Chapter 5
Diagnosis and Automata

Eric Fabre

5.1 Diagnosis vs. State Estimation

Automaton. Our starting point is a nondeterministic automaton $\mathscr{A} = (S, \Sigma, I, \delta)$, with S the finite 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 one. As usual, the action alphabet is partitioned into $\Sigma = \Sigma_o \uplus \Sigma_u$, observable and unobservable (or silent, or invisible) labels, respectively The transition set of $\mathscr{A}$ 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 $s^-(t) = s, s^+(t) = s'$, and $\sigma(t) = \alpha$. If $|I| = 1$ and $\forall (s, \alpha) \in S \times \Sigma$, $|\delta(s, \alpha)| \leq 1$, automaton $\mathscr{A}$ is said to be deterministic.

A path or trajectory π of $\mathscr{A}$ 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 notations $s^-(\pi) = s^-(t_1)$, $s^+(\pi) = s^+(t_n)$, $|\pi| = n$, the length of π, $\sigma(\pi) = \sigma(t_1) \dots \sigma(t_n)$ 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 $\mathscr{A}$ is $\mathscr{L}(\mathscr{A}) = \{\sigma(\pi) : \pi \text{ path of } \mathscr{A}\}$, and its observable language is $\mathscr{L}_o(\mathscr{A}) = \Pi_{\Sigma_o}(\mathscr{L}(\mathscr{A}))$. $\mathscr{A}$ is Σ_o-live iff from every state of $\mathscr{A}$, one can reach a transition labeled by Σ_o.

State estimation. Assume the system $\mathscr{A}$ performs some hidden run π, over which one only gets a partial knowledge by means of the observed sequence of labels $w = \sigma_o(\pi)$ produced by π. A natural question is *'What are the possible current states of $\mathscr{A}$ given that w was observed?'* So one wishes to build a function $f : \Sigma_o^* \to 2^S$ such that $f(w) = \{s^+(\pi) : \pi \text{ path of } \mathscr{A}, \sigma_o(\pi) = w\}$. There exists an obvious way of building f, recusively on the length of π: from $f^n : \Sigma_o^n \to 2^S$ solving the problem for observed sequences of length n, one can derive f^{n+1} by the so-called 'guided simulation' of $\mathscr{A}$.

Eric Fabre
INRIA Rennes Bretagne Atlantique, Campus de Beaulieu, 35042 Rennes cedex, France
e-mail: `eric.fabre@inria.fr`

Alternately, one can build an *observer* $Obs(\mathscr{A})$ of $\mathscr{A}$ as a pair $(\mathscr{O}, \phi)$, where $\mathscr{O} = (Q, \Sigma_o, q_0, \delta')$ is a *deterministic* automaton over alphabet Σ_o, and $\phi : Q \to 2^S$ is a label function over its states. For any observed $w \in \Sigma_o^*$, let $q = \delta'(q_0, w)$ be the unique state reached by w in $\mathscr{O}$, then ϕ satisfies $\phi(q) = f(w) \subseteq S$. An observer can thus be seen as a precompiled and finite version of the recursive function f. The derivation of observers is detailed later in this chapter, which also examines some of their properties.

Diagnosis. The problem is usually stated as follows. One first associates types to the transitions of $\mathscr{A}$. This is done simply by setting $T = T_1 \cup \ldots \cup T_K$, where each T_k gathers transitions of 'type k'. Sets T_k need not be disjoint, although the literature generally makes this assumption [3, 15]: transition types are usually interpreted as distinct failure modes, with one of them, say T_1, representing the 'safe' (*i.e.* non faulty) transitions. On wishes to build K diagnosis functions $f_k : \Sigma_o^* \to \{y, a, n\}$, $1 \leq k \leq K$, such that

$$f_k(w) = \begin{cases} y & \text{if } \forall \pi \in \sigma_o^{-1}(w), \pi \notin (T \setminus T_k)^* \\ n & \text{if } \forall \pi \in \sigma_o^{-1}(w), \pi \in (T \setminus T_k)^* \\ a & \text{otherwise} \end{cases} \tag{5.1}$$

In other words, $f_k(w)$ answers 'yes' if all runs of $\mathscr{A}$ explaining w use a transition of T_k, it answers 'no' if none of these runs uses a transition of T_k, and answers 'ambiguous' otherwise. A *diagnoser* of $\mathscr{A}$ is now a pair $(\mathscr{D}, \psi)$ where $\mathscr{D} = (Q, \Sigma_o, q_0, \delta')$ is again a deterministic automaton over alphabet Σ_o, and $\psi : \{1, ..., K\} \times Q \to \{y, a, n\}$ is a (collection of) label function(s) over its states. For any observed $w \in \Sigma_o^*$, let $q = \delta'(q_0, w)$ be the unique state reached by w in $\mathscr{D}$, then $\psi(k, q) = f_k(w)$, $1 \leq k \leq K$. A diagnoser is thus a finite and precompiled version of the K diagnosis functions.

Relation between the two problems. Despite an apparent difference, observers and diagnosers are similar objects. To build a diagnoser, the first step consists in augmenting the states of $\mathscr{A}$ with some memory $\mu \subseteq \{1, ..., K\}$ to keep track of transition types that are fired by $\mathscr{A}$ along its trajectory. This yields $\bar{\mathscr{A}} = (\bar{S} = S \times 2^{\{1,...,K\}}, \Sigma, I \times \{\emptyset\}, \bar{\delta})$ where

$$(s', \mu') \in \bar{\delta}((s, \mu), \alpha) \Leftrightarrow \begin{cases} s' \in \delta(s, \alpha) \\ \mu' = \mu \cup \{k : (s, \alpha, s') \in T_k\} \end{cases} \tag{5.2}$$

In words, this 'state augmentation trick' does the following: as soon as $\mathscr{A}$ fires a transition of T_k, the memory set μ stores index k (forever). Equivalently, the above construction can be seen as computing the synchronous product (see Section 5.5 for a definition) of $\mathscr{A}$ with K elementary memory automata[1].

[1] The memory automaton for T_k only has two states 0 and 1, and $\{1, ..., K\}$ as label set. It is deterministic and complete, and the only transition from 0 to 1 is labeled by k. Transitions of $\mathscr{A}$ 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.

Observe now that deriving a diagnoser for $\mathscr{A}$ is equivalent to computing an observer for the augmented automaton $\bar{\mathscr{A}}$. Given $w \in \mathscr{L}_o(\mathscr{A}) \subseteq \Sigma_o^*$ and the unique state q reached by w in $Obs(\bar{\mathscr{A}})$, the diagnosis function ψ is then given by

$$\psi(k,q) = \begin{cases} y & \text{if } \forall (s,\mu) \in q,\ k \in \mu \\ n & \text{if } \forall (s,\mu) \in q,\ k \notin \mu \\ a & \text{otherwise} \end{cases} \tag{5.3}$$

Given this similarity, the chapter focuses on the derivation of observers and will only mention diagnosers to comment specific aspects.

5.2 Observer and Diagnoser

Given $\Sigma = \Sigma_o \uplus \Sigma_u$, an observer of $\mathscr{A}$ is obtained by first performing an ε-reduction, and then determinizing the result: $Obs(\mathscr{A}) = Det(Red(\mathscr{A}))$. The label function ϕ is trivial (the identity), and thus can be omitted.

ε-reduction. The ε-reduction $\mathscr{A}' = Red(\mathscr{A}) = (S, \Sigma_o, I', \delta')$ amounts to bypassing all transitions of $\mathscr{A}$ labeled by Σ_u (or equivalently the generic silent label ε). It can be performed either to the left of visible transitions, or to their right. Without loss of generality, in this chapter we focus on the reduction to the right. The ε-reduction to the right (see Fig. 5.1) is defined by $\delta'(s,\alpha) = \delta(s, \alpha\Sigma_u^*) = \cup_{w \in \alpha\Sigma_u^*} \delta(s,w)$. For the initial states, one has $I' = \delta(I, \Sigma_u^*) = \cup_{s \in I} \delta(s, \Sigma_u^*)$. Observe that the resulting automaton $\mathscr{A}'$ has the same states as $\mathscr{A}$, operates on the reduced alphabet Σ_o, but is generally nondeterministic. By construction, one has $\mathscr{L}(\mathscr{A}') = \mathscr{L}_o(\mathscr{A})$. By contrast, the ε-reduction to the left would take $I' = I$ and $\delta'(s,\alpha) = \delta(s, \Sigma_u^*\alpha) = \cup_{w \in \Sigma_u^*\alpha} \delta(s,w)$, still preserving $\mathscr{L}(\mathscr{A}') = \mathscr{L}_o(\mathscr{A})$.

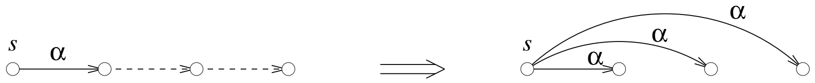

Fig. 5.1 Epsilon-reduction to the right. Dashed arrows represent silent (epsilon) transitions

Determinization. The determinization $\mathscr{A}'' = Det(\mathscr{A}') = (Q, \Sigma_o, q_0, \delta'')$ of $\mathscr{A}'$ is obtained by the standard subset construction. One has $Q = 2^S$, $q_0 = I'$, and for $q \in Q$ and $\alpha \in \Sigma_o$, the unique state $q' = \delta''(q,\alpha)$ in $\mathscr{A}''$ 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. 5.2). Determinization obviously has an exponential space complexity, in the worst case. Automaton $\mathscr{A}''$ directly yields a state estimator, or an observer of $\mathscr{A}$, by taking the identity for ϕ. By abuse of notation, one thus say that $\mathscr{A}''$ is an observer of $\mathscr{A}$, rather than $(\mathscr{A}'', \phi)$.

