Under consideration for publication in Formal Aspects of Computing

Layered Reasoning for Randomized Distributed Algorithms

Mani Swaminathan[1] and Joost-Pieter Katoen[2] and Ernst-Rüdiger Olderog[1]

[1]University of Oldenburg, Germany
[2]RWTH Aachen University, Germany

Abstract. This paper adopts the communication closed layer (CCL) concept of Elrad and Francez to the formal reasoning of randomized distributed algorithms. We do so by enriching probabilistic automata (PA) with a layered composition operator, an intermediate between parallel and sequential composition. Layered composition is used to establish probabilistic counterparts of the CCL laws that exploit independence and / or precedence conditions between the constituent PA. The probabilistic CCL laws enable partial order (po-) equivalence when layered composition is replaced by sequential composition. Such po-equivalence induces a purely syntactic partial-order state space reduction via layered separation in compositions of PA while preserving probabilistic next-free linear-time properties. The feasibility of such layered separation is demonstrated on a randomized mutual exclusion algorithm by Kushilevitz and Rabin, complementing an algebraic approach (for analyzing this algorithm) by McIver, Gonzalia, Cohen, and Morgan.

Keywords: Probabilistic Automata, Layered Composition and Separation, Communication Closedness, Partial Order Equivalence, Randomized Mutual Exclusion

1. Introduction

Randomized distributed algorithms are intricate. Randomization is of paramount importance in the setting of distributed algorithms. It is used to break the symmetry between identical process components in leader election and mutual exclusion, for routing purposes, or for obtaining consensus—a problem that is known to be unsolvable in a deterministic setting. The design and analysis of randomized distributed algorithms is however highly non-trivial. Lehmann and Rabin, e.g., argue that "proofs of correctness for probabilistic distributed systems are extremely slippery" [LR81], and various flawed versions of randomized distributed algorithms exist, cf. [Seg00]. This is mainly due to the fact that the stochastic process describing the evolution of a randomized distributed algorithm changes depending on the generally unknown scheduling policies and relative speeds of the individual process components, entailing a complex interplay between randomization and nondeterminism.

Correspondence and offprint requests to: Mani Swaminathan : mani.swaminathan@informatik.uni-oldenburg.de . This work is supported by the German Research Foundation through the Trans-Regio Collaborative Research Center (SFB/TR 14) AVACS (www.avacs.org), and by the EU through the FP7 project MoVeS (www.movesproject.eu).

Why probabilistic automata? Probabilistic automata [SL95, Seg00] (PA) constitute an operational framework for the modelling and analysis of discrete systems that exhibit both nondeterministic and randomized behavior, such as randomized distributed algorithms. An I/O-variant of PA (PIOA) has been used to successfully analyze intricate randomized distributed algorithms such as the Aspnes-Herlihy randomized consensus algorithm [PSL00] and the IEEE Firewire protocol [SV99]. Extensions of PIOA have been used for specifying and verifying security protocols [CCK$^+$08]. In the context of concurrency theory, PA are used as semantic models for stochastic process algebras, and have been equipped with (bi)simulation notions [SL95].

Layering. Despite the presence of modular verification techniques for PA [Seg00], the correctness proofs of randomized distributed algorithms remain difficult and require substantial human ingenuity. This paper attempts to simplify their reasoning by enriching probabilistic automata with the concept of *layering*. The main underlying idea is that the computations of randomized distributed algorithms often exhibit a *sequential* (i.e., layered) structure. The idea of using such sequential structure to simplify the verification of distributed algorithms was originally proposed in the eighties by Elrad and Francez [EF82], and has been extended, formalized [SdR94], and applied to intricate distributed algorithms such as the minimal spanning tree algorithm [JZ92] about a decade later. Layered reasoning (though in a different way as in this paper) has been recently used to obtain tighter bounds for asynchronous randomized consensus [AC08]; earlier work on applying layering to bound analysis appeared in [MR02]. Most recently, a notion of layering for distributed real-time systems (modelled as networks of timed automata) has been investigated in [OS10]. To our knowledge, layering has not been applied to ease the verification of randomized distributed algorithms. We therefore study in this paper layered reasoning for randomized distributed algorithms, with PA (enriched with shared data variables) as the underlying operational model. Our layered reasoning here builds on that investigated in [OS10] for the non-randomized, real-time setting.

Our contributions. For simplifying the formal reasoning of randomized distributed algorithms, we introduce layered composition $\mathcal{P} \bullet \mathcal{Q}$ of PA $\mathcal{P}$ and $\mathcal{Q}$; the PA $\mathcal{P} \bullet \mathcal{Q}$ behaves like $\mathcal{P}||\mathcal{Q}$, the parallel composition of $\mathcal{P}$ and $\mathcal{Q}$, except that for all actions a of $\mathcal{P}$ and b of $\mathcal{Q}$ that depend on each other (e.g., as both actions affect the same shared variable), a is executed before b. Layered composition is thus a kind of asymmetric parallel composition. We obtain a probabilistic version of the *communication closed layer* (CCL) law that allows us to identify $(\mathcal{P}_1 \bullet \mathcal{P}_2)||(\mathcal{Q}_1 \bullet \mathcal{Q}_2)$ and $(\mathcal{P}_1||\mathcal{Q}_1) \bullet (\mathcal{P}_2||\mathcal{Q}_2)$, provided $\mathcal{P}_1, \mathcal{Q}_2$ and $\mathcal{P}_2, \mathcal{Q}_1$ respect pair-wise certain *independence* or *precedence* conditions. This CCL law enables us to transform a randomized distributed algorithm into an equivalent layered one so as to permit easier verification. This verification is technically enabled using a partial-order (po) equivalence on probabilistic automata; in particular we show that $\mathcal{P} \bullet \mathcal{Q}$ and $\mathcal{P}; \mathcal{Q}$ are po-equivalent. The notion of po-equivalence is shown to preserve probabilistic next-free linear temporal logic properties. The CCL law together with the po-equivalence of $\bullet$ and ; allows the transformation of $(\mathcal{P}_1; \mathcal{P}_2)||(\mathcal{Q}_1; \mathcal{Q}_2)$ –via intermediate representations $(\mathcal{P}_1 \bullet \mathcal{P}_2)||(\mathcal{Q}_1 \bullet \mathcal{Q}_2)$ and $(\mathcal{P}_1||\mathcal{Q}_1) \bullet (\mathcal{P}_2||\mathcal{Q}_2)$– finally to $(\mathcal{P}_1 \| \mathcal{P}_2); (\mathcal{Q}_1 \| \mathcal{Q}_2)$ under the aforementioned independence or precedence conditions. This yields a *syntactic (partial order) state-space reduction* that may be applied prior to automated model checking of randomized distributed algorithms (as in [KN02]). We illustrate the feasibility of probabilistic layering on the (non-trivial) randomized mutual exclusion algorithm of Kushilevitz and Rabin [KR92]. This algorithm improves an earlier version proposed by Rabin in [Rab82], whose correctness proof was demonstrated as being flawed in [Sai92], due to the comparison of two probabilities that were not defined within the same probability space [Seg00].

Relation to the work of Carroll Morgan. Since the mid-nineties, the research of Carroll Morgan has largely been centered on the (algebraic) specification and verification of probabilistic programs described in a *probabilistic guarded command language* (pGCL). This research has been consolidated in the monograph co-authored with Annabelle McIver [MM04]. More recently, Morgan (along with his co-authors McIver, Gonzalia, and Cohen) advanced the pGCL-based algebraic specification and verification approach by a *probabilistic Kleene Algebra* (pKA) [MGCM08], with application to randomized distributed algorithms. This pGCL- / pKA- based approach compares to our layering as follows:

- Central to the pGCL/pKA approach in [MGCM08] is the exploitation of the *separation* theorems introduced earlier in [Coh00] for the non-randomized setting. Such separation theorems simplify the (algebraic) reasoning of (randomized) distributed systems by reducing complex interleavings into "separated" behaviours that admit individual analysis [MGCM08]. Our layering principle is similar in spirit: we exploit

structural properties (formalized by means of suitable independence and precedence conditions that induce communication closedness) in complex randomized distributed systems so as to admit a *layered separation* that consequently simplifies verification.

- The work in [MGCM08] gives an *axiomatic* specification of some key properties of the randomized mutual exclusion algorithm of Kushilevitz and Rabin [KR92]. These axioms then enable simplified reasoning via separation theorems expressed in pKA. While the randomized mutual exclusion algorithm of [KR92] is likewise considered in our paper for demonstrating the applicability and potential of our layered separation, we however provide a more detailed *operational* perspective of the algorithm's intricate behaviour via suitable PA models (cf. Figures 1-4). An analysis of these PA models then enables us to *derive* properties of the algorithm that were algebraically specified in Figure 7 of [MGCM08], as a consequence of our layered separation (cf. Proposition 7 and Section 6).
- Our layered separation and the pKA-based separation in [MGCM08] differ in their respective operational models and the verification techniques that they consequently admit. While the pKA-based separation aims for easier *assertional reasoning* or *interactive theorem proving* using pGCL in which probabilities can be treated as parameters, our layered separation is focussed on Segala's PA, aiming to achieve a *syntactic (partial order) state space reduction* prior to *model checking*.

Other related work. Partial-order reductions based on ample-set constructions for Markov Decision Processes (MDPs) have been investigated in [BGC04, DN04] and shown to preserve next-free probabilistic linear- and branching-time properties. As MDPs are basically PA in which the transition relation is viewed as a function, notions like action independence from [BGC04, DN04] can be easily adopted in our setting. An alternative symbolic on-the-fly partial-order reduction (termed confluence reduction) has been proposed in [TSvdP11] for the preservation of probabilistic branching bisimulation, based on a process-algebraic framework for data-enriched PA [KvST12].

Outline of this paper. Section 2 recalls the PA framework and the notion of trace distributions. It also defines sequential (denoted by ;) and parallel composition (denoted by ||). Section 3 defines the notion of layered composition (denoted by $\bullet$), precedence relations for PA, and formulates the probabilistic CCL laws. Section 4 defines po-equivalence $\equiv^*_{po}$ and shows its preservation when $\bullet$ is replaced by ; within the CCL laws. It also demonstrates the preservation by $\equiv^*_{po}$ of probabilistic next-free linear temporal properties. Section 5 shows the applicability of our approach to the randomized mutual exclusion algorithm of Kushilevitz and Rabin [KR92]. Section 6 concludes the paper with additional perspectives on our results (particularly in relation to the pGCL- / pKA- based analysis in [MGCM08]) and some directions for future work.

2. Probabilistic Automata

In this section, we introduce *Probabilistic Automata* (PA) [Seg00, Sto02] enriched with *shared data variables* as operational models for randomized distributed algorithms. Such PA exhibit both *probabilistic* and *non-deterministic* behaviour. The non-deterministic behaviours of PA are resolved using *adversaries*. *Trace distributions* characterize the *probability spaces* associated to such adversaries. The behaviours of PA are compared in terms of their trace distributions. To enable *modular reasoning* of PA, we also introduce the concepts of *sequential* and *parallel composition*. The *non-deterministic selection* of a component during distributed execution is modelled by a corresponding operator on PA.

Actions, data variables, and probability distributions. Let Σ be a finite *alphabet* of (communication) *channels*. A typical element of Σ is denoted $\alpha, \beta, \cdots$. There are two *actions* for each channel $\alpha \in \Sigma$: $\alpha?$ denotes an *input* on α, while $\alpha!$ denotes the corresponding *output* on α, where $\alpha?, \alpha! \notin \Sigma$. We denote by τ an action resulting from *synchronization*, with $\tau \notin \Sigma$. By $\Sigma_{?!} = \{\alpha? \mid \alpha \in \Sigma\} \cup \{\alpha! \mid \alpha \in \Sigma\} \cup \{\tau\}$, we denote the set of all actions over the alphabet Σ. A typical element of $\Sigma_{?!}$ is denoted $a, b, \cdots$. In the context of parallel composition, input and output are *complementary* actions that can synchronize yielding τ. For an action $a \in \Sigma_{?!} \setminus \{\tau\}$, its complementary action is denoted by $\overline{a}$, i.e., $\overline{\alpha?} = \alpha!$ and vice-versa. We stipulate that τ-actions may result only from the synchronization of complementary input and output actions.

Let V be a finite set of *data variables* (ranged over by v) that take values in some finite range D. By $\Psi(V)$ we denote the set of *data updates* with typical element ψ. The left-hand side of an update is a variable in V

whereas its right-hand side is an expression involving the variables of V and the usual arithmetic operators $+, -, \cdots$. For each $\psi \in \Psi(V)$ each variable occurs at most once on the left-hand side of an update. The set $\Phi(V)$ of *data constraints* over V with typical element ϕ is the set of Boolean constraints over variables in V involving the usual arithmetic $(+, -, \cdots)$ and relational $(<, \leq, >, \geq)$ operators. A *data valuation* assigns a value in D to each data variable in V. If $|V| = m$, a data valuation is identified with a point in D^m, denoted typically by $\vec{u}, \vec{v}$ etc. Applying an update $\psi \in \Psi(V)$ on data valuation $\vec{u}$ yields the data valuation $\psi(\vec{u})$. For valuation $\vec{u}$ and boolean constraint ϕ, let $\vec{u} \models \phi$ denote that the valuation $\vec{u}$ satisfies the data constraint ϕ.

Let S be a countable set. The function $\mu : S \to [0, 1]$ is a *distribution* on S if $\sum_{s \in S} \mu(s) = 1$. Let $Dist(S)$ denote the set of distributions on S and $supp(\mu) = \{s \in S \mid \mu(s) > 0\}$ be the *support* of μ.

Definition 1 (Probabilistic automata (PA)). PA are tuples $\mathcal{P} = \langle L, l_0, l_f, V, \Sigma, prob \rangle$, where:

- L is a finite, non-empty set of *locations*, ranged over by l, l', $\cdots$.
- $l_0 \in L$ is the *start location*.
- $l_f \in L$ is the *final location* with $l_0 \neq l_f$.
- V is a finite set of *data variables* taking on values in a finite range D.
- Σ is a finite *alphabet* of *channels*.
- $prob \subseteq L \setminus \{l_f\} \times \Sigma_{?!} \times \Phi(V) \times Dist(\Psi(V) \times L)$ is a *probabilistic transition relation*.

In the above definition, *prob* relates a location $l \neq l_f$, action a and guard $\phi \in \Phi(V)$ to a *distribution* over $\Psi(V) \times L$, i.e., an update and a target location. The intuitive operational meaning of $(l, a, \phi, \mu) \in prob$ is as follows. Given the current location l, action a, and a data valuation in l satisfying the guard ϕ, with probability $\mu(\psi, l')$ a transition to location l' is made while updating the data variables according to the update ψ. In case $(l, a, \phi, \mu) \in prob$ and $(l, a, \phi, \nu) \in prob$, on action a and satisfaction of guard ϕ a non-deterministic choice between distributions μ and ν is made. Tuples $(l, a, \phi, \mu) \in prob$ are called *edges* of the PA. Let $edges(l, a, \phi)$ denote the set of edges $\{(l, a, \phi, \mu) \in prob\}$. Note that by the above definition, $edges(l_f, a, \phi) = \emptyset$ for all $a \in \Sigma_{?!}$ and all $\phi \in \Phi(V)$.

