On the construction of probabilistic diagnosers ⋆

Eric Fabre * **Loig Jezequel** **

* *INRIA Rennes - Bretagne Atlantique, Campus de Beaulieu, 35042 Rennes cedex, France (e-mail: fabre@irisa.fr)*
** *ENS Cachan Bretagne, Rennes, France, (e-mail: loig.jezequel@eleves.bretagne.ens-cachan.fr)*

Abstract: This paper revisits the notions of observer and diagnoser, and adapts them to probabilistic automata, in a setting of weighted automata computations. In the non stochastic case, observers and diagnosers are obtained by standard elementary steps, as state augmentation, epsilon-reduction and determinization. It is shown that these steps can be adapted to probabilistic automata, and algorithms to perform them efficiently are provided. In particular, the determinization is related to a standard filtering equation that recursively computes the conditional distribution of the current state given past observations. New notions of probabilistic observers and diagnosers are provided and compared to previous constructions, and simpler derivations of the latter are proposed.

1. INTRODUCTION

Consider a plant modeled as a finite state machine (FSM) or automaton $\mathcal{A}$, with action alphabet Σ. As usual, we assume that the plant is partially observed, *i.e.* the actions in $\Sigma_o \subseteq \Sigma$ are observed when they fire, while those in $\Sigma_u = \Sigma \setminus \Sigma_o$ fire silently. When $\mathcal{A}$ runs, it produces the visible word $w \in \Sigma_o^*$ formed by concatenating the visible actions that were fired in this run. An *observer* of $\mathcal{A}$ is a deterministic FSM $\mathcal{O}$ on alphabet Σ_o, and a labelling function ϕ on its states, satisfying the following property: given any $w \in \Sigma_o^*$ produced by $\mathcal{A}$, let q be the unique state of $\mathcal{O}$ reached by following w, then the label $\phi(q) \subseteq S$ gives all possible states where $\mathcal{A}$ could be given that w has been observed. In other words, $(\mathcal{O}, \phi)$ is a state estimator of $\mathcal{A}$ under the form of a FSM. A *diagnoser* is a slightly more subtle notion. Assuming that some transitions of $\mathcal{A}$ are faulty and the others are safe, the diagnoser $\mathcal{D}$ is another deterministic FSM on alphabet Σ_o, with labelling function ψ on its states. On the unique state q of $\mathcal{D}$ reached by $w \in \Sigma_o^*$, the label $\psi(q)$ now gives a diagnosis value: "fault" if all trajectories of $\mathcal{A}$ that could have produced w contain a faulty transition, "safe" if none of them contain a faulty transition, and "uncertain" in the remaining cases. The diagnoser was initially proposed in Sampath et al. (1995) as a simple way to test diagnosability, *i.e.* the fact that a fault occurrence can be detected with a bounded number of observations after it has occurred. More efficient (polynomial) techniques were later proposed for this test in Yoo and Lafortune (2002), based on the so-called twin-machine, and avoiding the expensive construction of a diagnoser (exponential).

This paper examines the extension of the notions of observer and diagnoser to probabilistic automata: the objective is thus to build a deterministic FSM that outputs the *probability distribution* on states, or on the diagnosis value, given any observed sequence $w \in \Sigma_o^*$. Of course, for a given w, one could (and should) rather use recursive algorithms to compute these conditional distributions, for example the discrete event system version of the Kalman filter. Nevertheless, as "precompiled" versions of these filters, the probabilistic observers and diagnosers have some theoretical interest. At least for defining notions like probabilistic observability/diagnosability. Their construction sheds some light on techniques for performing elementary transforms on weighted automata, that in turn clarify previous contributions to the topic and allow their generalization to more complex settings.

Previous constructions of probabilistic diagnosers were proposed in Thorsley and Teneketzis (2005); Thorsley et al. (2008). They proceed by attaching transition probability matrices to a classical diagnoser. This is a half way to a true probabilistic diagnoser, since conditional distributions still have to be computed for a given $w \in \Sigma_o^*$ (while in a classical diagnoser, one just *reads out* the desired value). Our objective here is to show that the elementary steps in the derivation of an ordinary diagnoser can be recast in the setting of probabilistic automata, with minor and meaningful adaptations. In passing, this will clarify previous constructions, and will take us further by precomputing as well the conditional probabilities that can be reached. The probabilistic diagnoser proposed here is thus a standard probabilistic automaton, with transition weights taken in a well defined semiring, and of the same nature as $\mathcal{A}$, at the expense of not always being a finite state machine.

The paper first recalls basic steps on the construction of observers and diagnosers (Section 2), then studies their counterpart for weighted (probabilistic) automata, showing that they lead to the desired conditional distributions (Section 3). The relation to previous works is then detailed in Section 4.

⋆ This work was partly supported by the European Community's 7th Framework Programme under project DISC (DIstributed Supervisory Control of large plants), Grant Agreement INFSO-ICT-224498.

2. AUTOMATA, OBSERVERS, DIAGNOSERS

This section decomposes the construction of a diagnoser into three elementary steps, and shows that diagnosers and observers are similar objects.