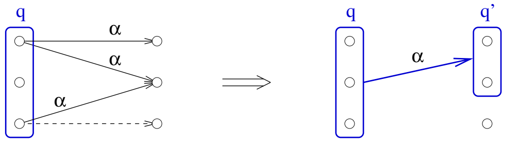

Fig. 5.2 Determinization. The dashed arrow represents a transition not labeled by α in $\mathscr{A}'$

Remarks

1. As mentioned in the previous section, building a diagnoser boils down to building an observer. Without loss of generality, for the diagnosis problem one can directly assume that states of $\mathscr{A}$ are partitioned into $S = S_1 \uplus \ldots \uplus S_L$ with $L = 2^K$, corresponding to the 2^K possible values of the memory in $\bar{\mathscr{A}}$. The diagnosis then reduces to checking whether all final states compatible with observation $w \in \mathscr{L}_o(\mathscr{A})$ 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 $\mathscr{A}$ with a simpler binary memory, or equivalently by assuming a partition $S = S_s \uplus S_f$ into safe and faulty states. In terms of complexity, this clearly saves an exponential in K. The diagnoser derived in the previous section is much more powerful, since it can also test for the simultaneous presence of several transition types in each trajectory explaining an observed word w.
3. Surprisingly, the ε-reduction to the right is frequent in the literature about state estimation (see the notion of 'unobservable reach'), but the ε-reduction to the left is preferred to derive a diagnoser. In other words, it is admitted that the supervised system changes its state silently, but not that it produce a fault silently. For diagnosis, 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), which can be considered as an optimistic assumption. This choice is not bothering for Σ_o-live systems, since it only delays by one observation the detection of a fault occurrence, and in particular it does not change the notions of diagnosability. However, it makes a difference for non Σ_o-live systems, in the case where some states necessarily lead to a failure after which no more observation is collected (system crash).
4. Some contributions introduced so-called 'observation filters' [16, 17]: rather than a partition $\Sigma = \Sigma_o \uplus \Sigma_u$, one gives a filter $\lambda : S \times \Sigma \times S \to 2^{\Lambda \cup \{\varepsilon\}}$, and when $t = (s, \alpha, s')$ is fired, one label $\beta \in \lambda(t) \subseteq \Lambda \cup \{\varepsilon\}$ is observed (possibly none if $\beta = \varepsilon$). 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.
5. An extended notion of diagnoser was proposed in [9]. It tests the occurrence (or not) of more complex properties on the partially observed trajectory of $\mathscr{A}$, such as the crossing of specific states interleaved with the crossing of specific

transitions. As long as these properties are regular and can thus be described by an automaton, the ideas of this chapter adapt naturally: one simply has to replace the simple state augmentation described previously by the product of $\mathscr{A}$ with the automaton describing the property to check.

5.3 Probabilistic Observer

Probabilistic automata [14] form a subclass of weighted automata. The idea is that the transitions rooted at any state are associated to firing probabilities. The state estimation problem now evolves into computing the probability to stop in state $s \in S$ given that $w \in \Sigma_o^*$ was observed (for diagnosis one wishes the conditional probability that a transition in T_k was fired). Again, one can design a recursive way to compute this conditional distribution over S; it takes the form of a filtering equation 'à la Kalman.' This section rather examines the precompiled version of this filter, that has great similarities with the non stochastic case.

Probabilistic automaton. We define it as $\mathscr{A} = (S, \Sigma, \mathbb{P}_0, \mathbb{P})$ where $\mathbb{P}_0 : S \to [0,1]$ is an initial probability on states, with initial states $I = supp(\mathbb{P}_0)$[2], 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 next labels and next states, given the current state s. Transitions are given by $T = supp(\mathbb{P})$, and the transition function by $\delta(s,\alpha) = supp(\mathbb{P}(s,\alpha,\cdot))$. This definition assumes that $\mathscr{A}$ is live, for simplicity[3]. $\mathscr{A}$ is said to be deterministic iff its support $supp(\mathscr{A}) = (S, \Sigma, I, \delta)$ is a deterministic (non stochastic) automaton. The probability of a path $\pi = t_1 ... t_n$ is equal to the product of the probabilities of its events $\mathbb{P}(\pi) = \mathbb{P}(t_1)...\mathbb{P}(t_n)$. And the language of $\mathscr{A}$ is defined as the formal power series $\mathscr{L}(\mathscr{A}) = \sum_{w \in \Sigma^*} \mathscr{L}(\mathscr{A}, w) \cdot w$ where coefficients are given by $\mathscr{L}(\mathscr{A}, w) = \sum_{\pi, \sigma(\pi) = w} \mathbb{P}_0(s^-(\pi))\mathbb{P}(\pi)$.

Observable (or stopped) language. To define the observable language of $\mathscr{A}$ as a weighted language, one must choose an appropriate notion of stopping time for $\mathscr{A}$, in order to define where runs should stop when they perform silent transitions. We adopt the following : $\mathscr{A}$ stops immediately before the production of the next observation, assuming $\mathscr{A}$ is Σ_o-live *i.e.* can reach an observable transition from any reachable state. Equivalently, $\mathscr{A}$ stops when it has been decided that the next step would produce an observation, but it is not yet decided which one[4]. 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

[2] *supp* = support of; this operation selects the elements with non zero probability

[3] One can easily extend this setting to include stopping probabilities at each state, just like standard weighted automata include stopping weights.

[4] 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.

'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 (as in [16] for example).

We define the stopped language of $\mathscr{A}$ 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 $s^+(\pi)$. Observe that a path cannot stop at a state which has no observable outgoing transition, *i.e.* such a path has a vanishing probability. Therefore some states in $\mathscr{A}$ are 'unstable.' Then $\mathscr{L}^s(\mathscr{A},w) = \sum_{\pi,\sigma(\pi)=w}\mathbb{P}^s(\pi)$ and the stopped language of $\mathscr{A}$ is $\mathscr{L}^s(\mathscr{A}) = \sum_{w\in\Sigma^*}\mathscr{L}^s(\mathscr{A},w)\cdot w$. The observable language of $\mathscr{A}$ is given by $\mathscr{L}_o(\mathscr{A}) = \sum_{w\in\Sigma_o^*}\mathscr{L}_o(\mathscr{A},w)\cdot w$ where

$$\mathscr{L}_o(\mathscr{A},w) = \sum_{v\in\Sigma^*,\Pi_{\Sigma_o}(v)=w} \mathscr{L}^s(\mathscr{A},v). \tag{5.4}$$

Notice that the support of the stopped language may be strictly smaller than the support of the ordinary language, since some states of $\mathscr{A}$ may forbid stopping. However, the observed stopped and non-stopped languages of $\mathscr{A}$ have identical supports.

Probabilistic observer. Given partitions $\Sigma = \Sigma_o \uplus \Sigma_u$ and $S = S_1 \uplus \ldots \uplus S_L$, the objective is to derive a deterministic probabilistic automaton $\mathscr{O} = (Q,\Sigma_o,\mathbb{P}_0^{\mathscr{O}},\mathbb{P}^{\mathscr{O}})$, and a labeling $\phi : Q \to \mathscr{P}(L)$ of its states, where $\mathscr{P}(L)$ is the set of probability distributions over $\{1,...,L\}$. Given $w \in \Sigma_o^*$ produced by $\mathscr{A}$, and $q \in Q$ the unique state reached by w in $\mathscr{O}$, we want $(\phi(q))(l) = \mathbb{P}(\mathscr{A}$ stops in $S_l \mid w$ was observed$)$ for $l \in \{1,...,L\}$. Every such probabilistic observer can trivially be derived from a universal one assuming the finest partition of S, *i.e.* we actually aim at building an observer that computes $\mathbb{P}(\mathscr{A}$ stops in $s \mid w$ was observed$)$ for every $s \in S$.

In fact, the probabilistic observer derived below exhibits more properties than requested above: it guarantees that $\mathscr{L}(\mathscr{O}) = \mathscr{L}_o(\mathscr{A})$, *i.e.* for any observed word $w \in \Sigma_o^*$, it can also compute the probability of this observed word in $\mathscr{A}$. If one is simply interested in the conditional distribution over S given w, the labeling function ϕ is sufficient and $\mathscr{O}$ can be reduced to its support.

ε-reduction. We look for a probabilistic automaton $\mathscr{A}' = Red(\mathscr{A}) = (S,\Sigma_o,\mathbb{P}'_0,\mathbb{P}')$ such that $\mathscr{L}(\mathscr{A}') = \mathscr{L}_o(\mathscr{A}') = \mathscr{L}_o(\mathscr{A})$. 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 $\mathscr{A}$ decides to stop (and commits to fire a visible transition at the next step). This requires to integrate probabilities over a possibly infinite set of silent paths. The difficulty can addressed by different methods, of equivalent complexities. We give two of them below; see [8] for a graphical one.

