Trace-based Semantics for Probabilistic Timed I/O Automata*

Sayan Mitra and Nancy Lynch

Computer Science and AI Laboratory,
Massachusetts Inst. of Technology,
32 Vassar Street, Cambridge, MA 02139
{lynch, mitras}@csail.mit.edu

Abstract. We propose the Probabilistic Timed I/O Automaton (PTIOA) framework for modelling and analyzing discretely communicating probabilistic hybrid systems. State transition of a PTIOA can be nondeterministic or probabilistic. Probabilistic choices can be based on continuous distributions. Continuous evolution of a PTIOA is purely nondeterministic. PTIOAs can communicate through shared actions. By supporting external nondeterminism, the framework allows us to model arbitrary interleaving of concurrently executing automata. The framework generalizes several previously studied automata models of its class. We develop the trace-based semantics for PTIOAs which involves measure theoretic constructions on the space of executions of the automata. We introduce a new notion of external behavior for PTIOAs and show that PTIOAs have simple compositionality properties with respect this external behavior.

1 Introduction

Probabilistic automata with continuous state spaces provide a mathematical framework for modeling and verifying computing systems that interact with uncertain environments. Particularly in cases where uncertainties, such as processor failures and message delays, find accurate description in terms of stochastic events. In systems that are also distributed, pure nondeterminism is necessary for allowing construction of implementation free abstract models through underspecification, and arbitrary interleaving of concurrently executing processes. For a detailed discussion on the need for nondeterminism in probabilistic automata we refer the reader to Chapter 4 of [17] and the introduction of [11]. Therefore, in order to verify systems, such as Sensor Networks and Mobile-robots, that have traits of both hybrid and distributed systems we need a framework supporting continuous dynamics, probabilistic transitions and nondeterminism. The interplay between probability and nondeterminism makes the development of semantics such frameworks challenging [26,23,8,4]. Introduction of continuous state spaces and distributions adds another layer of complexity to the problem [6,28,9].

* Supported by the MURI project:DARPA/AFOSR MURI F49620-02-1-0325 grant, NSF Awards CNS-0614414 and CCR-0121277 USAF,AFRL Award FA9550-04-1-0121.

Several continuous state probabilistic automaton models have been proposed. In Labelled Markov Processes (see e.g., [9,28]) state transitions can give rise to continuous probability distributions. Stochastic Hybrid Systems [18,3] have transitions triggered by discrete or continuous time Markov chains, and trajectories described by stochastic differential equations. In Piecewise Deterministic Markov Processes [10] discrete transitions are probabilistic and the continuous evolution of state in between those transitions is deterministic. These models do not permit *internal nondeterminism*. That is, choice of an action uniquely determines a transition, which in turn gives a probability distribution over the states. Modeling frameworks that support composition of automata have to resolve *external nondeterminism*, that is, the choice of which automaton gets to make the next move. This nondeterminism can be replaced by a race between the automata [14,27], else it can be explicitly resolved by a *scheduler* [6,8,4]. Nondeterminism can also be allowed by treating the probabilistic and nondeterministic transitions as separate kinds of objects [17]. Our *Probabilistic Timed Input/Output Automata (PTIOA)* framework shares certain features with the Stochastic Transition Systems (STS) of [6]. Both frameworks allow continuous state spaces and nondeterminism. However, a STS does not have notions of time or trajectories. This leads to very different semantics for the two frameworks and also important technical differences in the underlying construction of probability spaces. We discuss these issues in Remark 1.

The PTIOA framework proposed in this paper supports continuous evolution, nondeterminism, probabilistic transitions, and discrete communication between components. A PTIOA can capture continuous evolution of state through *trajectories*. Discrete state transition of a PTIOA can be nondeterministic or probabilistic. Probabilistic choices can be based on continuous distributions. Thus, the PTIOA framework generalizes several existing automata models, including Timed I/O Automaton [21], Probabilistic I/O Automaton [26,4] and its timed extension presented in [26], and discrete state Markov Decision Processes [11,1,22,19]. We define the parallel composition and hiding operations for PTIOAs, and show that the class is closed under these operations.

For constructing a probability measure over the executions of a set of communicating PTIOAs, we have to (a) resolve internal and external nondeterminism, and (b) ensure that all sets of reasonable executions are measurable. We resolve internal nondeterminism with *local schedulers*. For external nondeterminism, we equip PTIOAs with a partition over actions called the *task structure* and then use an *oblivious task scheduler* [5,4]. To ensure condition (b) we impose the following measurability conditions on a PTIOA that has a measurable space $(X, \mathcal{F}_X)$ as its state space: (1) for any action, the set of states in which the action is enabled is a measurable set, and (2) for measurable subsets $R \subseteq \mathbb{R}_{\geq 0}, Y \subseteq X$, the set of states from which there exits a *maximal trajectory* with length in R and final state in Y, is a measurable set. With these structures in place, each scheduler defines a probability measure over the space of executions of a PTIOA. We call such a measure a *probabilistic execution*. A probabilistic execution defines a unique probability measure on the space of traces, that is a unique *trace distribution*, provided the *trace* function is measurable. We show that this is indeed the case. We use a simple, but intuitive notion of *external behavior* for PTIOAs: for a given automaton $\mathcal{A}$, its external behavior is a function that maps

each closing environment $\mathcal{E}$ of $\mathcal{A}$ to the set of all possible trace distributions of the composition of $\mathcal{A}$ and $\mathcal{E}$. We show that the *implementation relation* defined in terms of the above notion of external behavior is compositional. Indeed, considering closed automata and using this functional definition of external behavior lets us circumvent some of the difficulties that underlie compositionality in the probabilistic setting. However, viewing external behavior as a mapping from environments as opposed to a set of trace distributions is natural in many applications, including analysis of security protocols [5]. Our longer term goal is to develop a suite of analysis techniques for PTIOAs for proving probabilistic safety, stability [7] and approximate implementation relations [25].

The paper is self-contained and we give the necessary mathematical background in Section 2. For the purpose of clear exposition, in Section 3 we first introduce a restricted type of PTIOA that limits nondeterministic choices to the choices over tasks. We define parallel composition and hiding operations for this restricted class of PTIOAs, and present the construction of measures over executions and traces in Section 4. We illustrate important definitions with an example—a randomized consensus protocol adapted from [2] with stochastic message delays. Finally, in Section 5 we show how the results for this restricted class of PTIOAs carry over to general PTIOAs by adding local schedulers. Further examples and proofs of the results appear in the full version of the paper which is available as [24].

2 Preliminaries

The complement of a set A is denoted by A^c. The union of a collection $\{A_i\}_{i\in I}$ of pairwise disjoint sets indexed by a set I is written as $\biguplus_{i\in I} A_i$. The domain of a function f by $f.dom$. For a set $S \subseteq f.dom$, we write $f \lceil S$ for the restriction of f to S. If f is a function whose range is a set of functions, each having domain Y and $S \subseteq Y$, then we write $f \downarrow S$ for the function g with $g.dom = f.dom$ such that $g(c) := f(c) \lceil S$, for every $c \in g.dom$.

