Layered Reduction for Abstract Probabilistic Automata[*]

Arpit Sharma[1] Joost-Pieter Katoen[1]

[1]Software Modeling and Verification Group, RWTH Aachen University, Germany

Abstract

Abstract probabilistic automata (APAs) constitute a complete abstraction and specification theory for probabilistic automata (PAs) [7, 8]. APA specifications support compositionality together with a step-wise refinement methodology, and thus are useful for component-oriented design and analysis of randomized distributed systems. This paper proposes a state-space reduction technique for such systems that are modeled as a network of acyclic APAs. Our technique is based on the notion of layered transformation. We propose a layered composition operator for acyclic APAs, and prove the communication closed layer (CCL) laws. Next, we define a partial order (po) equivalence between acyclic APAs, and show that it enables performing layered transformation within the framework of CCL laws. We also show the preservation of extremal reachability probabilities under this transformation.

1 Introduction

A probabilistic automaton (PA) resembles a labeled transition system (LTS) [23], but its transitions target probability distributions over states instead of single states. PAs have been developed by Segala [28, 29] and are compositional–a parallel composition operator allows one to construct a complex PA from several component PAs running in parallel, thus allowing to model complex systems in a modular way. PAs are widely used for the modeling and verification of randomized distributed algorithms and networking protocols. They have been used as semantic model for amongst others probabilistic process algebras [27] and the PIOA language [6].

Recently, *abstract* probabilistic automata (APAs) [7, 8] have been proposed as a powerful abstraction and specification formalism for (sets of) PAs. In an APA, sets of distributions are abstracted by using constraint functions. Action-labeled transitions of an APA are typed either "must" or "may". Hence, APAs can be seen as a combination of modal transition systems (MTSs) [21] and constraint Markov chains (CMCs) [5]. The theory of APAs is equipped with parallel and conjunction operators, and allows comparing two APAs using a refinement relation. A satisfaction relation is used to check whether a PA is an implementation of a given APA. APA specifications are useful for component-oriented design and analysis of randomized distributed systems. In this setting, a high-level model of the system which abstracts from the implementation details is constructed and used for the verification of interesting properties. A correct implementation can be obtained by applying a series of refinement steps. Model construction involves composing several components in parallel, where each component usually has multiple sub-components that are executed in a sequential manner. Components cooperate through their synchronization over common actions and through their respective action dependencies. Action dependencies between sub-components can be either explicitly stated or derived from the operations performed on data variables that are updated during an action execution (in case of APA with data). Some example systems that have this structure are distributed algorithms such as the randomized mutual exclusion algorithm by Kushilevitz and Rabin [19], Fischer's real-time mutual exclusion protocol [16], the two phase commit protocol [4], and the distributed minimum weight spanning tree algorithm [12]. Composing several components using parallel composition naturally leads to the problem of *state-space explosion*, where the number of states grows exponentially in the number of parallel components. The importance of probabilistic specification theories, and techniques for combating the state-space explosion problem has also been pointed out by Henzinger and Sifakis in [14].

In [32], a layered analysis of Kushilevitz and Rabin's randomized mutual exclusion algorithm [19] has been carried out. The underlying model on which layered transformations have been applied is a PA. We modeled this case study using the PRISM model checker [20]. The results for 3 processes and 5 rounds are summarised in Table 1 and show a state-space reduction by a factor three. These results

[*]This work is supported by the EU FP7 SENSATION project.

clearly indicate that layered reasoning can significantly reduce the state space of system models capturing the behavior of randomized distributed algorithms. In addition, layering has been successfully applied to obtain easier correctness proofs for various distributed systems [15, 16, 17, 18]. Motivated by this, we propose a state-space reduction technique for a network of acyclic APAs based on the notion of layering. The main principle is illustrated in Fig. 1. Here two APA components $\mathcal{M}$ and $\mathcal{N}$ are composed in parallel (left), where each component consists of n sub-components which are executed in a sequential manner (denoted by ;). The system obtained after performing layered transformation is shown in Fig. 1 (right). Note that all the sub-components of $\mathcal{M}$ and $\mathcal{N}$ need to be acyclic, but we do allow top-level recursion in $\mathcal{M}$ and $\mathcal{N}$ (as indicated by *). In other words, every component can have multiple rounds of execution, where a new round is started only when the last sub-component of the previous round has been completed. This is important as deadlock states are usually considered to be undesirable for models of distributed algorithms.

