Switched Probabilistic I/O Automata

Ling Cheung, Nancy Lynch, Roberto Segala, Frits Vaandrager

Nijmegen Institute for Computing and Information Sciences/

NIII-R0437 September 2004

Nijmegen Institute for Computing and Information Sciences
Faculty of Science
Catholic University of Nijmegen
Toernooiveld 1
6525 ED Nijmegen
The Netherlands

Switched Probabilistic I/O Automata*

Ling Cheung[1], Nancy Lynch[2], Roberto Segala[3], and Frits Vaandrager[1]

[1] Nijmegen Institute for Computing and Information Sciences
University of Nijmegen
P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
{lcheung,fvaan}@cs.kun.nl**

[2] MIT Computer Science and Artificial Intelligence Laboratory
Cambridge, MA 02139, USA
lynch@theory.csail.mit.edu***

[3] Dipartimento di Informatica, Università di Verona
Strada Le Grazie 15, 37134 Verona , Italy
roberto.segala@univr.it†

Abstract. A *switched probabilistic I/O automaton* is a special kind of probabilistic I/O automaton (PIOA), enriched with an explicit mechanism to exchange control with its environment. Every closed system of switched automata satisfies the key property that, in any reachable state, at most one component automaton is active. We define a trace-based semantics for switched PIOAs and prove it is compositional. We also propose *switch extensions* of an arbitrary PIOA and use these extensions to define a new trace-based semantics for PIOAs.

1 Introduction

Probabilistic automata [Seg95,SL95,Sto02] constitute a mathematical framework for modeling and analyzing probabilistic systems, specifically, systems of asynchronously interacting components capable of nondeterministic and probabilistic choices. This framework has been successfully adopted in the studies of distributed algorithms [LSS94,PSL00,Agg94] and practical communication protocols [SV99]. It also appears to be useful for modeling and analyzing security protocols.

An important part of a system modeling framework is a notion of *visible behavior* of system components. Such a notion is used to derive implementation and equivalence relations among components. For example, one can define the

* An extended abstract of this paper will appear as [CLSV04].

** Supported by DFG/NWO bilateral cooperation project 600.050.011.01 Validation of Stochastic Systems (VOSS).

*** Supported by DARPA/AFOSR MURI Award #F49620-02-1-0325, MURI AFOSR Award #F49620-00-1-0327, NSF Award #CCR-0326277, and USAF, AFRL Award #FA9550-04-1-0121.

† Supported by MURST project Constraint-based Verification of reactive systems (CoVer).

visible behavior of a nondeterministic automaton to be its set of *traces*— sequences of visible actions that arise during executions of the automaton [LT89]. This induces an implementation (resp. equivalence) relation on nondeterministic automata, namely inclusion (resp. equality) of sets of traces.

Perhaps the most important property of an implementation relation is *compositionality*: if P implements Q, then for every context R, one should be able to infer that $P\|R$ implements $Q\|R$. This greatly facilitates correctness proofs of complex systems by reducing properties of a large system to properties of smaller subsystems. In the setting of security analysis, for instance, compositionality ensures that plugging secure components into a security preserving context results again in a secure component [SM03].

Generalizing the notion of traces, Segala [Seg95] defines the visible behavior of a probabilistic automaton as its set of *trace distributions*, where each trace distribution is induced by a probabilistic *scheduler* which resolves all nondeterministic choices. This gives rise to implementation and equivalence relations as inclusion and equality of sets of trace distributions, respectively. It turns out that this notion of implementation relation is not compositional. A simple counterexample is illustrated in Figure 1.

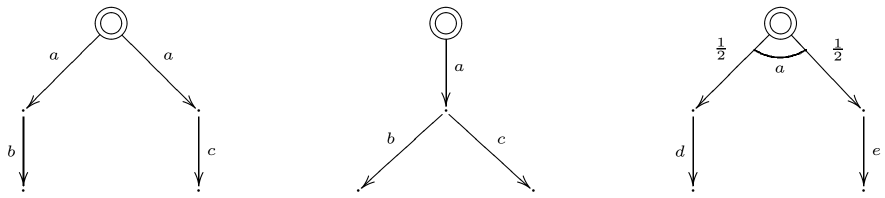


Fig. 1. Probabilistic automata Early, Late and Toss

As their names suggest, automaton Early forces its scheduler to choose between b and c as it chooses one of the two available a-transitions, whereas automaton Late allows its schedulers to make this decision after the a-transition. Clearly, these two automata have the same set of trace distributions, but they can be distinguished by the context Toss. The automaton Toss has a probabilistic a-transition leading to a uniform distribution on two states, one of which enables a d-transition while the other enables an e-transition. The composed system Late $\|$ Toss has a trace distribution D_0 that assigns probability $\frac{1}{2}$ to each of these traces: adb and aec (see Figure 2). Such total correlations between actions d and b, and between actions e and c, cannot be achieved by the composite Early $\|$ Toss.

Inspired by this example, we establish in [LSV03] that the coarsest precongruence refining trace distribution preorder coincides with the probabilistic simulation preorder. In other words, probabilistic contexts are capable of exposing internal branching structures of other components. This suggests to us a serious limitation in the composition mechanism of probabilistic automata.

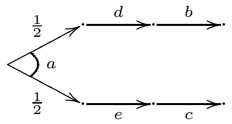


Fig. 2. Non-substitutivity of trace distribution inclusion

Namely, nondeterministic choices are resolved after the two automata are composed, allowing the global scheduler to make decisions in one component using state information of the other. This phenomenon can be viewed as a form of "information leakage": the global scheduler channels private information from one component to the other (from Toss to Late in the previous example).

In this paper, we present a composition mechanism where local scheduling decisions are based on strictly local information. That is, (i) local nondeterministic choices of each component are resolved by that component alone; (ii) global nondeterministic choices (i.e., inter-component choices) are resolved by some independent means. To address the first issue, we introduce an input/output distinction to our model and pair each automaton with an *input/output scheduler*. For the second, we introduce a control-passage[1] mechanism, which eliminates global scheduling conflicts.

Before describing our model in greater detail, we take a quick look at related proposals in the existing literature[2]. For purely synchronous, variable-based models, global nondeterministic choices are resolved by "avoidance": in each transition of the global system, all components may take a step. This intrinsic feature of synchronous models allows De Alfaro, Henzinger and Jhala [dAHJ01] to successfully define a compositional, trace-based semantics for their model of probabilistic reactive modules. For asynchronous models such as probabilistic automata, global nondeterministic choices must be resolved explicitly in order to assign a probability mass to each possible interleaving of actions. Wu, Smolka and Stark [WSS94] propose a compositional model based on probabilistic input/output automata. In that model, global nondeterminism is resolved by a "race" among components: each component draws a delay from an exponential distribution (thus leaving the realm of discrete distributions). Assuming independence of these random draws, the probability of two components drawing the same delay is zero, therefore there is almost always a unique winner. These races are history-independent, in the sense that outcomes depend on current states of components, but not on computation history of the overall environment.

[1] Throughout this paper, the term *control* is used in the spirit of "control flow" in sequential programming: a component is said to possess the control of a system if it is scheduled to actively perform the next action. This should not be confused with the notion of controllers for plants, as in control theory.

[2] We refer to [SV04] for a comparative study of various probabilistic models.

Our treatment of global nondeterminism finds its root in the very meaning of the term "interleaving semantics". In the non-probabilistic case, concurrent behaviors are captured by considering all possible interleavings of actions performed by the various components. However, with the introduction of probabilistic choices, it is no longer obvious what one means by the term "a possible interleaving". Thus, we provide a framework in which "a possible interleaving" has a concrete meaning and therefore compositional reasoning is sound on the level of trace distributions. We will also argue that, despite the appearance of single-threading, this framework is sufficiently expressive to model concurrently executing, communicating components.

We introduce the model of *switched probabilistic I/O automata* (or *switched automata* for short). This augments the probabilistic I/O automata model with some additional structures and axioms. In particular, we add a predicate active on the set of states, indicating whether the automaton is active or inactive[3]. We require that locally controlled actions are enabled only if the automaton is active. In other words, an inactive automaton must be quiescent and can only accept inputs from the environment.

A switched automaton changes its activity status by performing special control input and control output actions. Control inputs switch the machine from inactive to active and vice versa for control outputs. All other actions must leave the activity status unchanged. It is important that all control communications are "handshakes": at most two components may participate in a synchronization labeled by a control action. Together with an appropriate initialization condition, this ensures that at most one component is active at any point of an execution. Intuitively, we model a network of processes passing a single token among them, with the property that a process enables a locally controlled transition if and only if it possesses the token.

The main technical result of this paper is compositionality of a trace-based semantics for switched probabilistic I/O automata (Section 6, Theorem 1). Sections 2 and 3 are devoted to the basic theory. There we introduce new technical notions such as I/O schedulers, scheduled automata and parallel composition of scheduled automata. In Section 4, we define pseudo probabilistic executions and pseudo trace distributions for automata with open inputs, and prove important projection and pasting results. Section 5 treats two standard operators: renaming and hiding. In Section 7, we propose the notion of *switch extensions* for PIOAs, which can be used to derive a new form of composition for the original PIOA model. Concluding discussions follow in Section 8.

2 Preliminaries

In this section, we define probabilistic I/O automata and some related notions. This is a straightforward combination of the Input/Output Automata model

[3] A similar distinction appears in [dAH01]: *running* states vs. *waiting* states. It arises as part of an on-the-fly state space reduction technique for checking compatibility of interface automata.

of Lynch and Tuttle [LT89] and the Simple Probabilistic Automata model of Segala [Seg95].

2.1 PIOAs

A *discrete probability (resp. sub-probability) measure* over a set X is a measure μ on $(X, 2^X)$ such that $\mu(X) = 1$ (resp. $\mu(X) \leq 1$). With slight abuse of notation, we write $\mu(x)$ for $\mu(\{x\})$. The set of all discrete probability measures over X is denoted $\mathsf{Disc}(X)$; similarly for $\mathsf{SubDisc}(X)$. Moreover, we use $\mathsf{Supp}(\mu)$ to denote the *support* of a discrete measure μ: the set of elements in X to which μ assigns nonzero measure. Given $x \in X$, the *Dirac distribution* on x is the unique measure assigning probability 1 to x, denoted $(x \mapsto 1)$.

A *probabilistic I/O automaton (PIOA) P* consists of:

- a set $\mathsf{States}(P)$ of states and a *start state* $s^0 \in \mathsf{States}(P)$;
- a set $Act(P)$ of action symbols, partitioned into: I (*input* actions), O (*output* actions) and H (*hidden* actions);
- a transition relation $\rightarrow \subseteq \mathsf{States}(P) \times Act(P) \times \mathsf{Disc}(\mathsf{States}(P))$.

An action is *visible* if it is not hidden. It is *locally controlled* if it is non-input (i.e., either output or hidden); we define $L := O \cup H$. We write $s \xrightarrow{a} \mu$ for $\langle s, a, \mu \rangle \in \rightarrow$, and $s \xrightarrow{a} s'$ if there exists μ with $s \xrightarrow{a} \mu$ and $s' \in \mathsf{Supp}(\mu)$. A state is *quiescent* if it enables only input actions. A PIOA is *closed* if its set of input actions is empty. It is *deterministic* if, for each state s and action symbol a, there is at most one a-transition leaving s. As with I/O automata, we always assume *input enabling*: $\forall s \in \mathsf{States}(P)\ \forall a \in I\ \exists \mu : s \xrightarrow{a} \mu$.