Remark 1. According to the above definition, *multiple* distributions can be associated with a given location l, action a, and guard ϕ, as *prob* is a relation. This enables a conceptually cleaner notion of non-deterministic choice between probabilistic automata, as will be defined later (cf. Definition 13).

The semantics of a PA is given in terms of a *probabilistic transition system* (PTS). A PTS is basically a labelled transition system where the target of labelled transitions are distributions over states, rather than just simply states. A *state* of a PA with $|V| = m$ is a pair $(l, \vec{u}) \in L \times D^m$, denoted typically by s, consisting of a location l and a data valuation $\vec{u}$.

Definition 2 (Induced probabilistic transition system). The PA $\mathcal{P} = \langle L, l_0, l_f, V, \Sigma, prob \rangle$ with $|V| = m$ induces the PTS $[\![\mathcal{P}]\!] = \langle S, s_0, \Sigma, \Delta \rangle$, where

- $S = L \times D^m$ is the state space.
- $s_0 = (l_0, \vec{v_0})$, where $\vec{v_0} \in D^m$ denotes a designated initial data valuation.
- $\Delta \subseteq S \times \Sigma_{?!} \times Dist(S)$ is the transition relation defined by:

$$((l, \vec{u}), a, \nu) \in \Delta \text{ iff } (l, a, \phi, \mu) \in prob \text{ and } \vec{u} \models \phi, \text{ with } \nu((l', \vec{v})) = \sum \{\!|\, \mu(\psi, l') \mid \vec{v} = \psi(\vec{u}) \,|\!\},$$

  where $\{\!|\, \ldots \,|\!\}$ denotes a multi-set.

The transition relation Δ thus contains triples (s, a, ν) with $s = (l, \vec{u})$ whenever there is an edge (l, a, ϕ, μ) in the PA such that the valuation $\vec{u}$ satisfies the guard ϕ. The probability to move to the state $s' = (l', \vec{v})$ is the cumulative probability $\mu(\psi, l')$ where the valuation $\vec{v}$ is obtained from $\vec{u}$ by an update according to ψ. The multi-set is needed as there may be several branches of the edge (l, a, ϕ, μ) that lead with the same likelihood to the next state $s' = (l', \vec{v})$. A transition $(s, a, \mu) \in \Delta$ is denoted $s \xrightarrow{a} \mu$.

Remark 2. Note that the induced PTS considered in this paper are all *finite state*, owing to the range D of the data variables being finite.

Paths and traces. The PA model thus incorporates both *non-determinism* and *probabilistic choice*, and a possible behaviour reflected in the corresponding PTS results from the resolution of non-deterministic and probabilistic choices, described in terms of paths. A *path* π of a PTS $[\mathcal{P}] = \langle S, s_0, \Sigma, \Delta \rangle$ is a (possibly infinite) sequence of the form $\pi = s_0 a_1 \mu_1 s_1 a_2 \mu_2 s_2 a_3 \mu_3 s_3 \ldots$ where $\forall n : s_n \xrightarrow{a_{n+1}} \mu_{n+1}$, and $\mu_{n+1}(s_{n+1}) > 0$. Let $last(\pi)$ denote the last state of π (if π is finite), $\pi(n)$ the n^{th} state of π, and $|\pi|$ the length (i.e., number of actions) of π. For a given PTS $[\mathcal{P}]$, let $Path^*([\mathcal{P}])$ be the set of all finite paths of $[\mathcal{P}]$, and $Path^\omega([\mathcal{P}])$ the set of all (possibly infinite) paths of $[\mathcal{P}]$. Also, we denote by $Path^n([\mathcal{P}])$ the set of paths $[\mathcal{P}]$ of length upto n. For a given path $\pi = s_0 a_1 \mu_1 s_1 a_2 \mu_2 s_2 a_3 \mu_3 s_3 \ldots$, the n^{th} transition is $s_{n-1} \xrightarrow{a_n} \mu_n$, and its *trace* is given by $trace(\pi) = a_1 a_2 a_3 \ldots$, obtained by omitting all states and distributions from π.

Definition 3 (Adversary). For a given PTS $[\mathcal{P}] = \langle S, s_0, \Sigma, \Delta \rangle$, an *adversary* $\mathcal{A}$ of $[\mathcal{P}]$ is a function $\mathcal{A} : Path^*([\mathcal{P}]) \mapsto (\Sigma_{?!} \times Dist(S)) \cup \{\bot\}$ that maps every finite path π of $[\mathcal{P}]$ to a pair (a, μ) or to $\bot$, such that if $\mathcal{A}(\pi) = (a, \mu)$ for some $a \in \Sigma_{?!}$ and $\mu \in Dist(S)$, then $(last(\pi), a, \mu) \in \Delta$ and if there is no $a \in \Sigma_{?!}$ and $\mu \in Dist(S)$ such that $(last(\pi), a, \mu) \in \Delta$ then $\mathcal{A}(\pi) = \bot$, where $\bot$ denotes a special action for termination.

Thus, an adversary resolves all non-deterministic choices of the PTS $[\mathcal{P}]$, so that under a given adversary $\mathcal{A}$ of $[\mathcal{P}]$, the behaviour of $\mathcal{P}$ is *purely probabilistic*. Let $Adv([\mathcal{P}])$ be the set of all adversaries of the PTS $[\mathcal{P}]$.

A path $\pi = s_0 a_1 \mu_1 s_1 a_2 \mu_2 s_2 \ldots \in Path^\omega([\mathcal{P}])$ under an adversary $\mathcal{A}$ is such that $\mathcal{A}(pref_n(\pi)) = (a_{n+1}, \mu_{n+1})$ and $\mu_{n+1}(s_{n+1}) > 0$ for all $0 \leq n \leq |\pi|$. Here, $pref_n(\pi)$ denotes the prefix of π of length n. We denote by $Path^*_{\mathcal{A}}([\mathcal{P}])$ (resp. $Path^\omega_{\mathcal{A}}([\mathcal{P}])$) the set of all finite (resp. possibly infinite) paths of the PTS $[\mathcal{P}]$ under a given adversary $\mathcal{A}$. Similarly, $Path^n_{\mathcal{A}}([\mathcal{P}])$ denotes the set of paths upto length n of $[\mathcal{P}]$ under $\mathcal{A}$. Note that states having the final location l_f of the PA $\mathcal{P}$ as their location component do not admit further actions (cf. Definition 1), so that $\forall \mathcal{A} \in Adv([\mathcal{P}]), \forall \pi \in Path^*_{\mathcal{A}}([\mathcal{P}]) : last(\pi) = (l_f, \vec{u})$ we have that $\mathcal{A}(\pi) = \bot$.

A path π is said to be *maximal* under a given adversary $\mathcal{A}$ if either π is an infinite path in $Path^\omega_{\mathcal{A}}([\mathcal{P}])$, or if π is a finite path in $Path^*_{\mathcal{A}}([\mathcal{P}])$ and $\mathcal{A}(\pi) = \bot$. That is to say, a maximal path under adversary $\mathcal{A}$ is a path that cannot be prolonged under $\mathcal{A}$. We denote by $Path^{max}_{\mathcal{A}}([\mathcal{P}])$ the set of all maximal paths in $[\mathcal{P}]$ under the adversary $\mathcal{A}$.

We now formulate a notion of *equivalence on adversaries* for comparing the behaviours of PA that are defined over the same alphabet, the same state space, and the same set of distributions. Such PA arise when considering the probabilistic communication closed equivalences that will be detailed in the next section.

Definition 4 (Equivalence $\equiv$ on adversaries). Given PA $\mathcal{P}_1$ and $\mathcal{P}_2$ defined over the same alphabet Σ, the same state space S, and the same set of distributions $Dist(S)$, with $\mathcal{A}_i \in Adv([\mathcal{P}_i])$. Then $\mathcal{A}_1 \equiv \mathcal{A}_2$ iff $\forall \pi_i \in Path^*_{\mathcal{A}_i}([\mathcal{P}_i])\ \exists \pi_{3-i} \in Path^*_{\mathcal{A}_{3-i}}([\mathcal{P}_{3-i}]) : |\pi_i| = |\pi_{3-i}| \wedge last(\pi_i) = last(\pi_{3-i}) \wedge \mathcal{A}_{3-i}(\pi_{3-i}) = \mathcal{A}_i(\pi_i)$ where $i \in \{1, 2\}$.

Thus, equivalent adversaries induce the same possible non-deterministic choices for all same-length finite paths with identical last-states. The *probability* that a given resolution of non-determinism —as specified by an adversary $\mathcal{A}$ of $[\mathcal{P}]$— results in a path $\pi \in Path^*([\mathcal{P}])$ is given by a function $\mathbf{Q}_{\mathcal{A}} : Path^*([\mathcal{P}]) \mapsto [0, 1]$. It is defined inductively as follows: $\mathbf{Q}_{\mathcal{A}}(s_0) = 1$ and if $\mathcal{A}(\pi) = (a, \mu)$ for some $a \in \Sigma_{?!}$ and $\mu \in Dist(S)$, then $\mathbf{Q}_{\mathcal{A}}(\pi a \mu s) = \mathbf{Q}_{\mathcal{A}}(\pi) \cdot \mu(s)$, and if $\mathcal{A}(\pi) = \bot$, then $\mathbf{Q}_{\mathcal{A}}(\pi \bot) = \mathbf{Q}_{\mathcal{A}}(\pi)$.

This probability is embedded within a *probability space* associated to the given adversary $\mathcal{A}$. Note that it is necessary to reason about the (probabilistic) behaviour of adversaries in terms of their respective *probability spaces* —the mere assignment of probabilities to paths via distributions does not suffice. [1]

Definition 5 (Probability space). A *probability space* $\langle \Omega, \mathcal{F}, \mathbf{P} \rangle$ consists of

- Ω, the *sample space*.
- $\mathcal{F} \subseteq 2^\Omega$, a *$\sigma$-field*, i.e., $\mathcal{F}$ contains Ω, and is closed under countable union and complementation.
- $\mathbf{P} : \mathcal{F} \mapsto [0, 1]$, a *probability measure* on $\mathcal{F}$, such that $\mathbf{P}(\Omega) = 1$, and $\mathbf{P}(\cup_i X_i) = \sum_i \mathbf{P}(X_i)$, where the X_i are pair-wise disjoint subsets of $\mathcal{F}$.

This leads to the notion of a probability space associated to an adversary $\mathcal{A}$, by considering for each finite path π generated by $\mathcal{A}$ the corresponding *cylinder* $cyl(\pi)$ containing all *maximal* paths with π as prefix.

[1] In fact, the comparison of two probabilities that were not defined in the same probability space resulted in an erroneous proof of correctness [Seg00] for the earliest randomized algorithm for mutual exclusion [Rab82].

Definition 6 (Probability space associated to an adversary). The *probability space* associated to an adversary $\mathcal{A}$ of a PTS $[\mathcal{P}]$ is $\langle \Omega_{\mathcal{A}}, \mathcal{F}_{\mathcal{A}}, \mathbf{P}_{\mathcal{A}} \rangle$, where

- $\Omega_{\mathcal{A}} = Path^{max}_{\mathcal{A}}([\mathcal{P}])$
- $\mathcal{F}_{\mathcal{A}}$ is the smallest σ-field containing the *cylinder sets* $\{\ cyl(\pi)\ \mid\ \pi \in Path^{*}_{\mathcal{A}}([\mathcal{P}])\}$, where $cyl(\pi) = \{\pi' \in \Omega_{\mathcal{A}}\ \mid\ \pi \text{ prefix of } \pi'\}$
- $\mathbf{P}_{\mathcal{A}}$ is the unique measure on $\mathcal{F}_{\mathcal{A}}$ such that $\mathbf{P}_{\mathcal{A}}(cyl(\pi)) = \mathbf{Q}_{\mathcal{A}}(\pi)$ for all $\pi \in Path^{*}_{\mathcal{A}}([\mathcal{P}])$.

Measure-theoretic arguments ensure that $\langle \Omega_{\mathcal{A}}, \mathcal{F}_{\mathcal{A}}, \mathbf{P}_{\mathcal{A}} \rangle$ is indeed a probability space. We are now positioned to characterize the semantics of a given PA $\mathcal{P}$ in terms of the *trace distribution* for some adversary $\mathcal{A}$ of its PTS $[\mathcal{P}]$. Such a trace distribution is obtained from the probability space associated to paths under the adversary $\mathcal{A}$ by omitting all states and distributions.

Definition 7 (Trace distribution). The *trace distribution* $\mathcal{T} = trdist_{\mathcal{A}}([\mathcal{P}])$ of an adversary $\mathcal{A}$ of a PTS $[\mathcal{P}] = \langle S, s_0, \Sigma, \Delta \rangle$ is the probability space $\langle \Omega_{\mathcal{T}}, \mathcal{F}_{\mathcal{T}}, \mathbf{P}_{\mathcal{T}} \rangle$, where

- $\Omega_{\mathcal{T}} = \Sigma^{*}_{?!} \cup \Sigma^{\omega}_{?!}$.
- $\mathcal{F}_{\mathcal{T}}$ is the smallest σ-field containing $\{cyl(w)\ \mid\ w \in \Sigma^{*}_{?!}\}$, where $cyl(w) = \{w' \in \Omega_{\mathcal{T}}\ \mid\ w \text{ prefix of } w'\}$.
- $\mathbf{P}_{\mathcal{T}}(X) = \mathbf{P}_{\mathcal{A}}(\{\pi \in Path^{max}_{\mathcal{A}}([\mathcal{P}])\ \mid\ trace(\pi) \in X\})$ for all $X \in \mathcal{F}_{\mathcal{T}}$.

The sample space of $trdist_{\mathcal{A}}([\mathcal{P}])$ are sets of finite and infinite action-sequences, referred to as traces. Measurable elements are sets of traces obtained via a standard cylinder construction. The measure of set X is the probability of the set of maximal paths of PTS $[\mathcal{P}]$ under adversary $\mathcal{A}$ yielding a trace in X. Measure-theoretic arguments ensure the well-definedness of the above probability space. We denote by $trdist([\mathcal{P}])$ the set of the trace distributions of the PTS $[\mathcal{P}]$ under all possible adversarial resolutions.

Thus: $trdist([\mathcal{P}]) = \{\, trdist_{\mathcal{A}}([\mathcal{P}]) \mid \mathcal{A} \in Adv([\mathcal{P}]) \,\}$.

Definition 8 (Trace distribution equivalence). $\mathcal{P}_1 \equiv_{TD} \mathcal{P}_2$ iff $trdist([\mathcal{P}_1]) = trdist([\mathcal{P}_2])$.

This *trace distribution equivalence* $\equiv_{TD}$ enables the comparison of the behaviours of PA via the following proposition that gives a sufficient condition relating $\equiv_{TD}$ to the adversaries of the PA being compared.