Time and Trajectories. We define $\mathbb{T} = \mathbb{R}_{\geq 0} \cup \{\infty\}$ to be the *time axis*. A *trajectory in* X is a Let X be the set of states. function $\tau : J \to X$, where J is a left closed interval in $\mathbb{T}$ with left endpoint 0. A trajectory τ with $\tau.dom = \{0\}$, and $\tau(0) = x$, is called the *point trajectory* at x and is written as $\wp(x)$. A trajectory τ is *finite* if $\tau.dom$ has finite length. It is *closed* if it is finite and $\tau.dom$ is right closed. The *first state* of τ, $\tau.fstate$ is $\tau(0)$ and the *limit time* of τ, $\tau.ltime$, is $\sup\{\tau.dom\}$. If τ is closed then the *limit state* of τ, $\tau.lstate$, is $\tau(\tau.ltime)$. Given a trajectory τ and $t \in \mathbb{T}$, the time shifted function $(\tau + t) : (\tau.dom + t) \to X$ is defined as $(\tau + t)(t') := \tau(t' - t)$, for each $t' \in \{u + t \mid u \in \tau.dom\}$. Given two trajectories τ_1 and τ_2, τ_1 is a *prefix* of τ_2, written as $\tau_1 \leq \tau_2$, if $\tau_1 = \tau_2 \lceil \tau_1.dom$. Also, τ_1 is a *suffix* of τ_2 if $\tau_1 = (\tau_2 \lceil [t, \infty]) - t$, for some $t \in \tau_2.dom$. If τ_1 is a closed trajectory with $\tau_1.ltime = t$ and $\tau_2.fstate = \tau_1.lstate$, then the function $\tau_1 \frown \tau_2 : \tau_1.dom \cup (\tau_2.dom + t) \to X$ is defined as $\tau_1(t)$ if $t \leq u$ and $\tau_2(t - u)$ otherwise. Given a set of trajectories $\mathcal{T}$, we denote the subset of trajectories starting from x by $\mathcal{T}(x)$. $\mathcal{T}$ is *deterministic* if for all $x \in X$, $\tau_1, \tau_2 \in \mathcal{T}(x)$, either $\tau_1 \leq \tau_2$ or $\tau_2 \leq \tau_1$.

Let $\mathcal{T}$ be a set of deterministic trajectories for X; $(T(x), \leq)$ is a total order. A trajectory τ in $\mathcal{T}$ is said to be *maximal* if it is the supremal element

of $\mathcal{T}(\tau.fstate)$. If a maximal $\tau \in \mathcal{T}(x)$ exists then it is unique. We define $maxtime_{\mathcal{T}} : X \to \mathbb{T}$ as $maxtime_{\mathcal{T}}(x) := \tau.ltime$ if there exists a closed maximal $\tau \in \mathcal{T}(x)$, otherwise $maxtime_{\mathcal{T}}(x) : \infty$. Similarly, we define $maxstate_{\mathcal{T}} : X \to X$ as $maxstate_{\mathcal{T}}(x) := \tau.lstate$ if there exists a closed maximal $\tau \in \mathcal{T}(x)$, otherwise $maxstate_{\mathcal{T}}(x) : x$.

Measurability and measures. We follow the standard notations as found in any text book on measure theory, as for instance [13]. A measurable space is denoted by $(X, \mathcal{F}_X)$, where X is a set and $\mathcal{F}_X$ is a σ-algebra over X. Whenever the sets $\mathbb{R}, \mathbb{R}_{\geq 0}$ and $\mathbb{T}$ are viewed as measurable spaces, it is assumed that they are equipped with their usual Borel σ-algebras. The set of probability measures over $(X, \mathcal{F}_X)$ is denoted by $\mathcal{P}(X, \mathcal{F}_X)$. A function $f : (X, \mathcal{F}_X) \to (Y, \mathcal{F}_Y)$ is said to be *measurable* if $f^{-1}(E) \in \mathcal{F}_X$ for every $E \in \mathcal{F}_Y$. If $f : (X, \mathcal{F}_X) \to (Y, \mathcal{F}_Y)$ is a measurable function, and μ is a measure on X, then the *image measure of* μ *under* f is a measure φ on Y defined as $\varphi(E) = \mu(f^{-1}(E))$, for each $E \in \mathcal{F}_Y$.

A collection $\mathscr{C}$ of subsets of X, is a *semi-ring* if $X, \emptyset \in \mathscr{C}$, and for any $A, B \in \mathscr{C}$ $A \cap B \in \mathscr{C}$, and that there exists a finite collection of disjoint sets $\{C\}_{i=1}^{n}$ in $\mathscr{C}$ such that $A \setminus B = \biguplus_{i=1}^{n} C_i$. It is well known (see, e.g. [13]) that a measure μ defined over a semi-ring $\mathscr{C}$ can be uniquely extended to a measure over the σ-algebra generated by $\mathscr{C}$ by defining $\mu(\cup_{i=1}^{n} C_i) = \sum_{i=1}^{n} \mu(C_i)$.

In constructing measures over the space of executions of a PTIOA, we have to integrate over the space of probability distributions over the state space X, therefore we need to define a σ-algebra over $\mathcal{P}(X, \mathcal{F}_X)$. For this, we use the following construction due to Giry [15]: for each $A \in \mathcal{F}_X$, let the function $p_A : \mathcal{P}(X, \mathcal{F}_X) \to [0, 1]$ be defined as $p_A(\mu) = \mu(A)$. The σ-algebra on $\mathcal{P}(X, \mathcal{F}_X)$, then is the smallest σ-algebra such that all p_A's are measurable.

3 Probabilistic Timed I/O Automata

Definition 1. *A* pre-PTIOA *is a 6-tuple* $\mathcal{A} = ((X, \mathcal{F}_X), \bar{x}, A, \mathcal{R}, \mathcal{D}, \mathcal{T})$ *where: (1)* $(X, \mathcal{F}_X)$ *is a measurable space called the* state space*. (2)* $\bar{x} \in X$ *is the* start state*. (3)* A *is a countable set of* actions*, partitioned into* internal H*,* input I *and* output O *actions.* $L = O \cup H$ *is the set of* local actions *and* $E = O \cup I$ *is the set of* external actions*. (4)* $\mathcal{R}$ *is an equivalence relation on* L*; the equivalence classes of* $\mathcal{R}$ *are called* tasks*. A task* T *is called an* output task *if* $T \subseteq O$*. (5)* $\mathcal{D} \subseteq X \times A \times \mathcal{P}(X, \mathcal{F}_X)$ *is the set of* transitions*. (6)* $\mathcal{T}$ *is a set of deterministic trajectories for* X *that is closed under prefix, suffix, concatenation and contains* $\wp(x)$ *for every* $x \in X$*.*

The determinism assumption on $\mathcal{T}$ is relaxed in Section 5 to include nondeterministic trajectories, however, trajectories where stochastic choices are made continuously over a an interval of time are currently excluded from the PTIOA framework. This will be investigated in the future.

A pre-PTIOA is *closed* if its set of input actions is empty. If (x, a, μ) is an element of $\mathcal{D}$, we write $x \xrightarrow{a} \mu$ and action a is said to be *enabled* at x. The set of states in which at least one action from the set $B \subseteq A$ is enabled is denoted by E_B and the set of actions enabled at x is denoted by $enabled(x)$. If a single action a is enabled at x and $x \xrightarrow{a} \mu$, then this μ is denoted by μ_x. For $R \subseteq \mathbb{R}_{\geq 0}$ and $Y \subseteq X$, we define $E_{R,Y} = \{x \in X \mid maxtime_{\mathcal{T}}(x) \in R \ \wedge \ maxstate_{\mathcal{T}}(x) \in Y\}$.