An *execution* of P is a (possibly finite) sequence $p = s_0 a_1 \mu_1 s_1 a_2 \mu_2 s_2 \ldots$,[4] such that:

- each s_i (resp., a_i, μ_i) denotes a state (resp., action, distribution over states);
- $s_0 = s^0$ and, if p is finite, then p ends with a state;
- for each non-final i, $s_i \xrightarrow{a_{i+1}} \mu_{i+1}$ and $s_{i+1} \in \mathsf{Supp}(\mu_{i+1})$.

Given a finite execution p, we use $\mathsf{last}(p)$ to denote the last state of p. A state s is *reachable* if there exists a finite execution p such that $\mathsf{last}(p) = s$. We write $\mathsf{Exec}(P)$ for the set of all executions of P and $\mathsf{Exec}^{<\omega}(P)$ for the set of finite executions. Given an execution p, the sequence of visible action symbols in p is called the *(visible) trace* of p, denoted $\mathsf{tr}(p)$.

A finite set of PIOAs $\{P_1, \ldots, P_n\}$ is said to be *compatible* if for all $i \neq j$, $O_i \cap O_j = Act(P_i) \cap H_j = \emptyset$. Such a set is *closed* if $\bigcup_{1 \leq i \leq n} I_i \subseteq \bigcup_{1 \leq i \leq n} O_i$. We define $P = \|_{1 \leq i \leq n} P_i$ as usual with synchronization of shared actions:

- $\mathsf{States}(P) := \prod_{1 \leq i \leq n} \mathsf{States}(P_i)$ and the start state of P is $\langle s_1^0, \ldots, s_n^0 \rangle$;
- $I := \bigcup_{1 \leq i \leq n} I_i \setminus \bigcup_{1 \leq i \leq n} O_i$, $O := \bigcup_{1 \leq i \leq n} O_i$, and $H := \bigcup_{1 \leq i \leq n} H_i$;

[4] Some authors define executions to be sequences of states and actions in alternating fashion, thus omitting the target distributions. The current style allows for a more straightforward definition of probabilistic executions.

– given a state $\langle s_1, \ldots, s_n\rangle$, an action a and a target distribution μ, there is a transition $\langle s_1, \ldots, s_n\rangle \xrightarrow{a} \mu$ if and only if μ is of the form $\mu_1 \times \ldots \times \mu_n$ and for all $1 \leq i \leq n$,
  - either $a \in Act(P_i)$ and $s_i \xrightarrow{a} \mu_i$,
  - or $a \notin Act(P_i)$ and $\mu_i = (s_i \mapsto 1)$.

Notice $\|$ is commutative and associative for PIOAs. Moreover, given a finite execution p of a composite $\|_{1\leq i\leq n} P_i$, we define its *i-th projection* recursively as follows:

– $\pi_i(\langle s_1^0, \ldots, s_n^0\rangle) := s_i^0$;
– $\pi_i(pa(\mu_1 \times \ldots \times \mu_n)\langle s_1', \ldots, s_n'\rangle)$ equals
  - $\pi_i(p)a\mu_i s_i'$, if $a \in Act(P_i)$;
  - $\pi_i(p)$, otherwise.

2.2 I/O Schedulers

The notion of (probabilistic) schedulers for a PIOA P is introduced as a means to resolve all nondeterministic choices in P. Each scheduler consists of an input component and an output component. Given a finite history of the automaton, the output scheduler chooses probabilistically the next locally controlled transition, whereas the input scheduler responds to inputs from the environment and chooses probabilistically a transition carrying the correct input symbol.

Definition 1. *An* input scheduler σ *for* P *is a function*

$$\sigma : \mathsf{Exec}^{<\omega}(P) \times I \longrightarrow \mathsf{Disc}(\rightarrow)$$

such that for all $\langle p, a\rangle \in \mathsf{Exec}^{<\omega}(P) \times I$ *and transition* $(s \xrightarrow{b} \mu) \in \mathsf{Supp}(\sigma(p, a))$*, we have* $s = \mathsf{last}(p)$ *and* $b = a$*. An* output scheduler ρ *for* P *is a function*

$$\rho : \mathsf{Exec}^{<\omega}(P) \longrightarrow \mathsf{SubDisc}(\rightarrow)$$

such that for all $p \in \mathsf{Exec}^{<\omega}(P)$ *and transition* $(s \xrightarrow{a} \mu) \in \mathsf{Supp}(\rho(p))$*, we have* $s = \mathsf{last}(p)$ *and* $a \in L$*. An* I/O *scheduler for* P *is then a pair* $\langle\sigma, \rho\rangle$ *where* σ *is an input scheduler for* P *and* ρ *is an output scheduler for* P*.*

Notice input schedulers must return a discrete probability distribution, reflecting the requirement that each input issued by the environment is received with probability 1. (This is always possible because of the input enabling assumption.) In particular, if P is deterministic, then input schedulers for P always return Dirac distributions. In contrast, output schedulers may choose to halt with an arbitrary probability θ by returning a proper sub-distribution whose total probability mass is $1 - \theta$.

An I/O scheduler $\langle\sigma, \rho\rangle$ is said to be *deterministic* if both σ and ρ always return Dirac distributions. Moreover, we write $\sigma(p, a)(\mu)$ as a shorthand for $\sigma(p, a)(\mathsf{last}(p) \xrightarrow{a} \mu)$ and $\rho(p)(a, \mu)$ for $\rho(p)(\mathsf{last}(p) \xrightarrow{a} \mu)$.

Consider a closed PIOA P. Obviously, any I/O scheduler for P has a trivial input component (i.e., the empty function). Every output scheduler ρ thus induces a purely probabilistic behavior, which is captured by the following notion of probabilistic executions.

Definition 2. *Let P be a closed PIOA and let ρ be an output scheduler for P. The* probabilistic execution *induced by ρ is the function $Q_\rho : \mathsf{Exec}^{<\omega}(P) \longrightarrow [0,1]$ defined recursively by:*

- *$Q_\rho(s^0) := 1$, where s^0 is the initial state of P;*
- *$Q_\rho(p') := Q_\rho(p) \cdot \rho(p)(a,\mu) \cdot \mu(s')$, where p' is of the form $pa\mu s'$.*

A probabilistic execution Q_ρ induces a probability space over the sample space $\Omega_P := \mathsf{Exec}(P)$ as follows. Let $\sqsubseteq$ denote the prefix ordering on sequences. Each $p \in \mathsf{Exec}^{<\omega}(P)$ generates a *cone* of executions: $\mathbf{C}_p := \{p' \in \mathsf{Exec}(P) \mid p \sqsubseteq p'\}$. Let $\mathcal{F}_P$ denote the smallest σ-field generated by the collection $\{\mathbf{C}_p \mid p \in \mathsf{Exec}^{<\omega}(P)\}$. There exists a unique measure $\mathbf{m}_\rho$ on $\mathcal{F}_P$ with $\mathbf{m}_\rho[\mathbf{C}_p] = Q_\rho(p)$ for all p in $\mathsf{Exec}^{<\omega}(P)$; therefore Q_ρ gives rise to a probability space $(\Omega_P, \mathcal{F}_P, \mathbf{m}_\rho)$.

2.3 Trace Distributions

Trace distributions are obtained from probabilistic executions by removing non-visible elements. In our case, these are states, hidden actions and distributions of states. To state this precisely, we need the notion of minimal executions: a finite execution p of P is said to be *minimal* if every proper prefix of p has a strictly shorter trace. Notice, the empty execution (i.e., the sequence containing just the initial state) is minimal. Moreover, if p is nonempty and finite, then p is minimal if and only if the last transition in p has a visible action label. For each $\alpha \in Act(P)^{<\omega}$, let $\mathsf{tr}^{-1}_{\min}(\alpha)$ denote the set of minimal executions of P with trace α.

Now we define a lifting of the trace operator $\mathsf{tr} : \mathsf{Exec}^{<\omega}(P) \longrightarrow Act(P)^{<\omega}$. Given a function $Q : \mathsf{Exec}^{<\omega}(P) \longrightarrow [0,1]$, define $\mathsf{tr}(Q) : Act(P)^{<\omega} \longrightarrow [0,1]$ by

$$\mathsf{tr}(Q)(\alpha) := \sum_{p \in \mathsf{tr}^{-1}_{\min}(\alpha)} Q(p).$$

Given an output scheduler ρ of a closed PIOA P, the *trace distribution* induced by ρ (denoted D_ρ) is simply the result of applying tr to the probabilistic execution Q_ρ. That is, $D_\rho := \mathsf{tr}(Q_\rho)$. We often use variables D, D', etc. for trace distributions, thus leaving the scheduler ρ implicit.

Similar to the case of probabilistic executions, each D_ρ induces a probability measure on the sample space $\Omega := Act(P)^{\leq\omega}$. There the σ-field $\mathcal{F}$ is generated by the collection $\{\mathbf{C}_\alpha \mid \alpha \in Act(P)^{<\omega}\}$, where $\mathbf{C}_\alpha := \{\alpha' \in \Omega \mid \alpha \sqsubseteq \alpha'\}$. The measure $\mathbf{m}^\rho$ on $\mathcal{F}$ is uniquely determined by the equations $\mathbf{m}^\rho[\mathbf{C}_\alpha] = D_\rho(\alpha)$ for all $\alpha \in Act(P)^{<\omega}$.

In the literature, most authors define probabilistic executions (resp. trace distributions) to be the probability spaces $\langle \Omega_P, \mathcal{F}_P, \mathbf{m}_\rho \rangle$ (resp. $\langle \Omega, \mathcal{F}, \mathbf{m}^\rho \rangle$).

Here we find it more natural to reason with the functions Q_ρ and D_ρ, rather than the induced measures. We refer to [Seg95] for these alternative definitions and proofs that they are equivalent to our versions.

3 Switched Probabilistic I/O Automata

As we argued in Section 1, one must distinguish between global and local nondeterministic choices and must resolve them separately. Our solution lies in an explicit mechanism of control exchange among parallel components.

The notion of control exchange in fact arises naturally in asynchronous models of currency with input/output distinction, although it is often left implicit. For example, in the I/O automata model of [LT89], control exchange takes place between an automaton and its environment whenever an input transition follows a locally controlled transition, or vice versa. In this view, each trace of a composite nondeterministic automaton represents a single-threaded activity in a system of components. Therefore, in order to properly generalize the notion of traces, we should incorporate the notion of single-threaded activity into our definition of trace distributions. This is precisely the idea that leads to our explicit treatment of control exchange.

In our framework, each active component is allowed to run to completion, in the sense that control exchange takes place only when the active component schedules a control output transition, which uniquely determines the next active component. In other words, the local scheduler for an active component decides not only when to give up the token, but also to whom the token is transfered. Therefore, as soon as a local scheduler is chosen for every component, we have fully specified a "possible interleaving" of the composed system. Concurrent behaviors of the system are then captured by varying the choice of local schedulers for individual components.

The rest of this section is organized as follows: (i) first we define *pre-switched automata*, where we describe control action signatures and the Boolean-valued state variable active; (ii) then we introduce the notion of *input well-behaved executions* of a pre-switched automaton and state four axioms defining *switched automata*; (iii) finally, we introduce the notion of a *scheduled automaton*, essentially a switched automaton paired with an I/O scheduler.

3.1 Pre-Switched Automata

For technical simplicity, we assume a universal set Act of action symbols such that $Act(P) \subseteq Act$ for every PIOA P. Moreover, Act is partitioned into two sets: $BAct$ (*basic actions*) and $CAct$ (*control actions*). Both sets are assumed to be countably infinite, so we can rename hidden actions using fresh symbols whenever necessary (see Section 5).

