^-,q^-}$ in bel' and $\frac{\beta,\mathsf{Y},r}{\beta^-,\mathsf{Y},r}$ in bel''. The value of Y follows the rules of tag updates.

Illustration. In Figure 5 several transitions correspond to the transition $(q_0, \bot_0, in, q_0, \bot_0\gamma)$ of $\mathcal{V}$, including $(\mathsf{run}, \{\frac{\bot_0,\mathsf{U},q_0}{\bot_0,\mathsf{U},q_0}\}, \mathtt{in}, \mathsf{run}, \{\frac{\bot_0,\mathsf{U},q_0}{\bot_0,\mathsf{U},q_0}\}\{\frac{\gamma,\mathsf{U},q_0}{\bot_0,\mathsf{U},q_0}\})$ and several transitions correspond to the transition $(q_0, \gamma, \mathtt{in}, q_0, \gamma\gamma)$ of $\mathcal{V}$, including $(\mathsf{run}, \{\frac{\gamma,\mathsf{U},q_0}{\bot_0,\mathsf{U},q_0}\}, \mathtt{in}, \mathsf{run}, \{\frac{\gamma,\mathsf{U},q_0}{\bot_0,\mathsf{U},q_0}\}\{\frac{\gamma,\mathsf{U},q_0}{\gamma,\mathsf{U},q_0}\})$. Here, the specification of the tag updates is straightforward since it does not involve faulty runs. The runs represented in Figure 6 use these two transitions from the initial state.

Local transitions. Given an observed local event $o \in \Sigma_{o,\natural}$, from the control state run with top stack symbol bel, there is a looping local transitions $(\mathsf{run}, bel, o, \mathsf{run}, bel')$ in $\mathcal{A}(\mathcal{V})$ that encodes the possible signalling runs with observation o in $\mathcal{V}$. More precisely for every transition sequence $(q, \alpha) \overset{o}{\Longrightarrow} (r, \beta)$ in $\mathcal{V}$ (*i.e.* a sequence of unobservable local events ended by an event e with $\mathcal{P}(e) = o$) and $\frac{\alpha,\mathsf{X},q}{\alpha^-,\mathsf{X}^-,q^-} \in bel$ one inserts $\frac{\beta,\mathsf{Y},r}{\alpha^-,\mathsf{X}^-,q^-}$ in bel'. The value of Y follows the rules of tag updates.
Illustration. In the VPA $\mathcal{A}(\mathcal{V})$ of Figure 5 there are several transitions corresponding to transition $(q_0, \gamma, \mathtt{serve}, q_1, \gamma)$ of $\mathcal{V}$ including $(\mathsf{run}, \{\frac{\gamma,\mathsf{U},q_0}{\gamma,\mathsf{U},q_0}\}, \mathtt{serve}, \mathsf{run}, \{\frac{\gamma,\mathsf{U},q_1}{\gamma,\mathsf{U},q_0}\})$. The runs represented in Figure 6 use this transition.
Pop transitions. Given an observed local event $o \in \Sigma_{o,\flat}$, from the control state run with top stack symbol bel, the "pop operation" is performed by a sequence of two transitions: a pop transition labelled by o that keeps in the next state all the information needed by the next (local) transition labelled by ε to move back to state run with a consistent stack symbol. Given an intermediate stack symbol, there is exactly one possible such transition. Thus despite these transitions, $\mathcal{A}(\mathcal{V})$ is still deterministic. The first transition $(\mathsf{run}, bel, o, \ell, \varepsilon)$ in $\mathcal{A}(\mathcal{V})$ is specified as follows. The next state ℓ is a set of items of the following shape $\frac{\mathsf{X},q}{\alpha^-,\mathsf{X}^-,q^-}$. More precisely for every transition sequence $(q, \alpha) \overset{o}{\Longrightarrow} (r, \varepsilon)$ in $\mathcal{V}$ (i.e. a sequence of unobservable local events ended by an event e with $\mathcal{P}(e) = o$) and $\frac{\alpha,\mathsf{X},q}{\alpha^-,\mathsf{X}^-,q^-} \in bel$ one inserts $\frac{\mathsf{Y},r}{\alpha^-,\mathsf{X}^-,q^-}$ in ℓ. The value of Y follows the rules of tag updates. A transition $(\ell, bel, \varepsilon, \mathsf{run}, bel')$ is specified as follows. For every $\frac{\mathsf{X}',q'}{\gamma,\mathsf{X},q}$ in ℓ and $\frac{\gamma,\mathsf{X},q}{\gamma^-,\mathsf{X}^-,q^-}$ in bel (*i.e.* the denominator of the first fraction and the numerator of the second fraction match), one inserts $\frac{\gamma,\mathsf{X}',q'}{\gamma^-,\mathsf{X}^-,q^-}$ in bel'.