Definition 2. *A* PTIOA $\mathcal{A}$ *is a pre-PTIOA that satisfies the following axioms:*

- **M0** (Measurability) *For all* $B \subseteq A$, E_B *is measurable. For measurable sets* $R \subseteq \mathbb{R}_{\geq 0}$, $Y \in \mathcal{F}_X$, $E_{R,Y}$ *is measurable.*
- **D0** (Input action enabling) *Input actions are enabled in all states.*
- **D1** (Time-action determinism) *For any state* x *at most one of the following may exist: (1) a local action* a *such that* $x \in E_a$ *(2) a non-point trajectory* $\tau \in \mathcal{T}(x)$.
- **D2** (Task determinism) *For any state* x, *if there are actions* a, b *in the same task* T *and* $x \xrightarrow{a} \mu_1$ *and* $x \xrightarrow{b} \mu_2$ *then* $a = b$ *and* $\mu_1 = \mu_2$.

At this point some explanation of the of the axioms may be in order. The **M0** axiom was described in the introduction; it is necessary to ensure measurability of reasonable sets of executions. **D0** is a non-blocking axiom standard in I/O automata literature. Axiom **D1** allows resolution of nondeterminism in a structured manner; as we show in Section 5, by adding local schedulers, this axiom can be removed. According to **D1**, from any state x, either a local action is enabled or some non-zero amount of time can elapse. It prevents an action to remain enabled while time elapses and is similar to the maximal progress assumption found in real-time process algebras, e.g. [16]. If the time can elapse from x, then the state evolves according to the[1] maximal trajectory τ in $\mathcal{T}(x)$. If local actions are enabled at x then time cannot elapse and the automaton nondeterministically chooses one action a from the set of enabled actions. This nondeterministic choice is resolved by a *task scheduler*, which we shall define shortly. If a task T is specified then **D2** implies that at x there can be at most one enabled action in T, and at most one probabilistic transition corresponding to that action.

Theorem 1. *Suppose* $\mathcal{A} = ((X, \mathcal{F}_X), \bar{x}, A, \mathcal{R}, \mathcal{D}, \mathcal{T})$ *is a pre-PTIOA satisfying* **D0** *and* **D1** *such that for every transition* $x \xrightarrow{a} \mu$, μ *is a Dirac measure. Then* $(X, (E_L \cup E_{\mathbb{R}_{\geq 0}, X}), \{\bar{x}\}, I, O, H, \mathcal{D}, \mathcal{T})$ *is a timed I/O automaton.*

Example. *(Randomized Consensus)* The Ben-Or consensus protocol [2] is a randomized algorithm for n fault-prone processors to agree on a valid value by communicating over an asynchronous network. The algorithm proceeds in a sequence of stages in each of which nonfaulty processes send and receive messages based on coin-flips and comparison of values. With probability $\frac{1}{2^n}$, a stage ends successfully and all nonfaulty processes agree on a value, and after one communication round of a successful stage the consensus value is disseminated. An unsuccessful stage is followed by the beginning of the next stage.

The Consensus PTIOA of Figure 1 specifies the termination behavior of the Ben-Or protocol in a language that is a simple extension of the TIOA Language [20]. The *stage* variable represents the current stage of the protocol. The *phase* variable is 0 at the beginning of a stage, 1 when a stage completes successfully, and 2 when the protocol terminates at all nonfaulty processes. The try

[1] There exists a unique maximal trajectory because the set of trajectories is deterministic.

action models the computation and communication within a stage. With probability $1 - \frac{1}{2^n}$ it leads to the next stage and with probability $\frac{1}{2^n}$ it leads to phase 1 of the current stage. The decide action marks the termination of the protocol. The **Trajectories** section specifies that along any trajectory, *timer* increases at the same rate as real time, and that all other variables remain constant. In general, more complicated differential and algebraic equations can be used to specify the trajectories of a PTIOA. The amount of time that elapses in phase 0 owing to message delays is modeled by an exponential distribution with parameter λ_0. Specifically, the *delay* variable is assigned a value chosen from this distribution and **stop when** condition together with the **pre**condition of try forces the action to occur when *timer* equals *delay*. Likewise, the amount of time that elapses in phase 1 is modeled by an exponential distribution with parameter λ_1. The choice of exponential distributions here is somewhat arbitrary; other, more appropriate distributions can be used as well. The final section of the code specifies the two tasks of the automaton.

Consensus($n \in \mathbb{N}, \lambda_0, \lambda_1, p : \mathbb{R}$)
  where $\lambda_0, \lambda_1 > 0$, $0 < p < 1$
Variables:
  $stage : \mathbb{N}$ **initially** 1
  $phase : \{0, 1, 2\}$ **initially** 0
  $timer : \mathbb{R}_{\geq 0}$ **initially** 0
  $delay : \mathbb{R}_{\geq 0} \cup \{\infty\}$ **initially** t_0

Actions:
  output try, decide

Transitions:
  output decide
  pre $timer = delay \wedge phase = 1$
  eff $timer := 0$
    $phase$:= 2
    $delay$:= ∞
  output try
  pre $timer = delay \wedge phase = 0$
  eff $timer := 0$
    $phase$:= **choose** {1,0} **with prob** $\{\frac{1}{2^n}, 1 - \frac{1}{2^n}\}$
    if $phase = 0$
    then $stage$:= $stage + 1$;
      $delay$:= **choose** Exp(λ_0)
    else $delay$:= **choose** Exp(λ_1) **fi**

Trajectories:
  Trajdef *normal*
  stop when $timer = delay$
  evolve $d(timer) = 1$

Tasks: {try}{decide}

Fig. 1. Randomized consensus with exponential message delays.

Executions and traces. An *execution fragment* of an PTIOA $\mathcal{A}$ is an alternating sequence of actions and trajectories $\alpha = \tau_0 a_1 \tau_1 a_2 \ldots$, where each $\tau_i \in \mathcal{T}$, $a_i \in A$ and a_i is enabled at $\tau_{i-1}.lstate$. The *first state* of an execution fragment α, $\alpha.fstate$, is $\tau_0.fstate$. An execution fragment α is an *execution* of $\mathcal{A}$ if $\alpha.fstate = \bar{x}$. The *length* of a finite execution fragment α is the number of actions in α. An execution fragment is *closed* if it is a finite sequence and the last trajectory is closed. Given a closed execution fragment $\alpha = \tau_0 a_1 \ldots \tau_n$, its *limit state*, $\alpha.lstate$, is $\tau_n.lstate$ and its *limit time* is defined to be $\sum_i^n \tau_i.ltime$. Proposition 1 follows from axiom **D1**.

Proposition 1. *In any execution fragment of a closed PTIOA all trajectories, except possibly the last trajectory (of a finite fragment) are maximal.*

The *trace* of an execution α represents its externally visible part, namely the external actions and time passage. It is obtained by removing internal actions, concatenating consecutive trajectories, and replacing all the trajectories with

their limit times.

$$trace(\alpha) = \tau.ltime \quad \text{if } \alpha = \tau,$$
$$trace(\alpha a \tau) = \begin{cases} trace(\alpha) a\ \tau.ltime & \text{if } a \in E, \\ trace(\alpha')\ (\tau' \frown \tau).ltime & \text{where } \alpha = \alpha'\tau', \text{ otherwise.} \end{cases}$$

Remark 1. Concatenating consecutive trajectories hides information about the time of occurrence of internal actions in the trace of a PTIOA. In STS [6] there is no notion of time or trajectories and a trace is obtained by simply removing the internal (invisible) actions and states from an execution. Hence our notion of trace differs significantly that in STS. One obvious approach for modeling time passage in STS, is to treat a transition labeled by a real number r as a time passage action of duration r. The traces of STS that one would obtain using this approach would contain information about the point of occurrence of internal actions over an interval of time. Consequently, proving that the *trace* function is measurable for PTIOAs requires more work compared to the corresponding proof in the STS setting.