Definition 3. *A pre-switched automaton P is a PIOA endowed with a function* $\mathsf{active} : \mathsf{States}(P) \longrightarrow \{0, 1\}$ *and a set $Sync \subseteq O \cap CAct$ of* synchronized control *actions.*

We use variables P, Q, etc. to denote pre-switched automata. Given a pre-switched automaton P, we further classify its action symbols:

- $BI := I \cap BAct$ (*basic inputs*);
- $BO := O \cap BAct$ (*basic outputs*);
- $CI := I \cap CAct$ (*control inputs*);
- $CO := (O \cap CAct) \setminus Sync$ (*control outputs*).

Essentially, we have a partition $\{BI, BO, H, CI, CO, Sync\}$ of $Act(P)$. We say that P is *initially active* if $\mathsf{active}(s^0) = 1$. Otherwise, it is *initially inactive*.

As described in Section 1, the Boolean-valued function active on the states of P indicates whether P is active or inactive, while control actions allow P to exchange control with its environment. The designation of synchronized control actions helps to achieve the "handshake" condition on control synchronizations: whenever we compose two automata, we classify the shared control actions as "synchronized", so that they are no longer available for further synchronization with a third component. This is made precise in the definitions of compatibility and composition for pre-switched automata.

A finite set of pre-switched automata $\{P_1, \ldots, P_n\}$ is said to be *compatible* if (i) $\{P_1, \ldots, P_n\}$ is a compatible set of PIOAs; (ii) for all $i \neq j$, $Act(P_i) \cap Sync_j = CI_i \cap CI_j = \emptyset$; (iii) at most one P_i is initially active. Notice that such a set is compatible if and only if for all $i \neq j$, P_i and P_j are compatible. The *parallel composition* of $\{P_1, \ldots, P_n\}$, denoted $\|_{1\leq i\leq n} P_i$, is the result of composing $P_1, \ldots, P_n$ as PIOAs, together with:

- $Sync := \bigcup_{1\leq i\leq n} Sync_i \cup \bigcup_{1\leq i,j\leq n} (CI_i \cap CO_j)$;
- $\mathsf{active}(s_1, \ldots, s_n) = 1$ if and only if for some i, $\mathsf{active}_i(s_i) = 1$.

Clearly, the composite $\|_{1\leq i\leq n} P_i$ is again a pre-switched automaton. We consider some basic properties of the n-ary operator $\|_{1\leq i\leq n}$.

Lemma 1. *The following equalities hold:*

- $BI = \bigcup_{1\leq i\leq n} BI_i \setminus \bigcup_{1\leq i\leq n} BO_i$;
- $CI = \bigcup_{1\leq i\leq n} CI_i \setminus \bigcup_{1\leq i\leq n} CO_i$;
- $BO = \bigcup_{1\leq i\leq n} BO_i$;
- $CO = \bigcup_{1\leq i\leq n} CO_i \setminus \bigcup_{1\leq i\leq n} CI_i$.

Proof. By definition, $I = \bigcup_{1\leq i\leq n} I_i \setminus \bigcup_{1\leq i\leq n} O_i$. Since $BAct$ and $CAct$ are disjoint, we have the desired properties about BI and CI.

Similarly, $O = \bigcup_{1\leq i\leq n} O_i$. Thus $BO = \bigcup_{1\leq i\leq n} BO_i$ and $O \cap CAct = \bigcup_{1\leq i\leq n} CO_i$. Applying the definitions of CO and $Sync$, we have $CO = \bigcup_{1\leq i\leq n} CO_i \setminus \bigcup_{1\leq i\leq n} CI_i$. □

In the binary case, we write $P_1 \| P_2$ as shorthand for $\|_{1\leq i\leq 2} P_i$. Observe that $P_1 \| P_2 \cong P_2 \| P_1$; that is, composition of pre-switched automata is commutative up to isomorphism. Next we check that composition is also associative on the class of pre-switched automata. Lemma 2 says, if an automaton is compatible with a composite, then it is compatible with every component in that composite. Conversely, Lemma 3 says, if an automaton is compatible with each component in a composite, then it is compatible with the composite.

Lemma 2. *Let P_1, P_2 and P_3 be pre-switched automata. Assume that P_1 is compatible with P_2 and P_3 is compatible with $P_1\|P_2$. Then P_1 is compatible with P_3. (By symmetry, P_2 is also compatible with P_3.)*

Proof. By definition of active in a composite, we know that at most one P_i is initially active. It remains to check the signatures are compatible:

- $Act_1 \cap H_3 \subseteq Act_{12} \cap H_3 = \emptyset$;
- $Act_3 \cap H_1 \subseteq Act_3 \cap H_{12} = \emptyset$;
- $O_1 \cap O_3 \subseteq O_{12} \cap O_3 = \emptyset$;
- $Act_1 \cap Sync_3 \subseteq Act_{12} \cap Sync_3 = \emptyset$;
- $Act_3 \cap Sync_1 \subseteq Act_3 \cap Sync_{12} = \emptyset$;
- $CI_1 \cap CI_3 \subseteq (CI_{12} \cup Sync_{12}) \cap CI_3 = \emptyset$. □

Corollary 1. *Let P_1, P_2 and P_3 be pre-switched automata. Assume that P_1 is compatible with P_2 and P_3 is compatible with $P_1\|P_2$. Then $\{P_1, P_2, P_3\}$ is also a compatible set.*

Lemma 3. *Let P_1, P_2 and P_3 be pre-switched automata. Assume that $\{P_1, P_2, P_3\}$ is a compatible set. Then P_1 is compatible with P_2 and P_3 is compatible with $P_1\|P_2$.*

Proof. The first claim is trivial. Since there is at most one initially active P_i, either $P_1\|P_2$ or P_3 is initially active, but not both. For the signatures, we have:

- $Act_{12} \cap H_3 = (Act_1 \cap H_3) \cup (Act_2 \cap H_3) = \emptyset$;
- $Act_3 \cap H_{12} = (Act_3 \cap H_1) \cup (Act_3 \cap H_2) = \emptyset$;
- $O_{12} \cap O_3 = (O_1 \cap O_3) \cup (O_2 \cap O_3) = \emptyset$;
- $Act_{12} \cap Sync_3 = (Act_1 \cap Sync_3) \cup (Act_2 \cap Sync_3) = \emptyset$;
- $CI_{12} \cap CI_3 \subseteq (CI_1 \cup CI_2) \cap CI_3 = \emptyset$.

It remains to check $Act_3 \cap Sync_{12} = \emptyset$. Clearly Act_3 is disjoint from $Sync_1 \cup Sync_2$. Suppose $a \in CI_1 \cap CO_2$. Then $a \in CAct$. By compatibility, $a \notin Sync_3 \cup CI_3 \cup CO_3$, therefore $a \notin Act_3$. By symmetry, $Act_3 \cap (CI_2 \cap CO_1)$ is also empty. □

This allows us to conclude that composition is associative.

Lemma 4. *Let P_1, P_2 and P_3 be pre-switched automata. Assume P_1 is compatible with P_2, and P_3 is compatible with $P_1\|P_2$. Then P_2 is compatible with P_3, and P_1 is compatible with $P_2\|P_3$. Moreover, $(P_1\|P_2)\|P_3 \cong P_1\|(P_2\|P_3)$.*

Proof. The compatibility claim follows from Corollary 1 and Lemma 3. The second claim follows from the definitions of $Sync$ and active in a parallel composition and the fact that $\|$ is associative for PIOAs. □

3.2 Input Well-behaved Executions

Recall that switched automata are intended to be composed in such a way that at most one component is active at any point of an execution. Consider a system consisting of two pre-switched automata P and Q placed in parallel with an environment E. Suppose E activates P and Q via two consecutive control outputs (which are control inputs for P and Q, respectively). In this case, the environment E exhibits improper behavior: after performing the first control output it should become inactive and thus should not enable the second control output. As a consequence, the composite $P \| Q$ reaches a point where both P and Q are active, violating the intended property of composition. This example suggests that, in some cases, correctness claims about an automaton must be conditional upon correctness of its environment. In other words, we should restrict our attention to those executions in which the environment also follows the rules of control exchange.

This leads to the definition of *input well-behavedness.* Let P be a pre-switched automaton. An input transition $s \xrightarrow{a} \mu$ is *well-behaved* if $\mathsf{active}(s) = 0$. An execution p of P is *input well-behaved* if all input transitions occurring in p are well-behaved. Let $\mathsf{Exec}^{<\omega}_{\mathsf{iwb}}(P)$ denote the set of finite, input well-behaved executions of P. Moreover, we say that a state s is *input well-behaved reachable*, notation $\mathsf{iwbr}(s)$, if there exists an input well-behaved execution p such that $s = \mathsf{last}(p)$. Clearly, the empty execution is input well-behaved and thus the initial state is always input well-behaved reachable. If P is closed (i.e., $I = \emptyset$), then every execution of P is trivially input well-behaved and every reachable state is input well-behaved reachable.

Returning to the previous example, $P \| Q$ becomes active after the first control input from E, thus the second control input is not well-behaved. Later on (in Lemma 6), we will prove that the last state of this execution, where both P and Q are active, is not input well-behaved reachable.

3.3 Switched Automata

We are now prepared to define the notion of switched probabilistic I/O automata. This is done by specifying a set of axioms that relate the defining features of pre-switched automata, namely, the Boolean-valued state variable active and control signatures.

Definition 4. *A* switched (probabilistic I/O) automaton *is a pre-switched automaton P that satisfies the following axioms.*

$$s \xrightarrow{a} \mu \ \wedge \ \mathsf{active}(s) = 0 \quad \Rightarrow \quad a \in I \tag{1}$$

$$s \xrightarrow{a} s' \ \wedge \ a \in CI \quad \Rightarrow \quad \mathsf{active}(s') = 1 \tag{2}$$

$$s \xrightarrow{a} s' \ \wedge \ a \notin CI \cup CO \quad \Rightarrow \quad \mathsf{active}(s) = \mathsf{active}(s') \tag{3}$$

$$\mathsf{iwbr}(s) \ \wedge \ s \xrightarrow{a} s' \ \wedge \ a \in CO \quad \Rightarrow \quad \mathsf{active}(s') = 0 \tag{4}$$

These four axioms formalize our intuitions about control passage. Axiom (1) requires all inactive states to be quiescent. Axioms (2) and (4) say that control inputs lead to active states and control outputs to inactive states. Axiom (3) says that non-control transitions and synchronized control transitions do not change the activity status. Together, they describe an "activity cycle" for the automaton P: (i) while in inactive mode, P does not enable locally controlled transitions, although it may still receive inputs from its environment; (ii) when P receives a control input it moves into active mode, where it may perform hidden or output transitions, possibly followed by a control output; (iii) via this control output P returns to inactive mode.

Notice that Axiom (4) is required for input well-behaved reachable states only. Without this relaxation, the composition of two switched automata may not satisfy Axiom (4). A simple example is the automaton $P \| Q$ described in Section 3.2, where both P and Q have been activated by their environment E. At that point, if either P or Q performs a control output, the composite remains active.

We proceed to confirm that the class of switched automata is closed under the parallel composition operator for pre-switched automata. A set $\{P_1, \ldots, P_n\}$ of switched automata is *compatible* if the set of underlying pre-switched automata is compatible. Define the composite, $\|_{1 \leq i \leq n} P_i$, to be the result of composing the switched automata as pre-switched automata. Lemma 5 below verifies the first three axioms.

Lemma 5. *Let $\{P_1, \ldots, P_n\}$ be a compatible set of switched automata. The composite $\|_{1 \leq i \leq n} P_i$ satisfies Axioms* (1)*,* (2) *and* (3)*.*

Proof. Let $s = \langle s_1, \ldots, s_n \rangle$ be a state of the composite. Recall that $\mathsf{active}(s)$ is defined to be the disjunction $\bigvee_{1 \leq i \leq n} \mathsf{active}_i(s_i)$. Furthermore, every control input (resp. locally controlled action) of the composite is a control input (resp. locally controlled action) of some component, thus Axioms (1) and (2) follow easily.