The transition probability of automaton $\mathscr{A}'$ can be expressed as $\mathbb{P}'(s,\alpha,s') = \sum_{s''\in S}\mathbb{P}(s,\alpha,s'')\mathbb{P}^{\varepsilon}(s'',s')\mathbb{P}(s',\Sigma_o,S)$, where

$$\mathbb{P}^{\varepsilon}(s,s') = \mathbb{P}(s,\Sigma_u^*,s') = \sum_{\substack{\pi,\ \sigma(\pi)\in\Sigma_u^* \\ s^-(\pi)=s,\ s^+(\pi)=s'}} \mathbb{P}(\pi) \tag{5.5}$$

and similarly for the initial probability: $\mathbb{P}'_0(s') = \sum_{s''\in S}\mathbb{P}_0(s'')\mathbb{P}^{\varepsilon}(s'',s')\mathbb{P}(s',\Sigma_o,S)$. Notice that (5.5) does not represent the probability to reach s' from s after an arbitrary number of silent steps, because s' can be met several times along such paths. One must take into account a stopping or exit probability at s' to turn this quantity into a probability, as it is done in the definition of $\mathbb{P}'(s,\alpha,s')$: the term $\mathbb{P}^{\varepsilon}(s'',s')\mathbb{P}(s',\Sigma_o,S)$ does correspond to the probability of going from s'' to s' through an arbitrary number of silent steps *and* to exit at s' towards a visible label. Similarly, $\mathbb{P}^{\varepsilon}(s,s)\geq 1$ yields the inverse of the probability to leave state s for a visible label (after performing an arbitrary number of silent loops around s).

The *pseudo* transition matrix $\mathbb{P}^{\varepsilon}$ can be obtained through a Floyd-Warshall procedure. It 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 [4, 13]. Specifically, denoting $S=\{s_1,...,s_N\}$, one defines $\mathbb{P}^{\varepsilon}_n(s,s')$ as in (5.5), excepted that the sum is limited to paths that go through states in $\{s_1,...,s_n\}$. The desired $\mathbb{P}^{\varepsilon}$ corresponds to $\mathbb{P}^{\varepsilon}_N$, and $\mathbb{P}^{\varepsilon}_0(s,s')=\mathbb{P}(s,\Sigma_u,s')$: probability of a direct silent step from s to s'. The $\mathbb{P}^{\varepsilon}_n$ satisfy the following recursion: for $s\neq s_{n+1}\neq s'$

$$\mathbb{P}^{\varepsilon}_{n+1}(s,s') = \mathbb{P}^{\varepsilon}_n(s,s_{n+1})\,\mathbb{P}^{\varepsilon}_n(s_{n+1},s_{n+1})^*\,\mathbb{P}^{\varepsilon}_n(s_{n+1},s') + \mathbb{P}^{\varepsilon}_n(s,s') \tag{5.6}$$

$$\mathbb{P}^{\varepsilon}_n(s_{n+1},s_{n+1})^* = \sum_{k\geq 0}\mathbb{P}^{\varepsilon}_n(s_{n+1},s_{n+1})^k = \frac{1}{1-\mathbb{P}^{\varepsilon}_n(s_{n+1},s_{n+1})} \tag{5.7}$$

where (5.7) integrates over paths that make an arbitrary number of silent loops around s_{n+1}. For completeness, one must add to (5.6) three following specific cases: $\mathbb{P}^{\varepsilon}_{n+1}(s_{n+1},s_{n+1}) = \mathbb{P}^{\varepsilon}_n(s_{n+1},s_{n+1})^*$, then $\mathbb{P}^{\varepsilon}_{n+1}(s,s_{n+1}) = \mathbb{P}^{\varepsilon}_n(s,s_{n+1})\mathbb{P}^{\varepsilon}_n(s_{n+1},s_{n+1})^*$, and finally $\mathbb{P}^{\varepsilon}_{n+1}(s_{n+1},s') = \mathbb{P}^{\varepsilon}_n(s_{n+1},s_{n+1})^*\,\mathbb{P}^{\varepsilon}_n(s_{n+1},s')$. The complexity of the ε-reduction by this method is $\mathscr{O}(|S|^3)$.

Instead of the Floyd-Warshall procedure, one may also consider a fix-point relation satisfied by matrix $\mathbb{P}^{\varepsilon}$. Let $\bar{\mathbb{P}}^{\varepsilon}=\mathbb{P}^{\varepsilon}-I$; this corresponds to Definition (5.5) where Σ_u^* is replaced by Σ_u^+, *i.e.* paths must cross at least one unobservable transition. One then has

$$\forall s,s'\in S,\quad \bar{\mathbb{P}}^{\varepsilon}(s,s') = \sum_{s''\in S}\bar{\mathbb{P}}^{\varepsilon}(s,s'')\mathbb{P}(s'',\Sigma_u,s') + \mathbb{P}(s,\Sigma_u,s') \tag{5.8}$$

Still denoting by $\mathbb{P}^{\varepsilon}_0$ the matrix with entries $\mathbb{P}(s,\Sigma_u,s')$, (5.8) means $\bar{\mathbb{P}}^{\varepsilon}=\bar{\mathbb{P}}^{\varepsilon}\cdot\mathbb{P}^{\varepsilon}_0+\mathbb{P}^{\varepsilon}_0$ (notice that $\bar{\mathbb{P}}^{\varepsilon}=\mathbb{P}^{\varepsilon}_0\cdot\bar{\mathbb{P}}^{\varepsilon}+\mathbb{P}^{\varepsilon}_0$ holds as well). This entails $\mathbb{P}^{\varepsilon}=I+\mathbb{P}^{\varepsilon}\cdot\mathbb{P}^{\varepsilon}_0=I+\mathbb{P}^{\varepsilon}_0\cdot\mathbb{P}^{\varepsilon}$ whence $\mathbb{P}^{\varepsilon}=(I-\mathbb{P}^{\varepsilon}_0)^{-1}$ (assuming invertibility holds). Deriving $\mathbb{P}^{\varepsilon}$ by this methods reveals again a generic complexity in $\mathscr{O}(|S|^3)$, due to the matrix inversion.

Determinization. To determinize a probabilistic automaton $\mathscr{A}'=(S,\Sigma_o,\mathbb{P}'_0,\mathbb{P}')$, one can rely on the standard determinization procedure of weighted automata, that adapts the recursive subset construction given in the previous section [2, 10, 11]. One has $\mathscr{A}'' = Det(\mathscr{A}') = (Q, \Sigma_o, \mathbb{P}''_0, \mathbb{P}'')$ 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 $1 \le n \le N$

$$p''_n = \sum_{1\le m\le M} p_m \cdot \mathbb{P}'(s_m, \alpha, s'_n) \tag{5.9}$$

$$p'_n = p''_n / C \quad \text{where} \quad C = \sum_{1\le k\le N} p''_k = \mathbb{P}''(q, \alpha, q') \tag{5.10}$$

Proposition 5.1. *Let $w \in \Sigma_o^*$ and $\delta''(q_0, w) = q = \{(s_1, p_1), ..., (s_M, p_M)\}$ in $\mathscr{A}''$, then*

$$p_m = \mathbb{P}(\mathscr{A}' \text{ is in state } s_m | w \text{ was observed}) \tag{5.11}$$

Proof. This is obviously true at q_o for $w = \varepsilon$. Assume it is true at $q = \delta''(q_o, w)$ and let $q' = \delta''(q, \alpha)$. Eq. (5.9) is a standard filtering equation for $\mathscr{A}'$ (based on Bayes rule and the Markov property), so p''_n is the probability that $\mathscr{A}'$ 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 $(p'_n)_{1\le n\le N}$ give the conditional probability of the current state of $\mathscr{A}'$ given the observed sequence $w\alpha$. □

Corollary 5.1. *The probabilistic automaton $Det(Red(\mathscr{A}))$ built above yields a universal probabilistic observer. For state $q = \{(s_1, p_1), ..., (s_M, p_M)\}$, the index function $\phi(q) \in \mathscr{P}(S)$ is defined by $[\phi(q)](s_m) = p_m$, for $1 \le m \le M$, and by $[\phi(q)](s) = 0$ otherwise.*

Proof. If $\mathscr{A}' = Red(\mathscr{A})$, p_m is the probability that $\mathscr{A}$ stops in s_m given that w was observed. To make $\mathscr{A}''$ an observer for partition $S = S_1 \uplus ... \uplus S_L$, one simply has to take the distribution defined by the $[\phi(q)](S_l)$, for $1 \le l \le L$. Notice also that $\mathscr{L}(\mathscr{A}'') = \mathscr{L}(\mathscr{A}') = \mathscr{L}_o(\mathscr{A})$. □

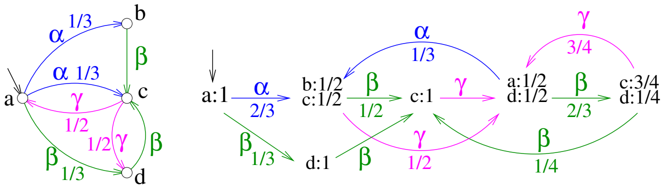

Fig. 5.3 A probabilistic automaton (left) and its determinized version (right). Transition probabilities are only mentioned when they differ from 1

Example 5.1. Figure 5.3 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. 5.4. While for weighted automata taking values in the $(\mathbb{R}^+, \min, +)$ semiring there exist sufficient conditions to guarantee termination (see the twin property in [12]), to our knowledge it is still not clear what these conditions could be for probabilistic automata. ■

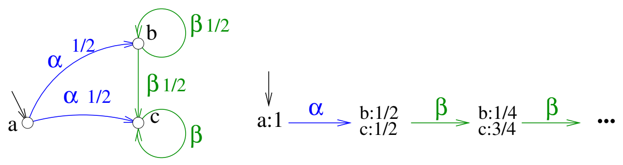

Fig. 5.4 Determinization may not terminate

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

$$\bar{\mathbb{P}}((s,\mu),\alpha,(s',\mu')) = \mathbb{P}(s,\alpha,s') \cdot \mathbb{I}_{\mu'=\mu\cup\{k:(s,\alpha,s')\in T_k\}} \tag{5.12}$$

where $\mathbb{I}$ is the indicator function. From the conditional distribution on states of $\bar{\mathscr{A}}$ 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 $\mathscr{A}$. Again, if one is only interested in this probability distribution, the observer can be turned into an ordinary (non stochastic) deterministic automaton, by taking the support of $\mathscr{O}$.