Having defined traces and executions, we state the final axiom for PTIOAs. This axiom is used to prove the measurability of the trace function. Henceforth, we assume that a PTIOA satisfies **D3**.

D3 A fragment α with $\alpha.ltime < \infty$ has finite internal actions occur in α.

We denote the set of execution fragments, the set of executions, and the set of traces of PTIOA $\mathcal{A}$ by $\mathsf{Frags}_{\mathcal{A}}$, $\mathsf{Execs}_{\mathcal{A}}$ and $\mathsf{Traces}_{\mathcal{A}}$. The set of finite fragments, finite executions and finite traces are denoted by $\mathsf{Frags}^*_{\mathcal{A}}$, $\mathsf{Execs}^*_{\mathcal{A}}$ and $\mathsf{Traces}^*_{\mathcal{A}}$.

Composition and Hiding. The composition operation allows a PTIOA representing a complex system to be constructed by composing PTIOAs representing smaller subsystems. PTIOA components can communicate discretely through shared actions. In the future we will extend the framework to support communication through shared variables. The hiding operation "hides" a set of external actions by reclassifying them as internal. Thus prevents these actions from being used for further communication.

Definition 3. *Two PTIOAs $\mathcal{A}_1$ and $\mathcal{A}_2$ are said to be* compatible *if $H_1 \cap A_2 = H_2 \cap A_2 = O_1 \cap O_2 = \emptyset$.*

Definition 4. *The* composition *of two compatible PTIOAs $\mathcal{A}_1$ and $\mathcal{A}_2$, denoted by $\mathcal{A}_1||\mathcal{A}_2$, is the tuple $\mathcal{A} = ((X, \mathcal{F}_X), \bar{x}, A, \mathcal{R}, \mathcal{D}, \mathcal{T})$, where: (1) $(X, \mathcal{F}_X) = (X_1 \times X_2, \mathcal{F}_{X_1} \otimes \mathcal{F}_{X_2})$, (2) $\bar{x} = (x_1, x_2)$, (3) $A = A_1 \cup A_2, I = (I_1 \cup I_2) \setminus (O_1 \cup O_2), O = O_1 \cup O_2$, and $H = H_1 \cup H_2$, (4) $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$, (5) $\mathcal{D} \subseteq X \times A \times \mathcal{P}(X, \mathcal{F}_X)$ is the set of triples $((x_1, x_2), a, \mu_1 \otimes \mu_2)$ such that for $i \in \{1, 2\}$ if $a \in A_i$ then $(x_i, a, \mu_i) \in \mathcal{D}_i$, otherwise $\mu_i = \delta_{x_i}$. (6) $\mathcal{T} = \{\tau \in trajs(X) \mid \tau \downarrow X_i \in \mathcal{T}_i, i \in \{1, 2\}\}$.*

Proposition 2. *Suppose $\mathcal{A} = \mathcal{A}_1||\mathcal{A}_2$. For $i \in \{1, 2\}$, and measurable sets $R \subseteq \mathbb{R}_{\geq 0}$, $Y \subseteq X$, let $E^i_{R,Y}$ be the subset of X_i such that for each $x \in E^i_{R,Y}$, $maxtime_{\mathcal{T}_i}(x) \in R$ and $maxstate_{\mathcal{T}_i}(x) \in Y \lceil Y_i$. Then,*

$$E_{R,Y} = (E^1_{R,Y \lceil Y_1} \times Y_2) \bigcup (Y_1 \times E^2_{R,Y \lceil Y_2}).$$

Theorem 2. *If $\mathcal{A}_1, \mathcal{A}_2$ are compatible PTIOAs then $\mathcal{A} = \mathcal{A}_1||\mathcal{A}_2$ is a PTIOA.*

The proof of this theorem has several parts and it appears in [24]. Here we check that $\mathcal{A}$ satisfies the axiom **M0** in two parts. We require to show that $E_{R,Y}$ is a $\mathcal{F}_{X_1} \otimes \mathcal{F}_{X_2}$-measurable set for any $R \subseteq \mathbb{R}_{\geq 0}$, $Y \subseteq X$. This follows from Proposition 2. Next, we check that for any action $a \in A$, the set of states E_a where a is enabled in $\mathcal{A}$ is $\mathcal{F}_{X_1} \otimes \mathcal{F}_{X_2}$-measurable. Suppose a is a local action of $\mathcal{A}_1$, and let $E_a^1 \subseteq X_1$ be the set of states of $\mathcal{A}_1$ where a is enabled. The set of states of $\mathcal{A}$ where a is enabled is $E_a = E_a^1 \times X_2$. $\mathcal{A}_1$ satisfies **M0**, therefore $E_a^1 \in \mathcal{F}_{X_1}$ and $E_a = E_a^1 \times X_2 \in \mathcal{F}_{X_1} \otimes \mathcal{F}_{X_2}$. For any $B \subseteq A$, we get E_B by taking countable union of E_a's over actions in B.

Definition 5. *Let $\mathcal{A}$ be a PTIOA and $\mathcal{O}$ be a set of output tasks of $\mathcal{A}$. Let $S = \cup_{T \in \mathcal{O}} T$, that is, S is the set of all actions in the tasks in $\mathcal{O}$. Then,* $\mathsf{ActHide}(\mathcal{A}, S)$ *is defined as PTIOA $\mathcal{B}$ that is identical to $\mathcal{A}$ except that $O_{\mathcal{B}} = O_{\mathcal{A}} \setminus S$ and $H_{\mathcal{B}} = H_{\mathcal{A}} \cup S$.*

Theorem 3. *If $\mathcal{A}$ be a PTIOA, $\mathcal{O}$ a set of output tasks of $\mathcal{A}$ and $S = \cup_{T \in \mathcal{O}} T$. Then, $\mathcal{B} = \mathsf{ActHide}(\mathcal{A}, S)$ is also a PTIOA.*

Theorem 4 states the standard projection property.

Theorem 4. *Suppose $\mathcal{A} = \mathcal{A}_1||\mathcal{A}_2$. If α is an execution of $\mathcal{A}_1||\mathcal{A}_2$, then for $i \in \{1, 2\}$, the restriction of α to states and actions of $\mathcal{A}_i$ is an execution for $\mathcal{A}_i$.*

4 Probability Measure Over Executions and Traces

In order to construct a probability measure over the set of executions of a given PTIOA $\mathcal{A}$, we have to first define the measurable sets in Execs_A. The standard approach for probabilistic automata with discrete state spaces [26,5,4,22] is to define the σ-algebra as the collection of sets of the form $C_\alpha := \{\alpha' \mid \alpha \text{ is a prefix of } \alpha'\}$. One then defines the probability of each C_α as the product of the probabilities of the transition sequence in α. It is well known (see, e.g., generalization of Markov processes in [12]) that this approach does not work when the transitions give continuous probability distributions because the probability of occurrence of any particular finite sequence of transitions is typically 0. Instead of considering a set of executions that extend a *single* prefix, we consider a set containing executions that extend an any prefix from a "cylinder" or *base* of prefixes.