Illustration. Let us describe how the $\mathtt{pop}$ event is performed by two transitions in the runs of the VPA of Figure 6 from the state reached after event $\mathtt{serve}$. From q_1 with γ as top of the stack there are two transitions whose observation is $\mathtt{pop}$: $(q_1, \gamma, \mathtt{out}, q_1, \varepsilon)$ and $(q_1, \gamma, \mathbf{f}, f_1, \varepsilon)$. Thus starting from run with top stack symbol $\{\frac{\gamma,\mathsf{U},q_1}{\gamma,\mathsf{U},q_0}\}$, one reaches state $\ell = \{\frac{\mathsf{U},q_1}{\gamma,\mathsf{U},q_0}, \frac{\mathsf{W},f_1}{\gamma,\mathsf{U},q_0}\}$. The faulty run is tagged with W as there was no tag W in the former top stack symbol. In the next configuration, the top stack symbol is $\{\frac{\gamma,\mathsf{U},q_0}{\bot_0,\mathsf{U},q_0}\}$. So the transition labelled by ε moves back to state run with updated top stack symbol $\{\frac{\gamma,\mathsf{U},q_1}{\bot_0,\mathsf{U},q_0}, \frac{\gamma,\mathsf{W},f_1}{\bot_0,\mathsf{U},q_0}\}$.

4.2.2 Product VPA

To recover the probabilistic behaviour of $\mathcal{V}$, we need to construct a synchronised product of $\mathcal{V}$ and the deterministic VPA $\mathcal{A}(\mathcal{V})$. In order to track the presence of a fault in a run of this product, we first enrich $\mathcal{V}$ to track occurrences of $\mathbf{f}$. We thus define the POpVPA $\widehat{\mathcal{V}}$ whose set of states $\widehat{Q}$ is a duplication of Q in correct states Q_c and faulty states Q_f. Given a transition of $\mathcal{V}$ starting from q leading to q', there is in $\widehat{\mathcal{V}}$ a transition starting from q_f leading to q'_f and a transition starting from q_c leading either to q'_c if the event is not $\mathbf{f}$ or to q'_f otherwise. We then construct $\mathcal{V}_{\mathcal{A}(\mathcal{V})} = \widehat{\mathcal{V}} \times \mathcal{A}(\mathcal{V})$ the product automaton of $\widehat{\mathcal{V}}$ and $\mathcal{A}(\mathcal{V})$ synchronised on the alphabet of observed events Σ_o. The transitions of $\widehat{\mathcal{V}}$ labelled by unobservable events do not change the second component of the state and the transitions of $\mathcal{A}(\mathcal{V})$ labelled by

ε do not change the first component of the state. Due to the determinism of $\mathcal{A}(\mathcal{V})$, $\mathcal{V}_{\mathcal{A}(\mathcal{V})}$ has the same probabilistic behaviour as the one of $\mathcal{V}$ except that it memorises additional information along the run. More precisely, let ρ be a run of $\mathcal{V}$, then $\bar{\rho}$, a run of $\mathcal{V}_{\mathcal{A}(\mathcal{V})}$, is obtained from ρ by following the same transitions and adding the single $\ominus$ transition firable after any pop transition. One immediately gets $\mathbb{P}_{\mathcal{V}_{\mathcal{A}(\mathcal{V})}}(\bar{\rho}) = \mathbb{P}_{\mathcal{V}}(\rho)$.