Consider Axiom (3). It is trivial if $a \in BAct$ and if $a \in Sync_i$ for some i. Otherwise, by compatibility, there exist a unique i and j such that $a \in CI_i \cap CO_j$. Since P_j satisfies Axiom (1) and a is locally controlled by P_j, we know that $\mathsf{active}_j(s_j) = 1$, thus $\mathsf{active}(s) = 1$. By Axiom (2) for P_i, we have $\mathsf{active}_i(s'_i) = 1$, therefore $\mathsf{active}(s') = 1 = \mathsf{active}(s)$. □

Axiom (4) follows from Lemma 6 below.

Lemma 6. *Let $\{P_1, \ldots, P_n\}$ be a compatible set of switched automata. For each finite, input well-behaved execution p of $\|_{1 \leq i \leq n} P_i$, we have:*

(i) for all i, $\pi_i(p)$ is also input well-behaved;
(ii) there is at most one i such that $\mathsf{active}_i(\pi_i(\mathsf{last}(p))) = 1$.

Proof. Induction on the length p. Let s denote $\mathsf{last}(p)$. By definition of compatibility, the empty execution satisfies Property (ii). Property (i) is trivial.

Take an input well-behaved execution p' of the form $pa\mu s'$. Assume the induction hypothesis and consider the following cases.

- $a \in H_i \cup Sync_i$ for some i. Notice that the input well-behavedness condition cannot be violated by appending a hidden or synchronized transition, thus Property (i) is satisfied. Property (ii) holds because, by Axiom (3), hidden and synchronized transitions don't change the status of an automaton.
- $a \in CI_i \cap CO_j$ for some i, j. By definition of compatibility, a is not in the signature of P_k for k distinct from i, j. Thus components other than P_i and P_j do not participate in this a-transition. By Axiom (1), $\mathsf{active}_j(\pi_j(s)) = 1$. By I.H, $\pi_k(p)$ is input well-behaved for all k and $\mathsf{active}_k(\pi_k(s)) = 0$ for k distinct from j. This implies both $\pi_i(p')$ and $\pi_j(p')$ are input well-behaved and thus Property (i) holds for p'. Moreover, since P_i and P_j satisfy Axioms (2) and (4), they exchange status after the a-transition. Therefore Property (ii) also holds.
- $a \in BI$. By input well-behavedness of p' and Axiom (3), we conclude that $\mathsf{active}(s) = \mathsf{active}(s') = 0$, which implies that $\mathsf{active}_i(\pi_i(s)) = \mathsf{active}_i(\pi_i(s')) = 0$ for all i. Therefore both Properties (i) and (ii) are satisfied.
- $a \in BO$. By compatibility and Lemma 1, we may choose a unique i such that a is locally controlled by P_i. Then the induction hypothesis guarantees that $\pi_i(p')$ is input well-behaved and that for all $j \neq i$, $\mathsf{active}_j(\pi_j(s)) = 0$. Therefore $\pi_j(p')$ is input well-behaved for all $j \neq i$. Property (ii) holds due to Axiom (3).
- $a \in CI$. By input well-behavedness of p', $\mathsf{active}(s) = 0$. This implies Property (i). By definition of compatibility, we may choose a unique i such that $a \in CI_i$. By Axiom (2), $\mathsf{active}_i(\pi_i(s')) = 1$. Since no other component participates in this a-step, Property (ii) is also satisfied.
- $a \in CO$. By compatibility and Lemma 1, we may choose a unique i such that a is in the signature of P_i. Property (i) is then immediate. Since a is locally controlled by P_i, we know that $\mathsf{active}_i(\pi_i(s)) = 1$. By the induction hypothesis, $\mathsf{active}_j(\pi_j(s)) = 0$ for all $j \neq i$. Since $\pi_j(s) = \pi_j(s')$ for all such j, Property (ii) also holds. □

Corollary 2. *Let $\{P_1, \ldots, P_n\}$ be a compatible set of switched automata. The composite $\|_{1 \leq i \leq n} P_i$ satisfies Axiom (4) and is therefore also a switched automaton.*

Proof. Let $s \xrightarrow{a} s'$ be a transition such that s is input well-behaved reachable and $a \in CO$. Choose input well-behaved p with $s = \mathsf{last}(p)$ and choose μ such that $s \xrightarrow{a} \mu$ and $s' \in \mathsf{Supp}(\mu)$. Then $p' = pa\mu s'$ is an input well-behaved execution. By compatibility and Lemma 1, we may choose a unique i such that a is in the signature of P_i. Applying Lemma 6, we have that $\pi_i(p)$ is input well-behaved; thus we may apply Axiom (4) to P_i and conclude that $\mathsf{active}_i(\pi_i(s')) = 0$. On the other hand, since a is locally controlled by P_i, we know that $\mathsf{active}_i(\pi_i(s)) = 1$. By Lemma 6, $\mathsf{active}_j(\pi_j(s)) = 0$ for all $j \neq i$. For every such j, P_j does not participate in this a-transition, therefore its activity status remains 0. This gives $\mathsf{active}(s') = 0$.

Since the first three axioms follow from Lemma 5, we may conclude that $\|_{1 \leq i \leq n} P_i$ is a switched automaton. □

To summarize, $\|_{1 \leq i \leq n}$ is a well-defined n-ary operator for switched automata and, in the binary case, associativity follows from Lemma 4.

The following corollary of Lemma 6 verifies our claim that, once a switched automaton P is placed in a closing environment (i.e., an environment capable of providing all inputs to P), it will never follow an execution that is not input well-behaved.

Corollary 3. *Let $\{P_1, \ldots, P_n\}$ be a closed and compatible set of switched automata. For every execution p of $\|_{1 \leq i \leq n} P_i$ and $1 \leq i \leq n$, $\pi_i(p)$ is input well-behaved.*

Intuitively, a closed and compatible set of switched automata represents a network of processes passing a single token among them. The basic assumption is that a component enables a locally controlled transition only if it is in possession of the token. Therefore, in any reachable state, there is a unique component in possession of the token and this active component never receives any inputs. As we shall see in Section 6, the behavior of an automaton is always parameterized by a closing environment; thus we are well justified in restricting our attention to input well-behaved executions.

3.4 Scheduled Automata

Next we turn to scheduling decisions. The notion of I/O schedulers for switched automata is inherited from that of its underlying PIOA.

Definition 5. *A* scheduled *automaton is a triple $\langle P, \sigma, \rho\rangle$ such that P is a switched automaton and $\langle \sigma, \rho\rangle$ is an I/O scheduler for P.*

We use letters S, T, etc. to denote scheduled automata. For each $1 \leq i \leq n$, let S_i denote a scheduled automaton $\langle P_i, \sigma_i, \rho_i\rangle$. The set $\{S_i \mid 1 \leq i \leq n\}$ is said to be *compatible* if $\{P_i \mid 1 \leq i \leq n\}$ is compatible as a set of switched automata. Given such a compatible set of scheduled automata, we obtain its composite by combining the I/O schedulers $\{\langle \sigma_i, \rho_i\rangle \mid 1 \leq i \leq n\}$ into an I/O scheduler $\langle \sigma, \rho\rangle$ for the switched automaton $\|_{1 \leq i \leq n} P_i$.

Definition 6. *Suppose $\{S_i \mid 1 \leq i \leq n\}$ is a compatible set of scheduled automata, where $S_i = \langle P_i, \sigma_i, \rho_i\rangle$ for each i. We construct from this set a* composite *scheduled automaton $\|_{1 \leq i \leq n} S_i := \langle P, \sigma, \rho\rangle$ as follows.*

- *$P := \|_{1 \leq i \leq n} P_i$.*
- *For every finite execution p of P with $\mathsf{last}(p) = s$ and for every $a \in I$,*
  - *$\sigma(p, a)(t \xrightarrow{b} \mu) := 0$ if $t \neq s$ or $b \neq a$;*
  - *otherwise, $\sigma(p, a)(s \xrightarrow{a} \mu_0 \times \ldots \times \mu_n) := \Pi_i c_i$, where c_i equals*
    - *$\sigma_i(\pi_i(p), a)(\mu_i)$, if $a \in I_i$;*
    - *1, otherwise.*
- *For every finite execution p of P with $\mathsf{last}(p) = s$,*
  - *$\rho(p)(t \xrightarrow{a} \mu) := 0$ if p is not input well-behaved, $t \neq s$, or $a \notin L$;*

- *otherwise,* $\rho(p)(s \xrightarrow{a} \mu_0 \times \ldots \times \mu_n) := \Pi_i c_i$*, where* c_i *equals*
  - $\rho_i(\pi_i(p))(a, \mu_i)$*, if* $a \in L_i$*;*
  - $\sigma_i(\pi_i(p), a)(\mu_i)$ *if* $a \in I_i$*;*
  - 1*, otherwise.*

The next two lemmas verify that $\langle \sigma, \rho \rangle$ is in fact an I/O scheduler for P.

Lemma 7. *The function* σ *in Definition 6 is in fact an input scheduler for* P*.*

Proof. Let p be a finite execution of P with $\mathsf{last}(p) = s$ and let $a \in I$ be given. By definition, $\sigma(p, a)$ assigns nonzero probability only to transitions of the form $s \xrightarrow{a} \mu$.

Let $i_1, \ldots, i_N$ be an enumeration of the set of indices i such that $a \in I_i$. For each $1 \leq k \leq N$, let X_k denote the set $\{\mu \mid \pi_{i_k}(s) \xrightarrow{a} \mu\}$. Let $X := X_1 \times \ldots \times X_N$. Then we have

$$\begin{aligned}
&\sum_{\mu : s \xrightarrow{a} \mu} \sigma(p, a)(\mu) \\
&= \sum_{\langle \mu_1, \ldots, \mu_N \rangle \in X} \prod_{k=1}^{N} \sigma_{i_k}(\pi_{i_k}(p), a)(\mu_k) && \text{definition of composition} \\
&= \sum_{\mu_1 \in X_1} \cdots \sum_{\mu_N \in X_N} \prod_{k=1}^{N} \sigma_{i_k}(\pi_{i_k}(p), a)(\mu_k) && \text{Cartesian product} \\
&= \prod_{k=1}^{N} \sum_{\mu_k \in X_k} \sigma_{i_k}(\pi_{i_k}(p), a)(\mu_k) && \text{factorization } k \text{ times} \\
&= 1. && \sigma_{i_k}(\pi_{i_k}(p), a) \text{ discrete distribution}
\end{aligned}$$

□

Lemma 8. *The function* ρ *in Definition 6 is in fact an output scheduler for* P*.*

Proof. Let p be a finite execution of $\|_{1 \leq i \leq n} P_i$ with $\mathsf{last}(p) = s$. By definition, $\rho(p)$ has empty support if p is not input well-behaved. Therefore, we may assume p is input well-behaved. By Lemma 6, there is at most one j such that $\mathsf{active}_j(\pi_j(s)) = 1$. Hence, by Axiom (1), there is at most one j such that $\pi_j(s)$ enables a locally controlled transition. If such j does not exist, then s does not enable any locally controlled transitions and, by definition, $\rho(p)$ must have an empty support.