Definition 6. *A* base *is a finite sequence of the form $\Lambda = X_0 R_0 X_1 A_1 R_1 X_2 A_2 R_2 \ldots X_m A_m R_m X_{m+1}$, where for every $i \in \{0, \ldots, m+1\}$, $X_i \in \mathcal{F}_X$, R_i is a measurable set in $\mathbb{R}_{\geq 0}$ and for every $i \in \{1, \ldots, m\}$, $A_i \subseteq A$. The* length *of a base is the number of sets of actions in the sequence. The* basic set *corresponding to a base Λ is a set of execution fragments of $\mathcal{A}$,*

$$C_\Lambda = \{\, \tau_0 a_1 \tau_1 \ldots \tau_m \alpha \in \mathsf{Frags}_{\mathcal{A}} \mid \tau_0.fstate \in X_0, \forall i \in \{0, \ldots, m\} \quad \tau_i.ltime \in R_i, \\ \tau_i.lstate \in X_{i+1},\ \forall i \in \{1, \ldots, m\}, \quad a_i \in A_i\}. \tag{1}$$

With some abuse of notation we will abbreviate a base $\Lambda = X_0 R_0 X_1 \ldots X_m A_m R_m X_{m+1}$, as $\Lambda_1 X_{m+1}$ or as $\Lambda_2 A_m R_m X_{m+1}$, where Λ_1 and Λ_2 are the appropriate prefixes of Λ.

Lemma 1. *The collection $\mathscr{C}$ of all basic sets of $\mathcal{A}$ is a semi-ring.*

The σ-algebra generated by $\mathscr{C}$ is denoted by $\mathcal{F}_{\mathsf{Frags}_{\mathcal{A}}}$. The collection of sets obtained by taking the intersection of each element in $\mathscr{C}$ with $\mathsf{Execs}_{\mathcal{A}}$ is a semi-ring in $\mathsf{Execs}_{\mathcal{A}}$. We denote the σ-algebra generated by this semi-ring by $\mathcal{F}_{\mathsf{Execs}_{\mathcal{A}}}$. We define the measurable space of executions of $\mathcal{A}$ to be $(\mathsf{Execs}_{\mathcal{A}}, \mathcal{F}_{\mathsf{Execs}_{\mathcal{A}}})$. Likewise, we can define finite basic sets, a σ-algebra on $\mathsf{Frags}^*_{\mathcal{A}}$, and a measurable space $(\mathsf{Execs}^*_{\mathcal{A}}, \mathcal{F}_{\mathsf{Execs}^*_{\mathcal{A}}})$ of finite executions.

Definition 7. *A* trace base *is a finite sequence of the form $\Lambda = R_0 E_1 \dots E_n$ where $\forall i \in \{0, \dots, n-1\}$, R_i is a measurable set in $\mathbb{R}_{\geq 0}$ and and $\forall j \in \{1, \dots, n\}$, $E_j \subseteq E$. The* length *of a trace base is the number of sets of actions in the sequence. The* trace basic set *corresponding to the base Λ is a set of traces of $\mathcal{A}$ defined as: $C_\Lambda = \{r_0 a_1 r_1 \dots a_n \beta \in \mathsf{Traces}_{\mathcal{A}} \mid \forall i \in \{0, \dots, n\} \;\; r_i \in R_i, a_i \in E_i\}$.*

The collection $\mathscr{D}$ of all trace basic sets of $\mathcal{A}$ is a semi-ring. The σ-algebra $\mathcal{F}_{\mathsf{Traces}}$ on the set of traces of $\mathcal{A}$ is defined as the σ-algebra generated by the collection of trace basic sets; the measurable space of traces is denoted by $(\mathsf{Traces}_{\mathcal{A}}, \mathcal{F}_{\mathsf{Traces}_{\mathcal{A}}})$.

A Probability Measure Over Executions. In order to obtain a probability distribution over the set of executions of $\mathcal{A}$, we have to combine $\mathcal{A}$ with a *scheduler* that resolves nondeterministic choice over enabled actions. Various types of schedulers are possible. A scheduler may be oblivious, Markovian, or dependent complete history. Further, a scheduler may choose the an action, from the set of enabled action, either deterministically or according to some porbability distribution. In this paper use the task mechanism and an *oblivious task scheduler* [5,4]. An Oblivious scheduler chooses the next action deterministically and independent of the information produced during an execution.

Definition 8. *A* task schedule *for a closed PTIOA $\mathcal{A} = ((X, \mathcal{F}_X), \bar{x}, A, \mathcal{R}, \mathcal{D}, \mathcal{T})$ is a finite or infinite sequence $\rho = T_1 T_2 \dots$ of tasks in $\mathcal{R}$.*

A task schedule resolves nondeterministic choices by repeatedly scheduling tasks, each of which determines at most one transition for the PTIOA. A task schedule for $\mathcal{A}$ determines a probability measure over $(\mathsf{Execs}_{\mathcal{A}}, \mathcal{F}_{\mathsf{Execs}_{\mathcal{A}}})$. We define an operation that "applies" a task schedule to a PTIOA. Given any task schedule ρ the corresponding probability distribution $\mathsf{apply}(\delta_{\bar{x}}, \rho)$ over $\mathsf{Exec}_{\mathcal{A}}$, is called a *probabilistic execution* of $\mathcal{A}$.

For each basic set Λ and each $B \subseteq A$, we define $g_{\Lambda,B} : \mathsf{Execs}^*_{\mathcal{A}} \to \{0, 1\}$ as:

$$g_{\Lambda,B} = \begin{cases} 1 & \text{if } \alpha.lstate \in E_B \text{ and } \alpha \in C_\Lambda \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

Observe that $g_{B,\Lambda}$ is a measurable function. There are three cases to consider: if $\Lambda = Y$, for some $Y \in \mathcal{F}_X$, then $g^{-1}_{B,\Lambda}(1) = C_{Y \cap E_B}$, if $\Lambda = Y' R Y$, for some $Y', Y \in \mathcal{F}_X$, $R \in \mathcal{F}_{\mathbb{R}_{\geq 0}}$, then $g^{-1}_{B,\Lambda}(1) = C_{Y' R (Y \cap E_B)}$, and finally if $\Lambda = \Lambda' C R Y$, for some $C \subseteq A$, $S \in \mathcal{F}_{\mathbb{R}_{\geq 0}}$, $Y \in \mathcal{F}_X$, then $g^{-1}_{B,\Lambda}(1) = C_{\Lambda' C R (Y \cap E_B)}$.

Definition 9. *Let $\mathcal{A} = ((X, \mathcal{F}_X), \bar{x}, A, \mathcal{R}, \mathcal{D}, \mathcal{T})$ be a PTIOA. Given a task schedule ρ for $\mathcal{A}$ and a probability measure $\mu \in \mathcal{P}(\mathsf{Execs}^*_{\mathcal{A}}, \mathcal{F}_{\mathsf{Execs}^*_{\mathcal{A}}})$, $\mu' = \mathsf{apply}(\mu, \rho)$ is a probability measure in $\mathcal{P}(\mathsf{Execs}_{\mathcal{A}}, \mathcal{F}_{\mathsf{Execs}_{\mathcal{A}}})$, defined recursively as follows:*

1. $\mathsf{apply}(\mu, \lambda) := \mu$, *where λ denotes the empty sequence of tasks.*
2. $\mathsf{apply}(\mu, T) := \mu'$, *where T is a task in $\mathcal{R}$ and μ' is defined as follows:*

$$\mu'(C_Y) = \begin{cases} 1 & \text{if } \bar{x} \in Y \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

$$\mu'(C_{Y'RY}) = \begin{cases} 1 & \text{if } \bar{x} \in Y', maxtime(x) \in R, maxstate(\bar{x}) \in Y \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

$$\mu'(C_{\Lambda BRY}) = \int_{\alpha \in \Lambda} g_{\Lambda, B\cap T}(\alpha) \mu_{\alpha.lstate}(E_{R,Y}) \mu(d\alpha) + \mu(C_{\Lambda BR(Y \cap E^c_T)}) \tag{5}$$