Proposition 1 (From $\equiv$ on adversaries to $\equiv_{TD}$). Given PA $\mathcal{P}_1$ and $\mathcal{P}_2$ over the same alphabet Σ, the same state space S, and the same set of distributions $Dist(S)$. If $\forall \mathcal{A}_i \in Adv([\mathcal{P}_i])\ \exists \mathcal{A}_{3-i} \in Adv([\mathcal{P}_{3-i}]) : \mathcal{A}_i \equiv \mathcal{A}_{3-i}$ (where $i \in \{1, 2\}$), then $\mathcal{P}_1 \equiv_{TD} \mathcal{P}_2$.

Proof. The PA $\mathcal{P}_1$ and $\mathcal{P}_2$ have the same alphabet Σ, the same state space S, and the same set of distributions $Dist(S)$, where each of Σ, S, and $Dist(S)$ is *finite*, and thus the induced PTS $[\mathcal{P}_1]$ and $[\mathcal{P}_2]$ are also finite (cf. Definition 1 and Remark 2). This finiteness rules out the existence of *non-cyclic infinite paths* in both $Path^{max}_{\mathcal{A}_1}([\mathcal{P}_1])$ and $Path^{max}_{\mathcal{A}_2}([\mathcal{P}_2])$, which might otherwise arise owing to König's Lemma [Koe36] for infinite graphs. Thus, reasoning about maximal paths in $Path^{max}_{\mathcal{A}_1}([\mathcal{P}_1])$ and $Path^{max}_{\mathcal{A}_2}([\mathcal{P}_2])$ reduces to reasoning about their finite prefixes of arbitrary length in $Path^{*}_{\mathcal{A}_1}([\mathcal{P}_1])$ and $Path^{*}_{\mathcal{A}_2}([\mathcal{P}_2])$, owing to all infinite paths in $Path^{max}_{\mathcal{A}_1}([\mathcal{P}_1])$ and $Path^{max}_{\mathcal{A}_2}([\mathcal{P}_2])$ being necessarily *cyclic*. In particular, two equivalent adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ –with $\mathcal{A}_1 \in Adv([\mathcal{P}_1])$ and $\mathcal{A}_2 \in Adv([\mathcal{P}_2])$– will induce the same set of maximal paths, i.e., $Path^{max}_{\mathcal{A}_1}([\mathcal{P}_1]) = Path^{max}_{\mathcal{A}_2}([\mathcal{P}_2])$ whenever $\mathcal{A}_1 \equiv \mathcal{A}_2$. Now, the trace distribution of a PTS for a given adversary is uniquely specified by the channel alphabet and the set of maximal paths under that adversary (Definition 7). Equivalent adversaries will therefore result in equal trace distributions, i.e., $\mathcal{A}_1 \equiv \mathcal{A}_2 \Rightarrow trdist_{\mathcal{A}_1}([\mathcal{P}_1]) = trdist_{\mathcal{A}_2}([\mathcal{P}_2])$. Further, we have that $\forall \mathcal{A}_i \in Adv([\mathcal{P}_i])\ \exists \mathcal{A}_{3-i} \in Adv([\mathcal{P}_{3-i}]) : \mathcal{A}_i \equiv \mathcal{A}_{3-i}$ (where $i \in \{1, 2\}$), thus entailing $trdist([\mathcal{P}_1]) = trdist([\mathcal{P}_2])$. By Definition 8, this means $\mathcal{P}_1 \equiv_{TD} \mathcal{P}_2$. □

Composing PA. We have thus far considered the semantics of PA that operate in isolation. In practice, however, PA need to be able to *communicate* with each other in order to effectively model the inter-component interactions within a randomized distributed system. We now define sequential, parallel, and choice composition of PA (denoted ;, ‖, and +, respectively) for assembling PA into a composite system.

Definition 9 (Sequential composition). Given PA $\mathcal{P}_i = \langle L_i, l_{0_i}, l_{f_i}, V_i, \Sigma_i, prob_i \rangle$, where $i \in \{1, 2\}$ with $L_1 \cap L_2 = \emptyset$. Their *sequential composition*, denoted $\mathcal{P}_1; \mathcal{P}_2$, is the PA $\langle L, l_0, l_f, V_1 \cup V_2, \Sigma_1 \cup \Sigma_2, prob \rangle$, where $L = (L_1 \setminus \{l_{f_1}\}) \cup L_2$ with $l_0 = l_{0_1}, l_f = l_{f_2}$ and $prob = prob'_1 \cup prob_2$.

Here $prob_1' = prob_1[l_{0_2} \leftarrow l_{f_1}]$ is defined by $(l, a, \phi, \mu) \in prob_1$ iff $(l, a, \phi, \nu) \in prob_1'$ with

$$\begin{array}{rcll} \nu(\psi, l') & = & \mu(\psi, l') & \text{if } l' \neq l_{f_1}, \text{ and} \\ \nu(\psi, l_{0_2}) & = & \mu(\psi, l_{f_1}) & \text{otherwise.} \end{array}$$

The PA $\mathcal{P}_1; \mathcal{P}_2$ behaves first like $\mathcal{P}_1$ and subsequently like $\mathcal{P}_2$. To establish this, the final location l_{f_1} of $\mathcal{P}_1$ is amalgamated with the initial location l_{0_2} of $\mathcal{P}_2$. This is reflected in the construction of *prob*, where basically all edges to l_{f_1} are redirected to l_{0_2}. This construction models sequential composition, since the final location l_{f_1} of $\mathcal{P}_1$ has no outgoing edges, cf. Definition 1.

While sequential composition describes the evolution of one PA followed by that of another, parallel composition $\|$ captures the concurrent evolution of two PA. We adopt the CCS-style composition [Mil89], i.e., parallel PA *synchronize* on common actions and act autonomously on all other actions —the latter is modelled by *interleaving*. In order to avoid any read-write and write-write conflicts w.r.t. the shared variables in the parallel PA, we require that edges corresponding to synchronizing actions are *non-interfering*. This notion is defined as follows. Consider edge $e = (l, a, \phi, \mu)$. Let the *write-set* of e, denoted $wr(e)$, be the set of variables occurring on the left-hand side of an update ψ with $\mu(\psi, l') > 0$ for some l'. The *read-set* of edge e, denoted $rd(e)$, consists of all data variables that appear in the guard ϕ or on the right-hand side of an update ψ with $\mu(\psi, l') > 0$ for some l'.

Definition 10 (Non-interfering edges). Let E_1, E_2 be sets of edges with $e_1 \in E_1$ and $e_2 \in E_2$. The *non-interference relation* $\not\frown \subseteq E_1 \times E_2$ is defined by:

$$e_1 \not\frown e_2 \text{ iff } rd(e_1) \cap wr(e_2) = wr(e_1) \cap rd(e_2) = wr(e_1) \cap wr(e_2) = \emptyset.$$

Thus, two edges are non-interfering whenever it is excluded that some variable that may change in one edge is read (or may change) in the other edge. The relation $\not\frown$ is then canonically lifted to sets of edges: $E_1 \not\frown E_2$ iff for all $e_1 \in E_1$ and $e_2 \in E_2$ we have $e_1 \not\frown e_2$. In the following, let $edges(a) = \{(l, a, \phi, \mu) \in prob \mid \exists l \in L\}$ denote the set of a-labelled edges emanating from some location $l \in L$. Two PA are now called non-interfering if their *synchronized* edges are non-interfering with each other.

Definition 11 (Non-interfering PA). PA $\mathcal{P}_1$ and $\mathcal{P}_2$ over alphabet Σ_1 and Σ_2 respectively, are *non-interfering*, denoted $\mathcal{P}_1 \not\frown_{sync} \mathcal{P}_2$, if

$$\forall a : a \in \Sigma_{i?!} \ \wedge \ \overline{a} \in \Sigma_{3-i?!}, \quad \text{it holds that } edges_i(a) \not\frown edges_{3-i}(\overline{a}),$$

where $edges_i(a)$ is the set of a-edges in PA $\mathcal{P}_i$, for $i \in \{1, 2\}$.

Remark 3. The above notion of non-interference is only w.r.t *synchronizing* actions on common channels. Note that this does not preclude shared-variable dependencies between actions on disjoint channels.

We now define parallel composition for such non-interfering PA.

Definition 12 (Parallel composition). Let PA $\mathcal{P}_1$ and $\mathcal{P}_2$ with $\mathcal{P}_i = \langle L_i, l_{0i}, l_{f_i}, V_i, \Sigma_i, prob_i \rangle$ for $i \in \{1, 2\}$ with $\mathcal{P}_1 \not\frown_{sync} \mathcal{P}_2$ and $L_1 \cap L_2 = \emptyset$. The *parallel composition* of $\mathcal{P}_1$ and $\mathcal{P}_2$, denoted $\mathcal{P}_1 \| \mathcal{P}_2$, is the PA $\langle L, l_0, l_f, V_1 \cup V_2, \Sigma_1 \cup \Sigma_2, prob \rangle$, where

- $L = L_1 \times L_2$ with $l_0 = (l_{01}, l_{02})$ and $l_f = (l_{f_1}, l_{f_2})$.
- $((l_1, l_2), a, \phi, \mu) \in prob$ iff $(l_i, a_i, \phi_i, \mu_i) \in prob_i$, $i \in \{1, 2\}$ and either:
  1. *Synchronization*
     $a_i \in \Sigma_{i?!} \ \wedge \ \overline{a_i} \in \Sigma_{3-i?!}$, $a = \tau$, $\phi = \phi_1 \wedge \phi_2$, and $\mu(\psi, (l_1', l_2')) = \mu_1(\psi\lceil_{V_1}, l_1') \cdot \mu_2(\psi\lceil_{V_2}, l_2')$, or
  2. *Interleaving*
     $a_i \in \Sigma_{i?!}$, $a = a_i$, $\phi = \phi_i$, and $\mu(\psi, (l_1', l_2')) = \begin{cases} \mu_i(\psi\lceil_{V_i}, l_i') & \text{if } \psi\lceil_{V_{3-i}} = id_{V_{3-i}} \wedge l_{3-i}' = l_{3-i}, \\ 0 & \text{otherwise} \end{cases}$

where $\psi\lceil_{V_i}$ restricts ψ to the domain V_i, while id_{V_i} denotes the identity valuation over V_i, for $i \in \{1, 2\}$.

The parallel composition of $\mathcal{P}_1$ and $\mathcal{P}_2$ is thus the product of these PA, where the probability of synchronizing on a common channel is the product of the individual probabilities of performing the corresponding input/output actions in $\mathcal{P}_1$ and $\mathcal{P}_2$. Note that (as in CCS) we always allow each component to perform autonomous actions, where the other component idles with unit probability.

During the execution of randomized distributed algorithms, one often encounters non-deterministic selection between various components that are to be subsequently executed. Such non-deterministic selection is modelled by the operator + on PA.

Definition 13 (Non-deterministic choice). Let PA $\mathcal{P}_i = \langle L_i, l_{0i}, l_{f_i}, V_i, \Sigma_i, prob_i \rangle$, for $i \in \{1,2\}$ with $(L_1 \setminus \{l_{01}, l_{f_1}\}) \cap (L_2 \setminus \{l_{02}, l_{f_2}\}) = \emptyset$, $l_{01} = l_{02} = l_0$ and $l_{f_1} = l_{f_2} = l_f$. The *non-deterministic choice* of $\mathcal{P}_1$ and $\mathcal{P}_2$, denoted $\mathcal{P}_1 + \mathcal{P}_2$, is the PA $\langle L_1 \cup L_2, l_0, l_f, V_1 \cup V_2, \Sigma_1 \cup \Sigma_2, prob_1 \cup prob_2 \rangle$.

Note that in the above definition, we require the initial and final locations of the component PA to be identical, while imposing disjointness of all other locations.

The compositional constructs $\|$ and + are both symmetric, and do not impose the precedence of one component over the other based on the dependencies between the individual PA. In the next section, we will examine an asymmetric compositional operator and some relations that exploit such dependencies.

3. Action Independence, Precedence, Layering

A randomized distributed algorithm often consists of (sequential) phases that execute in parallel on different components, wherein a transition within a given phase can execute only after all *dependent* transitions in each preceding phase have been executed. In this section, we introduce the notion of *action independence* in PA and its corresponding PTS along the lines of action independence in Markov Decision Processes (MDPs) proposed in [BGC04, DN04]. This notion forms the basis for a *layered composition* operator, denoted $\bullet$, on PA that is intermediate between parallel and sequential composition. The $\bullet$ -operator is then used to formulate the communication-closed layer (CCL) laws in a probabilistic setting. For a PTS $[\mathcal{P}] = \langle S, s_0, \Sigma, \Delta \rangle$, let $act(s) = \{a \in \Sigma_{?!} \mid \exists \mu : (s, a, \mu) \in \Delta\}$ denote the set of enabled actions in state s.

Definition 14 (Action independence). Let $[\mathcal{P}] = \langle S, s_0, \Sigma, \Delta \rangle$ be a PTS. The actions $a, b \in \Sigma_{?!}$ are *independent* in $[\mathcal{P}]$, denoted $a \leftrightsquigarrow b$, iff for all states $s \in S$ with $a, b \in act(s)$ it holds that:

1. For any $s' \in S$: if $s \xrightarrow{a} \mu$ and $\mu(s') > 0$, then $b \in act(s')$.
2. For any $s' \in S$: if $s \xrightarrow{b} \nu$ and $\nu(s') > 0$, then $a \in act(s')$.
3. For any $s'' \in S$:

$$\sum \{\mu(s') \cdot \nu(s'') \mid s \xrightarrow{a} \mu \wedge s' \xrightarrow{b} \nu\} = \sum \{\mu(s') \cdot \nu(s'') \mid s \xrightarrow{b} \mu \wedge s' \xrightarrow{a} \nu\}.$$

Stated in words, actions a and b are independent whenever for every state s in which both actions are enabled, (1.) the occurrence of a does not disable b, (2.) and vice versa. Moreover, (3.) the total probability of reaching s'' from s by either performing a followed by b, or by performing b followed by a, coincides. Two distinct actions a and b are dependent, denoted $a \nleftrightsquigarrow b$, iff they are not independent. The relation $\leftrightsquigarrow$ is lifted to sets of actions in the standard manner. Notice that action independence is a semantic notion as it is defined on the underlying PTS $[\mathcal{P}]$ of the PA $\mathcal{P}$. The following proposition shows that the non-interference relation $\not\frown$, which can be determined by a simple syntactic analysis of $\mathcal{P}$, is a sufficient condition for action independence. Let $a \not\frown b$ whenever $edges(a) \not\frown edges(b)$.

Proposition 2 (Sufficient condition for action independence). Let PA $\mathcal{P}_1$ and $\mathcal{P}_2$ be over the alphabets Σ_1 and Σ_2, respectively. Then for $a_i \in \Sigma_{i?!}$, $i \in \{i, 2\}$, $a_1 \not\frown a_2$ implies $a_1 \leftrightsquigarrow a_2$ in $[\mathcal{P}_1 \| \mathcal{P}_2]$.