Let us explain how to transform the paths formulae $\mathfrak{f}$, $\mathfrak{U}$ and $\mathfrak{W}$ into atomic propositions on the pairs $((q, \mathsf{run})(\gamma, bel))$ consisting of a control state of $\mathcal{V}_{\mathcal{A}(\mathcal{V})}$ together with a top stack contents. For path formula $\mathfrak{f}$, we define the corresponding atomic proposition ν_f by $\nu_f((q, \mathsf{run})(\gamma, bel)) = \mathsf{true}$ if and only if $q \in Q_f$. Let $bel \subseteq (\Gamma \times \mathsf{Tg} \times Q)^2$, we say that X *occurs* in bel if there exists $\frac{\gamma, \mathsf{X}, q}{\gamma^-, \mathsf{X}^-, q^-} \in bel$. We define atomic propositions ν_u and ν_w by: $\nu_u((q, \mathsf{run})(\gamma, bel)) = \mathsf{true}$ if and only if U occurs in bel; and $\nu_w((q, \mathsf{run})(\gamma, bel)) = \mathsf{true}$ if and only if W occurs in bel.

Given a run ρ of $\mathcal{V}_{\mathcal{A}(\mathcal{V})}$, we write $\mathsf{last}(\rho)$ for the pair formed of the control state and top stack symbol in $\mathcal{V}_{\mathcal{A}(\mathcal{V})}$ after ρ. The atomic propositions ν_f and ν_u perfectly reflect the paths formula $\mathfrak{f}$ and $\mathfrak{U}$, and ν_w is eventually forever true if and only if $\mathfrak{W}$ is.

▸ **Proposition 14.** *Let ρ be an infinite run of $\mathcal{V}$. Then:*
- *For all $k \in \mathbb{N}$, $\mathfrak{f}(\rho_{\downarrow k}) \Leftrightarrow \nu_f(\mathsf{last}(\bar{\rho}_{\downarrow k}))$ and $\mathfrak{U}(\rho_{\downarrow k}) \Leftrightarrow \nu_u(\mathsf{last}(\bar{\rho}_{\downarrow k}))$;*
- *$\rho \models \Diamond \Box \mathfrak{W} \Leftrightarrow \exists K \forall k \geq K.\ \nu_w(\mathsf{last}(\bar{\rho}_{\downarrow k})) = \mathsf{true}$.*

4.2.3 Complexity of diagnosability for POpVPA

Thanks to the relationships between the paths formulae and the atomic propositions, and using the characterisations from Section 3, we manage to reduce the FF-, IF- and IA-diagnosis to the model checking of a pLTL formula on the product VPA $\mathcal{V}_{\mathcal{A}(\mathcal{V})}$. Model checking qualitative pLTL for probabilistic pushdown automata is achievable in polynomial space in the size of the model [8]. In our case, $\mathcal{V}_{\mathcal{A}(\mathcal{V})}$ is exponential in the size of $\mathcal{V}$. We thus obtain the decidability and a complexity upper-bound for the diagnosability problems for POpVPA.

▸ **Theorem 15.** FF*-diagnosability,* IF*-diagnosability and* IA*-diagnosability are decidable in* EXPSPACE *for POpVPA.*

Proof. For $\mathcal{V}$ a POpVPA, $\mathcal{V}$ and $\mathcal{V}_{\mathcal{A}(\mathcal{V})}$ have the same probabilistic behaviour. Therefore, using the relation between path formulae and atomic propositions from Proposition 14, we reformulate Propositions 6 and 7 into pLTL characterisations of diagnosability:
- $\mathcal{V}$ is FF-diagnosable iff $\mathcal{V}_{\mathcal{A}(\mathcal{V})} \models \mathbb{P}^{=0}(\Diamond \Box (\nu_f \wedge \nu_u))$;
- $\mathcal{V}$ is IA-diagnosable iff $\mathcal{V}_{\mathcal{A}(\mathcal{V})} \models \mathbb{P}^{=0}(\Diamond \Box (\nu_u \wedge \nu_w))$.

Moreover, since the POpLTS generated by POpPDA are finitely-branching, IF-diagnosability coincides with FF-diagnosability [3] (see also Theorem 5). The two above qualitative pLTL formulae can be checked on general probabilistic pushdown automata (beyond visibly pushdown ones) thanks to [8]. More precisely, one can transform $\mathcal{V}_{\mathcal{A}(\mathcal{V})}$ into a recursive Markov chain (the transformation is linear) [7]. Then, the model checking of qualitative pLTL on recursive Markov chains is doable in PSPACE in the size of the Recursive Markov Chain and EXPTIME in the size of the formulae [8]. In our case, the product VPA $\mathcal{V}_{\mathcal{A}(\mathcal{V})}$ is exponential in the size of $\mathcal{V}$ and the size of the formulae is constant. This yields an EXPSPACE algorithm for checking diagnosability of POpVPA. ◂

Reducing the universality problem for VPA, which is known to be EXPTIME-complete [1], we obtain the EXPTIME-hardness of all diagnosability variants for POpVPA.