3. $\mathsf{apply}(\mu, \rho) := \mathsf{apply}(\mathsf{apply}(\mu, \rho'), T)$, *if ρ is finite and of the form $\rho' T$.*
4. $\mathsf{apply}(\mu, \rho) := \lim_{i \to \infty}(\mu_i)$, *where ρ is infinite ρ_i is the length-i prefix of ρ, and $\mu_i = \mathsf{apply}(\mu, \rho_i)$.*

For any task schedule any basic set of the form C_Y or $C_{Y'RY}$ have measure 0 or 1 depending on the conditions in (3-4). This is a because the initial condition of a PTIOA is a Dirac distribution at $\bar{x}$. The probability measure of an arbitrary basic set $C_{\Lambda BRY}$ can be written as the the sum of probabilities of $C_{\Lambda BR(Y\cap E_T)}$ and $C_{\Lambda BR(Y\cap E^c_T)}$. The former can be further broken down as the "sum" over all executions $\alpha \in \Lambda$, of the probability that an action $a \in B \cap T$ can occur at $\alpha.lstate$ leading to a state in $E_{R,Y}$. In (5) this sum becomes an integral. The integral is well defined because the integrand is a product of two measurable functions g and $\mu_\alpha.lstate$. Measurability of $\mu_{\alpha.lstate}$ in Equation (5) follows from **D2**, the Giry construction described in 2 and the fact that $E_{R,Y}$ is in $\mathcal{F}_X$.

Proposition 3. *Let μ be a probability measure on $(\mathsf{Execs}^*_{\mathcal{A}}, \mathcal{F}_{\mathsf{Execs}^*_{\mathcal{A}}})$ and ρ be a task schedule for $\mathcal{A}$. Then $\mathsf{apply}(\mu, \rho)$ is a probability measure on $(\mathsf{Execs}_{\mathcal{A}}, \mathcal{F}_{\mathsf{Execs}_{\mathcal{A}}})$.*

Example. *(continued)* A typical execution of Consensus is a sequence $\alpha = \tau_0$ try τ_1 try τ_2 decide τ_3, were each τ_i, $i \in \{0, \ldots, 3\}$, is a trajectory over which *timer* increases monotonically at the same rate as real time and all other variables remain constant. The corresponding trace is $(\tau_0.ltime)$try$(\tau_1.ltime)$ try $(\tau_2.ltime)$ decide $(\tau_3.ltime)$.

Let us examine how Definition 9 assigns probability measures to the cones generated by the Consensus automaton. A state of Consensus is an ordered 4-tuple specifying the values of the 4 variables in the order in which they appear in Figure 1. So, $\bar{x} = (1, 0, 0, t_0)$ and we define $\bar{x}' = maxstate(\bar{x}) = (1, 0, t_0, t_0)$. Given a state x, we refer to the value of variable *var* at x by $x.var$.

Suppose $\mu_1 = \mathsf{apply}(\delta_{\bar{x}}, \lambda)$, where λ is the empty task schedule, $\Lambda_1 = \{\bar{x}\}R_0\{\bar{x}'\}$, $X_0 \in \mathcal{F}_X$, such that $\bar{x} \in X_0$. Then, $\mu_1(C_{\{X_0}) = 1$ and $\mu_1(C_{\Lambda_1}) = 1$. For any base $\Lambda_1 A_1 R_1 X_2$ that extends Λ_1, $\mu_1(C_{\Lambda_1 A_1 R_1 X_2}) = 0$.

Next, suppose $\mu_2 = \mathsf{apply}(\mu_1, \{\mathsf{try}\})$. Let $\Lambda_2 = \Lambda_1\{\mathsf{try}\}R_1X_2$, $\Lambda'_2 = \Lambda_1\{\mathsf{try}\}R_1X'_2$, where $R_1 = [0, r_1]$ for some $r_1 \in \mathbb{R}_{\geq 0}$, $X_2 = \{x \mid x.stage = 2, x.phase = 0, x.timer = x.delay\}$, and $X'_2 = \{x \mid x.stage = 0, x.phase = 1, x.timer = x.delay\}$. Then,

$$\mu_2(C_{\Lambda_2}) = \int_{s \in G} \mu_s(E_{R_1,X_2}) \mu_1(d\alpha) = \frac{1}{2^n}(1 - e^{\lambda_0 r_1}),$$

And likewise $\mu_2(C_{\Lambda_2})$ is $\left(1-\frac{1}{2^n}\right)(1-e^{\lambda_0 r_1})$. The integrals are over $G := \{x \mid g_{\Lambda_1,\{a\}}(x)\}$. Since $\mu_1(\alpha) = 1$ for a single execution which is a trajectory starting from $\bar{x}$ and ending at $\bar{x}'$, in this case the integrals reduce to $\mu_{\bar{x}'}(E_{R_1,X_2})$ and $\mu_{\bar{x}'}(E_{R_1,X_2'})$, respectively. From the above we can deduce that $\mu_2(C_{\Lambda_1}) = 1$.

Consider another step. Suppose $\mu_3 = \mathsf{apply}(\mu_2, \{\mathsf{decide}\})$ and $R_2 = [0, r_2]$, and $X_3 = \{x \mid x.stage = 0, x.phase = 2, x.timer = x.delay\}$, where $r_2 \in \mathbb{R}_{\geq 0}$. From the first part of (5) we can calculate $\mu_3(C_{\Lambda_2'\{\mathsf{decide}\}R_2X_3}) = \left(1-\frac{1}{2^n}\right)(1-e^{\lambda_0 r_1})(1-e^{\lambda_1 r_2})$, and from the second part of (5) $\mu_3(C_{\Lambda_2}) = \mu_2(C_{\Lambda_2}) = \frac{1}{2^n}(1-e^{\lambda_a r_1})$.

Trace Distributions. For any probabilistic execution μ of a PTIOA, we want there to be a unique corresponding measure on the space of traces. Formally, the image measure of μ with respect to the *trace* function should be well defined. Therefore we require the function $trace : (\mathsf{Execs}, \mathcal{F}_{\mathsf{Execs}}) \rightarrow (\mathsf{Traces}, \mathcal{F}_{\mathsf{Traces}})$ to be measurable. Consider a simple trace base $[0, r]\{a\}$, where r is a positive real and a is an external action. Then, $\mathcal{E} = trace^{-1}(C_{[0,r]\{a\}})$ is the set of all finite executions of the form $\tau_1 h_1 \tau_2 h_2 \ldots h_{n-1}\tau_n a$, such that all the h_i's are internal actions and $\sum_{i=1}^{n} \tau_i.ltime \leq r$. For the *trace* function to be measurable $\mathcal{E}$ must be in $\mathcal{F}_{\mathsf{Execs}}$, that is, $\mathcal{E}$ should be expressible as a countable union of basic sets. Showing this requires some work because the condition on the sum of the τ_i's makes them interdependent. In what follows, we state and explain the key lemmas that go into proving this fact.

A trace base Γ of the form $[0, b_0)E_1[0, b_1)E_2 \ldots E_n$, where each $b_i \in \mathbb{R}_{\geq 0}$ and each $E_i \subseteq E$, is said to be a *canonical trace base*. Lemma 2 states that for proving measurability of *trace*, it suffices to show that for any canonical trace base Γ, $trace^{-1}(C_\Gamma)$ is in the σ-algebra of executions.