We may now define the *layered composition*, denoted $\bullet$, of two PA. The operational interpretation of $\mathcal{P}_1 \bullet \mathcal{P}_2$ is that it behaves like the parallel composition of $\mathcal{P}_1$ and $\mathcal{P}_2$ except that an action a in $\mathcal{P}_2$ can only occur if all the actions in $\mathcal{P}_1$ on which a depends (in the sense of $\nleftrightsquigarrow$) have already occurred. Stated differently, dependent actions in $\mathcal{P}_1$ and $\mathcal{P}_2$ are treated as in the sequential composition $\mathcal{P}_1; \mathcal{P}_2$ whereas independent actions are handled as in interleaving. For location l in PA $\mathcal{P}$, let $l \xrightarrow{*} l'$ denote that location l' is *syntactically reachable* from l through an arbitrary finite sequence of edges. Let $act(l) = \{a \in \Sigma_{?!} \mid (l, a, \phi, \mu)$ is an edge in $\mathcal{P}\}$ denote the set of enabled actions in location l.

Definition 15 (Layered composition). Given two PA $\mathcal{P}_i = \langle L_i, l_{0i}, l_{f_i}, V_i, \Sigma_i, prob_i \rangle$, where $i \in \{1,2\}$, with $L_1 \cap L_2 = \emptyset$ and $\mathcal{P}_1 \not\frown_{sync} \mathcal{P}_2$. The *layered composition* of $\mathcal{P}_1$ and $\mathcal{P}_2$, denoted $\mathcal{P}_1 \bullet \mathcal{P}_2$ is the PA $\langle L, l_0, l_f, V_1 \cup V_2, \Sigma_1 \cup \Sigma_2, prob \rangle$, where

- $L = L_1 \times L_2$ with $l_0 = (l_{01}, l_{02})$ and $l_f = (l_{f_1}, l_{f_2})$.
- $((l_1, l_2), a, \phi, \mu) \in prob$ iff $(l_i, a_i, \phi_i, \mu_i) \in prob_i$, $i \in \{1, 2\}$ and either:
  1. *Synchronization*
     $a_i \in \Sigma_{i?!} \ \wedge \ \overline{a_i} \in \Sigma_{3-i?!}$, $a = \tau$, $\phi = \phi_1 \wedge \phi_2$, and $\mu(\psi, (l_1', l_2')) = \mu_1(\psi\lceil_{V_1}, l_1') \cdot \mu_2(\psi\lceil_{V_2}, l_2')$, or
  2. *Interleaving* $\mathcal{P}_1$
     $a = a_1 \in \Sigma_{1?!}$, $\phi = \phi_1$, and $\mu(\psi, (l_1', l_2')) = \begin{cases} \mu_1(\psi\lceil_{V_1}, l_1') & \text{if } \psi\lceil_{V_2} = id_{V_2} \wedge l_2' = l_2, \\ 0 & \text{otherwise} \end{cases}$, or
  3. *Interleaving* $\mathcal{P}_2$
     $a = a_2 \in \Sigma_{2?!}$, $\phi = \phi_2$, and $\mu(\psi, (l_1', l_2')) = \begin{cases} \mu_2(\psi\lceil_{V_2}, l_2') & \text{if } \psi\lceil_{V_1} = id_{V_1} \wedge l_1' = l_1 \wedge \\ & \quad \forall l_1^* : l_1 \xrightarrow{*} l_1^* : act(l_1^*) \leftrightarrows a \text{ in } [\mathcal{P}_1 \| \mathcal{P}_2], \\ 0 & \text{otherwise.} \end{cases}$

In the above definition, the first two clauses are the same as for synchronization and interleaving in parallel composition, whereas the third clause restricts the autonomous execution of actions by $\mathcal{P}_2$ to actions that are ensured to be independent of those in $\mathcal{P}_1$. PA $\mathcal{P}_1$ and $\mathcal{P}_2$ are *independent*, denoted $\mathcal{P}_1 \leftrightarrows \mathcal{P}_2$, iff for all $a_1 \in \Sigma_{1?!}$ and for all $a_2 \in \Sigma_{2?!}$ it holds that $a_1 \leftrightarrows a_2$ in the PTS $[\mathcal{P}_1 \| \mathcal{P}_2]$. Otherwise, $\mathcal{P}_1$ and $\mathcal{P}_2$ are dependent, denoted $\mathcal{P}_1 \leftrightsquigarrow \mathcal{P}_2$. In the presence of a shared variable dependence between $\mathcal{P}_1$ and $\mathcal{P}_2$, a related notion is that of *precedence* for PA, defined as follows:

Definition 16 (Precedence $\prec$). For PA $\mathcal{P}_1$ and $\mathcal{P}_2$, $\mathcal{P}_1 \prec \mathcal{P}_2$ iff $\mathcal{P}_1 \| \mathcal{P}_2 \equiv_{TD} \mathcal{P}_1; \mathcal{P}_2$.

The relation $\prec$ is transitive and enforces a precedence of $\mathcal{P}_1$ over $\mathcal{P}_2$ by requiring that $\mathcal{P}_1$ and $\mathcal{P}_2$ do not synchronize on common channels, and that $trdist([\mathcal{P}_2; \mathcal{P}_1]) = \emptyset$, which is ensured at the semantic level by appropriate guards querying the data variables shared between $\mathcal{P}_1$ and $\mathcal{P}_2$. This is illustrated in the analysis of the randomized mutual exclusion algorithm in Section 5. The following proposition establishes the behaviour of the operator $+$ w.r.t. the independence $\leftrightarrows$ and precedence $\prec$ relations for PA, and follows quite straightforwardly from the definitions.

Proposition 3 (Relating $+$ with $\leftrightarrows$ and $\prec$). For PA $\mathcal{P}_1$ and $\mathcal{P}_2$ with $\mathcal{P}_1 = \mathcal{R}_1 + \mathcal{U}_1$ and $\mathcal{P}_2 = \mathcal{R}_2 + \mathcal{U}_2$,

- $\mathcal{R}_1 \leftrightarrows \mathcal{R}_2 \ \wedge \ \mathcal{R}_1 \leftrightarrows \mathcal{U}_2 \ \wedge \ \mathcal{U}_1 \leftrightarrows \mathcal{R}_2 \ \wedge \ \mathcal{U}_1 \leftrightarrows \mathcal{U}_2$ if and only if $\mathcal{P}_1 \leftrightarrows \mathcal{P}_2$, and
- $\mathcal{R}_1 \prec \mathcal{R}_2 \ \wedge \ \mathcal{R}_1 \prec \mathcal{U}_2 \ \wedge \ \mathcal{U}_1 \prec \mathcal{R}_2 \ \wedge \ \mathcal{U}_1 \prec \mathcal{U}_2$ if and only if $\mathcal{P}_1 \prec \mathcal{P}_2$.

We then use layered composition and appropriate *independence* and *precedence* side-conditions for formulating the following communication closed layer (CCL) equivalences for PA.

Theorem 1 (CCL laws for PA). For PA $\mathcal{P}_1, \mathcal{P}_2, \mathcal{Q}_1$, and $\mathcal{Q}_2$, with ($\mathcal{P}_1 \leftrightarrows \mathcal{Q}_2$ or $\mathcal{P}_1 \prec \mathcal{Q}_2$) and ($\mathcal{Q}_1 \leftrightarrows \mathcal{P}_2$ or $\mathcal{Q}_1 \prec \mathcal{P}_2$), the following *communication closed layer* (CCL) equivalences hold:

1. $\mathcal{P}_1 \bullet \mathcal{Q}_2 \ \equiv_{TD} \ \mathcal{P}_1 \| \mathcal{Q}_2$ *(IND)*
2. $(\mathcal{P}_1 \bullet \mathcal{P}_2) \| \mathcal{Q}_2 \ \equiv_{TD} \ \mathcal{P}_1 \bullet (\mathcal{P}_2 \| \mathcal{Q}_2)$ *(CCL-L)*
3. $(\mathcal{P}_1 \bullet \mathcal{P}_2) \| \mathcal{Q}_1 \ \equiv_{TD} \ (\mathcal{P}_1 \| \mathcal{Q}_1) \bullet \mathcal{P}_2$ *(CCL-R)*
4. $(\mathcal{P}_1 \bullet \mathcal{P}_2) \| (\mathcal{Q}_1 \bullet \mathcal{Q}_2) \ \equiv_{TD} \ (\mathcal{P}_1 \| \mathcal{Q}_1) \bullet (\mathcal{P}_2 \| \mathcal{Q}_2)$ *(CCL)*

Proof. We sketch a proof of the law CCL-L. The proofs of the other CCL laws are similar. Given PA $\mathcal{P}_1, \mathcal{P}_2, \mathcal{Q}_2$, with $\mathcal{P}_1 \leftrightarrows \mathcal{Q}_2$ or $\mathcal{P}_1 \prec \mathcal{Q}_2$, we aim to show $\mathcal{R} \equiv_{TD} \mathcal{U}$, where $\mathcal{R} = (\mathcal{P}_1 \bullet \mathcal{P}_2) \| \mathcal{Q}_2$ and $\mathcal{U} = \mathcal{P}_1 \bullet (\mathcal{P}_2 \| \mathcal{Q}_2)$. Note that $\mathcal{R}$ and $\mathcal{U}$ are both defined over the same channel alphabet Σ, state space S, and set of distributions $Dist(S)$, each of which is finite. It therefore suffices by Proposition 1 to show that $\forall \mathcal{A} \in Adv([\mathcal{R}]) \ \exists \mathcal{A}' \in Adv([\mathcal{U}]) : \mathcal{A} \equiv \mathcal{A}'$, and vice versa. We show such adversarial equivalence by induction on path lengths n for finite paths in $Path^*_{\mathcal{A}}([\mathcal{R}])$ and $Path^*_{\mathcal{A}'}([\mathcal{U}])$. The existence of an equivalent adversary $\mathcal{A}$ in $Adv([\mathcal{R}])$ for each adversary $\mathcal{A}'$ in $Adv([\mathcal{U}])$ is not hard to see intuitively, as the parallel composition operator $\|$ dominates in $\mathcal{R}$ and the layered composition operator $\bullet$ in $\mathcal{U}$. The dominance of $\bullet$ induces fewer interleavings in $[\mathcal{U}]$ on the basis of the respective dependencies or precedences.

We now show: $\forall \mathcal{A} \in Adv([\mathcal{R}]) \ \exists \mathcal{A}' \in Adv([\mathcal{U}]) : \mathcal{A} \equiv \mathcal{A}'$. By Definition 4, this amounts to showing: $\forall \mathcal{A} \in Adv([\mathcal{R}]) \ \forall \pi \in Path^*_{\mathcal{A}}([\mathcal{R}]) \ \exists \mathcal{A}' \in Adv([\mathcal{U}]) \ \exists \pi' \in Path^*_{\mathcal{A}'}([\mathcal{U}])$: $|\pi| = |\pi'| \ \wedge \ last(\pi) = last(\pi') \ \wedge \ \mathcal{A}'(\pi') = \mathcal{A}(\pi)$.

For an adversary $\mathcal{A} \in Adv([\mathcal{R}])$ we proceed by induction on the length n of a path π generated by $\mathcal{A}$.

Induction Basis. For the case $n = 0$, we have $\pi = s_0$, where $s_0 = ((l_{0P1}, l_{0P2}, l_{0Q2}), \vec{v_0})$ [2] is the initial state of both $\mathcal{R}$ and $\mathcal{U}$. Thus we take $\pi' = \pi = s_0$ and choose $\mathcal{A}' = \mathcal{A}$, as $\mathcal{R}$ and $\mathcal{U}$ have the same alphabet Σ, the same state-space S, and the same set of distributions $Dist(S)$.

Induction Step. Consider a path $\pi_{n+1} \in Path^{n+1}_{\mathcal{A}}([\mathcal{R}])$ (of length $n+1$) with $pref_n(\pi_{n+1}) = \pi_n$ such that $last(\pi_n) = s_n = ((l_{P1}, l_{P2}, l_{Q2}), \vec{u})$ and $\mathcal{A}(\pi_n) = (a_n, \mu_n)$ for $a_n \in \Sigma_{?!}$ and $\mu_n \in Dist(S)$. This choice (a_n, μ_n) could have been performed in $\mathcal{R}$ by either (a1) $\mathcal{P}_1$, (a2) $\mathcal{P}_2$, (a3) $\mathcal{Q}_2$ individually, or as a synchronization involving either (b1) $\mathcal{P}_2$ and $\mathcal{Q}_2$, (b2) $\mathcal{P}_1$ and $\mathcal{Q}_2$, or (b3) $\mathcal{P}_1$ and $\mathcal{P}_2$.

The cases (a1) and (a2) are relatively straightforward. We now consider case (a3). This means that there exists an edge $e = (l_{Q2}, a_n, \phi, \mu_n)$ in $\mathcal{Q}_2$ with $\mu_n(\psi, l'_{Q2}) > 0$ for some $l'_{Q2} \in L_{Q2}$ and $\vec{u} \models \phi$, leading to a state $s_{n+1} = ((l_{P1}, l_{P2}, l'_{Q2}), \vec{u}')$ where $\vec{u}' = \psi(\vec{u})$. From the induction hypothesis, there exists $\mathcal{A}' \in Adv([\mathcal{U}])$ resulting in a path $\pi'_n \in Path^n_{\mathcal{A}'}([\mathcal{U}])$ with $last(\pi'_n) = s_n$ and $\mathcal{A}'(\pi_n) = (a_n, \mu_n)$. As either $\mathcal{P}_1 \leftrightsquigarrow \mathcal{Q}_2$ or $\mathcal{P}_1 \prec \mathcal{Q}_2$, this necessarily means the existence of the same edge e in $\mathcal{U}$ leading to the same state s_{n+1}. We have thus shown that $last(\pi_{n+1}) = last(\pi'_{n+1}) = s_{n+1}$ where π'_{n+1} is a path in $Path^{n+1}_{\mathcal{A}'}([\mathcal{U}])$ obtained by continuing from π'_n along the adversarial choice (a_n, μ_n) of $\mathcal{A}'$. Now let $\mathcal{A}(\pi_{n+1}) = (a_{n+1}, \mu_{n+1})$. Then the possibilities for the choice (a_{n+1}, μ_{n+1}) are again as in (a1)–(a3) and (b1)–(b3). Thus for the case (a3) (and also straightforwardly for (a1) and (a2)), using again the fact that either $\mathcal{P}_1 \leftrightsquigarrow \mathcal{Q}_2$ or $\mathcal{P}_1 \prec \mathcal{Q}_2$, we see that $(last(\pi'_{n+1}), a_{n+1}, \mu_{n+1}) \in \Delta_{\mathcal{U}}$, and thus take $\mathcal{A}'(\pi'_{n+1}) = (a_{n+1}, \mu_{n+1})$.

On the other hand, when $s_n = ((l_{f\,P1}, l_{f\,P2}, l_{f\,Q2}), \vec{u})$, we have $\mathcal{A}(\pi_n) = \mathcal{A}'(\pi'_n) = \bot$.