Roughly speaking, layering exploits the independence between sub-components to transform the system under consideration from a distributed representation to a layered representation. A layered composition operator "$\bullet$" is used to denote the layered representation of the system. Informally, $\mathcal{N}_1 \bullet \mathcal{N}_2$ allows synchronization on common actions and interleaving on disjoint actions, except when some action a of $\mathcal{N}_2$ depends on one or more actions of $\mathcal{N}_1$; in this case, a can be executed only after all the actions of $\mathcal{N}_1$ on which it depends have been executed. This new composition operator allows formulating *Communication Closed Layer* (CCL) laws [11, 15], which are required to carry out the structural transformations and establish an equivalence between the two systems. Since the sub-components within a component are executed sequentially, a partial order relation is proposed to relate the $\bullet$ and ; (sequential composition) operators. The reduced system obtained as a result of applying layered transformation can be used for analysis, provided it preserves a rich class of properties of interest. Probabilistic reachability is one of the most important quantitative properties in the area of probabilistic model checking [2]. Therefore, we focus on proving that the reduced system preserves maximum (resp. minimum) probability to reach its set of final states. As a result, probabilistic reachability properties can be checked on the layered, typically smaller, model.

Contributions. More specifically, the main contributions of this paper are as follows:

- We define the notions of *abstract execution* and *realisation*, which are subsequently used to compare the behaviour of acyclic APAs.
- We define the layered ($\bullet$) and sequential ($;$) composition operator, and formulate communication closed layer (CCL) laws for acyclic APAs.
- We define the partial order (po) equivalence between acyclic APAs, show that $\bullet$ is po-equivalent to ;, and prove that po-equivalence preserves maximum (resp. minimum) probabilities to reach the set of final states.
- Finally, we show how state space reduction can be achieved by replacing $\bullet$ with ; within the framework of CCL laws.

	Parallel	Layered
Build time (s)	898.70	90.39
# States	198063	71619
# Transitions	351432	128920

Table 1. Parallel vs. layered composition

Related work

APA. Abstract Probabilistic Automata (APAs) were originally defined in [7, 8]. In [8], a complete abstraction and specification theory for PAs was proposed. This theory was later extended to support specifications over dissimilar alphabets, some additional operators, and an APA-embedding of Interface Automata [9]. A tool implementing this theory was reported in [10]. Recently, compositional abstraction techniques for APAs have been proposed, which are based on the notion of common combined transitions [31]. The theory presented in this paper is based on the APA model introduced in [8].

Layering. The decomposition of a distributed program into communication closed layers to simplify its analysis has been originally proposed in [11]. In [17], a layered composition operator and various algebraic transformation rules have been introduced to simplify the analysis of distributed database systems. Some other examples where layering techniques have been applied for the analysis of distributed systems can be found in [15, 16, 18]. An extension of layering operator and CCL laws to the real-time setting has been proposed in [16]. Layered composition for timed automata has been investigated in [25]. In [26], layering based structural transformations have been applied for easier verification of Fischer's real-time mutual exclusion protocol. In the probabilistic context, layering has been applied to the consensus problem to prove complexity lower bounds [24]. The layered composition operator and probabilistic counterparts of the CCL laws have been defined for the PA model [32]. As mentioned before, the feasibility of this approach has been demonstrated on a randomized mutual exclusion algorithm. Most recently, the theory of layering has

$$\begin{bmatrix} \mathcal{M}_1 \\ ; \\ \mathcal{M}_2 \\ ; \\ \vdots \\ ; \\ \mathcal{M}_n \end{bmatrix}^* \; \Big\| \; \begin{bmatrix} \mathcal{N}_1 \\ ; \\ \mathcal{N}_2 \\ ; \\ \vdots \\ ; \\ \mathcal{N}_n \end{bmatrix}^* \xRightarrow[\text{transformation}]{\text{Layered}} \begin{bmatrix} \mathcal{M}_1 \parallel \mathcal{N}_1 \\ ; \\ \mathcal{M}_2 \parallel \mathcal{N}_2 \\ ; \\ \vdots \\ ; \\ \mathcal{M}_n \parallel \mathcal{N}_n \end{bmatrix}^*$$

Figure 1. Layered reduction

been extended to modal transition systems (MTSs) [30]. In [30], it has been shown that layering can be used for state-space reduction of distributed systems that are modeled as a network of acyclic MTSs. Our results can be viewed as an extension of the layering for PAs [32] and MTSs [30] to APAs.

Organisation of the paper. Section 2 briefly recalls the basic concepts of PAs and APAs. Section 3 presents the satisfaction and refinement relations for APAs. Section 4 discusses the composition operators for APAs, and introduces CCL laws. Section 5 defines po-equivalence between APAs, and proves that po-equivalence preserves maximum (resp. minimum) reachability probabilities. Section 6 discusses the possible extensions of our results. Finally, Section 7 concludes the paper.