2.1 Automata

Our starting point is a non-deterministic automaton $\mathcal{A} = (S, \Sigma, I, \delta)$, with S the state set, $I \subseteq S$ possible initial states, action alphabet Σ and transition function $\delta : S \times \Sigma \to 2^S$. The latter extends naturally into $\delta : 2^S \times \Sigma^* \to 2^S$ by union on the first variable and by iteration on the second. As usual, the action alphabet is partitioned into $\Sigma = \Sigma_o \uplus \Sigma_u$, observable and unobservable (or silent, or invisible) labels, resp. The transition set of $\mathcal{A}$ is denoted as $T = \{(s, \alpha, s') \in S \times \Sigma \times S : s' \in \delta(s, \alpha)\}$, and for a transition $t = (s, \alpha, s')$, we denote $s^-(t) = s, s^+(t) = s', \sigma(t) = \alpha$. A path or trajectory π of $\mathcal{A}$ is a sequence of transitions $\pi = t_1 \dots t_n$ such that $s^-(t_1) \in I$ and $s^+(t_i) = s^-(t_{i+1})$, for $1 \leq i < n$. We adopt notation $s^-(\pi) = s^-(t_1)$, $s^+(\pi) = s^+(t_n)$, $\sigma(\pi) = \sigma(t_1) \dots \sigma(t_n)$ and $\sigma_o(\pi) = \Pi_{\Sigma_o}(\sigma(\pi))$ where Π_{Σ_o} is the natural projection of Σ^* on Σ_o^*. The language of $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = \{\sigma(\pi) : \pi \text{ path of } \mathcal{A}\}$, and its observable language is $\mathcal{L}_o(\mathcal{A}) = \Pi_{\Sigma_o}(\mathcal{L}(\mathcal{A}))$.

2.2 Observer

Given $\Sigma = \Sigma_o \uplus \Sigma_u$, an observer of $\mathcal{A}$ is obtained by first performing an ϵ-reduction, and then determinizing the result: $Obs(\mathcal{A}) = Det(Red(\mathcal{A}))$.

ϵ-reduction. The ϵ-reduction $\mathcal{A}' = Red(\mathcal{A}) = (S, \Sigma_o, I', \delta')$ amounts to bypassing all transitions of $\mathcal{A}$ labeled by Σ_u (or equivalently the generic silent label ϵ). It can be performed either to the left (of visible events), or to the right. Without loss of generality, we present the latter here (see Fig. 1). It is defined by $\delta'(s, \alpha) = \delta(s, \alpha\Sigma_u^*) \triangleq \cup_{w \in \alpha\Sigma_u^*} \delta(s, w)$ where $\delta(s, w)$ is the extension by iteration of δ to the sequence of actions w: $\delta(s, wv) = \delta(\delta(s, w), v)$. For the initial states, one has $I' = \delta(I, \Sigma_u^*) \triangleq \cup_{s \in I} \delta(s, \Sigma_u^*)$. Observe that the resulting automaton $\mathcal{A}'$ has the same states as $\mathcal{A}$, operates on the reduced alphabet Σ_o, but is generally non-deterministic. By construction, one has $\mathcal{L}(\mathcal{A}') = \mathcal{L}_o(\mathcal{A})$.

The ϵ-reduction to the right accounts for the fact that after some visible label, system $\mathcal{A}$ may evolve silently and thus change state. It is thus suited for state estimation. For diagnosis purposes, the reduction to the left is often preferred, to account for properties of trajectories stopped immediately after the last observable event (see discussion below). Technically, this is a minor difference that does not impact the constructions below.

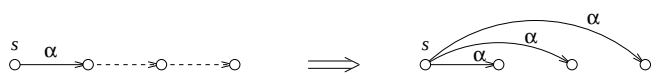

Figure 1. *Epsilon-reduction to the right. Dashed arrows represent silent (epsilon) transitions.*

Determinization. The determinization $\mathcal{A}'' = Det(\mathcal{A}') = (Q, \Sigma_o, q_0, \delta'')$ of $\mathcal{A}'$ is obtained by the standard subset construction. One has $Q = 2^S$, $q_0 = I'$, and the unique new state $q' = \delta''(q, \alpha)$ is defined as $q' = \delta'(q, \alpha) \triangleq \cup_{s \in q} \delta'(s, \alpha)$. Not all states in 2^S are reachable, so one often directly takes for Q the reachable part of 2^S, starting from $q_0 = I'$ and exploring recursively the $\delta'(q, \alpha)$ for all $\alpha \in \Sigma_o$ until no new q is found (Fig. 2). This step is known to have an exponential space complexity, in the worst case. Automaton $\mathcal{A}''$ directly yields a state estimator, or an observer of $\mathcal{A}$, by taking $\phi(q) = q$.

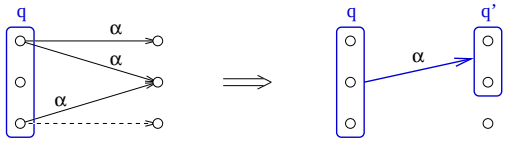

Figure 2. *Determinization. The dashed arrow represents a transition not labeled by α.*

2.3 Diagnoser

For diagnosers, one first associates types to the transitions of $\mathcal{A}$. This is done simply by setting $T = T_1 \cup \dots \cup T_K$, where each T_k gathers transitions of "type k". Notice that the T_k need not be disjoint, although the literature generally makes this assumption (Cassandras and Lafortune (1999); Sampath et al. (1995)): transition types are usually interpreted as distinct failure modes.