Otherwise, choose a unique j such that s enables locally controlled transitions of P_j. Fix $a \in L_j$. Let $Y := \{\mu \mid \pi_j(s) \xrightarrow{a} \mu\}$ and let $i_1, \ldots, i_N$, $X_1, \ldots, X_k$, X

be given as in Lemma 7. Then we have

$$\begin{aligned}\sum_{\overline{\mu}:s\xrightarrow{a}\overline{\mu}} \rho(p)(a,\overline{\mu}) &= \sum_{\langle \mu,\mu_1,\ldots,\mu_N\rangle\in Y\times X} \rho_j(\pi_j(p))(a,\mu)\cdot\prod_{k=1}^{N}\sigma_{i_k}(\pi_{i_k}(p),a)(\mu_k)\\ &= \sum_{\mu\in Y}\rho_j(\pi_j(p))(a,\mu)\cdot\sum_{\langle \mu_1,\ldots,\mu_N\rangle\in X}\prod_{k=1}^{N}\sigma_{i_k}(\pi_{i_k}(p),a)(\mu_k)\\ &= \sum_{\mu\in Y}\rho_j(\pi_j(p))(a,\mu),\end{aligned}$$

where the last equality follows the proof of Lemma 7.

Now we sum over all $a \in L_j$:

$$\sum_{a\in L_j}\sum_{\overline{\mu}:s\xrightarrow{a}\overline{\mu}}\rho(p)(a,\overline{\mu}) = \sum_{a\in L_j}\sum_{\mu:\pi_j(s)\xrightarrow{a}\mu}\rho_j(\pi_j(p))(a,\mu).$$

This is always at most 1 because $\rho_j(\pi_j(p))$ is a discrete sub-distribution. □

Corollary 4. *The parallel composition operator* $\|_{1\leq i\leq n}$ *is well-defined for scheduled automata.*

Proof. By Corollary 2 and Lemmas 7 and 8.

As usual, we write $S_1\|S_2$ for $\|_{1\leq i\leq 2}S_i$, provided S_1 and S_2 are compatible. Associativity of $\|$ for scheduled automata follows from that for switched automata and a routine check on the I/O schedulers. Finally, the notions of probabilistic executions and trace distributions for closed scheduled automata are inherited from those of PIOAs. In particular, we write Q_S (respectively, D_S) for the probabilistic execution (respectively, trace distribution) induced by the output scheduler of a closed scheduled automaton S.

4 Projection and Pasting

In this section, we study projection and pasting of probabilistic behaviors. Such results are essential elements in constructing a compositional modeling framework. We begin by introducing the notion of regular executions, which will be used to define pseudo trace distributions for automata with open inputs. In Lemma 14, we prove that the pseudo distribution of a composite is uniquely determined by those of its components. Finally, we prove the main pasting lemma for closed automata (Lemma 16), which plays a crucial role in the proof of our main compositionality theorem (Theorem 1).

4.1 Regular Executions

Given an execution p of a switched automaton P, we say that p is *regular* if it is both minimal and input well-behaved. Given a finite sequence α of visible actions in P, let $\mathsf{tr}^{-1}_{\mathsf{reg}}(\alpha)$ denote the set of regular executions of P with trace α. Notice that regularity coincides with minimality in case P is closed. Moreover, regularity is preserved under the operation of appending input transitions.

We make the following observation about regular executions ending with an input transition.

Lemma 9. *Let p' be a nonempty regular execution and let p be the one-step prefix of p'. That is, $p' = pa\mu s'$ for some a, μ and s'. If a is an input action, then p is also regular.*

Proof. Notice that any prefix of an input well-behaved execution is input well-behaved. It remains to show p is minimal. Without loss, assume p is nonempty. For contradiction, suppose that p is of the form $qb\overline{\mu}s$ where b is a hidden action. By Axioms (1) and (3), we have $\mathsf{active}(\mathsf{last}(q)) = \mathsf{active}(s) = 1$. Since a is an input action, this contradicts the assumption that p' is input well-behaved. □

For the next three lemmas, let P_1 and P_2 be compatible scheduled automata and let α be a finite sequence of visible action symbols of $P_1 \| P_2$. Lemma 10 states that regular executions of a parallel composite always project to regular executions of the components. Lemma 11 states that regular executions of the two components in a composition can be "zipped" together in a unique way, provided they have matching traces. Finally, Lemma 12 states that, given a fixed trace, there is a bijective correspondence between the set of regular executions of the composite and the Cartesian product of the sets of regular executions of the two components. It follows directly from Lemma 10 and Lemma 11.

Lemma 10. *For every regular execution p of $P_1 \| P_2$, both $\pi_1(p)$ and $\pi_2(p)$ are regular executions.*

Proof. By Corollary 3, both projections are input well-behaved. We prove minimality by induction on the length of p. Note that empty executions are always minimal.

Consider a nonempty regular execution p and let a be the label of the last transition on p. Without loss of generality, we consider only $\pi_1(p)$. By minimality of p, a is a visible action. If a is in the signature of P_1, then $\pi_1(p)$ is minimal. Otherwise, let q be the unique prefix of p such that q is minimal and $\mathsf{tr}(p) = \mathsf{tr}(q)a$. Then p is of the form $qa_1\mu_1 s_1 \dots s_n a\mu s$, where each a_i is in $H_{P_1} \cup H_{P_2}$. Since q is a prefix of p, q must also be input well-behaved, thus it follows from the induction hypothesis that $\pi_1(q)$ is minimal. We claim that $\pi_1(p) = \pi_1(q)$.

There are two cases.

- a is an input of $P_1 \| P_2$. By Lemma 9, the one-step prefix of p is regular, thus minimal. Therefore it must coincide with q, so $\pi_1(p) = \pi_1(q)$.
- a is locally controlled by P_2. A (backwards) inductive argument using Lemma 6 and Axioms (1) and (3) shows that for all $1 \leq i \leq n$, $a_i \in H_{P_2}$. Again this implies $\pi_1(p) = \pi_1(q)$. □

Lemma 11. *Let p be a regular execution of P_1 such that $\mathsf{tr}(p) = \pi_1(\alpha)$. Similarly for q in P_2. There is a unique regular execution r of $P_1 \| P_2$ such that $\pi_1(r) = p$, $\pi_2(r) = q$, and $\mathsf{tr}(r) = \alpha$.*

Proof. We proceed by induction on the length of α. If α is empty, then, by minimality, p and q are both empty. Take r to be the empty execution of $P_1 \| P_2$.

Consider αa. Let p' be a regular execution of P_1 with visible trace $\pi_1(\alpha a)$ and let p denote the unique minimal prefix of p with visible trace $\pi_1(\alpha)$. Similarly for $q \sqsubseteq q'$ in P_2. By induction hypothesis, choose a unique regular execution r such that $\pi_1(r) = p$, $\pi_2(r) = q$, and $\mathsf{tr}(r) = \alpha$.

First assume that a is locally controlled by P_1. Suppose a is in the signature of P_2. In that case, $a \in I_{P_2}$ and q ends with an a-transition. By Lemma 9, we know that the one-step prefix of q' is minimal, thus coincides with q. Take r' to be the unique extension of r, in which P_1 follows p' and P_2 idles after r until the last step (i.e., the a-step).

If a is not in the signature of P_2, then $\pi_2(\alpha) = \pi_2(\alpha a)$. Therefore $q = q'$ and we take r' to be the unique extension of r in which P_1 follows p' and P_2 idles after q.

The case in which a is locally controlled by P_2 is symmetric. It remains to consider the case where a is an input of $P_1 \| P_2$. Again, if a is not in the signature of P_1, then $p = p'$; otherwise, we apply Lemma 9 to conclude that p is the one-step prefix of p'. Similarly for q and q'. Take r' to be the unique (one-step) extension of r in which 1. P_i takes an a-step after r, if a is in the signature of P_i, and 2. P_i idles after r otherwise. ☐

Lemma 12. *Let X denote $\mathsf{tr}^{-1}_{\mathsf{reg}}(\alpha)$ in $P_1 \| P_2$. Let Y and Z denote $\mathsf{tr}^{-1}_{\mathsf{reg}}(\pi_1(\alpha)$ in P_1 and $\mathsf{tr}^{-1}_{\mathsf{reg}}(\pi_2(\alpha))$ in P_2, respectively. There exists an isomorphism $\mathsf{zip} : Y \times Z \longrightarrow X$ whose inverse is $\langle \pi_1, \pi_2 \rangle$.*

4.2 Pseudo Probabilistic Executions and Pseudo Trace Distributions

Next we introduce a notion of *pseudo* probabilistic execution for automata with open inputs. The definition itself is completely analogous to probabilistic executions for closed automata; however, a pseudo probabilistic execution does not always induce a probability measure, because it does not take into account the probabilities with which inputs are provided by the environment.

Definition 7. *Let $S = \langle P, \sigma, \rho \rangle$ be a scheduled automaton and let p be a finite execution of S. Define the* pseudo probabilistic execution *Q of S as follows:*

- *$Q(s^0) = 1$, where s^0 is the initial state of S;*
- *if p' is of the form $pa\mu s'$ with $a \in I$, then $Q(p') := Q(p) \cdot \sigma(p, a)(\mu) \cdot \mu(s')$;*
- *if p' is of the form $pa\mu s'$ with $a \in L$, then $Q(p') := Q(p) \cdot \rho(p)(a, \mu) \cdot \mu(s')$.*

Similarly, we define *pseudo* trace distributions.

Definition 8. *Let $S = \langle P, \sigma, \rho\rangle$ be a scheduled automaton and let α be a finite sequence of visible action symbols of S. The* pseudo trace distribution D *of S is defined by $D(\alpha) := \sum_{p \in \mathsf{tr}_{\mathsf{reg}}^{-1}(\alpha)} Q(p)$, where Q is the pseudo probabilistic execution of S.*

Notice that, if S is closed, then the pseudo probabilistic execution of S coincides with the probabilistic execution of S. Moreover, an execution of a closed automaton S is regular if and only if it is minimal, thus the pseudo trace distribution of S coincides with the trace distribution of S.

For the rest of this section, let S and T be a pair of compatible scheduled automata. Let $Q_{S\|T}$, Q_S and Q_T denote the pseudo probabilistic executions of $S\|T$, S and T, respectively. Similarly for pseudo trace distributions $D_{S\|T}$, D_S and D_T. Lemma 13 below says we can project a pseudo probabilistic execution of the composite to yield pseudo probabilistic executions of the components. Lemma 14 then combines Lemma 12 and Lemma 13 to show the analogous projection result for pseudo trace distributions.

Lemma 13. *For all finite executions p of $S\|T$, we have $Q_{S\|T}(p) = Q_S(\pi_1(p)) \cdot Q_T(\pi_2(p))$.*

Proof. If p is empty, $Q_{S\|T}(p) = 1 = Q_S(\pi_1(p)) \cdot Q_T(\pi_2(p))$. Consider $p' = pa\mu s'$, where $\mu = \mu_1 \times \mu_2$ and $s' = (s'_1, s'_2)$. Let c_1 denote

- $\rho_S(\pi_1(p))(a, \mu_1)$ if a is locally controlled by S;
- $\sigma_S(\pi_1(p), a)(\mu_1)$ if a is an input of S;
- 1 otherwise.