Lemma 2. *Consider a function $f : (\mathsf{Execs}, \mathcal{F}_{\mathsf{Execs}}) \rightarrow (\mathsf{Traces}, \mathcal{F}_{\mathsf{Traces}})$. If $f^{-1}(C_\Gamma) \in \mathcal{F}_{\mathsf{Execs}}$ for every canonical trace base Γ then f is measurable.*

The main construction in the proof, as provided by the next definition, is to express $\mathcal{E}$ as the countable union of a disjoint basic sets that partition $[0, r]$ into several intervals with dyadic rational endpoints. A base is then constructed by inserting internal actions in between the successive sub-intervals.

Definition 10. *Let $\Gamma = [0, 1)E_1$ be a canonical trace base of unit length and $n, k \in \mathbb{N}$, such that $2^k > n$. For a set of indices $j_1, \ldots, j_n \in \mathbb{N}$, satisfying $j_1 \leq 2^k - n - 1$, and for each $i = 2 \ldots n$, $j_i \leq 2^k - \left(1 + n + \sum_{s=1}^{i-1} j_s\right)$, we define:*

$$\Delta^k_{j_1,\ldots,j_n} := X[\frac{j_1}{2^k}, \frac{j_1+1}{2^k})XH[\frac{j_2}{2^k}, \frac{j_2+1}{2^k})X \ldots H[\frac{j_n}{2^k}, \frac{j_n+1}{2^k})X$$
$$H[0, 1 - \frac{n + \sum_{s=1}^{n} j_s}{2^k})XE_1[0, \infty)X.$$

The rational n-partition *of Γ, denoted by Q_n, is defined as follows:*

$$Q_n := \lim_{k\to\infty} \bigcup_{j_1} \bigcup_{j_2} \ldots \bigcup_{j_n} C_{\Delta^k_{j_1,\ldots,j_n}}$$

Q_n is a union over all possible choices of n dyadic rationals in $[0, 1)$, and therefore is a countable union. Geometrically, the $(n + 1)$ real intervals in each basic set of Q_n, define a rectangle in $\mathbb{R}^{n+1}$; the rectangles corresponding to distinct basic sets are disjoint and their union is an approximation of the unit simplex in $\mathbb{R}^{n+1}$. Using this construction we are able to prove the next Lemma.

Lemma 3. *If Γ is a canonical trace base then $trace^{-1}(C_\Gamma) = \bigcup_{n=1}^{\infty} Q_n$.*

From Lemmas 3 and 2, the next lemma follows immediately, and the measurability of the *trace* function is a corollary of this lemma.

Lemma 4. *$trace^{-1}(\Gamma)$ for a trace base Γ is a countable union of disjoint basic sets.*

The *trace distribution* corresponding to a probabilistic execution μ_ρ given by a task schedule ρ, is written as $tdist(\rho)$, and it defined as the image measure of μ_ρ under the *trace* function. More formally, $tdist(\rho) : \mathsf{Traces}_\mathcal{A} \rightarrow [0,1]$, is defined as $tdist(\rho)(E) = \mu_\rho(trace^{-1}(E))$, for any measurable set $E \in \mathcal{F}_{\mathsf{Traces}_\mathcal{A}}$. Note that $trace^{-1}(E) \in \mathcal{F}_{\mathsf{Execs}_\mathcal{A}}$ because *trace* is a measurable function. The *set of trace distributions* of $\mathcal{A}$, $tdists(\mathcal{A})$ is the set of $tdist(\rho)$'s for any task schedule ρ of $\mathcal{A}$.

We define a notion of *external behavior* for PTIOAs based on trace distributions, and show that the implementation relation based on this external behavior is compositional. We formulate the external behavior of a $\mathcal{A}$ as a mapping from possible "environments" for $\mathcal{A}$ to sets of trace distributions that can arise when $\mathcal{A}$ is composed with the given environment. Theorem thm:compositionality states that PTIOAs are compositional with respect to external behavior.

Definition 11. *An* environment *for PTIOA $\mathcal{A}$ is a PTIOA $\mathcal{E}$ such that $\mathcal{A}$ and $\mathcal{E}$ are compatible and their composition $\mathcal{A}||\mathcal{E}$ is closed. The* external behavior *of a PTIOA $\mathcal{A}$, written as $extbeh_\mathcal{A}$, is defined as a function that maps each environment PTIOA $\mathcal{E}$ for $\mathcal{A}$ to the set of trace distributions $tdists(\mathcal{A}||\mathcal{E})$.*

Definition 12. *Two PTIOAs $\mathcal{A}_1$ and $\mathcal{A}_2$ are* comparable *if $E_1 = E_2$. If $\mathcal{A}_1$ and $\mathcal{A}_2$ are comparable then $\mathcal{A}_1$ is said to* implement *$\mathcal{A}_2$, written as $A_1 \leq A_2$ if, for every environment PTIOA $\mathcal{E}$ for both $\mathcal{A}_1$ and $\mathcal{A}_2$, $extbeh_{\mathcal{A}_1}(\mathcal{E}) \subseteq extbeh_{\mathcal{A}_2}(\mathcal{E})$.*

Theorem 5. *Suppose $\mathcal{A}_1, A_2$ and $\mathcal{B}$ are PTIOAs, where $\mathcal{A}_1$ and $\mathcal{A}_2$ are comparable and $\mathcal{A}_1 \leq \mathcal{A}_2$. If $\mathcal{B}$ is compatible with $\mathcal{A}_1$ and $\mathcal{A}_2$ then $\mathcal{A}_1||\mathcal{B} \leq \mathcal{A}_2||\mathcal{B}$.*

5 Generalized PTIOAs and Local Schedulers

So far, we have restricted PTIOAs to have deterministic trajectories, and we have not allowed choice between enabled actions and non-trivial trajectories (axiom **D1**). In this section, relax these assumptions by adding *local schedulers*.

Definition 13. *A* Generalized PTIOA *is a tuple $\mathcal{A} = ((X, \mathcal{F}_X), \bar{x}, A, \mathcal{R}, \mathcal{D}, \mathcal{T})$, where the first five components are the same as in Definition 2. The set $\mathcal{T}$ is not necessarily deterministic and $\mathcal{A}$ does not necessarily satisfy* **D1**.

Thus, from a given state $x \in X$ of a generalized PTIOA, $\mathcal{A}$ there may be nondeterministic choice of actions that could be performed and also choice of distinct trajectories starting from x. A *local scheduler* for generalized PTIOA $\mathcal{A}$, is a PTIOA $S = ((X, \mathcal{F}_X), \bar{x}, A, \mathcal{R}, \mathcal{D}', \mathcal{T}')$ that is identical to $\mathcal{A}$ except that $\mathcal{D}' \subseteq D$ and $\mathcal{T}' \subseteq \mathcal{T}$. A *local scheduler* S satisfies **D1** and has deterministic trajectories.

A probabilistic system captures the notion of possible ways of resolving the nondeterminism in a generalized PTIOA. Formally, a *probabilistic-system* is a pair $\mathcal{M} = (\mathcal{A}, \mathcal{S})$, where $\mathcal{A}$ is a generalized PTIOA and $\mathcal{S}$ is a set of local schedulers for $\mathcal{A}$. An *environment* for $\mathcal{M}$ is any PTIOA $\mathcal{E}$ such that $\mathcal{A}||\mathcal{E}$ is closed. The notions of probabilistic

execution and trace distribution defined earlier for PTIOAs, carry over naturally to generalized PTIOAs. A probabilistic execution for $\mathcal{M}$ is defined to be any probabilistic execution of S, for any $S \in \mathcal{S}$. For probabilistic system $\mathcal{M} = (\mathcal{A}, \mathcal{S})$, we define the *external behavior* of $\mathcal{M}$ to be the total function $extbeh_{\mathcal{M}}$ that maps each environment PTIOA $\mathcal{E}$ for $\mathcal{M}$ to the set $\cup_{S' \in \mathcal{S}} tdists(S'||\mathcal{E})$. Thus for each environment, we consider the set of trace distributions that arise from the choices of the local scheduler of $\mathcal{M}$ and the task scheduler ρ. This leads to a notion of implementation of probabilistic systems, similar to that of PTIOAs.