Remark. The 'observation filters,' that randomly modify the labels of Σ produced by transitions of $\mathscr{A}$, can be processed in a similar manner as in Remark 4 of Section 5.2. The slight difference here is that a given observed label $\beta \in \Lambda \cup \{\varepsilon\}$ 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 $\mathbb{I}$ term in (5.12) by a 'randomized' memory. Specifically, given $T = T_1 \cup \ldots \cup T_K$ and for $\mu' = \mu \uplus \mu$", (5.12) 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 $k \notin \mu'$.

5.4 Diagnosability

For simplicity, and without loss of generality, this section assumes an automaton $\mathscr{A}$ with a partition $S = S_s \uplus S_f$ of its states into safe and faulty ones (recall the state augmentation trick), and such that no safe state is reachable from a faulty one. It is also assumed that $\mathscr{A}$ is Σ_o-live. For $w \in \Sigma_o^*$, let us consider again the diagnosis function $f(w)$ for $\mathscr{A}$ defined as

$$f(w) = \begin{cases} y & \text{if } \forall \pi \in \sigma_o^{-1}(w),\ s^+(\pi) \in S_f \\ n & \text{if } \forall \pi \in \sigma_o^{-1}(w),\ s^+(\pi) \in S_s \\ a & \text{otherwise} \end{cases} \tag{5.13}$$

Definition 5.1. *$\mathscr{A}$ is diagnosable iff there exists some integer N such that*

$$\forall \pi : s^+(\pi) \in S_f,\ \forall \pi' : s^-(\pi') = s^+(\pi),\ \ [\, |\pi'|_{\Sigma_o} > N \ \Rightarrow f(\sigma_o(\pi\pi')) = y\,]$$

where $|\pi'|_{\Sigma_o}$ denotes the number of visible transitions in path π'.

In other words, as soon as a path hits a faulty state, at most N observations later the fault will be diagnosed. Ambiguity cannot last forever.

Remarks

1. Some definitions rather take $|\pi'|$ rather than $|\pi'|_{\Sigma_o}$, with the assumption that $\mathscr{A}$ has no silent circuit, which yields an equivalent definition for this smaller class of systems. This restriction is not really necessary, and $|\pi'|_{\Sigma_o}$ makes more sense since it gives an observable criterion to position the fault.
2. Some authors do not require that $\mathscr{A}$ is Σ_o-live, and then extend the condition to extensions π' that contain $M \leq N$ observations after which no more visible transition is reachable (deadlock or silent live-lock). This generalization introduces minor technical changes, that are left to the reader.

Proposition 5.2. *If $\mathscr{A}$ is not diagnosable, then a diagnoser $(\mathscr{D}, \psi)$ of $\mathscr{A}$ necessarily contains a circuit of ambiguous states.*

Proof. In Def. 5.1, $f(\sigma_o(\pi\pi'))$ can only take values y or a. If $\mathscr{A}$ is not diagnosable, let N be greater than the number of states in $\mathscr{D}$. There exists π, π' with $s^+(\pi) \in S_f$, $|\pi'|_{\Sigma_o} > N$ and $f(\sigma_o(\pi\pi')) = a$. Let q be the unique state reached by $\sigma_o(\pi)$ in $\mathscr{D}$, then $\psi(q) = a$ and any state q' crossed by $\sigma_o(\pi')$ after q in $\mathscr{D}$ is also ambiguous, $\psi(q') = a$, since by construction the values of ψ can only evolve from n to a and then y. And $\sigma_o(\pi')$ necessarily crosses twice some state of $\mathscr{D}$. □

This proposition gives a sufficient condition for diagnosability, which unfortunately is not necessary. Consider the counter-example in Fig. 5.5, where safe states are represented as a white circle, and faulty states as a colored one. The diagnoser contains an ambiguous circuit, because for any sequence $(\alpha\beta)^n$ or $(\alpha\beta)^n\alpha$ it is not certain that a fault occured. However, any path π leading to the faulty state s_3 will necessarily produce a γ as second next observation, which characterizes the occurrence of the fault. So $\mathscr{A}$ is 2-diagnosable ($N = 2$).

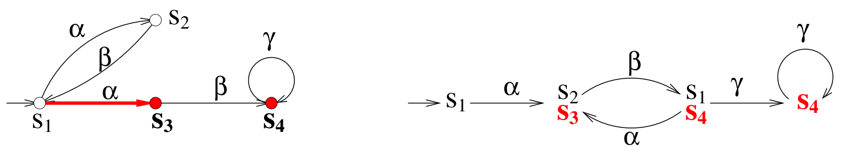

Fig. 5.5 A diagnosable system (left), and its observer/diagnoser (right)

Proposition 5.3. *$\mathscr{A}$ is not diagnosable iff, for any integer N,*

$$\exists \pi, \pi', \pi'' : \ s^+(\pi) = s^-(\pi') \in S_f, \ |\pi'|_{\Sigma_o} > N, \ s^+(\pi'') \in S_s, \ \sigma_o(\pi\pi') = \sigma_o(\pi'')$$

The proof is straightforward by logical inversion of Def. 5.1. One can easily reinforce the result into the existence of a faulty π (the same for all N) for which one can find arbitrarily long extensions π' such that there exists a non faulty π'' with the same visible signature: $\sigma_o(\pi\pi') = \sigma_o(\pi'')$ (hint: use the pumping lemma). Further, if the result holds for some large enough N, then it holds for any N (same trick).

This simple result has interesting practical consequences. First of all, it allows one to check if the ambiguous circuits of a diagnoser of $\mathscr{A}$ really entail non-diagnosability. Specifically, $\mathscr{A}$ is not diagnosable iff a diagnoser of $\mathscr{A}$ contains an *indeterminate cycle*, following the vocabulary in [15]. Such cycles are ambiguous circuits of $\mathscr{D}$ where both a safe run of $\mathscr{A}$ and a faulty one are nested. Specifically, the circuit from q to q in $\mathscr{D}$, following the visible sequence $w \in \Sigma_o^*$, is indeterminate iff there exist two circuits π_1, π_2 in $\mathscr{A}$ such that π_1 only crosses faulty states, π_2 only crosses safe states, and $\sigma_o(\pi_1) = \sigma_o(\pi_2) = w^n$ for some n. In such situations, using the pumping lemma, one can easily build the π, π', π'' of Prop. 5.3 that prove the non diagnosability. And conversely.

A second consequence of Prop. 5.3 is to provide a direct and more practical means of checking diagnosability (and actually polynomial rather than exponential). The idea is based on the *twin machine* construction. Consider $\mathscr{A}_s$, the restriction of $\mathscr{A}$ to the safe states S_s: all transitions involving states in S_f are discarded. The twin machine is obtained as the synchronous product of the ε-reductions of $\mathscr{A}$ and $\mathscr{A}_s$: $\mathscr{T} = Red(\mathscr{A}) \times Red(\mathscr{A}_s)$.

Proposition 5.4. *$\mathscr{A}$ is diagnosable iff no cycle of the twin machine $\mathscr{T} = Red(\mathscr{A}) \times Red(\mathscr{A}_s)$ contains a faulty state of $\mathscr{A}$.*

The proof is directly based on Prop. 5.3, using again the pumping lemma (details are left to the reader). Applied to the counter-example of Fig. 5.5, assuming all labels are observable, this yields the construction of Fig. 5.6, where the unique circuit crosses only safe states of $\mathscr{A}$, which makes $\mathscr{A}$ diagnosable.

Probabilistic diagnosability. For the sake of completeness, let us briefly examine how diagnosability extends to probabilistic automata. For simplicity, we assume that $\mathscr{A}$ is already ε-reduced, based on some stopping time definition.

As a (live) stochastic automaton, $\mathscr{A}$ defines a probability $\mathbb{P}_n$ on the set $\mathscr{F}_n$ of runs of length n. These $(\mathscr{F}_n)_n$ define a natural filtration over the set of infinite runs

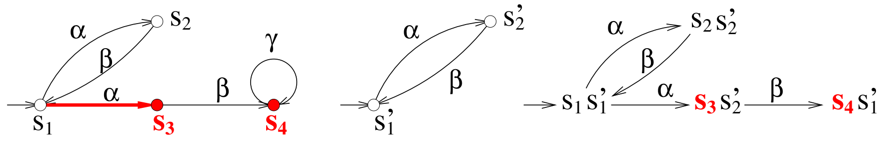

Fig. 5.6 System $\mathscr{A}$ (left), its safe restriction $\mathscr{A}_s$ (center), and the twin machine (right)

of $\mathscr{A}$, and since $\mathbb{P}_n$ is the restriction of $\mathbb{P}_{n+1}$ to $\mathscr{F}_n$, there exists a unique distribution $\mathbb{P}$ over the set $\mathscr{F}$ of infinite runs of $\mathscr{A}$, by the Kolmogorov extension theorem.

On this probability space, let us define two diagnosis indicators, as random variables. We denote by ω an infinite run (path) of $\mathscr{A}$, and by ω^n its restriction to the first n transitions. The first diagnosis indicator is $X_n(\omega) = \mathbb{P}(\{\omega' : \sigma_o(\omega'^n) = \sigma_o(\omega^n), s^+(\omega'^n) \in S_f\})$, which is an $\mathscr{F}_n$-measurable random variable. X_n is thus a failure likelyhood, and $X_n(\omega) = 0$ iff all runs of length n that produce the same observations as ω^n finish in S_{nf}. The second indicator is $D_n(\omega) \in \{0,1\}$, another $\mathscr{F}_n$-measurable random variable, defined by $D_n(\omega) = 0$ iff $\exists\omega', \sigma_o(\omega'^n) = \sigma_o(\omega^n) \wedge s^+(\omega'^n) \in S_s$. So $D_n(\omega)$ switches to one when *all* runs of length n that produce the same observation as ω^n contain a failure, which corresponds to the detection of that failure.