In (b1) there are edges with complementary actions enabled in $\mathcal{P}_2$ and $\mathcal{Q}_2$ individually that can synchronize to an edge labeled with τ in the context of $\mathcal{P}_2 \| \mathcal{Q}_2$. The part of the τ-edge stemming from $\mathcal{P}_2$ was possible in $\mathcal{P}_1 \bullet \mathcal{P}_2$, so it is possible also in $\mathcal{U} = \mathcal{P}_1 \bullet (\mathcal{P}_2 \| \mathcal{Q}_2)$. When $\mathcal{P}_1 \leftrightsquigarrow \mathcal{Q}_2$ or $\mathcal{P}_1 \prec \mathcal{Q}_2$, the part of the τ-edge stemming from $\mathcal{Q}_2$ is also enabled in $\mathcal{U}$, and when executed in $\mathcal{U}$ yields the same state as when executed in $\mathcal{R} = (\mathcal{P}_1 \bullet \mathcal{P}_2) \| \mathcal{Q}_2$, with the same possible further adversarial choices.

The case (b2) reduces to (a1) and (a3), and the case (b3) reduces to (a1) and (a2). □

Remark 4. Note that each of the equivalences of Theorem 1 in fact relates *isomorphic* PA, as expressed by the equivalence $\equiv$ on adversaries at each transition step.

The CCL laws (generalized to an arbitrary number of processes) together with the po-equivalences of the next section will be used in the analysis of the randomized mutex algorithm in Section 5.

4. Partial Order Equivalence and Linear Time Properties

We introduce in this section *partial order equivalence* ($\equiv^*_{po}$) for PA, and show that two po-equivalent PA satisfy the same probabilistic stutter-invariant (i.e., next-free) linear-time properties. As we are interested in comparing layered and sequential compositions of PA w.r.t $\equiv^*_{po}$, we first need to eliminate paths having τ-edges in the layered composition that have resulted from the synchronization of complementary actions.

For PA $\mathcal{P}_1$ and $\mathcal{P}_1$, let $\mathcal{P} = \mathcal{P}_1 \bullet \mathcal{P}_2$. Let $Path^*_{\setminus\{\tau\}}([\mathcal{P}])$ denote the set of finite paths of $\mathcal{P}$ with no τ-labelled transitions, i.e., $Path^*_{\setminus\{\tau\}}([\mathcal{P}]) = \{\pi \in Path^*([\mathcal{P}]) \mid \pi = s_0 a_1 \mu_1 s_1 a_2 \mu_2 s_3 \cdots \ \wedge \ \forall i : a_i \neq \tau\}$. Similarly, we denote by $Path^*{}_{\mathcal{A}\setminus\{\tau\}}([\mathcal{P}])$ the set of all finite paths without τ-edges under some adversary $\mathcal{A} \in Adv([\mathcal{P}])$. We then have the following proposition that relates the (probabilistic) behaviour of paths of $\mathcal{P}$ with those that do not contain τ-labelled edges.

Proposition 4 (Ignoring paths with τ-edges). For PA $\mathcal{P}_1$ and $\mathcal{P}_1$, let $\mathcal{P} = \mathcal{P}_1 \bullet \mathcal{P}_2$. Then we have $\forall \mathcal{A} \in Adv([\mathcal{P}]) \ \forall \pi \in Path^*_{\mathcal{A}}([\mathcal{P}]) : \exists \mathcal{A}_i \in Adv([\mathcal{P}]) \ \exists \pi_i \in Path^*{}_{\mathcal{A}i\setminus\{\tau\}}([\mathcal{P}]) : \mathbf{P}_{\mathcal{A}}(cyl(\pi)) = \mathbf{P}_{\mathcal{A}_i}(cyl(\pi_i))$, where $i \in \{1, 2\}$.

Proof. Let $\pi \in Path^*_{\mathcal{A}}([\mathcal{P}])$ be a finite path (under some adversary $\mathcal{A}$ of the layered composition $\mathcal{P}$) containing a τ-labelled edge. Then π is of the form $\pi = \pi' \tau \mu \pi''$, where π' and π'' are finite path-fragments. Then from Definition 15 and from our stipulation that τ-edges may only result from synchronization along complementary actions, there necessarily exist adversaries $\mathcal{A}_i$ and corresponding finite paths $\pi_i \in Path^*_{\mathcal{A}i\setminus\{\tau\}}([\mathcal{P}])$ such that $\pi_1 = \pi' a \mu' s_1 \overline{a} \mu'' \pi''$ and $\pi_2 = \pi' \overline{a} \mu'' s_2 a \mu' \pi''$, where $a \in \Sigma_{i?!}$, $\overline{a} \in \Sigma_{3-i?!}$ ($i \in \{1, 2\}$). Owing

[2] $l_{0\,P1}$ is the initial location of $\mathcal{P}_1$, etc. We identify nested pairs $((x, y), z)$ and $(x, (y, z))$ of locations with tuples (x, y, z).

to the layered composition $\bullet$ admitting a CCS-style interleaving on all actions, and owing to the non-interference condition imposed on complementary actions (cf. Definition 15), we have that $\mu'(s_1)\cdot\mu''(\pi''(0)) = \mu''(s_2)\cdot\mu'(\pi''(0)) = \mu(\pi''(0))$, where $\pi''(0)$ denotes the starting state of the path-fragment π''. As in the proof of Proposition 1, reasoning about the path probabilities along π, π_1 and π_2 may then be extended to reasoning about their possibly infinite continuations via the cylinder constructions, thus yielding $\mathbf{P}_{\mathcal{A}}(cyl(\pi)) = \mathbf{P}_{\mathcal{A}_1}(cyl(\pi_1)) = \mathbf{P}_{\mathcal{A}_2}(cyl(\pi_2))$. $\square$

Note that π_1 and π_2 appearing in (the proof of) Proposition 4 differ only in the permutative ordering of the independent transitions corresponding to a and $\overline{a}$, and they both preserve the validity of properties that are insensitive to superfluous intermediate states (such as s_1 and s_2). For such (stutter invariant, or next-free) properties, it therefore suffices to consider paths in $Path^*_{\setminus\{\tau\}}([\mathcal{P}])$. For relating paths in $Path^*_{\setminus\{\tau\}}([\mathcal{P}])$ with those of sequential composition, we introduce the partial order equivalence $\equiv^*_{po}$ on (finite paths of) PA.

Definition 17 (po-equivalence $\equiv^*_{po}$ on finite paths). For PA $\mathcal{P}_1$ and $\mathcal{P}_2$, let $\pi_1 \in Path^*_{\setminus\{\tau\}}([\mathcal{P}_1])$ and $\pi_2 \in Path^*_{\setminus\{\tau\}}([\mathcal{P}_2])$. Then $\pi_1 \equiv_{po} \pi_2$ iff there exist finite path fragments π, π' such that $\pi_1 = \pi a \mu s_1 b \mu' \pi'$ and $\pi_2 = \pi b \mu' s_2 a \mu \pi'$, where $a \leftrightsquigarrow b$. The *partial order equivalence* $\equiv^*_{po}$ on finite paths without τ-labelled edges is then the reflexive, transitive closure of $\equiv_{po}$.

Thus two paths related via $\equiv^*_{po}$ may be obtained from each other by repeated permutation of adjacent independent actions, modulo some superfluous intermediate states (such as s_1 and s_2 in the above definition). Note that by the independence $a \leftrightsquigarrow b$, we have $\mu(s_1)\cdot\mu'(\pi'(0)) = \mu'(s_2)\cdot\mu(\pi'(0))$. We then introduce a notion of *layered normal form* to relate via $\equiv^*_{po}$ the τ-eliminated paths of a layered composition.

Definition 18 (Paths of $\bullet$ in LNF). Given two PA $\mathcal{P}_i = \langle L_i, l_{0i}, l_{f_i}, V_i, \Sigma_i, prob_i\rangle$, where $i \in \{1,2\}$, with $L_1 \cap L_2 = \emptyset$ and $\mathcal{P}_1 \not\leftarrow_{sync} \mathcal{P}_2$. Then $\pi \in Path^*_{\setminus\{\tau\}}([\mathcal{P}_1 \bullet \mathcal{P}_2])$ is said to be in *layered normal form* (LNF) iff $\pi = s_0 a_1 \mu_1 s_1 ....$ such that $[\ \forall n : a_n \in \Sigma_{1?!} \ \wedge\ loc(s_n) = (l, l_{02})$ for some $l \in L_1\ ] \vee$
$[\ \exists n : loc(s_n) = (l_{f_1}, l_{02}) \ \wedge\ \forall m \leq n : a_m \in \Sigma_{1?!} \ \wedge\ \forall m > n : a_m \in \Sigma_{2?!}\ ]$.

Here, $loc(s_n)$ is the location component of the n^{th} state s_n of π. Thus, a path (in the PTS) of a layered composition $\mathcal{P}_1 \bullet \mathcal{P}_2$ is in LNF iff it first consecutively executes actions of $\mathcal{P}_1$ (until $\mathcal{P}_1$ has terminated in l_{f_1}), and thereafter consecutively executes actions of $\mathcal{P}_2$. We denote by $Path^*_{LNF}([\mathcal{P}_1 \bullet \mathcal{P}_2])$ the set of all paths of $[\mathcal{P}_1 \bullet \mathcal{P}_2]$ that are in LNF. The definition of the layered composition $\bullet$ permits the execution of $\mathcal{P}_2$ actions only after the execution of all dependent $\mathcal{P}_1$ actions, thus leading to the following proposition.

Proposition 5 (Relating paths in LNF via $\equiv^*_{po}$). For PA $\mathcal{P}_1$ and $\mathcal{P}_2$, the following holds:
$\forall \pi \in Path^*_{\setminus\{\tau\}}([\mathcal{P}_1 \bullet \mathcal{P}_2])\ \exists \pi' \in Path^*_{LNF}([\mathcal{P}_1 \bullet \mathcal{P}_2]) : \pi \equiv^*_{po} \pi'$.

Proof. This proposition follows from the definition of $\bullet$ and the construction of paths in LNF, given that an action of $\mathcal{P}_2$ is allowed to execute in $\mathcal{P}_1 \bullet \mathcal{P}_2$ only after all dependent actions of $\mathcal{P}_1$ have been executed, and any action of $\mathcal{P}_2$ in a path of $\mathcal{P}_1 \bullet \mathcal{P}_2$ is thus necessarily independent of all $\mathcal{P}_1$ actions it precedes, and may therefore be permuted repeatedly to yield a path of $\mathcal{P}_1 \bullet \mathcal{P}_2$ in LNF. $\square$

We now lift the notion of $\equiv^*_{po}$ from paths to PA.

Definition 19 ($\equiv^*_{po}$ for PA). For PA $\mathcal{P}_1$ and $\mathcal{P}_2$, we write $\mathcal{P}_1 \equiv^*_{po} \mathcal{P}_2$ iff the following holds for $i \in \{1,2\}$
$\forall \mathcal{A}_i \in Adv([\mathcal{P}_i])\ \forall \pi_i \in Path^*_{\mathcal{A}i\setminus\{\tau\}}([\mathcal{P}_i])\ \exists \mathcal{A}_{3-i} \in Adv([\mathcal{P}_{3-i}])\ \exists \pi_{3-i} \in Path^*_{\mathcal{A}3-i\setminus\{\tau\}}([\mathcal{P}_{3-i}]) : \pi_i \equiv^*_{po} \pi_{3-i}$

Remark 5. Proposition 4 ensures that it is sufficient for $\equiv^*_{po}$ to consider paths in $Path^*_{\mathcal{A}i\setminus\{\tau\}}([\mathcal{P}_i])$. As in Proposition 4, we have for the po-equivalent paths π_1 and π_2 above: $\mathbf{P}_{\mathcal{A}_1}(cyl(\pi_1)) = \mathbf{P}_{\mathcal{A}_2}(cyl(\pi_2))$.

A consequence of Definition 19 and Proposition 5 is that $\equiv^*_{po}$ holds between the compositions $\bullet$ and ; .

Theorem 2 ($\equiv^*_{po}$ between $\bullet$ and ;). For PA $\mathcal{P}_1$ and $\mathcal{P}_2$, $\mathcal{P}_1 \bullet \mathcal{P}_2 \ \equiv^*_{po}\ \mathcal{P}_1; \mathcal{P}_2$.

Proof. Let $\pi \in Path^*_{LNF}([\mathcal{P}_1 \bullet \mathcal{P}_2])$. Then by Definition 18, there exists a path $\pi' \in Path^*([\mathcal{P}_1; \mathcal{P}_2])$ such that $\forall n \geq 0 : loc(\pi(n)) \approx loc(\pi'(n)) \ \wedge\ val(\pi(n)) = val(\pi'(n)) \ \wedge\ \pi(n) \xrightarrow{a_{n+1}} \mu_{n+1} \ \Rightarrow\ \pi'(n) \xrightarrow{a_{n+1}} \mu_{n+1} \ \wedge\ \mu_{n+1}(\pi(n+1)) = \mu_{n+1}(\pi'(n+1))$, where $val(\pi(n))$ is the vector of data valuations in the n^{th} state $\pi(n)$ of path π, and$\approx$ is a relation defined as follows on the location spaces $L_1 \times L_2$ of $\mathcal{P}_1 \bullet \mathcal{P}_2$ and $(L_1 \cup L_2) \setminus \{l_{f_1}\}$

of $\mathcal{P}_1;\mathcal{P}_2 : \forall l_1 \in L_1$ with $l_1 \neq l_{f_1} : (l_1, l_{02}) \approx l_1$ and $\forall l_2 \in L_2 : (l_{f_1}, l_2) \approx l_2$. Every path of $\mathcal{P}_1 \bullet \mathcal{P}_2$ in LNF is therefore exactly mimicked by a path of $\mathcal{P}_1;\mathcal{P}_2$ (modulo the relation $\approx$ on the location spaces of $\mathcal{P}_1 \bullet \mathcal{P}_2$ and $\mathcal{P}_1;\mathcal{P}_2$), and vice-versa. We also have from Proposition 5 that every finite (τ-eliminated) path of $\mathcal{P}_1 \bullet \mathcal{P}_2$ is either in LNF, or is po-equivalent to another path in LNF. □

As the PA related by Theorem 1 are isomorphic, with the equivalence $\equiv$ on adversaries being stronger than $\equiv^*_{po}$, we then have the following corollary from Theorems 1 and 2.

Corollary 1. Replacement of $\bullet$ by ; in the CCL laws yields po-equivalences.

We now illustrate the probabilistic preservation of stutter invariant properties expressible in the Linear Temporal Logic (LTL) without the next operator (denoted LTL \ X) by the partial order equivalence $\equiv^*_{po}$.

Theorem 3 (Preservation by $\equiv^*_{po}$ of LTL \ X). Given $\mathcal{P}_1 \equiv^*_{po} \mathcal{P}_2$ and a formula φ in LTL \ X. Then the following holds: $\forall \mathcal{A}_i \in Adv([\mathcal{P}_i])\ \exists \mathcal{A}_{3-i} \in Adv([\mathcal{P}_{3-i}]) : \mathbf{P}_{\mathcal{A}_i}([\varphi]) = \mathbf{P}_{\mathcal{A}_{3-i}}([\varphi])$, where $\mathbf{P}_{\mathcal{A}_i}([\varphi]) = \mathbf{P}_{\mathcal{A}i}(\{\pi \in Path^{max}_{\mathcal{A}_i}([\mathcal{P}_i]) \ \mid \ \pi \models \varphi\})$, and $i \in \{1, 2\}$.