To build a diagnoser, the fisrt step consists in augmenting the states of $\mathcal{A}$ with some memory $\mu \subseteq \{1, \dots, K\}$ to keep track of transition types that have been fired along the trajectory. This yields $\bar{\mathcal{A}} = (\bar{S} = S \times 2^{\{1,\dots,K\}}, \Sigma, I \times \{\emptyset\}, \bar{\delta})$ where

$$(s', \mu') \in \bar{\delta}((s, \mu), \alpha) \Leftrightarrow \begin{cases} s' \in \delta(s, \alpha) \\ \mu' = \mu \cup \{k : (s, \alpha, s') \in T_k\} \end{cases} \quad (1)$$

Equivalently, this can be seen as computing the synchronous product of $\mathcal{A}$ with K elementary memory automata. The memory automaton for T_k only has two states 0 and 1, and $\{1, \dots, K\}$ as label set. It is deterministic and complete, and the only transition from 0 to 1 is labeled by k. Transitions of $\mathcal{A}$ must of course be relabeled by their type before the synchronous product can be computed, using types as labels. Details are left to the reader.

The second step simply consists in computing an observer for the augmented automaton $\bar{\mathcal{A}}$. Given $w \in \mathcal{L}_o(\mathcal{A}) \subseteq \Sigma_o^*$ and the unique state q reached by w in $Obs(\bar{\mathcal{A}})$, the K diagnosis functions ψ_k, $1 \leq k \leq K$, are defined by

$$\psi_k(q) = \begin{cases} T_k \text{ seen} & \text{if } \forall (s, \mu) \in q,\ k \in \mu \\ T_k \text{ not seen} & \text{if } \forall (s, \mu) \in q,\ k \notin \mu \\ T_k \text{ uncertain} & \text{otherwise} \end{cases} \quad (2)$$

Remarks

(1) This construction reveals that building a diagnoser boils down to building an observer. Without loss of generality, one can directly assume that states of $\mathcal{A}$ are partitioned into $S = S_1 \uplus \dots \uplus S_L$ with $L = 2^K$, corresponding to the 2^K possible values of the memory in $\bar{\mathcal{A}}$. The diagnosis then reduces to checking

whether all final states compatible with observation $w \in \mathcal{L}_o(\mathcal{A})$ lie into the union of some selected S_l.

(2) If one is only interested in diagnosing independently the occurrence of each T_k, it is simpler to build K diagnosers, one for each T_k, by augmenting $\mathcal{A}$ with a simpler binary memory. In terms of complexity, this saves an exponential in K. The diagnoser derived above is much more powerful, since it can also test for the simultaneous presence of several transition types in the trajectories explaining an observed word w.

(3) The ϵ-reduction to the left is often preferred to derive a diagnoser, since one is generally interested in the occurrence or not of some transition type before the last observation of w (and not necessarily in the silent moves that follow w). This is simply a matter of interpretation, since all silent transitions following w may impose that type T_k is fired at some point.

(4) Some contributions introduced so-called "observation filters" (Thorsley and Teneketzis (2005); Thorsley et al. (2008)): rather than a partition $\Sigma = \Sigma_o \uplus \Sigma_u$, one gives a filter $\lambda : S \times \Sigma \times S \to \Lambda \cup \{\epsilon\}$, and when $t = (s, \alpha, s')$ is fired, one label $\beta \in \lambda(t)$ is observed (possibly none if $\beta = \epsilon$). This does not change the expressive power of the model, that can be recoded in the classical setting by replacing (s, α, s') by (s, β, s') for every $\beta \in \lambda(t)$. The only difficulty introduced by such a recoding is that two versions of (s, β, s') may co-exist, one faulty and the other not. But this is captured by the possibility that a transition belong to several T_k.

3. PROBABILISTIC OBSERVERS

Given Remark (1) above, we limit ourselves to building probabilistic observers, assuming a partition $S = S_1 \uplus ... \uplus S_L$ on states of the probabilistic automaton $\mathcal{A}$. The goal is to derive another probabilistic automaton able to compute the probability that $\mathcal{A}$ reaches state type S_l, given some observed word $w \in \mathcal{L}_o(\mathcal{A})$. We show that this can be achieved as above, by extending ϵ-reduction and determinization to probabilistic automata.

3.1 Probabilistic automaton

We define it as $\mathcal{A} = (S, \Sigma, \mathbb{P}_0, \mathbb{P})$ where $\mathbb{P}_0 : S \to [0, 1]$ is an initial probability on states, with initial states $I = supp(\mathbb{P}_0)$ [1], and $\mathbb{P} : S \times \Sigma \times S \to [0, 1]$ a transition probability, *i.e.* $\forall s \in S$, $\mathbb{P}(s, \cdot, \cdot)$ is a probability distribution (over labels and next states, given the current state s). Transitions are given by $T = supp(\mathbb{P})$, and the transition function by $\delta(s, \alpha) = supp(\mathbb{P}(s, \alpha, \cdot))$. Notice that this definition assumes that $\mathcal{A}$ is live, for simplicity. One can generalize the setting to halting systems, by means of a special stopping label in Σ leading to a trap state s_t from which no more transition is allowed ($\mathbb{P}(s_t, \cdot, \cdot) = 0$). For a path $\pi = t_1...t_n$ one has $\mathbb{P}(\pi) = \mathbb{P}(t_1)...\mathbb{P}(t_n)$. And the language of $\mathcal{A}$ is defined as the formal power series $\mathcal{L}(\mathcal{A}) = \sum_{w \in \Sigma^*} \mathcal{L}(\mathcal{A}, w) \cdot w$ where $\mathcal{L}(\mathcal{A}, w) = \sum_{\pi,\, \sigma(\pi)=w} \mathbb{P}_0(s^-(\pi))\mathbb{P}(\pi)$.