In [16], two notions of diagnosability were proposed. The A-diagnosability corresponds to the following: for all k, conditioned on $X_k > 0$, D_{k+n} converges to 1 in probability. The AA-diagnosability only requires that X_{k+n} converges to 1 in probability, again conditioned on $X_k > 0$. The first criterion expresses that the 'hard' detector D_n will ultimately switch to 1 (certain detection) after a failure has occurred, while the second criterion means that the detection probability X_n will converge to 1 (the more one waits, the more the detection is certain). These convergences are in probability: the more one waits, the more these events are likely. (There is still space for defining and characterizing a diagnosability based on an almost sure convergence.)

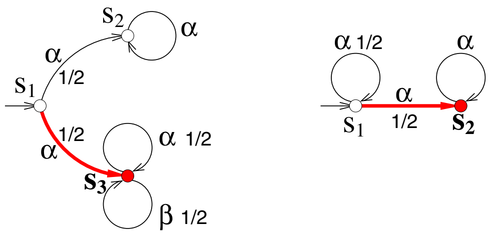

Fig. 5.7 Left: a non-diagnosable probabilistic automaton that is A-diagnosable. Right: an AA-diagnosable automaton that is not A-diagnosable

Fig. 5.7 (left) shows a probabilistic automaton that is not diagnosable, if probabilities are ignored: after the faulty state s_3 has been reached, one can observe an arbitrary long sequence of α^n that will not allow to discriminate between s_2 and s_3, *i.e.* safe and faulty. However, with probability 1 a β will ultimately be fired after the faulty state s_3, which will make the hard detector D_n jump to 1. The right hand side in the figure shows a probabilistic automaton that is not diagnosable nor A-diagnosable, since after the faulty transition, whatever the number of observations, one can not be certain to be in s_2. However, the longer one waits, the more likely the system jumped from s_1 to s_2. So with probability 1 the soft detector X_n will converge to 1.

One of the main results in [16] is to translate the A-diagnosability of $\mathscr{A}$ into a *structural* property of its *non probabilistic* diagnoser/observer. Equivalently, this amounts to first replacing $\mathscr{A}$ by its support $\bar{\mathscr{A}}$. The A-diagnosability criterion then observes the product $\mathscr{B} = Det(Red(\bar{\mathscr{A}})) \times Red(\bar{\mathscr{A}})$, where the ε-reduction corresponds to the chosen notion of stopping time in $\mathscr{A}$. This non-probabilistic automaton $\mathscr{B}$ has $Q \times S$ as state set. One then has to examine the recurrent states of $\mathscr{B}$, *i.e.* the states (q,s) that belong to a terminal connected component of $\mathscr{B}$, regarded as a directed graph. This is standard in convergence analysis of Markov chains, since with probability one the chain – here the probabilistic diagnoser of $\mathscr{A}$ – will terminate in one of its recurrent components, and A-diagnosability deals with the limit behaviors of the diagnoser of $\mathscr{A}$. Theorem 3 in [16] then expresses that $\mathscr{A}$ is A-diagnosable iff any recurrent state (q,s) in $\mathscr{B}$ with $s \in S_f$ satisfies $\psi(q) = y$ or equivalently $q \subseteq S_f$. In other words, after a faulty state s is crossed, the (probabilistic) diagnoser will terminate with probability one in states where the failure is unambiguous. As $\mathscr{B}$ requires a determinization, the complexity of the A-diagnosability test proposed in [16] is exponential. But as for standard diagnosability, one recovers a polynomial complexity by performing the same test on the recurrent states of the twin-machine derived for $\bar{\mathscr{A}}$.

5.5 Modular Observers

A compound system is obtained by assembling components by means of a composition operator. Here we chose the usual synchronous product of automata. The section proves that composition and derivation of an observer are two operations that commute, under some circumstances. This has important consequences to design efficient observers for some compound systems. Results are presented in the simple case of two components, but extend to larger compound systems through the notion of interaction graph between components (see for example [7] where this notion is used for distributed planning purposes).

Composition of Automata

Definition 5.2. *The* synchronous product *(see Fig. 5.8) of two automata $\mathscr{A}_1$ and $\mathscr{A}_2$ is the automaton $\mathscr{A}_1 \times \mathscr{A}_2 = (S,\Sigma,I,\delta)$ such that: $S = S_1 \times S_2, \Sigma = \Sigma_1 \cup \Sigma_2, I = I_1 \times I_2$, and the transition function δ is defined by $\forall (s_1,s_2) \in S, \forall \alpha \in \Sigma$, $\delta((s_1,s_2),\alpha) = \delta_1^+(s_1,\alpha) \times \delta_2^+(s_2,\alpha)$, where $\delta_i^+(s_i,\alpha)$ coincides with δ_i for $\alpha \in \Sigma_i$, and $\delta_i^+(s_i,\alpha) \triangleq \{s_i\}$ for $\alpha \notin \Sigma_i$.*

In other words, a transition carrying a shared label in one component must be fired jointly with a transition carrying the same label in the other component. By contrast, a transition carrying a private label only changes the state of one component, while the other remains idle. Observe that a firable sequence u of transitions in $\mathscr{A}_1 \times \mathscr{A}_2$ leads to a unique firable sequence u_1 in $\mathscr{A}_1$ (for example) by simply removing private moves of $\mathscr{A}_2$, *i.e.* transitions $((s_1,s_2),\alpha,(s_1,s_2'))$ with $\alpha \notin \Sigma_1$, and then erasing the states of component $\mathscr{A}_2$.

A plain synchronous product may yield an automaton that is not trimmed, *i.e.* that contains unaccessible states. So we define the composition as a synchronous product followed by a trimming operation, and still denote it by $\times$, with a light abuse of notation. This definition naturally extends to observers (or diagnosers) by simply gathering the label functions on states, for example $\phi(q_1,q_2) \triangleq \phi_1(q_1) \times \phi_2(q_2)$.

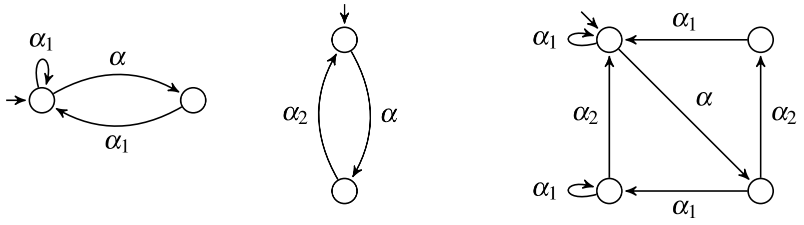

Fig. 5.8 Two automata (left) sharing only label α, and their synchronous product (right)

Observer of compound systems. We consider here the 'canonical' observers derived in the previous section by epsilon-reduction and determinization. Such an observer for $\mathscr{A}$ has $Q = 2^S$ as state set, if S represents states of $\mathscr{A}$.

Proposition 5.5. *Let $\mathscr{A}_1,\mathscr{A}_2$ be two automata, with $\mathscr{A}_i = (S_i,\Sigma_i,I_i,\delta_i)$ and $\Sigma_{o,i}$ as set of observable labels, $i \in \{1,2\}$. If all synchronizations are observable by each automaton,* i.e. $\Sigma_1 \cap \Sigma_2 = \Sigma_{o,1} \cap \Sigma_{o,2}$ *, then $Obs(\mathscr{A}_1 \times \mathscr{A}_2)$ and $Obs(\mathscr{A}_1) \times Obs(\mathscr{A}_2)$ are isomorphic.*

Proof. We consider the epsilon-reduction to the right. For the proof, we show by induction the bisimilarity of the two deterministic automata $Obs(\mathscr{A}_1 \times \mathscr{A}_2)$ and $Obs(\mathscr{A}_1) \times Obs(\mathscr{A}_2)$. Given the one to one correspondence of state sets, this will induce isomorphism.

Our recursion assumption is the following: let $w \in \Sigma_o^*$, and $w_i = \Pi_{\Sigma_{o,i}}(w)$, then $\bar{\delta}(I',w) = \bar{\delta}_1(I_1',w_1) \times \bar{\delta}_2(I_2',w_2)$. In other words, all reachable states q in $Obs(\mathscr{A}_1 \times \mathscr{A}_2)$ have a product form $q = q_1 \times q_2$.

This is obviously true for $w = \varepsilon$. If nothing has been observed, then only (unobservable) private events of the $\mathscr{A}_i$ can have been fired. Let u_i be any unobservable transition sequence in $\mathscr{A}_i$, starting from an initial state. So $s^-(u_i) = s_{i,0} \in I_i$ and $s^+(u_i) = s_i \in I_i' = \delta_i(I_i,\Sigma_{u,i}^*)$ where $\Sigma_{u,i} \triangleq \Sigma_i \setminus \Sigma_{o,i}$ denote unobservable labels of $\mathscr{A}_i$. Any interleaving u of sequences u_1 and u_2 is a firable sequence of transitions in

$\mathscr{A}_1 \times \mathscr{A}_2$, with $s^-(u) = (s_{1,0}, s_{2,0}) \in I$ and so $s^+(u) = (s_1, s_2) \in I' = \delta(I, \Sigma_u^*)$. Conversely, starting from an unobservable sequence u in $\mathscr{A}_1 \times \mathscr{A}_2$, one easily derives the associated u_i in $\mathscr{A}_i$. So this proves $I' = I'_1 \times I'_2$, *i.e.* that the initial state q_0 of $Obs(\mathscr{A}_1 \times \mathscr{A}_2)$ is the product of the initial states $q_{i,0}$ of the $Obs(\mathscr{A}_i)$.