Similarly for c_2 in T. Then we have

$$\begin{aligned} Q_{S\|T}(p') &= Q_{S\|T}(p) \cdot c_1 \cdot \mu_1(s'_1) \cdot c_2 \cdot \mu_2(s'_2) && \text{definition } S\|T \\ &= Q_S(\pi_1(p)) \cdot Q_T(\pi_2(p)) \cdot c_1 \cdot \mu_1(s'_1) \cdot c_2 \cdot \mu_2(s'_2) && \text{I.H.} \\ &= Q_S(\pi_1(p')) \cdot Q_T(\pi_2(p')). && \text{definition } Q_S,\ Q_T \end{aligned}$$

□

Lemma 14. *Let α be a finite sequence of visible action symbols of $S\|T$. Then $D_{S\|T}(\alpha) = D_S(\pi_1(\alpha)) \cdot D_T(\pi_2(\alpha))$.*

Proof. Let X denote $\mathsf{tr}_{\mathsf{reg}}^{-1}(\alpha)$ in $S\|T$. Let Y and Z denote $\mathsf{tr}_{\mathsf{reg}}^{-1}(\pi_1(\alpha))$ in S and $\mathsf{tr}_{\mathsf{reg}}^{-1}(\pi_2(\alpha))$ in T, respectively. We have

$$\begin{aligned} D_{S\|T}(\alpha) &= \sum_{r \in X} Q_{S\|T}(r) && \text{definition } D_{S\|T} \\ &= \sum_{r \in X} Q_S(\pi_1(r)) \cdot Q_T(\pi_2(r)) && \text{Lemma 13} \\ &= \sum_{p \in Y, q \in Z} Q_S(p) \cdot Q_T(q) && \text{Lemma 12} \\ &= (\sum_{p \in Y} Q_S(p)) \cdot (\sum_{q \in Z} Q_T(q)) && \text{factorization} \\ &= D_S(\pi_1(\alpha)) \cdot D_T(\pi_2(\alpha)). && \text{definition } D_S \text{ and } D_T \end{aligned}$$

□

To prove the main pasting lemma, we need yet another technical result; namely, inputs must be received with probability 1. This can be viewed as "input enabling" in the probabilistic sense and it follows from basic properties of target distributions and input schedulers.

Lemma 15. *Let α be a finite sequence of visible action symbols of $S\|T$ and let $a \in Act(S\|T)$ be given. If a is not locally controlled by T, then $D_T(\pi_2(\alpha)) = D_T(\pi_2(\alpha a))$.*

Proof. Let Z_α denote $\mathsf{tr}^{-1}_{\mathsf{reg}}(\pi_2(\alpha))$ in T. Similarly for $Z_{\alpha a}$. If a is not in the signature of T, then $Z_\alpha = Z_{\alpha a}$ and the claim is trivial. Otherwise, $a \in I_T$. Let q' in $Z_{\alpha a}$ be given. Let q be the one-step prefix of q' (i.e., $q' = qa\mu s'$ for some μ and s'). By Lemma 9, we know that q is regular, thus in Z_α. Therefore,

$$\begin{aligned}
\sum_{q' \in Z_{\alpha a}} Q_T(q') &= \sum_{q \in Z_\alpha} \sum_{\mu \in \mathsf{Supp}(\sigma_T(q,a))} \sum_{s' \in \mathsf{Supp}(\mu)} Q_T(q) \cdot \sigma_T(q,a)(\mu) \cdot \mu(s') \\
&= \sum_{q \in Z_\alpha} Q_T(q) \cdot \sum_{\mu \in \mathsf{Supp}(\sigma_T(q,a))} (\sigma_T(q,a)(\mu) \cdot \sum_{s' \in \mathsf{Supp}(\mu)} \mu(s')) \\
&= \sum_{q \in Z_\alpha} Q_T(q) \cdot \sum_{\mu \in \mathsf{Supp}(\sigma_T(q,a))} \sigma_T(q,a)(\mu) \\
&= \sum_{q \in Z_\alpha} Q_T(q).
\end{aligned}$$

The last two equalities are true because μ and $\sigma_T(p,a)$ are discrete probability distributions. □

4.3 Pasting Lemma

Two switched/scheduled automata are said to be *comparable* if they have the same visible signature and their start states have the same status. We are now ready for the main pasting lemma.

Lemma 16 (Pasting). *Let S_1, S_2, T_1 and T_2 be scheduled automata satisfying (i) S_1 and S_2 are comparable; (ii) $\{S_1, T_1\}$, $\{S_2, T_2\}$ and $\{S_1, T_2\}$ are compatible sets; (iii) the pseudo trace distributions $D_{S_1\|T_1}$ and $D_{S_2\|T_2}$ coincide (denoted D). Then D also coincides with the pseudo trace distribution $D_{S_1\|T_2}$.*

Proof. Let D' denote $D_{S_1\|T_2}$ and let $D_{S_1}, D_{S_2}, D_{T_1}$ and D_{T_2} denote the pseudo trace distributions of S_1, S_2, T_1 and T_2, respectively. Similarly for their pseudo probabilistic executions.

Notice that $S_1\|T_2$ and $S_2\|T_2$ have exactly the same visible signature, which can be partitioned into three sets: O_{S_1}, O_{T_2} and $I_{S1\|T_2}$. Let α be a finite sequence of actions from $O_{S_1} \cup O_{T_2} \cup I_{S_1\|T_2}$. We show by induction on the length of α that $D(\alpha) = D'(\alpha)$.

The base case is trivial, since $D(\epsilon) = 1 = D'(\epsilon)$. Consider αa. If $D(\alpha) = 0$, then by the induction hypothesis $D'(\alpha) = 0$. Therefore $D(\alpha a) = 0 = D'(\alpha a)$. Otherwise, $D(\alpha) = D'(\alpha) \neq 0$.

First consider the case in which $a \in O_{S_1}$. By Lemma 14,

$$D_{S_1}(\pi_1(\alpha)) \cdot D_{T_1}(\pi_2(\alpha)) = D(\alpha) = D'(\alpha) = D_{S_1}(\pi_1(\alpha)) \cdot D_{T_2}(\pi_2(\alpha)).$$

By assumption, $D(\alpha)$ is non-zero, hence $D_{T_1}(\pi_2(\alpha)) = D_{T_2}(\pi_2(\alpha))$. Since both T_1 and T_2 are compatible with S_1, a is not locally controlled by T_1 or T_2. It follows from Lemma 15 that $D_{T_1}(\pi_2(\alpha a)) = D_{T_2}(\pi_2(\alpha a))$. Again, applying Lemma 14, we have

$$D(\alpha a) = D_{S_1}(\pi_1(\alpha a)) \cdot D_{T_1}(\pi_2(\alpha a)) = D_{S_1}(\pi_1(\alpha a)) \cdot D_{T_2}(\pi_2(\alpha a)) = D'(\alpha a).$$

Now suppose $a \in O_{T_2}$. Then a is not locally controlled by S_1 (hence also not locally controlled by S_2). Again by the induction hypothesis and Lemma 14,

$$D_{S_2}(\pi_1(\alpha)) \cdot D_{T_2}(\pi_2(\alpha)) = D(\alpha) = D'(\alpha) = D_{S_1}(\pi_1(\alpha)) \cdot D_{T_2}(\pi_2(\alpha)).$$

By assumption, $D(\alpha)$ is non-zero, hence $D_{S_1}(\pi_1(\alpha)) = D_{S_2}(\pi_1(\alpha))$. Since a is not locally controlled by S_1 or S_2, we may apply Lemma 15 to obtain $D_{S_1}(\pi_1(\alpha a)) = D_{S_2}(\pi_1(\alpha a))$. Then, by Lemma 14, we have

$$D(\alpha a) = D_{S_2}(\pi_1(\alpha a)) \cdot D_{T_2}(\pi_2(\alpha a)) = D_{S_1}(\pi_1(\alpha a)) \cdot D_{T_2}(\pi_2(\alpha a)) = D'(\alpha a).$$

Finally, in case $a \in I_{S_1 \| T_2}$, a is also not locally controlled by S_1 or S_2. Thus the same argument applies. □

5 Renaming and Hiding

In this section, we consider the standard renaming and hiding operators. We start with an equivalence relation on switched automata: $P_1 \equiv_R P_2$ just in case there exists a bijection $f : H_1 \longrightarrow H_2$ such that P_2 can be obtained from P_1 by replacing every hidden action symbol $a \in H_1$ by $f(a) \in H_2$ (notation: $P_2 = f(P_1)$).

It is routine to check this is in fact an equivalence relation. If $P_1 \equiv_R P_2$, we say that P_2 can be obtained from P_1 by *renaming of hidden actions*. This operation also induces an equivalence relation on scheduled automata: $\langle P_1, \sigma_1, \rho_1 \rangle \equiv_R \langle P_2, \sigma_2, \rho_2 \rangle$ just in case there exists a renaming function f such that $P_1 \equiv_R P_2$ via f and $\langle \sigma_2, \rho_2 \rangle$ is obtained from $\langle \sigma_1, \rho_1 \rangle$ via f and f^{-1} (notation: $S_2 = f(S_1)$). Notice f^{-1} is used because $\mathsf{Exec}^{<\omega}(P)$ occurs negatively in the type of schedulers.

The following lemma says, as long as the renaming operation does not introduce incompatibility of signatures, it does not affect the behavior of an automaton in a closing context.

Lemma 17. *Let S and C be compatible scheduled automata with $S \| C$ closed. Suppose $S \equiv_R S'$ via the renaming function $f : H \longrightarrow H'$ with H' disjoint from $Act(C)$. Then $\{S', C\}$ is closed and compatible and $D_{S \| C} = D_{S' \| C}$.*

Next we consider the issue of hiding output actions. Let Hide denote the standard hiding operator for PIOA. This is also an operator for switched automata, provided we hide only basic outputs and synchronized control actions.

Lemma 18. *Let P be a switched automaton and let $\Omega \subseteq BO \cup Sync$ be given. Then $\mathsf{Hide}_\Omega(P)$ is again a switched automaton.*

Notice that every I/O scheduler for P is an I/O scheduler for $\mathsf{Hide}_\Omega(P)$. Therefore Hide can be extended to scheduled automata:

$$\mathsf{Hide}_\Omega\langle P, \sigma, \rho\rangle := \langle \mathsf{Hide}_\Omega(P), \sigma, \rho\rangle.$$

In the rest of this section we investigate the effect of Hide_Ω on (pseudo) trace distributions. Let $S = \langle P, \sigma, \rho\rangle$ be a scheduled automaton with signature $\langle I, O, H\rangle$. For convenience, write P' for $\mathsf{Hide}_\Omega(P)$, O' for $O \setminus \Omega$, and tr' for the trace operator for $\mathsf{Hide}_\Omega(P)$. (If we view Hide_Ω as an operator on traces, then tr' is precisely $\mathsf{Hide}_\Omega \circ \mathsf{tr}$.)

Moreover, for all $\beta' \in (I \cup O')^{<\omega}$, let $\mathcal{M}_{\beta'}$ denote the set of all minimal (w.r.t. $\sqsubseteq$) traces in $\mathsf{Hide}_\Omega{}^{-1}(\beta')$. That is, if β' is empty, then $\mathcal{M}_{\beta'}$ is the singleton set containing the empty trace ϵ; otherwise,

$$\mathcal{M}_{\beta'} := \{\beta \in (I \cup O)^{<\omega} \mid \mathsf{Hide}_\Omega(\beta) = \beta' \text{ and the last symbol on } \beta \text{ is not in } \Omega.\}$$

We make a simple observation about minimal executions of P and those of P'.

Lemma 19. *For all $\beta' \in (I \cup O')^{<\omega}$, the following two sets are equal:*

- $X := \bigcup_{\beta \in \mathcal{M}_{\beta'}} \{p \in \mathsf{Exec}^{<\omega}(P) \mid \mathsf{tr}(p) = \beta$ *and p minimal w.r.t.* $\mathsf{tr}\}$;
- $Y := \{p \in \mathsf{Exec}^{<\omega}(P') \mid \mathsf{tr}'(p) = \beta'$ *and p minimal w.r.t.* $\mathsf{tr}'\}$.

Proof. Note that $\mathsf{Exec}^{<\omega}(P) = \mathsf{Exec}^{<\omega}(P')$. Clearly, if β' is empty, then both X and Y coincide with the singleton set containing the empty execution. Thus we assume β' is nonempty.

Let $p \in Y$ be given and let $\beta := \mathsf{tr}(p)$. Note that $\beta' = \mathsf{tr}'(p) = \mathsf{Hide}_\Omega(\mathsf{tr}(p)) = \mathsf{Hide}_\Omega(\beta)$. Since p is minimal w.r.t. tr' the last action on p is not in $\Omega \cup H$, therefore the last action on β is not in Ω and p is minimal w.r.t. tr. Thus p is in X.