3.2 Probabilistic observer

Given partitions $\Sigma = \Sigma_o \uplus \Sigma_u$ and $S = S_1 \uplus ... \uplus S_L$, the objective is to derive a deterministic probabilistic automaton $\mathcal{O} = (Q, \Sigma_o, \mathbb{P}_0^{\mathcal{O}}, \mathbb{P}^{\mathcal{O}})$, and a labeling $\phi : Q \to \mathcal{P}(L)$ of its states, where $\mathcal{P}(L)$ is the set of probability distributions over $\{1, ..., L\}$. Given $w \in \Sigma_o^*$ produced by $\mathcal{A}$, and $q \in Q$ the unique state reached by w in $\mathcal{O}$, we want $\phi(q, l) = \mathbb{P}(\mathcal{A}$ stops in $S_l \mid w$ was observed).

To specify the meaning of "stops," one needs an appropriate definition of stopping time. We adopt the following: $\mathcal{A}$ stops immediately before the production of the next observation, assuming $\mathcal{A}$ is Σ_o-live *i.e.* can produce a new observation with positive probability from any reachable state. Specifically, to make this sound as a true stopping time, $\mathcal{A}$ stops when it has been decided that the next step would produce an observation, but it is not yet decided which one [2]. This definition allows one to consume all silent steps after each observation. It contrasts with the usual choice of stopping immediately after an observable transition, which is slightly easier to handle and thus has often been chosen. It corresponds to the "optimistic" assumption that the system does not evolve silently by itself. Or at least that this evolution is ignored until there is evidence of it. Technically, the only impact is on the ϵ-reduction below, performed to the right (our case) instead of to the left.

We define the stopped language of $\mathcal{A}$ as follows: for a path π we take $\mathbb{P}^s(\pi) = \mathbb{P}_0(s^-(\pi))\mathbb{P}(\pi)\mathbb{P}(s^+(\pi), \Sigma_o, S)$, where $\mathbb{P}(s^+(\pi), \Sigma_o, S)$ is the probability of firing an observable transition from state $s^+(\pi)$. Then $\mathcal{L}^s(\mathcal{A}, w) = \sum_{\pi,\, \sigma(\pi)=w} \mathbb{P}^s(\pi)$ and the stopped language of $\mathcal{A}$ is $\mathcal{L}^s(\mathcal{A}) = \sum_{w \in \Sigma^*} \mathcal{L}^s(\mathcal{A}, w) \cdot w$. The observable language of $\mathcal{A}$ is given by $\mathcal{L}_o(\mathcal{A}) = \sum_{w \in \Sigma_o^*} \mathcal{L}_o(\mathcal{A}, w) \cdot w$ where

$$\mathcal{L}_o(\mathcal{A}, w) = \sum_{v \in \Sigma^*, \Pi_{\Sigma_o}(v) = w} \mathcal{L}^s(\mathcal{A}, v). \tag{3}$$

3.3 ϵ-reduction

We look for a probabilistic automaton $\mathcal{A}' = Red(\mathcal{A}) = (S, \Sigma_o, \mathbb{P}'_0, \mathbb{P}')$ such that $\mathcal{L}(\mathcal{A}') = \mathcal{L}_o(\mathcal{A}') = \mathcal{L}_o(\mathcal{A})$. Structurally, the automaton will be the same as in the non probabilistic case, and obtained by ϵ-reduction to the right. The difficulty lies in the computation of transition probabilities, since an unbounded number of silent steps may be performed until $\mathcal{A}$ decides to stop (and then fire a visible transition). This requires to integrate probabilities over a possibly infinite set of silent paths. Equivalently, the sum appearing in Eq. (3) above may contain infinitely many terms. This difficulty can be circumvented in at least two simple manners.

Method 1. The first one is through a set of three graph rewriting rules, depicted in Fig. 3. The first and main one (top) removes at once all M silent transitions going out of a given state (b). To an incoming transition $t_k = (a_k, \alpha_k, p_k, b)$ one adds direct jumps to states $c_1, ..., c_M$ with respective probabilities $p_k q_1, ..., p_k q_M$, all of them carrying the same label α_k (possibly ϵ). Then the probability of t_k is updated into $p'_k = p_k * (1 - q)$ where

[1] *supp* = support of

[2] For systems that have final states and stopping probabilities, one can choose to assimilate (or not) the choice to terminate in some state to the production of an observation, for the definition of the stopping time.