For the recursion, let $q = \bar{\delta}(q_0, w) = (q_1, q_2)$ be an accessible state in $Obs(\mathscr{A}_1 \times \mathscr{A}_2)$, and let $\alpha \in \Sigma_o$. Three cases must be considered.

Case 1: $\alpha \in \Sigma_{o,1} \cap \Sigma_{o,2}$, which corresponds to a synchronization event. Let $s_i \in q_i$, $t_i = (s_i, \alpha, s'_i)$ be a transition of $\mathscr{A}_i$ and u_i be a sequence of silent transitions in $\mathscr{A}_i$, with $s^-(u_i) = s'_i$ and $s^+(u_i) = s''_i$. Then $((s_1, s_2), \alpha, (s'_1, s'_2))$ is a transition of $\mathscr{A}_1 \times \mathscr{A}_2$ and any interleaving u of sequences u_1, u_2 is firable in $\mathscr{A}_1 \times \mathscr{A}_2$ after (s'_1, s'_2) and leads to (s''_1, s''_2). This proves that $q' = \bar{\delta}(q, \alpha)$ in $Obs(\mathscr{A}_1 \times \mathscr{A}_2)$ contains $q'_1 \times q'_2$ where $q'_i = \bar{\delta}_i(q_i, \alpha)$ in $Obs(\mathscr{A}_i)$. For the converse inclusion, consider as above a visible transition $((s_1, s_2), \alpha, (s'_1, s'_2))$ in $\mathscr{A}_1 \times \mathscr{A}_2$ followed by some unobservable sequence u leading to (s''_1, s''_2), and split u into u_1 and u_2 as above. Then $(s_i, \alpha, s'_i)u_i$ is firable in $\mathscr{A}_i$ and leads from s_i to s''_i. This proves $q' \subseteq q'_1 \times q'_2$. So one concludes $\bar{\delta}(q, \alpha) = \bar{\delta}_1(q_1, \alpha) \times \bar{\delta}_2(q_2, \alpha)$.

Case 2: $\alpha \in \Sigma_{o,1} \setminus \Sigma_{o,2}$, which corresponds to a private observable event of $\mathscr{A}_1$. Let $s_1 \in q_1$, $t_1 = (s_1, \alpha, s'_1)$ be a private (observable) transition of $\mathscr{A}_1$ and u_1 be a sequence of silent transitions in $\mathscr{A}_1$, with $s^-(u_1) = s'_1$ and $s^+(u_1) = s''_1$. For any $s_2 \in q_2$, the private sequence $t_1 u_1$ of $\mathscr{A}_1$ is mapped into a sequence tu of $\mathscr{A}_1 \times \mathscr{A}_2$ leading from (s_1, s_2) to (s''_1, s_2), thus leaving $\mathscr{A}_2$ idle. So $q' = \bar{\delta}(q, \alpha) \supseteq q'_1 \times q_2$ where $q'_1 = \bar{\delta}_1(q_1, \alpha)$. And the converse inclusion is derived again as above, which proves $\bar{\delta}(q, \alpha) = \bar{\delta}_1(q_1, \alpha) \times q_2$.

Case 3: $\alpha \in \Sigma_{o,2} \setminus \Sigma_{o,1}$, which corresponds to a private observable event of $\mathscr{A}_2$. Similar to case 2.

The three cases above allow one to extend by one letter the recursion assumption, which induces the desired bisimilarity. □

Remarks

1. The above proposition assumed an epsilon-reduction to the right, but it remains valid with a reduction to the left.
2. Although canonical observers were assumed in the proof, it extends to general observers with minor modifications. One simply has that the product $Obs(\mathscr{A}_1) \times Obs(\mathscr{A}_2)$ yields *one* observer for $\mathscr{A}_1 \times \mathscr{A}_2$. In other words, the bisimilarity holds, but not necessarily the isomorphism.

Application. The application of this proposition to modular/distributed observation is direct: from a given observed sequence $w \in \Sigma_o^*$, derive projections $w_i = \Pi_{\Sigma_{o,i}}(w)$ and feed them to observers $Obs(\mathscr{A}_i)$ to get local state estimates q_i. Then assemble the latter by $q = q_1 \times q_2$ to get a state estimate of the global system. Reading this property in the reverse direction, this means that the interleaving of private events in w_1 and w_2 does not carry information to estimate the state of $\mathscr{A} = \mathscr{A}_1 \times \mathscr{A}_2$. In other words, it is equivalent to observe a total order of events, as sequence w or a *partial order* of events under the form of two partially synchronized sequences (w_1, w_2).

The assumption $\Sigma_1 \cap \Sigma_2 = \Sigma_{o,1} \cap \Sigma_{o,2}$ is crucial and one can check that the proof fails without this argument. Actually, without this assumption, it is possible to build examples where there exists no pair of *finite* deterministic machines that would play the role of local observers $Obs(\mathscr{A}_i)$, and such that their 'composition' in any way would have the same power as an observer of the global system. This is due to the possibly infinite number of assumptions that must be stored in each local observer about the way the two components synchronize in their unobserved sequences.

This is briefly illustrated by the example in Figure 5.9, where $\mathscr{A}_1$ and $\mathscr{A}_2$ are two automata such that $\Sigma_{o,1} = \{\alpha_1, \alpha_2\}$, $\Sigma_{o,2} = \{\beta_1, \beta_2\}$ and $\Sigma_1 \cap \Sigma_2 = \{\gamma_1, \gamma_2\} \subseteq \Sigma_{uo}$. Observe that, by construction, the production of α's and β's in $\mathscr{A}_1 \times \mathscr{A}_2$ alternate[5]. In other words, from a sequence w_1 of α's observed on $\mathscr{A}_1$ and a sequence w_2 of β's observed on $\mathscr{A}_2$, one can recover their interleaving w. So dealing with distributed observations here does not bother state estimation: no interleaving information is lost.

Then observe that as long as the indexes of the α's observed in w_1 match those observed on the β's in w_2, the component $\mathscr{A}_2$ will be in one of the states of its 'upper part,' $\{s'_o, s'_1, s'_2\}$. As soon as these two index sequences differ, $\mathscr{A}_2$ becomes trapped in the states of its lower part $\{s'_3, s'_4\}$. A global observer of $\mathscr{A}_1 \times \mathscr{A}_2$ fed with w is of course able to determine where $\mathscr{A}_2$ finished. But there is no pair of finite local observers for $\mathscr{A}_1$ and $\mathscr{A}_2$ that would have this power, since they would have to store the sequences of indexes of the α's and of the β's they have seen along w_1 and w_2 in order to compare them and decide.

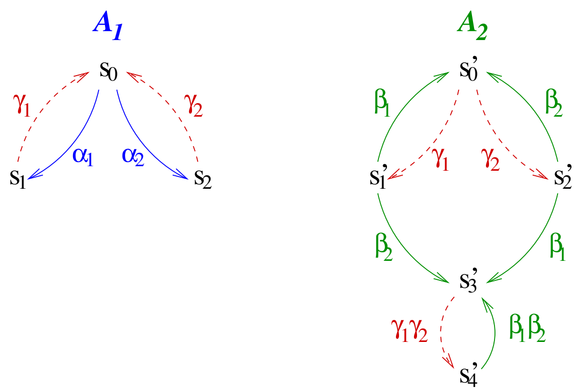

Fig. 5.9 For these two interacting components $\mathscr{A}_1$ and $\mathscr{A}_2$, with non observable interactions, there are no finite local observers that would jointly have the same power as an observer of their product $\mathscr{A}_1 \times \mathscr{A}_2$

[5] This is not strictly the case, since between two consecutive γ the labels α and β can appear in any order. This detail can be easily fixed, at the expense of a more complex example, and does not really bother the rest of the reasoning.

5.6 Distributed State Estimation

The lesson of the previous section is that distributed/modular state estimation (or diagnosis) is easy when synchronizations are observable. What about the general case? Assume a modular system $\mathscr{A} = \mathscr{A}_1 \times \mathscr{A}_2$, with $\Sigma_1 \cap \Sigma_2 \not\subseteq \Sigma_{o,1} \cap \Sigma_{o,2}$, performs a hidden run π that is observed by two sensors, one per component $\mathscr{A}_i$. This yields the pair of observed words (w_1, w_2), with $w_i = \Pi_{\Sigma_{o,i}}(\sigma(\pi))$, where shared events may be observed by only one or none of the two sensors. Notice that the exact interleaving of w_1 and w_2, that would be $w = \Pi_{\Sigma_{o,1} \cup \Sigma_{o,2}}(\sigma(\pi))$, is definitely lost, so all possible interleavings must be assumed to estimate the current state of $\mathscr{A}$. In other words, one truly observes a partial order of events, without the possibility to come back to a unique sequence as in Section 5.5. This constitutes a major change.