Conversely, let $p \in X$ be given. Again, $\mathsf{tr}'(p) = \mathsf{Hide}_\Omega(\mathsf{tr}(p)) = \mathsf{Hide}_\Omega(\beta) = \beta'$. Since p is minimal w.r.t. tr, the last action on p is not in H. By assumption on β, the last action on p is not in Ω. Thus p is minimal w.r.t. tr' and p must be in Y. □

Now consider the pseudo trace distribution D_S. Define the effect of Hide_Ω on D_S to be the following function from $O'^{<\omega}$ to $[0, 1]$:

$$\mathsf{Hide}_\Omega(D_S)(\beta') := \sum_{\beta \in \mathcal{M}_{\beta'}} D_S(\beta).$$

We have the following corollary of Lemma 19.

Corollary 5. *The pseudo trace distribution of* $\mathsf{Hide}_\Omega(S)$ *is precisely* $\mathsf{Hide}_\Omega(D_S)$. *That is,* $D_{\mathsf{Hide}_\Omega(S)} = \mathsf{Hide}_\Omega(D_S)$.

Proof. First note that the I/O scheduler for S is identical to that of S', thus $Q_S = Q_{S'}$. For each $\beta' \in (I \cup O')^{<\omega}$, write $X_{\beta'}$ for the set of regular executions p of P with $\mathsf{tr}(p) \in \mathcal{M}_{\beta'}$. Similarly, let $Y_{\beta'}$ denote the set of regular executions p of P' with $\mathsf{tr}'(p) = \beta'$. By Lemma 19, we have $X_{\beta'} = Y_{\beta'}$. Then for each $\beta' \in (I \cup O')^{<\omega}$,

$$\begin{aligned} D_{\mathsf{Hide}_\Omega(S)}(\beta') &= \sum_{p \in Y_{\beta'}} Q_{S'}(p) = \sum_{p \in X_{\beta'}} Q_S(p) = \sum_{\beta \in \mathcal{M}_{\beta'}} \sum_{p \in \mathsf{tr}_{\mathsf{reg}}^{-1}(\beta)} Q_S(p) \\ &= \sum_{\beta \in \mathcal{M}_{\beta'}} D_S(\beta) = \mathsf{Hide}_\Omega(D_S)(\beta'). \end{aligned}$$

□

Finally, we consider the effect of hiding in a parallel composition. We claim that the act of hiding in one component does not affect the behavior of the other, as long as the actions being hidden in the first component are not observable by the second component. This idea is captured in the following lemma, which follows from Corollary 5 and Lemma 14.

Lemma 20. *Let* S_1, S_2, T *be scheduled automata satisfying: (i)* S_1 *and* S_2 *are comparable and (ii)* T *is compatible with* S_1 *and* S_2. *Let* $\Omega \subseteq O_T$ *be given and let* T' *denote* $\mathsf{Hide}_\Omega(T)$. *If* T' *is compatible with* S_1 *(and thus with* S_2*), then*

$$D_{S_1 \| T} = D_{S_2 \| T} \Leftrightarrow D_{S_1 \| T'} = D_{S_2 \| T'}.$$

Proof. Since T' is compatible with S_1, it must be the case that Ω is disjoint from the signature of S_1 (and that of S_2); therefore, by the definition of $\|$, $S_1 \| \mathsf{Hide}_\Omega(T) = \mathsf{Hide}_\Omega(S_1 \| T)$ (and similarly for S_2). Thus the "only if" direction follows from Corollary 5.

For the converse, let D_{S_1}, D_{S_2}, D_T and $D_{T'}$ denote the pseudo trace distributions induced by S_1, S_2, T and T', respectively. Let β be a sequence of visible actions of $S_1 \| T$ and let β' be $\mathsf{Hide}_\Omega(\beta)$.

By Lemma 14, we have

- $D_{S_1 \| T}(\beta) = D_{S_1}(\pi_1(\beta)) \cdot D_T(\pi_2(\beta))$;
- $D_{S_2 \| T}(\beta) = D_{S_2}(\pi_1(\beta)) \cdot D_T(\pi_2(\beta))$;
- $D_{S_1 \| T'}(\beta') = D_{S_1}(\pi_1(\beta')) \cdot D_{T'}(\pi_2(\beta'))$;
- $D_{S_2 \| T'}(\beta') = D_{S_2}(\pi_1(\beta')) \cdot D_{T'}(\pi_2(\beta'))$.

If $D_{T'}(\pi_2(\beta')) = 0$, then we apply Corollary 5 to conclude $\mathsf{Hide}_\Omega(D_T)(\pi_2(\beta')) = 0$. Let β'' be the unique prefix of $\pi_2(\beta)$ such that $\beta'' \in \mathcal{M}_{\pi_2(\beta')}$. Then

$$0 = \mathsf{Hide}_\Omega(D_T)(\pi_2(\beta')) \geq D_T(\beta'') \geq D_T(\pi_2(\beta)).$$

Therefore $D_{S_1 \| T}(\beta) = D_{S_2 \| T}(\beta) = 0$.

Now suppose $D_{T'}(\pi_2(\beta')) \neq 0$. Since actions in Ω do not occur in the signatures of S_1 and S_2, we know that $\pi_1(\beta) = \pi_1(\beta')$. Using the assumption that $D_{S_1\|T'} = D_{S_2\|T'}$, we have

$$D_{S_1}(\pi_1(\beta)) = D_{S_1}(\pi_1(\beta')) = D_{S_2}(\pi_1(\beta')) = D_{S_2}(\pi_1(\beta)).$$

This implies $D_{S_1\|T}(\beta) = D_{S_2\|T}(\beta)$. □

6 Probabilistic Systems

In this section, we give a formal definition of our implementation preorder and prove compositionality. The basic approach is to model a system as a switched PIOA together with a set of I/O schedulers. Observable behavior is then defined in terms of trace distributions induced by the prescribed schedulers.

Formally, a *probabilistic system* $\mathcal{P}$ is a set of scheduled automata that share a common underlying switched automaton. (Equivalently, a probabilistic system is a pair $\langle P, \mathcal{S}\rangle$ where P is a switched automaton and $\mathcal{S}$ is a set of I/O schedulers for P.) Such a system is *full* if $\mathcal{S}$ is the set of all possible I/O schedulers for P.

Two probabilistic systems $\mathcal{P}_1 = \langle P_1, \mathcal{S}_1\rangle$ and $\mathcal{P}_2 = \langle P_2, \mathcal{S}_2\rangle$ are *compatible* just in case P_1 is compatible with P_2. The *parallel composite* of $\mathcal{P}_1$ and $\mathcal{P}_2$, denoted $\mathcal{P}_1\|\mathcal{P}_2$, is the probabilistic system: $\{S_1\|S_2 \mid S_1 \in \mathcal{P}_1 \text{ and } S_2 \in \mathcal{P}_2\}$. Notice the underlying automaton of the composite is $P_1\|P_2$.

Let S be a scheduled automaton. A *context* for S is a scheduled automaton C such that (i) C is compatible with S; (ii) S and C have complementary signatures (i.e., $I_C = O_S$ and $I_S = O_C$). Given probabilistic system $\mathcal{P} = \langle P, \mathcal{S}\rangle$, we say that D is a *trace distribution of* $\mathcal{P}$ just in case there exist scheduled automata $S \in \mathcal{P}$ and context C for S such that $D = D_{S\|C}$. We write $\mathsf{td}(\mathcal{P})$ for the set of trace distributions of $\mathcal{P}$.

Two probabilistic systems are *comparable* whenever the underlying switched automata are comparable. Given comparable systems $\mathcal{P}_1$ and $\mathcal{P}_2$, we define the *trace distribution preorder* by: $\mathcal{P}_1 \leq_{\mathsf{td}} \mathcal{P}_2$ whenever $\mathsf{td}(\mathcal{P}_1) \subseteq \mathsf{td}(\mathcal{P}_2)$. We are now ready to present our main theorem, namely, that the trace distribution preorder for probabilistic systems is compositional.

Theorem 1. *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be comparable probabilistic systems with $\mathcal{P}_1 \leq_{\mathsf{td}} \mathcal{P}_2$. Suppose $\mathcal{P}_3$ is compatible with both $\mathcal{P}_1$ and $\mathcal{P}_2$. Then $\mathcal{P}_1\|\mathcal{P}_3 \leq_{\mathsf{td}} \mathcal{P}_2\|\mathcal{P}_3$.*

Proof. Let D be a trace distribution of $\mathcal{P}_1\|\mathcal{P}_3$. Choose $S \in \mathcal{P}_1\|\mathcal{P}_3$ and context C for S such that $D = D_{S\|C}$. By definition of $\mathcal{P}_1\|\mathcal{P}_3$, S is of the form $S_1\|S_3$ for some $S_1 \in \mathcal{P}_1$ and $S_3 \in \mathcal{P}_3$. By Lemma 17, we may assume that the set of hidden actions of C is disjoint from that of $\mathcal{P}_2$.

By associativity of $\|$, we have $(S_1\|S_3)\|C \cong S_1\|(S_3\|C)$. Let Ω denote the set $O_{S_3\|C} \setminus I_{S_1}$. Then $\mathsf{Hide}_\Omega(S_3\|C)$ is a context for S_1 and, by Corollary 5,

$$\mathsf{Hide}_\Omega(D) = D_{\mathsf{Hide}_\Omega(S_1\|(S_3\|C))} = D_{S_1\|\,\mathsf{Hide}_\Omega(S_3\|C)} \in \mathsf{td}(\mathcal{P}_1).$$

Since $\mathcal{P}_1 \leq_{\mathsf{td}} \mathcal{P}_2$, we may choose $S_2 \in \mathcal{P}$ and context C' for S_2 such that $\mathsf{Hide}_\Omega(D) = D_{S_2\|C'}$.

We claim that S_2 is also compatible with $\mathsf{Hide}_\Omega(S_3\|C)$. Since S_1 and S_2 have the same visible signatures and S_1 is compatible with $\mathsf{Hide}_\Omega(S_3\|C)$, we may focus on hidden actions of S_2. By assumption, $\mathcal{P}_2$ is compatible with $\mathcal{P}_3$, thus H_{S_2} is disjoint from the alphabet of S_3. Moreover, $I_C \cup O_C = I_{S_1} \cup O_{S_1} = I_{S_2} \cup O_{S_2}$; therefore H_{S_2} is disjoint from $I_C \cup O_C$. Finally, we have chosen C so that H_C is disjoint from the alphabet of S_2. Now we have: (i) S_1 and S_2 are comparable; (ii) $\{S_1, \mathsf{Hide}_\Omega(S_3\|C)\}$, $\{S_2, \mathsf{Hide}_\Omega(S_3\|C)\}$ and $\{S_2, C'\}$ are compatible sets; (iii) $D_{S_1\|\,\mathsf{Hide}_\Omega(S_3\|C)} = \mathsf{Hide}_\Omega(D) = D_{S_2\|C'}$. Therefore we can apply Lemma 16 to conclude that

$$D_{S_1\|\,\mathsf{Hide}_\Omega(S_3\|C)} = D_{S_2\|\,\mathsf{Hide}_\Omega(S_3\|C)}.$$

By Lemma 20 and associativity of $\|$, this implies

$$D = D_{S_1\|(S_3\|C)} = D_{S_2\|(S_3\|C)} = D_{(S_2\|S_3)\|C} \in \mathsf{td}(\mathcal{P}_2\|\mathcal{P}_3).$$

$\square$

7 PIOA Revisited

Before concluding, we give an example in which switched automata are used to obtain a new trace-based semantics for general PIOAs. The idea is to convert a general PIOA to a switched PIOA by adding control actions and activity classification. We then hide all control actions in trace distributions generated by the resulting switched PIOA. In many cases, this yields a set of trace distributions strictly smaller than that considered by Segala [Seg95].