Proof. Consider two paths π'_1 and π'_2 with $\pi'_i \in Path^*_{\mathcal{A}_i \setminus \{\tau\}}([\mathcal{P}_i])$, where $i \in \{1, 2\}$, such that $\pi'_1 \equiv^*_{po} \pi'_2$. As π'_1 and π'_2 differ only in the permutative ordering of independent actions and in some superfluous intermediate states, π'_1 and π'_2 are *stutter equivalent* [BGC04, DN04] in the sense that any stutter invariant (i.e., next free) LTL formula φ (where the LTL atomic propositions correspond to assertions on the data valuations) will be satisfied by π'_1 iff it is satisfied by π'_2. As in the proof of Proposition 1, it suffices here to reason about arbitrary length finite prefixes in $Path^*_{\mathcal{A}_i \setminus \{\tau\}}([\mathcal{P}_i])$ of maximal paths in $Path^{max}_{\mathcal{A}_i}([\mathcal{P}_i])$ owing to finiteness of the induced PTS. Thus, all possibly infinite continuations of π'_1 and π'_2 via the cylinder constructions will be necessarily cyclic, and will induce the same probability measure via the respective adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$. The theorem then follows from Definition 19 and Remark 5. □

Methodology for layered separation. We conclude this section with an outline of the intended methodology for layered separation, illustrated on a schematic example. Suppose we want to verify for PA $\mathcal{P}_1, \mathcal{P}_2, \mathcal{Q}_1$ that the system $(\mathcal{P}_1;\mathcal{P}_2) \| \mathcal{Q}_1$ satisfies with a probability threshold p a formula φ expressible in LTL \ X. If $\mathcal{Q}_1 \rightsquigarrow \mathcal{P}_2$ or $\mathcal{Q}_1 \prec \mathcal{P}_2$ we can reduce the state space of the PA system by applying to it the equivalences $\equiv_{TD}$ and $\equiv^*_{po}$, as shown below.

$$
\begin{array}{ll}
 & (\mathcal{P}_1;\mathcal{P}_2) \| \mathcal{Q}_1 \\
\equiv^*_{po} & \quad \{ \text{ Corollary 1 } \} \\
 & (\mathcal{P}_1 \bullet \mathcal{P}_2) \| \mathcal{Q}_1 \\
\equiv_{TD} & \quad \{ \text{ CCL-R } \} \\
 & (\mathcal{P}_1 \| \mathcal{Q}_1) \bullet \mathcal{P}_2 \\
\equiv^*_{po} & \quad \{ \text{ Corollary 1 } \} \\
 & (\mathcal{P}_1 \| \mathcal{Q}_1); \mathcal{P}_2.
\end{array}
$$

From Theorem 3, it then suffices to check that $(\mathcal{P}_1 \| \mathcal{Q}_1); \mathcal{P}_2$ satisfies φ with the probability threshold p. If each of $\mathcal{P}_1, \mathcal{P}_2, \mathcal{Q}_1$ has 10 locations then –from the definitions of the compositional constructs $\|$, $\bullet$, and ;– we see that the original PA system has 190 locations while the transformed system via layered separation (using $\equiv_{TD}$ and $\equiv^*_{po}$) has 109 locations. These reductions in the number of discrete locations to be explored become more pronounced as the number of system components increases, as will be illustrated next.

5. Case Study : Randomized Mutual Exclusion

To demonstrate the benefit of the layered separation achieved by means of the CCL laws and the associated po-equivalences, we consider the (revised) randomized mutual exclusion algorithm $MUTEX$ for $N \geq 2$ processes by Kushilevitz and Rabin [KR92]. The authors in [KR92] describe their algorithm informally, partly in the running text and partly by pieces of pseudo code. This makes it difficult to obtain a complete picture of the algorithm. We present here (our view of) the essential ingredients of the algorithm in terms of PA composed sequentially, nondeterministically and in parallel. Our presentation is influenced by the algebraic analysis in the work of McIver, Gonzalia, Cohen, and Morgan [MGCM08] of parts of the algorithm in terms

of separation theorems expressed in a probabilistic Kleene Algebra. We however analyze the algorithm of [KR92] in this section by means of layered separation.

The algorithm proceeds in rounds. To avoid keeping a record of the unbounded round numbers, the algorithm uses a randomized round number chosen by the process entering the critical section. During a round, every process $\mathcal{P}_i$ with $i \in \{1, \ldots, N\}$ cycles through four phases: *drawing* or *voting* $\mathcal{V}_i$, *notification* or *testing* $\mathcal{T}_i$, *critical section* $\mathcal{C}_i$, and *remainder* $\mathcal{R}_i$. In the voting phase, the processes participate in a lottery where they use a geometric distribution to pick numbers in the set $\{1, \ldots, B\}$, where $B = log_2 N + 4$. The value k is drawn with probability $1/2^k$ for $k \leq B-1$, and with probability $1/2^{k-1}$ for $k = B$. The process that has drawn the highest number in the lottery will be notified as the winner in the subsequent notification round and will next enter the critical section. It stays thereafter in its remainder phase until it decides to participate in another voting phase.

The algorithm distinguishes between *even* and *odd* rounds, with different pseudo-code for voting, notification, and critical section. Between an access to the critical section in an even round (called even critical section) and the access to the critical section in the subsequent odd round (called odd critical section) both an odd notification phase (for entering the odd critical section) and an even voting phase (for the next even critical section) will occur, and vice versa, with the roles of even and odd exchanged.

The processes $\mathcal{P}_1, \ldots, \mathcal{P}_N$ share several variables: *b_even* and *b_odd* (maximal lottery value in even and odd rounds, respectively), *r_even* and *r_odd* (random round number bit in even and odd rounds, respectively), s (binary semaphore guarding the critical section), p (parity bit serving as guards of even and odd phases), and w (indicator that the winner of lottery is notified). Whereas *b_even*, *b_odd* range over $\{0, \ldots, B\}$, the variables s, p, w range over $\{0, 1\}$, and *r_even*, *r_odd* range over $\{nil, 0, 1\}$. Additionally, each process $\mathcal{P}_i$ manipulates some local variables: $b(i)$ (lottery value drawn by $\mathcal{P}_i$), $d(i)$ (difference contributed by $\mathcal{P}_i$ to the maximum lottery value), $r(i)$ (round number), $w(i)$ (indicator that $\mathcal{P}_i$ knows it has won the lottery), and $pc(i)$ (program counter). Here $b(i)$ and $d(i)$ range over $\{0, \ldots, B\}$, $r(i)$ ranges over $\{nil, 0, 1\}$, $w(i)$ over $\{0, 1\}$, and $pc(i)$ over $\{nil, rem, it\}$.

In Fig. 1–4 we show for process $\mathcal{P}_i$ the PA representing the even phases $\mathcal{V}_i, \mathcal{T}_i, \mathcal{C}_i$ (and explain how PA for the odd phases are obtained) as well as PA representing $\mathcal{R}_i$ and idling versions of voting and notification. In these PA, all edges are labelled uniquely by corresponding actions, which we omit in the graphic representation. Synchronization between the processes $\mathcal{P}_1, \ldots, \mathcal{P}_N$ does not take place via these actions, but via guards checking the values of the shared variables.

The algorithm is modeled by the following parallel composition $MUTEX$, where each process $\mathcal{P}_i$ is a loop (represented by $*$) built up from sequences of nondeterministic choices of the phases defined above:

$$MUTEX = (\mathcal{P}_1 \| \ldots \| \mathcal{P}_N) \quad \text{with} \quad P_i = \left(\begin{array}{c} odd\ \mathcal{V}_i + \mathcal{IV}_i + even\ \mathcal{C}_i \\ ; \\ odd\ \mathcal{T}_i + \mathcal{IT}_i + \mathcal{R}_i \\ ; \\ even\ \mathcal{V}_i + \mathcal{IV}_i + odd\ \mathcal{C}_i \\ ; \\ even\ \mathcal{T}_i + \mathcal{IT}_i + \mathcal{R}_i \end{array} \right)^*$$

The algorithm starts with the following initial values: $b\_even = b\_odd = 0$, $r\_even = r\_odd = nil$, $s = 0$, $p = 1$, $w = 1$, and for all $i \in \{1, \ldots, N\}$ we assume $b(i) = 0$, $d(i) = 0$, $r(i) = 0$, $pc(i) = nil$. Furthermore, we stipulate w.l.o.g that $w(1) = 1$ and $w(i) = 0$ for $i \in \{2, \ldots, N\}$.

The nondeterministic choices in $\mathcal{P}_i$ are restricted by the following sequencing constraints enforced via the local program counter $pc(i)$: once idle voting $\mathcal{IV}_i$ is chosen, only idle notification $\mathcal{IT}_i$ can follow due to the guard $pc(i) = it$; once the critical section $\mathcal{C}_i$ is chosen, only the remainder $\mathcal{R}_i$ can follow due to the guard $pc(i) = rem$. Once voting $\mathcal{V}_i$ is chosen and it ends with a contribution $d(i) > 0$, only notification $\mathcal{T}_i$ can follow. If $\mathcal{V}_i$ ends with $d(i) = 0$, only idle notification $\mathcal{IT}_i$ can follow.

Transformation into a layered representation. We now show that $MUTEX$ can be transformed into a layered representation. Since the CCL laws stated in Section 3 do not involve the $*$-operator on PA, we argue by considering finite, initial unfoldings of the $*$ and establish the layering for these unfoldings. Inferring a layered representation of the loops from a layered representation of the unfoldings is correct if the processes

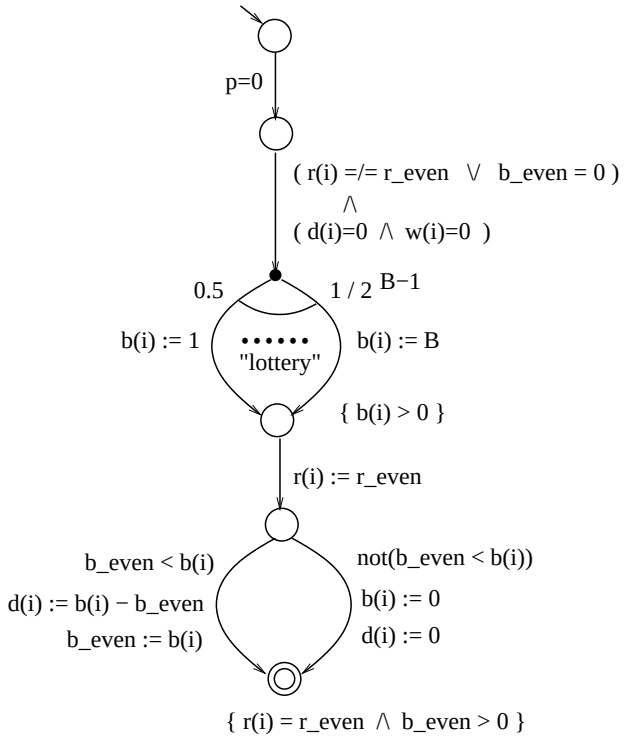

Figure 1. The PA for *even* voting $\mathcal{V}_i$ with guard $p = 0$. In the PA for *odd* voting this guard is replaced by $p = 1$, and *r_even* and *b_even* are replaced by *r_odd* and *b_odd*, respectively. Note that the lottery has B branches with the last two branches having the same probability $1/2^{B-1}$.

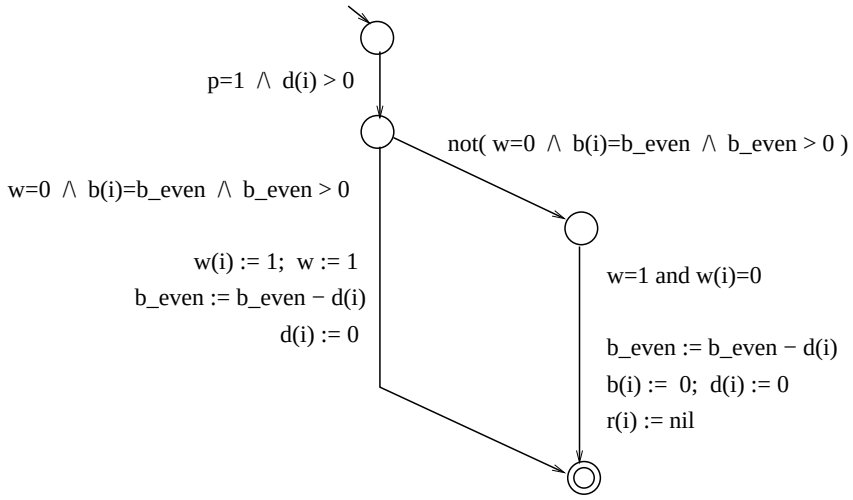

Figure 2. The PA for *even* notification $\mathcal{T}_i$. Note that the guard is $p = 1 \wedge d(i) > 0$. In the PA for *odd* notification this guard is replaced by $p = 0 \wedge d(i) > 0$ and *b_even* by *b_odd*. The right branch is taken when $\mathcal{P}_i$ is not the winner, where it waits in the intermediate state until notified of the winner $\mathcal{P}_j$ via an update $w := 1$ in the left branch of $\mathcal{T}_j$.

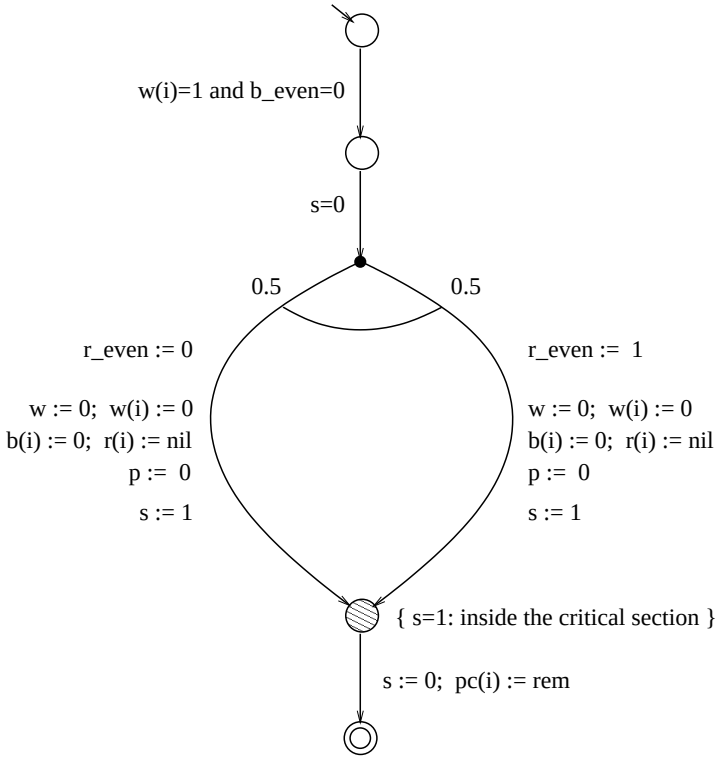

Figure 3. The PA for *even* critical section $\mathcal{C}_i$. In the PA for *odd* critical section the update $p := 0$ is replaced by $p := 1$, and *r_even* by *r_odd*, and *b_even* by *b_odd*.