In this section we consider automata with marked states $\mathscr{A} = (S, \Sigma, I, \delta, F)$ where $F \subseteq S$. A path π of $\mathscr{A}$ is accepted by $\mathscr{A}$ iff $s^-(\pi) \in I$ and $s^+(\pi) \in F$. The language of $\mathscr{A}$ becomes $\mathscr{L}(\mathscr{A}) = \{\sigma(\pi) : \pi \text{ is accepted by } \mathscr{A}\}$. And for the product of automata, one takes $F = F_1 \times F_2$. To address the state estimation (or diagnosis) problem, we extend it into computing all paths π of $\mathscr{A}$ that can explain (w_1, w_2). Without loss of generality, we also assume that $\mathscr{A}_1$ and $\mathscr{A}_2$ are deterministic, so the problem amounts to finding words of $\mathscr{L}(\mathscr{A})$ that match (w_1, w_2).

Product of languages

Definition 5.3. *Let the $\mathscr{L}_i \subseteq \Sigma_i^*$ be two languages, $i = 1, 2$, their product is defined as $\mathscr{L}_1 \times_L \mathscr{L}_2 = \Pi_{\Sigma_1}^{-1}(\mathscr{L}_1) \cap \Pi_{\Sigma_2}^{-1}(\mathscr{L}_2)$, where the $\Pi_{\Sigma_i} : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_i^*$ are the natural projections.*

For example, with $\mathscr{L}_1 = \{\alpha\gamma\alpha\} \subseteq \{\alpha, \gamma\}^*$ and $\mathscr{L}_2 = \{\gamma\beta, \gamma\beta\gamma\} \subseteq \{\beta, \gamma\}^*$, one has $\mathscr{L} = \mathscr{L}_1 \times_L \mathscr{L}_2 = \{\alpha\gamma\alpha\beta, \alpha\gamma\beta\alpha\} \subseteq \{\alpha, \beta, \gamma\}^*$. The second word in $\mathscr{L}_2$ matches no word of $\mathscr{L}_1$, while the first one interleaves in two different ways with the only word of $\mathscr{L}_1$. Observe the following result:

Lemma 5.1. *Let $\mathscr{L} = \mathscr{L}_1 \times_L \mathscr{L}_2$, and $\mathscr{L}_i' = \Pi_{\Sigma_i}(\mathscr{L})$, then $\mathscr{L}_i' \subseteq \mathscr{L}_i$ and one has $\mathscr{L} = \mathscr{L}_1' \times_L \mathscr{L}_2'$. The $\mathscr{L}_i'$ are the minimal sublanguages of the $\mathscr{L}_i$ that allow one to recover $\mathscr{L}$.*

The proof is left to the reader as an exercise. As a direct application of $\times_L$, observe that $\{w_1\} \times_L \{w_2\}$, denoted $w_1 \times_L w_2$ for short, yields all interleavings w of w_1 and w_2 that must be considered.

The products of automata and of languages are related by the following property:

Proposition 5.6. *Let $\mathscr{A} = \mathscr{A}_1 \times \mathscr{A}_2$, then $\mathscr{L}(\mathscr{A}) = \mathscr{L}(\mathscr{A}_1) \times_L \mathscr{L}(\mathscr{A}_2)$.*

The proof is left to the reader as an exercise. Notice that $\Pi_{\Sigma_i}(\mathscr{L}(\mathscr{A})) \subseteq \mathscr{L}(\mathscr{A}_i)$ represents the behaviors of $\mathscr{A}_i$ that remain possible once the other component is connected.

Application to hidden run recovery. Assume centralized observation: one collects $w = \Pi_{\Sigma_o}(\sigma(\pi)) \in \Sigma_o^*$. The runs (or equivalently the words) of $\mathscr{A}$ that explain w are given by $\mathscr{E} = \mathscr{L}(\mathscr{A}) \times_L w$. Let $\mathscr{W}$ be a deterministic automaton such that $\mathscr{L}(\mathscr{W}) = \{w\}$, one also has $\mathscr{E} = \mathscr{L}(\mathscr{A} \times \mathscr{W})$. So one can take $\mathscr{A} \times \mathscr{W}$ as a compact and finite representation of this possibly infinite language (set of runs).

With distributed observations, one has (see Prop. 5.6)

$$\mathscr{E} = \mathscr{L}(\mathscr{A}_1 \times \mathscr{A}_2) \times_L (w_1 \times_L w_2) \tag{5.14}$$

$$= (\mathscr{L}(\mathscr{A}_1) \times_L w_1) \times_L (\mathscr{L}(\mathscr{A}_2) \times_L w_2) \tag{5.15}$$

$$= \mathscr{E}_1 \times_L \mathscr{E}_2 \tag{5.16}$$

where $\mathscr{E}_i = \mathscr{L}(\mathscr{A}_i) \times_L w_i$ represents local explanations to observation w_i in component $\mathscr{A}_i$. One is actually interested in building a *distributed explanation*, under the form $(\mathscr{E}'_1, \mathscr{E}'_2)$, where $\mathscr{E}'_i = \Pi_{\Sigma_i}(\mathscr{E})$ represents the local view in component $\mathscr{A}_i$ of runs of $\mathscr{A}$ that explain all observations w_1 and w_2. Again, one has $\mathscr{E} = \mathscr{E}'_1 \times_L \mathscr{E}'_2$ (see Lemma 5.1).

Distributed computation of a distributed explanation. The objective here is to determine directly the elements $\mathscr{E}'_i$ of a distributed explanation... without computing $\mathscr{E}$ itself! This can be done in a distributed manner, by message exchanges between the local 'supervisors' in charge of each component. The key idea is a notion of *conditional independence* on languages:

Proposition 5.7. *Let $\mathscr{L}_i \subseteq \Sigma_i^*$, $i = 1,2$, be two languages, and let $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma' \subseteq \Sigma_1 \cup \Sigma_2$, then $\Pi_{\Sigma'}(\mathscr{L}_1 \times_L \mathscr{L}_2) = \Pi_{\Sigma'}(\mathscr{L}_1) \times_L \Pi_{\Sigma'}(\mathscr{L}_2)$.*

In our setting, taking $\Sigma' = \Sigma_1$, this induces for example

$$\mathscr{E}'_1 = \Pi_{\Sigma_1}(\mathscr{E}_1 \times_L \mathscr{E}_2) = \mathscr{E}_1 \times_L \Pi_{\Sigma_1 \cap \Sigma_2}(\mathscr{E}_2) \tag{5.17}$$

and symmetrically for $\mathscr{E}'_2$. Equation (5.17) expresses that the local view $\mathscr{E}'_1$ of global explanations $\mathscr{E}$ are obtained by synchronizing the local explanations $\mathscr{E}_1$ on component $\mathscr{A}_1$ with the *message* $\Pi_{\Sigma_1 \cap \Sigma_2}(\mathscr{E}_2)$ from component $\mathscr{A}_2$. This message propagates the constraints that explanations $\mathscr{E}_2$ impose on synchronizations. Given the small alphabet $\Sigma_1 \cap \Sigma_2$ and the projection operation that removes private events of $\mathscr{A}_2$, (5.17) generally involves smaller objects than $\mathscr{E}$.

Example 5.2. The above computations involve possibly infinite languages. But again, they can be translated into automata computations thanks to Prop. 5.6 and to the fact that projection as well preserves the regularity (recall the construction of observers in Section 5.2, where $\mathscr{L}(\mathscr{A}'') = \mathscr{L}(\mathscr{A}') = \Pi_{\Sigma_o}(\mathscr{L}(\mathscr{A}))$).

Consider the example in Fig. 5.10, with two components and a distributed observations (b,d) represented as two single word automata. Fig. 5.11 computes the local explanations $\mathscr{E}_i$ by product $\mathscr{E}_i = \mathscr{L}(\mathscr{A}_i) \times_L w_i$, represented as $\mathscr{A}_i \times \mathscr{W}_i$.

Eq. (5.17) is illustrated in Fig. 5.12 (top), where the central automaton (obtained by projection) represents the message from $\mathscr{A}_2$ to $\mathscr{A}_1$. The bottom figure illustrates the message propagation and integration in the reverse direction (bottom), *i.e.* the symmetric version of (5.17).

The final distributed explanations are obtained by taking the languages of the automata at the top left and bottom right in Fig. 5.12. One has $\mathscr{E}'_1 = \{a\alpha b\alpha, \beta b\}$ and

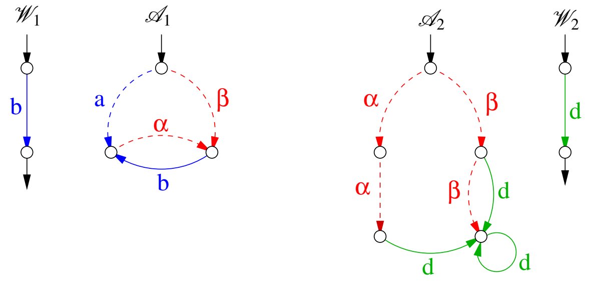

Fig. 5.10 Components $\mathscr{A}_1, \mathscr{A}_2$, on $\Sigma_1 = \{a, b, \alpha, \beta\}$ and $\Sigma_2 = \{\alpha, \beta, d\}$ resp. Only labels b and d are observable (dashed transitions are unobservable). All states are final in the $\mathscr{A}_i$. Automata $\mathscr{W}_1, \mathscr{W}_2$ encode the observed words w_1, w_2; only their bottom states are final

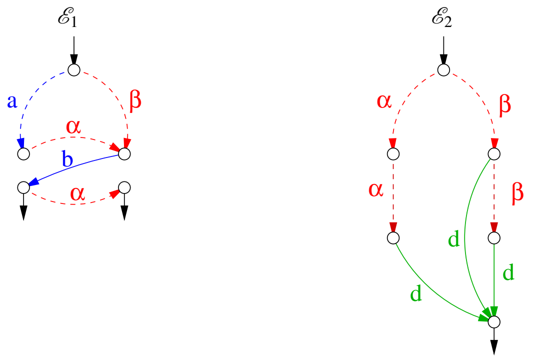

Fig. 5.11 Local explanations to the observed word of w_i in each component $\mathscr{A}_i$, represented as the language of $\mathscr{E}_i = \mathscr{A}_i \times \mathscr{W}_i$