▸ **Theorem 16.** *Diagnosability is* EXPTIME*-hard for POpVPA.*

The restriction to visibly pushdown automata is motivated by the unfeasibility of diagnosis for general probabilistic pushdown automata.

▸ **Theorem 17.** *Diagnosability is undecidable for POpPDA.*

The undecidability can be obtained by adapting the proof for diagnosis of *non-probabilistic* pushdown automata [12]. However, in order to show how robust the result is, we rather reduce from the Post Correspondence Problem and prove the undecidability of diagnosability for restricted classes of partially observable probabilistic pushdown automata. In particular, undecidability already holds for two (incomparable) subclasses of POpPDA [5] with restriction on what is observable and on the number of phases of any run, where a phase is a portion of run in which the stack either never decreases or never increases.

5 Conclusion

We studied the diagnosability problem for infinite-state probabilistic systems, both from a semantical perspective, and from an algorithmic one when considering probabilistic visibly pushdown automata. A natural research aim is to reduce the complexity gap for the diagnosability of POpVPA (currently EXPTIME-hard and in EXPSPACE). We could also investigate the diagnosability problem for other probabilistic extensions infinite state systems, such as lossy channel systems or VASS. Another research direction would be to consider the fault diagnosis problem for continuous-time probabilistic models, starting with CTMC.

References

1. R. Alur and P. Madhusudan. Visibly pushdown languages. In *Proc. STOC'04*, pages 202–211. ACM, 2004.
2. N. Bertrand, É. Fabre, S. Haar, S. Haddad, and L. Hélouët. Active diagnosis for probabilistic systems. In *Proc. FoSSaCS'14*, volume 8412 of *LNCS*, pages 29–42. Springer, 2014.
3. N. Bertrand, S. Haddad, and E. Lefaucheux. Foundation of diagnosis and predictability in probabilistic systems. In *Proc. FSTTCS'14*, volume 29 of *LIPIcs*, pages 417–429. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014.
4. N. Bertrand, S. Haddad, and E. Lefaucheux. Accurate approximate diagnosability of stochastic systems. In *Proc. LATA'16*, volume 9618 of *LNCS*, pages 549–561. Springer, 2016.
5. N. Bertrand, S. Haddad, and E. Lefaucheux. Diagnosis in infinite-state probabilistic systems (long version). Technical report, HAL Inria, 2016. `https://hal.inria.fr/hal-01334218`.
6. M. P. Cabasino, A. Giua, and C. Seatzu. Diagnosability of discrete-event systems using labeled Petri nets. *IEEE Trans. Automation Science and Engineering*, 11(1):144–153, 2014.
7. K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *Journal of the ACM*, 56(1), 2009.
8. K. Etessami and M. Yannakakis. Model checking of recursive probabilistic systems. *ACM Trans. Computational Logic*, 13(2):12, 2012.
9. S. Haar, S. Haddad, T. Melliti, and S. Schwoon. Optimal constructions for active diagnosis. In *Proc. FSTTCS'13*, volume 24 of *LIPIcs*, pages 527–539. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.
10. S. Jiang, Z. Huang, V. Chandra, and R. Kumar. A polynomial algorithm for testing diagnosability of discrete-event systems. *IEEE Trans. Automatic Control*, 46(8):1318–1321, 2001.
11. A. Kučera, J. Esparza, and R. Mayr. Model checking probabilistic pushdown automata. *Logical Methods in Computer Science*, 2(1), 2006.

12 C. Morvan and S. Pinchinat. Diagnosability of pushdown systems. In *Proc. HVC'09*, volume 6405 of *LNCS*, pages 21–33. Springer, 2009.

13 Y. N. Moschovakis. *Descriptive Set Theory.* Mathematical Surveys and Monographs. AMS, 2009.

14 M. Sampath, S. Lafortune, and D. Teneketzis. Active diagnosis of discrete-event systems. *IEEE Trans. Automatic Control*, 43(7):908–929, 1998.

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

16 D. Thorsley and D. Teneketzis. Diagnosability of stochastic discrete-event systems. *IEEE Trans. Automatic Control*, 50(4):476–492, 2005.