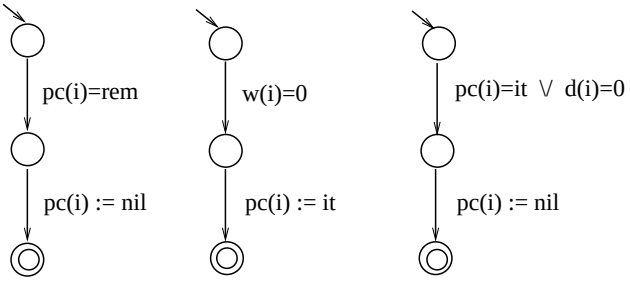

Figure 4. The PA for remainder $\mathcal{R}_i$ (left), idle voting $\mathcal{IV}_i$ (middle), and idle notification $\mathcal{IT}_i$ (right).

$\mathcal{P}_i$ of $MUTEX$ are prevented from pursuing infinite executions of idle voting and idle notification by a *fairness* assumption.

In Table 1 we display the unfolded $MUTEX$ up to round 4. To refer uniquely to different instances of the phases of each process $\mathcal{P}_i$ in the unfolding we attach a *round number* $r \in \mathbb{N}$ to them and thus write $\mathcal{V}_i^r$, $\mathcal{T}_i^r$, $\mathcal{C}_i^r$, $\mathcal{R}_i^r$, $\mathcal{IV}_i^r$, and $\mathcal{IT}_i^r$. The instances are defined as follows: $\mathcal{V}_i^r = \mathit{even}\ \mathcal{V}_i$ if r is even and $\mathcal{V}_i^r = \mathit{odd}\ \mathcal{V}_i$ if r is odd, and analogously for $\mathcal{T}_i^r$ and $\mathcal{C}_i^r$. Furthermore, for all $r \in \mathbb{N}$, $\mathcal{R}_i^r = \mathcal{R}_i$, $\mathcal{IV}_i^r = \mathcal{IV}_i$, and $\mathcal{IT}_i^r = \mathcal{IT}_i$. The initial unfolding up to round 4 is presented by the parallel composition in Table 1, with one sequential thread for each of the processes $\mathcal{P}_1, \ldots, \mathcal{P}_N$. The nondeterminism represented by $+$ is resolved during execution so that in each round r there is exactly one $i \in \{1, \ldots, N\}$ such that the critical section $\mathcal{C}_i^r$ is entered (thus guaranteeing mutual exclusion). Exactly for that i the remainder $\mathcal{R}_i^r$ will follow. All the other processes $j \neq i$ are already in their next round $r+1$ performing voting $\mathcal{V}_j^{r+1}$ and then notification $\mathcal{T}_j^{r+1}$. For a proper

$$
\begin{array}{rlll}
\text{odd } (r=1): & \{p=1\}\mathcal{V}_1^1 + \mathcal{IV}_1^1 + \mathcal{C}_1^0 & \mathcal{V}_2^1 + \mathcal{IV}_2^1 + \mathcal{C}_2^0 & \mathcal{V}_N^1 + \mathcal{IV}_N^1 + \mathcal{C}_N^0 \\
& \quad ; & \quad ; & \quad ; \\
& \{p=0\}\mathcal{T}_1^1 + \mathcal{IT}_1^1 + \mathcal{R}_1^0 & \mathcal{T}_2^1 + \mathcal{IT}_2^1 + \mathcal{R}_2^0 & \mathcal{T}_N^1 + \mathcal{IT}_N^1 + \mathcal{R}_N^0 \\
& \quad ; & \quad ; & \quad ; \\
\text{even } (r=2): & \{p=0\}\mathcal{V}_1^2 + \mathcal{IV}_1^2 + \mathcal{C}_1^1 & \mathcal{V}_2^2 + \mathcal{IV}_2^2 + \mathcal{C}_2^1 & \mathcal{V}_N^2 + \mathcal{IV}_N^2 + \mathcal{C}_N^1 \\
& \quad ; & \quad ; & \quad ; \\
& \{p=1\}\mathcal{T}_1^2 + \mathcal{IT}_1^2 + \mathcal{R}_1^1 & \mathcal{T}_2^2 + \mathcal{IT}_2^2 + \mathcal{R}_2^1 & \mathcal{T}_N^2 + \mathcal{IT}_N^2 + \mathcal{R}_N^1 \\
& \quad ; & \quad ; & \quad ; \\
\text{odd } (r=3): & \{p=1\}\mathcal{V}_1^3 + \mathcal{IV}_1^3 + \mathcal{C}_1^2 & \mathcal{V}_2^3 + \mathcal{IV}_2^3 + \mathcal{C}_2^2 & \mathcal{V}_N^3 + \mathcal{IV}_N^3 + \mathcal{C}_N^2 \\
& \quad ; & \quad ; & \quad ; \\
& \{p=0\}\mathcal{T}_1^3 + \mathcal{IT}_1^3 + \mathcal{R}_1^2 & \mathcal{T}_2^3 + \mathcal{IT}_2^3 + \mathcal{R}_2^2 & \mathcal{T}_N^3 + \mathcal{IT}_N^3 + \mathcal{R}_N^2 \\
& \quad ; & \quad ; & \quad ; \\
\text{even } (r=4): & \{p=0\}\mathcal{V}_1^4 + \mathcal{IV}_1^4 + \mathcal{C}_1^3 & \mathcal{V}_2^4 + \mathcal{IV}_2^4 + \mathcal{C}_2^3 & \mathcal{V}_N^4 + \mathcal{IV}_N^4 + \mathcal{C}_N^3 \\
& \quad ; & \quad ; & \quad ; \\
& \{p=1\}\mathcal{T}_1^4 + \mathcal{IT}_1^4 + \mathcal{R}_1^3 & \mathcal{T}_2^4 + \mathcal{IT}_2^4 + \mathcal{R}_2^3 & \mathcal{T}_N^4 + \mathcal{IT}_N^4 + \mathcal{R}_N^3
\end{array}
$$

Table 1. Unfolding of the parallel $MUTEX$ algorithm up to round 4. For clarity, we have indicated the round numbers r and as assertions the p-values required in the guards of voting $\mathcal{V}_1^r$ and notification $\mathcal{T}_1^r$ of process $\mathcal{P}_1$ in each round. The voting and notification phases $\mathcal{V}_i^r$ and $\mathcal{T}_i^r$ of the other processes $\mathcal{P}_i$ have the same guards. Note that $+$ binds stronger than $;$ and $\|$, and that $;$ binds stronger than $\|$.

initialization of the algorithm we stipulate an abridged round 0 in which one process, say $\mathcal{P}_1$, enters its critical section $\mathcal{C}_1^0$ without previous voting and testing.

We now establish crucial independence and precedence relations between the phases.

Proposition 6. The phases of the unfolded $MUTEX$ satisfy the following independence and precedence conditions, where $r, r1, r2 > 0$ and $i, j \in \{1, \ldots, N\}$.

1. Remainder $\mathcal{R}_i^r$, idle voting $\mathcal{IV}_i^r$ and idle notification $\mathcal{IT}_i^r$ are *independent* ($\leftrightsquigarrow$) of any other phase.
2. If one of the rounds $r1$ and $r2$ is even and the other odd, voting $\mathcal{V}_i^{r1}$ and notification $\mathcal{T}_i^{r1}$ are *independent* of both voting $\mathcal{V}_j^{r2}$ and notification $\mathcal{T}_j^{r2}$, so $\mathcal{V}_i^{r1} \leftrightsquigarrow \mathcal{V}_j^{r2}$, $\mathcal{V}_i^{r1} \leftrightsquigarrow \mathcal{T}_j^{r2}$, $\mathcal{T}_i^{r1} \leftrightsquigarrow \mathcal{V}_j^{r2}$, and $\mathcal{T}_i^{r1} \leftrightsquigarrow \mathcal{T}_j^{r2}$.
3. A critical section $\mathcal{C}_i^r$ can only start if notification $\mathcal{T}_i^r$ has been completed, which in turn can only start if voting $\mathcal{V}_i^r$ has been completed: $\mathcal{V}_i^r \prec \mathcal{T}_i^r$ and $\mathcal{T}_i^r \prec \mathcal{C}_i^r$.
4. A critical section $\mathcal{C}_i^r$ can only start if all processes j that contributed $d(j) > 0$ to *b_even* or *b_odd*, respectively, in their voting $\mathcal{V}_j^r$ have completed their notification $\mathcal{T}_j^r$, so $\mathcal{T}_j^r \prec \mathcal{C}_i^r$ for all these j.
5. Notification $\mathcal{T}_j^{r+1}$ and voting $\mathcal{V}_j^{r+2}$ can only start if some critical section $\mathcal{C}_i^r$ has been completed: $\mathcal{C}_i^r \prec \mathcal{T}_j^{r+1}$ and $\mathcal{C}_i^r \prec \mathcal{V}_j^{r+2}$.
6. The critical section $\mathcal{C}_1^0$ and all votings $\mathcal{V}_j^1$ with $j \in \{2, \ldots, N\}$ are not preceded by any other phase and can thus start immediately.

Proof. Claim 1. This is clear because $\mathcal{R}_i^r$, $\mathcal{IV}_i^r$, $\mathcal{IT}_i^r$ do not access any of the shared variables.

Claim 2. Let r_1 be even and r_2 be odd. Of the shared variables, voting and notification both read p, and moreover, even voting $\mathcal{V}_i^{r1}$ accesses *r_even*, *b_even*, and even notification $\mathcal{T}_i^{r1}$ accesses *b_even* and w, whereas odd voting $\mathcal{V}_j^{r2}$ accesses *r_odd*, *b_odd*, and odd notification $\mathcal{T}_j^{r2}$ accesses *b_odd* and w. However, both $\mathcal{T}_i^{r1}$ and $\mathcal{T}_j^{r2}$ have the same effect on the shared variable w, namely updating w to 1 (via the left branch on Figure 2) in case the corresponding process has won the lottery in the preceding voting phase. So even voting $\mathcal{V}_i^{r1}$ and notification $\mathcal{T}_i^{r1}$ are independent of odd voting $\mathcal{V}_j^{r2}$ and notification $\mathcal{T}_j^{r2}$. Similar arguments apply when r_1 is odd and r_2 is even.

Claim 3. An even critical section $\mathcal{C}_i^r$ can only start if in round r process i left its notification $\mathcal{T}_i^r$ with $w(i) = 1$, which in turn is only possible if in the prior voting phase $\mathcal{V}_i^r$ process i drew the highest number $b(i)$ yielding $b(i) = b\_even$. The corresponding precedence relation holds for an odd critical section.

Claim 4. An even critical section $\mathcal{C}_i^r$ can only start if $b\_even = 0$ holds, which is only possible if in round r all processes j that increased *b_even* by contributing $d(j) > 0$ to it in their voting phase $\mathcal{V}_j^r$ have deducted the value $d(j)$ again from *b_even* in their notification phase $\mathcal{T}_j^r$. Thus all these processes j must have completed

$$
\begin{array}{c}
(\mathcal{V}_1^1 + \mathcal{IV}_1^1 + \mathcal{C}_1^0 \parallel \mathcal{V}_2^1 + \mathcal{IV}_2^1 + \mathcal{C}_2^0 \parallel \cdots \parallel \mathcal{V}_N^1 + \mathcal{IV}_N^1 + \mathcal{C}_N^0) \\
; \\
(\mathcal{T}_1^1 + \mathcal{IT}_1^1 + \mathcal{R}_1^0 \parallel \mathcal{T}_2^1 + \mathcal{IT}_2^1 + \mathcal{R}_2^0 \parallel \cdots \parallel \mathcal{T}_N^1 + \mathcal{IT}_N^1 + \mathcal{R}_N^0) \\
; \\
(\mathcal{V}_1^2 + \mathcal{IV}_1^2 + \mathcal{C}_1^1 \parallel \mathcal{V}_2^2 + \mathcal{IV}_2^2 + \mathcal{C}_2^1 \parallel \cdots \parallel \mathcal{V}_N^2 + \mathcal{IV}_N^2 + \mathcal{C}_N^1) \\
; \\
(\mathcal{T}_1^2 + \mathcal{IT}_1^2 + \mathcal{R}_1^1 \parallel \mathcal{T}_2^2 + \mathcal{IT}_2^2 + \mathcal{R}_2^1 \parallel \cdots \parallel \mathcal{T}_N^2 + \mathcal{IT}_N^2 + \mathcal{R}_N^1) \\
; \\
(\mathcal{V}_1^3 + \mathcal{IV}_1^3 + \mathcal{C}_1^2 \parallel \mathcal{V}_2^3 + \mathcal{IV}_2^3 + \mathcal{C}_2^2 \parallel \cdots \parallel \mathcal{V}_N^3 + \mathcal{IV}_N^3 + \mathcal{C}_N^2) \\
; \\
(\mathcal{T}_1^3 + \mathcal{IT}_1^3 + \mathcal{R}_1^2 \parallel \mathcal{T}_2^3 + \mathcal{IT}_2^3 + \mathcal{R}_2^2 \parallel \cdots \parallel \mathcal{T}_N^3 + \mathcal{IT}_N^3 + \mathcal{R}_N^2) \\
; \\
(\mathcal{V}_1^4 + \mathcal{IV}_1^4 + \mathcal{C}_1^3 \parallel \mathcal{V}_2^4 + \mathcal{IV}_2^4 + \mathcal{C}_2^3 \parallel \cdots \parallel \mathcal{V}_N^4 + \mathcal{IV}_N^4 + \mathcal{C}_N^3) \\
; \\
(\mathcal{T}_1^4 + \mathcal{IT}_1^4 + \mathcal{R}_1^3 \parallel \mathcal{T}_2^4 + \mathcal{IT}_2^4 + \mathcal{R}_2^3 \parallel \cdots \parallel \mathcal{T}_N^4 + \mathcal{IT}_N^4 + \mathcal{R}_N^3)
\end{array}
$$

Table 2. Unfolding of the layered $MUTEX$ algorithm up to round 4.

their notification $\mathcal{T}_j^r$ before $\mathcal{C}_i^r$ can start. The corresponding precedence relation holds for an odd critical section.

Claim 5. Every even notification $\mathcal{T}_j^{r+1}$ in round $r+1$ and every odd voting $\mathcal{V}_j^{r+2}$ in round $r+2$ is guarded by $p = 1$, which can only be established by the update $p := 1$ of a previous odd critical section $\mathcal{C}_i^r$ in round r. Analogously, every odd notification $\mathcal{T}_j^{r+1}$ and every even voting $\mathcal{V}_j^{r+2}$ is guarded by $p = 0$, which can only be established by the update $p := 0$ of a previous even critical section $\mathcal{C}_i^r$.