$\mathscr{E}'_2 = \{\alpha\alpha d, \beta d\}$. Each word in $\mathscr{E}'_1$ matches at least one word of $\mathscr{E}'_2$, and *vice versa*. This yields two pairs of runs of $\mathscr{A}_1$ and $\mathscr{A}_2$ that explain the distributed observation (b, d): $(a\alpha b\alpha, \alpha\alpha d)$ and $(\beta b, \beta d)$. Observe that each pair can be interleaved in several manners to produce explaining runs of $\mathscr{A}_1 \times \mathscr{A}_2$. This shows the interest of distributed state/run computations: useless interleavings need not be explored, which can greatly reduce the search space. ■

Remarks

1. If component $\mathscr{A}_1$ is not deterministic, one can easily recover its explaining runs in the 'automaton version' of (5.17): $\mathscr{A}'_1 \triangleq \mathscr{A}_1 \times [\mathscr{W}_1 \times \Pi_{\Sigma_1 \cap \Sigma_2}(\mathscr{A}_2 \times \mathscr{W}_2)]$. The bracketted term simply constrains the runs of $\mathscr{A}_1$. Any run of $\mathscr{A}'_1$ (restricted to its component in $\mathscr{A}_1$) is a local view of a run π of $\mathscr{A}$ that explains the distributed observation (w_1, w_2).
2. From the runs of the $\mathscr{A}_i$ that match the distributed observation (w_1, w_2), one easily recovers the possible final states of $\mathscr{A}_i$, and consequently can establish a diagnosis, relying on the state augmentation trick.

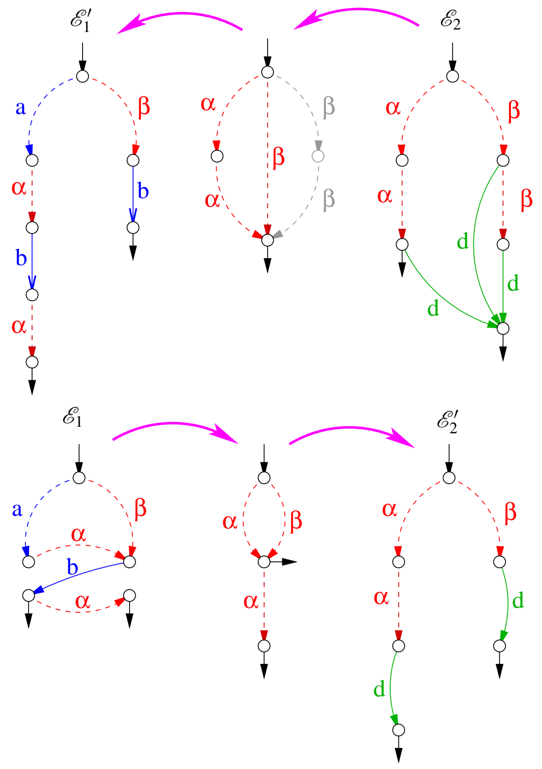

Fig. 5.12 Message propagation from $\mathscr{A}_2$ to $\mathscr{A}_1$ (top) and from $\mathscr{A}_1$ to $\mathscr{A}_2$ (bottom) to compute the local views $\mathscr{E}_1', \mathscr{E}_2'$ of global explanations

3. Observe that, by contrast with the counter-example of the previous section, we have a single observation and with limited length here. This does not prevent however having to check an infinite number of assumptions on the possible (unobserved) synchronizations that take place between the two components. What makes the approach work is that this possibly infinite set of explanations can be condensed into an automaton, thanks to its regularity.

5.7 Conclusion and Further Reading

What are the lessons of the above developments? First of all, diagnosis and state estimation are closely related problems, if not equivalent. Diagnosers and observers are obtained by simple and similar operations on automata. They extend without difficulty to weighted automata, and in particular to probabilistic automata, with the main difference that determinization may not yield a finite automaton. But if it is the case, they yield structures that output detection probabilities. Diagnosability, or observability of a state property, represents the ability to detect a fault/property not long after it occurs/becomes true. It can be checked on the observer/diagnoser, but is more efficiently tested by a direct method, using the so-called twin-machine. Observability/diagnosability extends to the probabilistic case, and takes the form of the convergence (with probability 1) of some numerical indicator after this property becomes true. In the simplest case, this indicator switches suddenly to 1 (A-diagnosability), while for AA-diagnosability it converges to 1. The first case is rather simple, and actually A-diagnosability can be translated into a structural property of the non-probabilistic observer, or of the twin-machine of $\mathscr{A}$.

Other important lessons relate to distributed or modular systems. Having distributed observations amounts to considering a global observation as a partial orders of events. And similarly, representing runs of a distributed system as a tuple of partially synchronized sequences amounts to considering them as partial orders of events. This somehow invisible change of semantics greatly saves in complexity when dealing with distributed systems. It still allows one to perform state estimation or diagnosis, possibly with distributed methods. Notice that for distributed diagnosis, the properties one wishes to characterize must also be expressible as a product of local properties (one per component). Or they should be separable, following the vocabulary of [20]. Distributed/modular diagnosability has been examined by different authors [18], assuming or not that synchronization events are observable, which greatly simpifies the problem, as mentioned above. Modular state estimation or diagnosis for probabilistic systems remains an open issue. A central difficulty in such settings is to define a probabilistic setting that is compatible with the concurrency of events: a not careful way of combining probabilistic automata generally produces weird phenomena, for example private events of some component may change the occurrence probability of private events in another component... A next step towards the management of distributed systems consists in adopting true concurrency semantics. See [6] for a discussion, and Chapter 15 or [1, 5] for a detailed treatment.

References

1. Benveniste, A., Fabre, E., Jard, C., Haar, S.: Diagnosis of asynchronous discrete event systems a net unfolding approach. IEEE Transactions on Automatic Control 48(5), 714–727 (2003)
2. Buchsbaum, A.L., Giancarlo, R., Westbrook, J.R.: On the determinization of weighted finite automata. SIAM Journal on Computing 30, 1502–1531 (1998)
3. Cassandras, C.G., Lafortune, S.: Introduction to Discrete Event Systems, 2nd edn. Springer (2008)
4. Cortes, C., Mohri, M., Rastogi, A., Riley, M.D.: Efficient Computation of the Relative Entropy of Probabilistic Automata. In: Correa, J.R., Hevia, A., Kiwi, M. (eds.) LATIN 2006. LNCS, vol. 3887, pp. 323–336. Springer, Heidelberg (2006)

5. Fabre, E., Benveniste, A., Haar, S., Jard, C.: Distributed monitoring of concurrent and asynchronous Systems. Journal of Discrete Event Systems 15(1), 33–84 (2005)
6. Fabre, E., Benveniste, A.: Partial order techniques for distributed discrete event systems: why you can't avoid using them. Journal of Discrete Events Dynamical Systems 17(3), 355–403 (2007)
7. Fabre, E., Jezequel, L.: Distributed optimal planning: an approach by weighted automata calculus. In: Proc. 48th Conference on Decision and Control, Shangai, China (2009)
8. Fabre, E., Jezequel, L.: On the construction of probabilistic diagnosers. In: Proc. 10th Workshop on Discrete Event Systems, Berlin, Germany (2010)
9. Jeron, T., Marchand, H., Pinchinat, S., Cordier, M.O.: Supervision patterns in discrete event systems diagnosis. In: Proc. 8th Workshop on Discrete Event Systems, Ann Arbor, Michigan (2006)
10. Kirsten, D., Murer, I.: On the determinization of weighted automata. Journal of Automata, Languages and Combinatorics 10(2/3), 287–312 (2005)
11. Mohri, M.: Weighted automata algorithms. In: Kuich, W., Vogler, H., Droste, M. (eds.) Handbook of Weighted Automata. Springer (2009)
12. Mohri, M.: Finite-state transducers in language and speech processing. Computational Linguistics 23, 269–311 (1997)
13. Mohri, M.: Generic epsilon-removal and input epsilon-renormalization algorithms for weighted transducers. International Journal on Foundations of Computer Sciences 13(1), 129–143 (2002)
14. Paz, A.: Introduction to Probabilistic Automata. Academic Press, New-York (1971)
15. Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., Teneketzis, D.: Diagnosability of discrete-event systems. IEEE Transactions on Automatic Control 40(9), 1555–1575 (1995)
16. Thorsley, D., Teneketzis, D.: Diagnosability of stochastic discrete-event systems. IEEE Transactions on Automatic Control 50(4), 476–492 (2005)
17. Thorsley, D., Yoo, T.S., Garcia, H.E.: Diagnosability of stochastic discrete-event systems under unreliable observations. In: Proc. American Control Conference, Seattle, USA (2008)
18. Ye, L., Dague, P.: An optimized algorithm for diagnosability of component-based systems. In: Proc. 10th Workshop on Discrete Event Systems, Berlin, Germany (2010)
19. Yoo, T.S., Lafortune, S.: Polynomial-time verification of diagnosability of partially observed discrete-event systems. IEEE Transactions on Automatic Control 47(9), 1491–1495 (2002)
20. Zhou, C., Kumar, R., Sreenivas, R.S.: Decentralized modular diagnosis of concurrent discrete event systems. In: 9th Workshop on Discrete Event Systems, Goteborg, Sweden (2008)