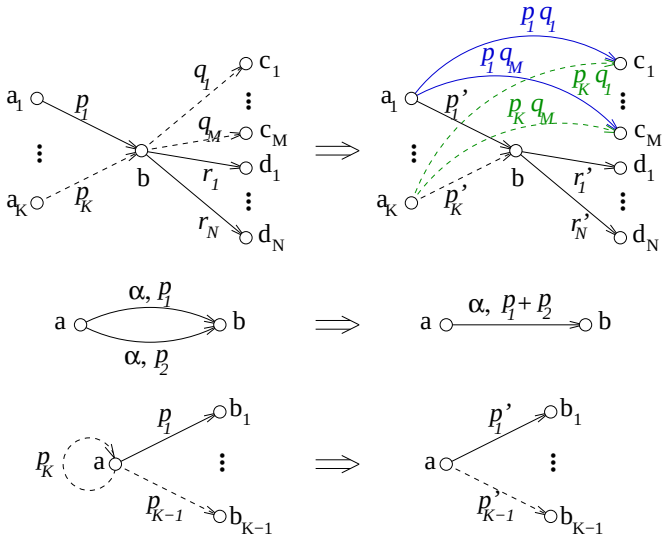

Figure 3. *Rewriting rules that perform the ϵ-reduction. Dashed lines represent silent transitions.*

$q = q_1 + ... + q_M$ is the probability to fire an ϵ-transition from b. One may have $p'_k = 0$, and in that case t_k vanishes. The probabilities of the N non-silent transitions going out of b, if there are any, are renormalized by $r'_n = r_n/(1-q)$ (notice that $N = 0$ when $q = 1$). The semantics of Rule 1 assumes that all transitions connected to b are processed at once; all transitions in the figure are distinct, but the states a_k, b, c_m, and d_n need not be different. Rule 1 also applies to update the initial probability, assuming for example that a_1 is a dummy initial node assigning to b its initial probability. The second rule (center) simply gathers two transitions that carry the same label into a single one, by summing their probabilities. This situation may occur after the application of Rule 1, and allows one to recover a graph depicting a true probabilistic automaton (technically, our definition does not capture duplicate transitions). Rule 2 applies also to silent transitions, to initial probabilities, and captures the case where a and b are the same state. Finally, Rule 3 (bottom) removes a silent loop of probability p_K at some state a, and renormalizes all the outgoing transitions of a, including silent ones, by $p'_i = p_i/(1-p_K)$. The Σ_o liveness of $\mathcal{A}$ guarantees that $p_K < 1$. Again, all outgoing transitions are distinct in the picture, but states $b_1, ..., b_{K-1}$ need not.

Theorem 1. Let automaton $\mathcal{A}^+$ be obtained from $\mathcal{A}$ by applying some rules of Fig. 3 (and such that Rule 2 is not applicable anymore). Then $\mathcal{L}_o(\mathcal{A}^+) = \mathcal{L}_o(\mathcal{A})$.

Proof. For this proof, we slightly enlarge the definition of a probabilistic automaton and allow the existence of several transitions with the same label between two given states. Denoting by $\mathcal{A}$ the initial automaton and by $\mathcal{A}^+$ the result obtained by firing one of the rules, the objective is to prove the preservation of the stopped language: $\mathcal{L}_o(\mathcal{A}) = \mathcal{L}_o(\mathcal{A}^+)$.

Clearly, Rule 2 does not change the stopped language of $\mathcal{A}$: for some $w \in \Sigma_o^*$, if $\mathcal{L}_o(\mathcal{A}, w)$ needs a path π of $\mathcal{A}$ using once the upper arrow (a, α, p_1, b), it needs as well path π' obtained by replacing this transition by (a, α, p_2, b), and conversely. The overall contribution to $\mathcal{L}_o(\mathcal{A}, w)$ of these two paths will be the same as if the two transitions are merged into $(a, \alpha, p_1 + p_2, b)$. The same reasoning holds for several consecutive uses of these transitions in a given path π.

We now examine Rule 3. Assume some path π_0 contributing to $\mathcal{L}_o(\mathcal{A}, w)$ crosses state a once and does not use transition $t_K = (a, \epsilon, p_K, a)$: so $\pi_0 = \pi'\pi''$, $s^+(\pi') = a = s^-(\pi'')$, where π'' contains at least one transition. Then all paths $\pi_n = \pi'(t_K)^n\pi''$ contribute as well to $\mathcal{L}_o(\mathcal{A}, w)$, and their total contribution is $\mathbb{P}_0(s^-(\pi'))\mathbb{P}(\pi')\frac{1}{1-p_K}\mathbb{P}(\pi'')\mathbb{P}(s^+(\pi''), \Sigma_o, S)$, which is what one would obtain after Rule 3 has been applied, since π'' is not empty: the coefficient $\frac{1}{1-p_K}$ is incorporated into the probability of the first transition of π''. If π_0 terminates in state a, that is if π'' is empty and $\pi_0 = \pi'$, let us assume that there exist visible transitions rooted at a (otherwise the contribution of π_0 to $\mathcal{L}_o(\mathcal{A}, w)$ vanishes). Then $\mathbb{P}^s(\pi_0) = \mathbb{P}_0(s^-(\pi_0))\mathbb{P}(\pi_0)\mathbb{P}(a, \Sigma_o, S)$. Again, the total contribution of the $\pi_n = \pi_0(t_K)^n$ will be $\mathbb{P}_0(s^-(\pi_0))\mathbb{P}(\pi_0)\frac{1}{1-p_K}\mathbb{P}(a, \Sigma_o, S)$, which is the same before and after the application of Rule 3, since the coefficient $\frac{1}{1-p_K}$ is introduced in the cost of the visible transitions rooted at state a. Again, the same reasoning holds if π_0 crosses several times state a.