Claim 6. Initially, we assume $w(1) = 1$ and $b\_even = 0$, so $\mathcal{C}_1^0$ can start immediately. Since initially also $p = 0$ and $w(j) = 0$, $d(j) = 0$ holds for all $j \in \{2, \ldots, N\}$, voting $\mathcal{V}_j^1$ can start for these j. □

We are now prepared for the layered restructuring.

Proposition 7. The unfolding of the parallel $MUTEX$ algorithm shown in Table 1 is po-equivalent ($\equiv_{po}^*$) to the unfolding of the layered $MUTEX$ algorithm shown in Table 2.

Proof. We check that the claims of Proposition 6 imply the necessary independence and precedence relations between phases in different rounds and processes so that Theorem 1 and Corollary 1 can be applied. Since by Claim 1, the phases for remainder, idle voting, and idle notification are independent of any other phase, it suffices to analyze voting, notification, and critical section. Of these we pick phases ph_j^r with a high round number r and argue that phases with a round number lower than r are either independent of ph_j^r or precede ph_j^r.

Case 1. Consider a voting $\mathcal{V}_j^4$ in round 4. Then in round 3, we have $\mathcal{V}_j^4 \nleftrightarrow \mathcal{T}_i^3$ and $\mathcal{V}_j^4 \nleftrightarrow \mathcal{V}_i^3$ by Claim 2. In round 2, we have $\mathcal{C}_i^2 \prec \mathcal{V}_j^4$ for the critical section of one process i due to Claim 5. Moreover, by Claim 4, we have $\mathcal{T}_k^2 \prec \mathcal{C}_i^2$ for all processes k that contributed $d(k) > 0$ to the maximum lottery value in their voting $\mathcal{V}_k^2$ before $\mathcal{T}_k^2$. All other processes l either abstained from voting before $\mathcal{C}_i^2$, which we can simulate by an independent idle voting $\mathcal{IV}_l^2$, or their voting $\mathcal{V}_l^2$ occurred before $\mathcal{C}_i^2$ but yielded only $d(l) = 0$. In both cases only an independent idle notification $\mathcal{IT}_l^2$ can follow. By Claim 5, any notification $\mathcal{T}_k^2$ in round 2 must be preceded by one critical section in round 1, say $\mathcal{C}_m^1 \prec \mathcal{T}_k^2$. Similar arguments apply when comparing phases in further lower numbered rounds with $\mathcal{V}_j^4$.

Case 2. Consider a notification $\mathcal{T}_j^3$ in round 3. Then in round 2, we have $\mathcal{C}_i^2 \prec \mathcal{T}_j^3$ for the critical section of one process i due to Claim 5. Note that the votings $\mathcal{V}_l^3$ of round 3 must have occurred before $\mathcal{C}_i^2$ as this changes the value of p. After $\mathcal{C}_i^2$ only idle voting $\mathcal{IV}_l^3$ of round 3 remain possible. Furthermore, in round 2 we have $\mathcal{T}_j^3 \nleftrightarrow \mathcal{V}_k^2$ and $\mathcal{T}_j^3 \nleftrightarrow \mathcal{T}_k^2$ due to Claim 2. By Claim 5, any notification $\mathcal{T}_k^2$ in round 2 must be preceded by one critical section in round 1, say $\mathcal{C}_m^1 \prec \mathcal{T}_k^2$. For $\mathcal{C}_m^1$ we argue similar to Case 1, starting with $\mathcal{C}_i^2$ in round 2.

Case 3. Consider a critical section $\mathcal{C}_i^3$ in round 3. Again, we argue similar to Case 1, starting with $\mathcal{C}_i^2$ in round 2. □

Proposition 7 holds for unfoldings up to any round number. This enables us to conclude that (under the assumption of fairness) the parallel $MUTEX$ is po-equivalent ($\equiv^*_{po}$) to the following layered version:

$$layered\_MUTEX = \left(\begin{array}{c} (odd\ \mathcal{V}_1 + \mathcal{IV}_1 + even\ \mathcal{C}_1 \parallel \cdots \parallel odd\ \mathcal{V}_N + \mathcal{IV}_N + even\ \mathcal{C}_N) \\ ; \\ (odd\ \mathcal{T}_1 + \mathcal{IT}_1 + \mathcal{R}_1 \quad \parallel \cdots \parallel \quad odd\ \mathcal{T}_N + \mathcal{IT}_N + \mathcal{R}_N) \\ ; \\ (even\ \mathcal{V}_1 + \mathcal{IV}_1 + odd\ \mathcal{C}_1 \parallel \cdots \parallel even\ \mathcal{V}_N + \mathcal{IV}_N + odd\ \mathcal{C}_N) \\ ; \\ (even\ \mathcal{T}_1 + \mathcal{IT}_1 + \mathcal{R}_1 \quad \parallel \cdots \parallel \quad even\ \mathcal{T}_N + \mathcal{IT}_N + \mathcal{R}_N) \end{array} \right)^*$$

For this layered version, we establish the following reduction in the number of locations. The number of locations $|L_P|$ for any of the constituent PA $\mathcal{P}$ is taken from Fig. 1–4. For nondeterministic choice we calculate $|L_{P+Q}| = |L_P| + |L_Q| - 2$, for sequential composition $|L_{P;Q}| = |L_P| + |L_Q| - 1$, and for parallel composition $|L_{P\parallel Q}| = |L_P| \cdot |L_Q|$, due to the definitions of these operators in Section 2.

	number of locations in ...		
N	... $MUTEX$	... $layered\_MUTEX$	reduction factor
3	15,625	1,453	≥ 10
4	390,625	10,781	≥ 36
5	9,765,625	81,085	≥ 120

Such reductions will speed up model checking the algorithm. Since both representations are (modulo fairness in $MUTEX$) po-equivalent, by Theorem 3, they satisfy the same probabilistic next-free linear-time properties – for example, some of the properties verified or computed for the original (flawed) randomized mutual exclusion algorithm by Rabin [Rab82] in a case study of the model checker PRISM [KNP04]. [3]

6. Conclusion

This paper adopted the concept of communication-closed layers to probabilistic automata, a popular operational framework for the specification and verification of randomized distributed algorithms. The focus was on the theoretical underpinnings of incorporating layered separation into the framework of PA, for the analysis of complex, distributed compositions of PA. The application of such layered separation has been shown in the modelling and analysis of a randomized mutual exclusion algorithm by Kushilevitz and Rabin [KR92].

McIver, Gonzalia, Cohen, and Morgan in [MGCM08] distilled from the description of this algorithm *algebraic axioms* that enabled them to simplify its reasoning by separation laws established for probabilistic Kleene Algebras. We start operationally, formalizing parts of the mutual exclusion algorithm of [KR92] in terms of probabilistic automata and combine them by nondeterministic, sequential and parallel composition. Then we check whether suitable independence and precedence relations allow us to restructure parallel computations into layered ones, aiming (via a state-space reduction) at a simpler verification.

It is interesting to ask whether our formalization satisfies the four axioms of [MGCM08], which we cite here in their informal versions:

(1) Voting and notification commute.
(2) Notification occurs when the critical section is free.
(3) Voting occurs when the critical section is busy.
(4) It's more likely to lose, the later the vote.

Following Kushilevitz and Rabin, we distinguish even and odd rounds. This difference is not addressed in [MGCM08]. Regarding (1), we show in Proposition 6 the *independence* ($\leftrightsquigarrow$) of voting and notification, but only when voting is performed in an even round and notification in an odd round (or vice versa). Of

[3] see `http://www.prismmodelchecker.org/casestudies/rabin.php`

course, independence implies commutativity. If voting $\mathcal{V}_i^{r1}$ and notification $\mathcal{T}_j^{r2}$ both occur in even rounds $r1, r2$ they interfere by writing to the shared variable b_even. Interestingly, $\mathcal{V}_i^{r1}$ and $\mathcal{T}_j^{r2}$ still commute, but in a degenerate sense: since $\mathcal{V}_i^{r1}$ is guarded by $p = 0$ and $\mathcal{T}_j^{r}$ by $p = 1$, and p is not changed by $\mathcal{V}_i^{r1}$ or $\mathcal{T}_j^{r2}$, both $\mathcal{V}_i^{r1}; \mathcal{T}_j^{r2}$ and $\mathcal{T}_j^{r2}; \mathcal{V}_i^{r1}$ end in a deadlock. The analogous statement is true for odd rounds. So axiom (1) is indeed satisfied.

Regarding (2) and (3), the unfolding of the layered $MUTEX$ in Table 2 shows that notification $\mathcal{T}_i^{r}$ takes place when no process is in its critical section and voting $\mathcal{V}_i^{r}$ takes place when one process is in its critical section. So axioms (2) and (3) are also satisfied.

Regarding (4), consider a sequence of even votings (cf. Fig. 1) by different processes in a given even round r, say $\mathcal{V}_1^r; \ldots; \mathcal{V}_m^r$ for $m \leq N$. Then after $\mathcal{V}_1^r$, process 1 will be the (temporary) winner with some value k drawn with probability $1/2^k$ for $1 \leq k \leq B-1$, and with probability $1/2^{k-1}$ for $k = B$, where $B = log_2 N + 4$, according to the geometric distribution assumed in the lottery. For process 2 to outperform process 1, it must draw a value $l > k$, but this is only possible with a probability of $1/2^l < 1/2^k$, etc. The same argument is true for odd votings. So axiom (4) is satisfied.

The satisfaction of the axioms (1)–(4) thus enables the application of the algebraic analysis of [MGCM08] to our operational model of Kushilevitz and Rabin's randomized mutual exclusion algorithm [KR92] in terms of probabilistic automata.

Future work consists in exploiting layered separation in the actual automated verification by probabilistic model-checkers. In addition, we would like to investigate the use of our notion of layering for the analysis of complexity bounds, such as in [AC08, MR02].

Acknowledgements. We wish to thank Annabelle McIver for her feedback via email on the paper [MGCM08]. The comments from the reviewers helped in improving the presentation of our paper.

References

[AC08] H. Attiya and K. Censor. Tight bounds for asynchronous randomized consensus. *J. ACM*, 55(5), 2008.

[BGC04] C. Baier, M. Größer, and F. Ciesinski. Partial order reduction for probabilistic systems. In *Quantitative Evaluation of Systems (QEST)*, pages 230–239. IEEE CS Press, 2004.

[CCK+08] R. Canetti, L. Cheung, D. K. Kaynar, M. Liskov, N. A. Lynch, O. Pereira, and R. Segala. Analyzing security protocols using time-bounded task-PIOAs. *Discrete Event Dynamic Systems*, 18(1):111–159, 2008.

[Coh00] E. Cohen. Separation and reduction. In R. C. Backhouse and J. N. Oliveira, editors, *Mathematics of Program Construction (MPC)*, volume 1837 of *LNCS*, pages 45–59. Springer, 2000.

[DN04] P. R. D'Argenio and P. Niebert. Partial order reduction on concurrent probabilistic programs. In *Quantitative Evaluation of Systems (QEST)*, pages 240–249. IEEE CS Press, 2004.

[EF82] T. Elrad and N. Francez. Decomposition of distributed programs into communication-closed layers. *Sci. Comput. Program.*, 2(3):155–173, 1982.

[JZ92] W. Janssen and J. Zwiers. From sequential layers to distributed processes: Deriving a distributed minimum weight spanning tree algorithm. In *Principles of Distributed Computing (PODC)*, pages 215–227. ACM Press, 1992.

[KN02] M. Z. Kwiatkowska and G. Norman. Verifying randomized Byzantine agreement. In D. Peled and M. Y. Vardi, editors, *Formal Description Techniques (FORTE)*, volume 2529 of *LNCS*, pages 194–209. Springer, 2002.

[KNP04] M. Z. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. *International Journal on Software Tools for Technology Transfer*, 6(2):128–142, 2004.

[Koe36] D. Koenig. *Theorie der Endlichen und Unendlichen Graphen: Kombinatorische Topologie der Streckenkomplexe.* Leipzig: Akad. Verlag, 1936.

[KR92] E. Kushilevitz and M. O. Rabin. Randomized mutual exclusion algorithms revisited. In *PODC*, pages 275–283, 1992.

[KvST12] J.-P. Katoen, J. C. van de Pol, M. I. A. Stoelinga, and M. Timmer. A linear process-algebraic format with data for probabilistic automata. *Theor. Comput. Sci.*, 413(1):36–57, 2012.

[LR81] D. J. Lehmann and M. O. Rabin. On the advantages of free choice: A symmetric and fully distributed solution to the dining philosophers problem. In *Principles of Programming Languages (POPL)*, pages 133–138. ACM Press, 1981.

[MGCM08] A. K. McIver, C. Gonzalia, E. Cohen, and C. C. Morgan. Using probabilistic Kleene algebra pKA for protocol verification. *J. Log. Algebr. Program.*, 76(1):90–111, 2008.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[MM04] A. K. McIver and C. C. Morgan. *Abstraction, Refinement And Proof For Probabilistic Systems.* Springer, 2004.

[MR02] Y. Moses and S. Rajsbaum. A layered analysis of consensus. *SIAM J. Comput.*, 31(4):989–1021, 2002.

[OS10] E.-R. Olderog and M. Swaminathan. Layered composition for timed automata. In K. Chatterjee and T. A Henzinger, editors, *Formal Modeling and Analysis of Timed Systems (FORMATS)*, volume 6246 of *LNCS*, pages 228–242. Springer, 2010.

[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.

[Rab82] M. O. Rabin. n-process mutual exclusion with bounded waiting by $4 \log n$ shared variables. *J. Comput. Syst. Sci.*, 25(1):66–75, 1982.

[Sai92] I. Saias. Proving probabilistic correctness statements: the case of Rabin's algorithm for mutual exclusion. In *Principles of Distributed Computing (PODC)*, pages 263–274. ACM Press, 1992.

[SdR94] F. A. Stomp and W.-P. de Roever. A principle for sequential reasoning about distributed algorithms. *Formal Aspects of Computing*, 6(6):716–737, 1994.

[Seg00] R. Segala. Verification of randomized distributed algorithms. In E. Brinksma, H. Hermanns, and J.-P. Katoen, editors, *Formal Methods and Performance Analysis*, volume 2090 of *LNCS*, pages 232–260. Springer, 2000.

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

[Sto02] M. Stoelinga. An introduction to probabilistic automata. *Bulletin of the EATCS*, 78:176–198, 2002.

[SV99] M. Stoelinga and F. W. Vaandrager. Root contention in IEEE 1394. In J.-P. Katoen, editor, *AMAST Workshop on Real-Time and Probabilistic Systems (ARTS)*, volume 1601 of *LNCS*, pages 53–74. Springer, 1999.

[TSvdP11] M. Timmer, M. Stoelinga, and J. van de Pol. Confluence reduction for probabilistic systems. In P. A. Abdulla and K. R. M. Leino, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 6605 of *LNCS*, pages 311–325. Springer, 2011.