Definition 14. *Let $\mathcal{M}_1 = (\mathcal{A}_1, \mathcal{S}_1)$ and $\mathcal{M}_2 = (\mathcal{A}_2, \mathcal{S}_2)$ be probabilistic systems such that $\mathcal{A}_1$ and $\mathcal{A}_2$ are comparable generalized PTIOAs. Then, $\mathcal{M}_1$ is said to* implement *$\mathcal{M}_2$ if for every environment $\mathcal{E}$ of $\mathcal{M}_1$ and $\mathcal{M}_2$, $extbeh_{\mathcal{M}_1}(\mathcal{E}) \subseteq extbeh_{\mathcal{M}_2}(\mathcal{E})$.*

Two probabilistic systems $\mathcal{M}_1 = (\mathcal{A}_1, \mathcal{S}_1)$ and $\mathcal{M}_2 = (\mathcal{A}_2, \mathcal{S}_2)$ are compatible if $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible, and their composition $\mathcal{M}_1||\mathcal{M}_2$ is defined as $(\mathcal{A}_1||\mathcal{A}_2, \mathcal{S})$, where $\mathcal{S}$ is the set of local schedulers $\{S_1||S_2 \mid S_1 \in \mathcal{S}_1 \text{ and } S_2 \in \mathcal{S}_2\}$. Theorem 6 gives the following sufficient condition for implementation of probabilistic systems: each local schedular for the concrete probabilistic system must always correspond to the same local scheduler for the abstract.

Theorem 6. *If $\mathcal{M}_1 = (\mathcal{A}_1, \mathcal{S}_1)$, $\mathcal{M}_2 = (\mathcal{A}_2, \mathcal{S}_2)$ are comparable and there exists $f : \mathcal{S}_1 \rightarrow \mathcal{S}_2$, such that for all $S_1 \in \mathcal{S}_1$, S_1 implements $f(S_1)$, then $\mathcal{M}_1$ implements M_2.*

Conclusions. In this paper, we have introduced PTIOA—a compositional framework for modelling and verifying discretely communicating probabilistic hybrid systems. PTIOAs support continuous distributions, nondeterminism, and probabilistic transitions. Using local and task schedulers and two key measurability assumptions we have constructed the probability measures over the space of executions and traces of PTIOAs. We have defined external behavior of a PTIOA as a mapping from closing environment PTIOAs to trace distributions. We have shown that this notion of external behavior is compositional. In the future we will extend PTIOAs to support shared variables and develop new proof techniques for verification of quantitative properties such as, probabilistic safety, and approximate implementations.

Acknowledgments We have immensely benefited from discussing this work with Sanjoy Mitter. We also thank the anonymous referees for their valuable comments on a previous version of this paper.

References

1. C. Baier, F. Ciesinski, and M. Grer. ProbMela and model checking markov decision processes. In *ACM Performance Evaluation Review on Performance and Verification*, 2006. to appear.
2. M. Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols. In *PODC'83*, pages 27–30, 1983.
3. M. Bujorianu and J. Lygeros. General stochastic hybrid systems: Modelling and optimal control. In *IEEE Conference on Decision and Control*, December 2004.
4. R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala. Task-structured probabilistic I/O automata. Tech. Report MIT-CSAIL-TR-2006-060, MIT, Cambridge, 2006.
5. R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala. Time-bounded task-PIOAs: a framework for analyzing security protocols. In *DISC '06*, 2006.

6. S. Cattani, R. Segala, M. Z. Kwiatkowska, and G. Norman. Stochastic transition systems for continuous state spaces and non-determinism. In *FoSSaCS*, pages 125–139, 2005.
7. D. Chatterjee and D. Liberzon. Stability analysis of deterministic and stochastic switched systems via a comparison principle and multiple lyapunov functions. *SIAM Journal on Control and Optimization*, 2005.
8. L. Cheung. *Reconciling nondeterministic and probabilistic choices*. PhD thesis, ICIS, Radboud University Nijmegen, The Netherlands, 2006.
9. V. Danos, J. Desharnais, F. Laviolette, and P. Panangaden. Bisimulation and cocongruence for probabilistic systems. *Information and Computation, Special issue for selected papers from CMCS04*, 2005.
10. M. H. A. Davis. *Markov Models and Optimization*. Chapman & Hall, 1993.
11. L. de Alfero. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, CA, 1997. Technical Report STAN-CS-TR-98-1601.
12. J. L. Doob. *Stochastic Processes*. John Wiley & Sons, Inc., New York, 1953.
13. R. M. Dudley. *Real Analysis and Probability*. Wadsworth, Belmont, Calif, 1989.
14. S. Smolka E. Stark, R. Cleaveland. A process-algebraic language for probabilistic I/O automata. In *Proc. CONCUR 03*, *LNCS* 2761:189–203, 2003.
15. M. Giry. A categorical approach to probability theory. In B. Banaschewski, editor, *Categorical Aspects ofTopology and Analysis*, *LNM* 915:68–85, 1981.
16. M. Hennessy and T. Regan. A process algebra for timed systems. *Information and Computation*, 117:221–239, 1995.
17. H. Hermanns. *Interactive Markov Chains : The Quest for Quantified Quality*. Springer, 2002.
18. J. P. Hespanha. Stochastic hybrid systems: Application to communication networks. In *HSCC 2004*, LNCS 2993:387 – 401, 2004.
19. A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *TACAS'06*, *LNCS* 3920, 2006.
20. D. Kaynar, N. Lynch, S. Mitra, and S. Garland. *TIOA Language*. MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, 2005.
21. D. Kaynar, N. Lynch, R. Segala, and F. Vaandrager. *The Theory of Timed I/O Automata*. Synthesis Lectures on Computer Science. Morgan Claypool, 2005.
22. M. Kwiatkowska, G. Norman, J. Sproston, and F. Wang. Symbolic model checking for probabilistic timed automata. In *FORMATS'04*, *LNCS* 3253:293–308. 2004.
23. M. W. Mislove, J. Ouaknine, and J. Worrell. Axioms for probability and nondeterminism. *ENTCS* 96(9):7–28, 2004.
24. S. Mitra and N. Lynch. Trace-based Semantics for Probabilistic Timed I/O Automata Submitted for review. Full version `http://theory.lcs.mit.edu/~mitras/research/PTIOA06-full.pdf`
25. S. Mitra and N. Lynch Approximate simulations for task-structured probabilistic I/O automata. In LICS Workshop on Probabilistic Automata and Logics, 2006.
26. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Laboratory for Computer Science, MIT, 1995.
27. E. Stark. On behavior equivalence for probabilistic I/O automata and its relationship to probabilistic bisimulation. *Journal of Automata, Languages, and Combinatorics*, 8(2):361–395, 2003.
28. F. van Breugel, M. W. Mislove, J. Ouaknine, and J. Worrell. Domain theory, testing and simulation for labelled markov processes. *Theoretical Computer Science*, 2005.