Regarding Rule 1, the same reasonings can again be applied to paths crossing state b, and either stopping there or continuing their way. □

Corollary 2. Starting from $\mathcal{A}$, the rules in Fig. 3 have a unique stationary point $\mathcal{A}' = Red(\mathcal{A})$, which is the ϵ-reduction of $\mathcal{A}$.

Proof. When no more rule is applicable, one has the graph of a true probabilistic automaton (Rule 2 does not apply), where no node has silent outgoing transitions. Ignoring probabilities, Rules 1,2,3 clearly compute the ϵ-reduction (to the right) of the non-stochastic case, which gives uniqueness of $\mathcal{A}'$. From Theorem 1 one has $\mathcal{L}_o(\mathcal{A}') = \mathcal{L}_o(\mathcal{A})$, but since $\mathcal{A}'$ has no silent transition, one has $\mathcal{L}_o(\mathcal{A}') = \mathcal{L}(\mathcal{A})$. □

The ϵ-reduction by this method has complexity $\mathcal{O}(|S|^3)$.

Method 2. An alternate method to perform the ϵ-reduction consists in computing the probabilities $\mathbb{P}^\epsilon(s, s')$ for any $s, s' \in S$, probabilities to reach s' from s in $\mathcal{A}$ through silent trantisions:

$$\mathbb{P}^\epsilon(s, s') = \sum_{\substack{\pi,\ \sigma(\pi) \in \Sigma_u^* \\ s^-(\pi) = s,\ s^+(\pi) = s'}} \mathbb{P}(\pi) \qquad (4)$$

This is actually the difficult step where infinite sums may appear. Automaton $\mathcal{A}'$ is then obtained by $\mathbb{P}'(s, \alpha, s') = \sum_{s'' \in S} \mathbb{P}(s, \alpha, s'')\mathbb{P}^\epsilon(s'', s')\mathbb{P}(s', \Sigma_o, S)$, and with a similar equation for the initial probability $\mathbb{P}'_0$.

The transition matrix $\mathbb{P}^\epsilon$ can be obtained easily through a Floyd-Warshall procedure. The latter is usually applied to compute minimum distances between all pairs of nodes in a graph. By replacing the $(min, +)$ setting by the $(+, *)$ setting, one obtains a simple way to integrate probabilities over all paths relating two nodes (Mohri (2002); Cortes et al. (2006)). Specifically, denoting $S = \{s_1, ..., s_N\}$, one defines $\mathbb{P}^\epsilon_n(s, s')$ as in (4), excepted that the sum is

limited to paths that go through states in $\{s_1, ..., s_n\}$. So $\mathbb{P}_1^\epsilon(s, s') = \mathbb{P}(s, \Sigma_u, s')$. One then has

$$\mathbb{P}_{n+1}^\epsilon(s, s') = \mathbb{P}_n^\epsilon(s, s_{n+1})\mathbb{P}_n^\epsilon(s_{n+1}, s_{n+1})^*\mathbb{P}_n^\epsilon(s_{n+1}, s') \quad (5)$$

$$\mathbb{P}_n^\epsilon(s_{n+1}, s_{n+1})^* = \frac{1}{1 - \mathbb{P}_n^\epsilon(s_{n+1}, s_{n+1})} \quad (6)$$

where (6) represents the probability of performing an arbitrary number of loops at s_{n+1}. Again, the complexity of the ϵ-reduction by this method is $\mathcal{O}(|S|^3)$.

3.4 Determinization

The determinization of a probabilistic automaton $\mathcal{A}' = (S, \Sigma_o, \mathbb{P}'_0, \mathbb{P}')$ can be derived from the standard determinization procedure of weighted automata, that adapts the recursive subset construction given in the previous section (Mohri (2009); Kirsten and Murer (2005); Buchsbaum et al. (1998)). One has $\mathcal{A}'' = Det(\mathcal{A}') = (Q, \Sigma_o, \mathbb{P}''_0, \mathbb{P}'')$ where $Q \subset 2^{S\times[0,1]}$ and can be infinite. $\mathbb{P}''_0$ assigns probability 1 to the unique state $q_0 = \{(s, \mathbb{P}'_0(s)) : s \in supp(\mathbb{P}'_0)\}$. Successive states are obtained recursively as follows. Let $q = \{(s_1, p_1), ..., (s_M, p_M)\} \in Q$ and $\alpha \in \Sigma_o$, one has $\delta''(q, \alpha) = q' = \{(s'_1, p'_1), ..., (s'_N, p'_N)\}$ iff $\{s'_1, ..., s'_N\} = \delta'(\{s_1, ..., s_M\}, \alpha) \neq \emptyset$, and for $1 \leq n \leq N$

$$p''_n = \sum_{1\leq m\leq M} p_m \cdot \mathbb{P}'(s_m, \alpha, s'_n) \quad (7)$$

$$p'_n = p''_n/C \quad \text{where} \quad C = \sum_{1\leq k\leq N} p''_k \quad (8)$$

$$\mathbb{P}''(q, \alpha, q') = C \quad (9)$$

Proposition 3. Let $\delta''(q_0, w) = q = \{(s_1, p_1), ..., (s_M, p_M)\} \in Q$ in $\mathcal{A}''$ for some $w \in \Sigma_o^*$, then

$$p_m = \mathbb{P}(\mathcal{A}' \text{ is in state } s_m | w \text{ was observed}) \quad (10)$$