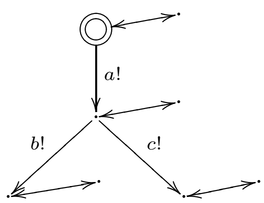
Let P be a PIOA and assume $Act(P) \subseteq BAct$. Let $\mathsf{go}, \mathsf{done} \in CAct$ be fresh symbols and let b_0 be a Boolean value. The *switch extension* of P with go, done and initialization b_0 (notation: $\mathcal{E}(P, \mathsf{go}, \mathsf{done}, b_0)$), is the switched automaton P' constructed as follows:

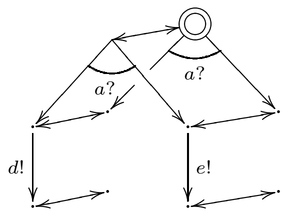
- $\mathsf{States}(P') = \mathsf{States}(P) \times \{0, 1\}$ and the start state of P' is $\langle s^0,\, b_0\rangle$;
- $I' = I \cup \{\mathsf{go}\}$, $O' = O \cup \{\mathsf{done}\}$, and $Sync' = \emptyset$;
- $\mathsf{active}'(s, b) = b$ for $b \in \{0, 1\}$;
- the transition relation is the union of the following:
  - $\{\langle\langle s,\, 1\rangle,\, a,\, \mu^1\rangle \mid s \xrightarrow{a} \mu \text{ in } P\}$,
  - $\{\langle\langle s,\, 0\rangle,\, a,\, \mu^0\rangle \mid s \xrightarrow{a} \mu \text{ in } P \text{ and } a \in I\}$,
  - $\{\langle\langle s,\, b\rangle,\, \mathsf{go},\, (\langle s,\, 1\rangle \mapsto 1)\rangle \mid s \in \mathsf{States}(P) \text{ and } b \in \{0, 1\}\}$,
  - $\{\langle\langle s,\, 1\rangle,\, \mathsf{done},\, (\langle s,\, 0\rangle \mapsto 1)\rangle \mid s \in \mathsf{States}(P)\}$,

  where μ^b denotes the distribution that assigns probability $\mu(t)$ to $\langle t,\, b\rangle$ and 0 to $\langle t,\, 1 - b\rangle$.

Roughly speaking, P' is obtained from P by (i) adding a Boolean flag active' to each state; (ii) enabling locally controlled transitions only if $\mathsf{active}' = 1$; and (iii) adding go and done transitions which update active' appropriately. It is not hard to check that P' satisfies all axioms of switched automata. Moreover, the pair $\langle\mathsf{go},\, \mathsf{done}\rangle$ can be easily generalized to a pair of disjoint sets of control actions.

Given any two compatible PIOAs, we can always extend them with complementary control actions and initialization statuses, resulting in a pair of compatible switched automata. As an example, we consider the automata Late and Toss in Figure 1. Actions a, b and c are considered outputs of Late, whereas action a is an input of Toss and actions e and f are outputs of Toss. The following diagrams illustrate $\mathcal{E}(\mathsf{Late}, \mathsf{go}, \mathsf{done}, 1)$ and $\mathcal{E}(\mathsf{Toss}, \mathsf{done}, \mathsf{go}, 0)$. For a clearer picture, we have omitted the probabilities on the input a-transition in Toss, as well as all nonessential input loops. The active region, which is identical to the original PIOA, is drawn in the foreground. The inactive region, in which all locally controlled transitions are removed, is in the background. Each two-headed arrow indicates a control output from active to inactive and a control input from inactive to active.





Now consider the problematic trace distribution D_0 of $\mathsf{Late} \| \mathsf{Toss}$, as described in Section 1. Let $\mathcal{P}_1$ and $\mathcal{P}_2$ denote the full probabilistic systems on $\mathcal{E}(\mathsf{Late}, \mathsf{go}, \mathsf{done}, 1)$ and $\mathcal{E}(\mathsf{Toss}, \mathsf{done}, \mathsf{go}, 0)$, respectively. As we compose these two systems, D_0 is no longer a trace distribution of $\mathcal{P}_1 \| \mathcal{P}_2$ (even after hiding go and done), because I/O schedulers in $\mathcal{P}_1$ have no way of knowing whether action d or action e was performed by $\mathcal{P}_2$, thus they cannot establish the correlations between actions d and b, and between actions e and c.

Interestingly, if we modify $\mathcal{P}_1$ by adding d, e to its input signature and adding d, e-loops to every state, the trace distribution D_0 is again possible. This shows that our trace distribution semantics for switched automata is very sensitive to the observational power of each automaton, that is, the ability of an automaton to observe activities taking place in its environment.

This leads to our proposal of a new notion of visible behaviors for PIOA. Let P be a PIOA and let $\mathcal{P}$ be the full probabilistic system on $\mathcal{E}(P, \mathsf{go}, \mathsf{done}, 0)$. A PIOA E is a *context* for P if $I_E = O_P$, $O_E = I_P$, and E is compatible with P. For each such E, write $\mathcal{P}_E$ for the full probabilistic system on $\mathcal{E}(E, \mathsf{done}, \mathsf{go}, 1)$. We say that D is a *trace distribution* of P if there exists a context E for P such that $D \in \mathsf{td}(\mathsf{Hide}_{\{\mathsf{go},\mathsf{done}\}}(\mathcal{P} \| \mathcal{P}_E))$, where Hide is lifted from scheduled automata to probabilistic systems.

We claim that this new semantics is at least as expressive as the trace semantics for I/O Automata. More precisely, we view an ordinary I/O automaton P as a PIOA in which every transition leads to a Dirac distribution and we claim that every trace α of P can be obtained as a trivial trace distribution. To do so, we first obtain a trace α' by inserting the symbol done whenever an input action follows a locally controlled action and vice versa with go (also prepending go if α starts with a locally controlled action). Let E be a context for P such that every state of E enables every output action of E. Then it is straightforward to find deterministic schedulers for $\mathcal{E}(P, \mathsf{go}, \mathsf{done}, 0)$ and $\mathcal{E}(E, \mathsf{done}, \mathsf{go}, 1)$ so that the composite generates precisely the trace α'. We omit the details.

8 Conclusions and Further Work

Our ultimate goal, of course, is to obtain a compositional semantics for PIOAs. The notion of switch extensions opens up an array of new options for that end. A promising approach is to model each system as a finite set of PIOAs, rather than a single PIOA. In that case, composition is simply set union, representing the act of placing two sets of processes in the same computing environment. Behavior is then defined in terms of switch extensions, which instantiate the system with a particular network topology for control passage. In that case, a behavior of a finite set $\mathcal{F}$ is determined by (i) a context E for $\mathcal{F}$; (ii) a combination of switch extensions of $\mathcal{F} \cup \{E\}$; (iii) a combination of I/O schedulers for these switch extensions. By ranging over all contexts and all extension-scheduler combinations, we capture all possible behaviors of the system represented by $\mathcal{F}$.

Another option is an *arbitrated* composition: we add an arbiter automaton which observes overall activities in the computing environment and resolves choices among components. Control is always passed between a component-arbiter pair (i.e., never directly between two components). In other words, each component is responsible for its local choices and the arbiter chooses (probabilistically) the next component to perform a locally controlled transition. Then the behavior of a system depends also on the choice of arbiters. It remains to be seen if such an arbitrated composition is more or less expressive compared to the arbiter-less version.

In other future work, we plan to apply our theory of composition for PIOAs to the task of verifying security protocols. For example, we will try to model typical Oblivious Transfer protocols within the PIOA framework and verify correctness in the style of Canetti's Universal Composability [Can01]. We will also explore the use of PIOAs as a semantic model for the probabilistic polynomial time process calculus of Lincoln, Mitchell, Mitchell and Scedrov [LMMS98].

References

[Agg94] S. Aggarwal. Time optimal self-stabilizing spanning tree algorithms. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, May 1994. Available as Technical Report MIT/LCS/TR-632.

[Can01] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computing*, pages 136–145, 2001.

[CLSV04] L. Cheung, N.A. Lynch, R. Segala, and F.W. Vaandrager. Switched probabilistic I/O automata. In *Proceedings First International Colloquium on Theoretical Aspects of Computing (ICTAC2004),* Guiyang, China, 20-24

September 2004, Lecture Notes in Computer Science. Springer-Verlag, 2004. To appear.

[dAH01] L. de Alfaro and T.A. Henzinger. Interface automata. In V. Gruhn, editor, *Proceedings of the Joint 8th European Software Engineering Conference and 9th ACM SIGSOFT Symposium on the Foundation of Software Engineering (ESEC/FSE-01)*, volume 26 of *Software Engineering Notes*, pages 109–120, New York, September 2001. ACM Press.

[dAHJ01] L. de Alfaro, T.A. Henzinger, and R. Jhala. Compositional methods for probabilistic systems. In K.G. Larsen and M. Nielsen, editors, *Proceedings CONCUR 01,* Aalborg, Denmark, August 20-25, 2001, volume 2154 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2001.

[LMMS98] P. Lincoln, J.C. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic polytime framework for protocol analysis. In *ACM Conference on Computer and Communications Security*, pages 112–121, 1998.

[LSS94] N.A. Lynch, I. Saias, and R. Segala. Proving time bounds for randomized distributed algorithms. In *Proceedings of the 13th Annual ACM Symposium on the Principles of Distributed Computing*, pages 314–323, Los Angeles, CA, August 1994.

[LSV03] N.A. Lynch, R. Segala, and F.W. Vaandrager. Compositionality for probabilistic automata. In R. Amadio and D. Lugiez, editors, *Proceedings 14th International Conference on Concurrency Theory (CONCUR 2003),* Marseille, France, volume 2761 of *Lecture Notes in Computer Science*, pages 208–221. Springer-Verlag, September 2003.

[LT89] N.A. Lynch and M.R. Tuttle. An introduction to input/output automata. *CWI Quarterly*, 2(3):219–246, September 1989.

[PSL00] A. Pogosyants, R. Segala, and N.A. Lynch. Verification of the randomized consensus algorithm of Aspnes and Herlihy: a case study. *Distributed Computing*, 13(3):155–186, 2000.

[Seg95] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, June 1995. Available as Technical Report MIT/LCS/TR-676.

[SL95] R. Segala and N.A. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.

[SM03] A. Sabelfeld and A.C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, January 2003.

[Sto02] M.I.A. Stoelinga. An introduction to probabilistic automata. *Bulletin of the European Association for Theoretical Computer Science*, 78:176–198, October 2002.

[SV99] M.I.A. Stoelinga and F.W. Vaandrager. Root contention in IEEE 1394. In J.-P. Katoen, editor, *Proceedings 5th International AMAST Workshop on Formal Methods for Real-Time and Probabilistic Systems,* Bamberg, Germany, volume 1601 of *Lecture Notes in Computer Science*, pages 53–74. Springer-Verlag, 1999.

[SV04] A. Sokolova and E.P. de Vink. Probabilistic automata: system types, parallel composition and comparison. In C. Baier et al., editor, *Validation of Stochastic Systems*, volume 2925 of *Lecture Notes in Computer Science*, pages 1–43. Springer-Verlag, 2004.

[WSS94] S.-H. Wu, S. A. Smolka, and E. W. Stark. Composition and behaviors of probabilistic i/o automata. In B. Jonsson and J. Parrow, editors, *Proceedings CONCUR 94,* Uppsala, Sweden, volume 836 of *Lecture Notes in Computer Science*, pages 513–528. Springer-Verlag, 1994.