Proof. This is obviously true at q_o for $w = \epsilon$. Assume it is true at $q = \delta''(q_o, w)$ and let $q' = \delta''(q, \alpha)$. Eq. (7) is a standard filtering equation for $\mathcal{A}'$ (based on Bayes rule and the Markov property), so p''_n is the probability that $\mathcal{A}'$ produces $\alpha \in \Sigma_o$ and reaches state $s_n \in S$ given that w was observed. Consequently, C is the probability to fire α given w was observed, and the p'_n give the conditional probability of the current state of $\mathcal{A}'$ given the observed sequence $w\alpha$. □

As a corollary, if $\mathcal{A}' = Red(\mathcal{A})$, p_m is also the probability that $\mathcal{A}$ stops in s_m given that w was observed, which almost makes $\mathcal{A}''$ an observer. Since one reads in q the conditional distribution on S given some observation $w \in \Sigma_o^*$, one easily derives the conditional distribution over the indexes $\{1, ..., L\}$ corresponding to the partition $S = S_1 \uplus ... \uplus S_L$. Notice also that $\mathcal{L}(\mathcal{A}'') = \mathcal{L}(\mathcal{A}') = \mathcal{L}_o(\mathcal{A})$.

Example. Figure 4 illustrates the determinization procedure. This simple example seems to suggest that the conditional probabilities appear as extra information attached to a standard (*i.e.* non-probabilistic) observer. This is not the case, and the determinization procedure may very well not terminate, as revealed by the counter-example in Fig. 5. While for weighted automata taking values in

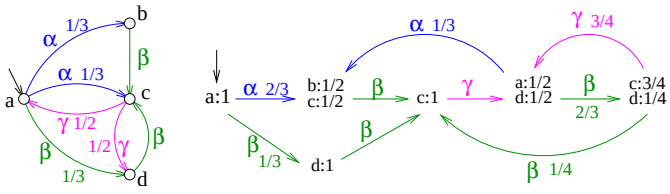

Figure 4. *A probabilistic automaton (left) and its determinized version (right).*

the $(\mathbb{R}^+, \min, +)$ semiring there exist sufficient conditions to guarantee termination (see the twin property in Mohri (1997)), to our knowledge it is still not clear what these conditions could be for probabilistic automata.

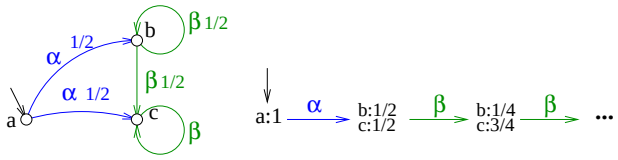

Figure 5. *Determinization may not terminate.*

3.5 Probabilistic diagnoser

Using the same technique as in the previous section, a probabilistic diagnoser for $\mathcal{A}$ is nothing else than a probabilistic observer on an augmented automaton $\bar{\mathcal{A}}$, that keeps track of which transitions types have been crossed along the run of $\mathcal{A}$:

$$\bar{\mathbb{P}}((s, \mu), \alpha, (s', \mu')) = \mathbb{P}(s, \alpha, s') \cdot \mathbb{1}_{\mu' = \mu \cup \{i:(s,\alpha,s')\in T_i\}} \quad (11)$$

From the conditional distribution on states of $\bar{\mathcal{A}}$ given some observation $w \in \Sigma_o^*$, one then easily derives the conditional distribution on memory values μ, and further on transition classes T_k that were crossed by $\mathcal{A}$.

Remark. The case of "observation filters," that randomly modify the labels of Σ produced by transitions of $\mathcal{A}$, can be processed in a similar manner as in Remark 4 of Section 2.3. The slight difference here is that a given observed label $\beta \in \Lambda \cup \{\epsilon\}$ may correspond to several underlying transition types T_k, that have different probabilities. This case is captured simply as follows: one replaces the deterministic memory represented by the $\mathbb{1}$ term in (11) by a "randomized" memory. Specifically, given $T = T_1 \cup ... \cup T_K$ and for $\mu' = \mu \uplus \mu$", (11) becomes $\bar{\mathbb{P}}((s, \mu), \beta, (s', \mu')) = \mathbb{P}(s, \beta, s') \cdot \mathbb{P}(\bigwedge_{k\in\mu"} T_k \wedge \bigwedge_{k\notin\mu'} \bar{T}_k \,|(s, \beta, s'))$. The first term is the probability to move from s to s' and produce label β, the second one is the (conditional) probability that this move crosses a transition lying in all T_k for $k \in \mu$", and in none of the T_k for $k \notin \mu'$.

4. RELATION TO PREVIOUS WORK

Alternate probabilistic diagnosers were proposed by Thorsley and Teneketzis (2005), as a way to extend the definition of diagnosability to the stochastic case. The latter expresses that, after some failure has occurred, the probability that it is not detected tends to zero as the number of observations increases. The authors also assumed noisy observations, *i.e.* random observation masks,

that randomly turn the label of a transition into some observed one (possibly ϵ, to account for a loss). As we have shown, both random observations and the specific aspects of diagnosis introduce no extra difficulty: the problem is essentially that of building a probabilistic observer, and the diagnosability issue can be expressed as an observability issue. Reformulated as such, the contruction proposed in Thorsley and Teneketzis (2005) corresponds to a standard (non-stochastic) observer, enriched with transition probability matrices. Specifically, between states $q = \{s_1, ..., s_M\}$ and $q' = \{s'_1, ..., s'_N\}$ of the observer, and for a transition (q, α, q'), the authors compute the $M \times N$ transition matrix containing elements $\mathbb{P}(s'_n, \alpha | s_m)$. The latter corresponds to the probability of jumping from s_m to s_n in $\mathcal{A}$ by first crossing some ϵ-transitions and then a visible transition producing α. So this corresponds to an ϵ-reduction to the left, or equivalently to taking as stopping time the firing of a visible transition. Technically, the probabilistic observer obtained in that way can be considered as a weighted automaton where the weight of each transition lies in a semiring of matrices. The advantage of this probabilistic observer is of course its finiteness. The drawback is that processings at this stage are not finished, and the diagnosability/observability test then amounts to exploring the recurrent components of Markov chains defined with the above transition probabilities. By contrast, the definition of probabilistic observer provided here is based on standard constructions for the ϵ-reduction and for the determinization of weighted automata. And we have simply emphasized that the recursion computing the determinization was exactly the stochastic filtering equation that one needs in order to compute posterior probabilities on states given observations. The drawback is of course that the construction may not yield a finite structure. It is not clear which definition of a probabilistic observer is best to check observability. Probably a mixture of both. Unless smarter (*i.e.* less complex) strategies can be found, as it is the case in the non-probabilistic setting.

For technical reasons, Thorsley and Teneketzis (2005) was limited to stochastic automata without unobservable cycles (and deterministic once ϵ-transitions are removed). These limitations were relaxed in Thorsley et al. (2008), although the construction is still hard to follow. The essential difficulty lies in the computation of the ϵ-reduction. The latter is classical for weighted automata. See for example Mohri (2002) for a generic form in the case of $k-$closed semirings, which unfortunately do not capture the case of probabilistic automata. More precise complexity bounds can also be found there. The idea of recycling minimum-length algorithms, that explore all paths, into integration algorithms, that sum quantities over all paths, is however present in Cortes et al. (2006), and adapts well to probabilistic automata. We have also proposed here a somehow simpler graphical way to perform this reduction. The determinization was proposed in Mohri (1997, 2009), see also Buchsbaum et al. (1998); Kirsten and Murer (2005). The finiteness of the resulting automaton, that is the "determinizability," depends very much on the nature of the underlying semiring. In the much studied tropical semiring $(\mathbb{R}^+, \min, +)$, sufficient conditions like the twin property guarantee that the algorithm terminates. However, to our knowledge, sufficient conditions that would be permissive enough are still missing for probabilistic automata.

5. CONCLUSION

We have shown that the construction of a diagnoser could be split into elementary steps, as the removal of observation filters, the introduction of memory, and then the construction of an observer. The latter being obtained by performing an ϵ-reduction, followed by a determinization. These operations translate almost immediately to the stochastic case, by considering probabilistic automata as weighted automata taking values in a specific semiring. We have shown that the ϵ-reduction could be performed in two ways, and proposed the less frequent reduction to the right, that allows the system to evolve silently after each observation, which was not the case in existing settings. We have proposed an alternate solution to compute the ϵ-reduction, and shown that the standard determinization procedure for weighted automata actually implements a filtering equation. Hopefully, this decomposition will allow one to get a clearer view on where the true difficulties lie in such constructions, and on what extensions can be reached simply. For example, by changing the $(\mathbb{R}, +, \times)$ semiring for $(\mathbb{R}, \max, +)$, one can easily derive another notion of diagnoser that would provide the maximum probability diagnosis given some observed string.

REFERENCES

Buchsbaum, A.L., Giancarlo, R., and Westbrook, J.R. (1998). On the determinization of weighted finite automata. *SIAM J. Comput*, 30, 2000.

Cassandras, C.G. and Lafortune, S. (1999). *Introduction to Discrete Event Systems.* Kluwer Academic Publishers.

Cortes, C., Mohri, M., Rastogi, A., and Riley, M. (2006). Efficient computation of the relative entropy of probabilistic automata. In *LATIN'06, LNCS 3887*, 323–336. Springer-Verlag.

Kirsten, D. and Murer, I. (2005). On the determinization of weighted automata. *Journal of Automata, Languages and Combinatorics*, 10(2/3), 287–312.

Mohri, M. (1997). Finite-state transducers in language and speech processing. *Computational Linguistics*, 23, 269–311.

Mohri, M. (2002). Generic epsilon-removal and input epsilon-renormalization algorithms for weighted transducers. *Int. J. of Foundations of Computer Sciences*, 13(1), 129–143.

Mohri, M. (2009). Weighted automata algorithms. *Handbook of weighted automata, Werner Kuich, Heiko Vogler, and Manfred Droste Edit., Springer.*

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

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

Thorsley, D., Yoo, T., and Garcia, H. (2008). Diagnosability of stochastic discrete-event systems under unreliable observations. In *American Control Conference*, 1158–1165.

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