2 Preliminaries

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 $supp(\mu) = \{s \in S | \mu(s) > 0\}$ be the support of μ.

2.1 Probabilistic Automata

Definition 1 [PA] A probabilistic automaton (PA) is a tuple $\mathcal{P} = (S, s_0, S_f, Act, V)$, where:

- S is a finite set of states,
- $s_0 \in S$ is the initial state,
- $S_f \subset S$ is the set of final states where $s_0 \notin S_f$,
- Act is a finite set of actions,
- $V : S \setminus S_f \times Act \times Dist(S) \to \mathbb{B}_2$ is a two-valued transition function.

Here $\mathbb{B}_2 = \{\bot, \top\}$, with $\bot < \top$. $V(s, a, \mu)$ identifies the transition of the automaton: $\top$ indicates its presence and $\bot$ indicates its absence. We write $s \xrightarrow{a} \mu$ whenever $V(s, a, \mu) = \top$. Intuitively, PAs are very similar to LTSs, with the only difference that the target of each transition is a distribution over states instead of just a single next state.

Let $act(s)$ denote the set of enabled actions from state s, i.e., $act(s) = \{a \in Act \mid \exists \mu : V(s, a, \mu) \neq \bot\}$. A possible behaviour of a PA is obtained from the resolution of non-deterministic and probabilistic choices, described in terms of paths. A path π of PA $\mathcal{P}$ is a (possibly infinite) sequence of the form $\pi = s_0 a_1 \mu_1 s_1 a_2 \mu_2 s_2 a_3 \mu_3 \ldots$ 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). Let $|\pi|$ be the length (number of actions) of a finite path π. For infinite path π and any $i \in \mathbb{N}$, let $\pi[i] = s_i$, the $(i+1)$-st state of π. For finite path π of length n, $\pi[i]$ is only defined for $i \leq n$ and defined as for infinite paths. Let $Paths_{fin}(\mathcal{P})$ be the set of all finite paths in PA $\mathcal{P}$, and $Paths_{inf}(\mathcal{P})$ the set of all infinite paths of $\mathcal{P}$ that start in state s_0. Let $Paths_{fin}^{S_f}(\mathcal{P})$ be the set of all finite paths of $\mathcal{P}$ that start in state s_0 and end in some state $s \in S_f$.

An adversary $\mathcal{D}$ of PA $\mathcal{P}$ maps a finite path π of $\mathcal{P}$ to a pair (a, μ) or to $\curlywedge$, such that if $\mathcal{D}(\pi) = (a, \mu)$ for some $a \in Act$ and $\mu \in Dist(S)$, then $last(\pi) \xrightarrow{a} \mu$, and if there is no $a \in Act$ and $\mu \in Dist(S)$ s.t. $last(\pi) \xrightarrow{a} \mu$ then $\mathcal{D}(\pi) = \curlywedge$, where $\curlywedge$ denotes the terminal action. We restrict the class of adversaries to the class of *admissible history-independent adversaries* [13]. An adversary $\mathcal{D}$ is called history-independent iff $last(\pi) = last(\pi') \Rightarrow \mathcal{D}(\pi) = \mathcal{D}(\pi')$ for any finite paths π and π'. Since our interest is in probabilistic reachability of PAs, it is sufficient to consider history-independent adversaries [2]. Such adversaries are also known as memoryless schedulers. Admissible adversaries avoid the problem of unrealistic upper and lower bounds for the probability values [13]. For PA $\mathcal{P}$, let $Paths_{fin}^{\mathcal{D}}(\mathcal{P})$ be the set of all finite paths, and $Paths_{inf}^{\mathcal{D}}(\mathcal{P})$ the set of all infinite paths of $\mathcal{P}$ under $\mathcal{D}$ that start in state s_0. Let $Adv(\mathcal{P})$ be the set of all admissible history-independent adversaries of PA $\mathcal{P}$. Let $Paths_{fin}^{\mathcal{D}, S_f}(\mathcal{P})$ be the set of all finite paths under $\mathcal{D}$ that start in state s_0 and end in some state $s \in S_f$. For $\mathcal{D} \in Adv(\mathcal{P})$ let the probability measure $Prob^{\mathcal{D}}$ be defined over $Paths_{inf}^{\mathcal{D}}(\mathcal{P})$ in the following way. Let function $\mathbb{A} : Paths_{fin}^{\mathcal{D}}(\mathcal{P}) \times Paths_{fin}^{\mathcal{D}}(\mathcal{P}) \to [0,1]$ be defined for two finite paths $\pi, \pi' \in Paths_{fin}^{\mathcal{D}}(\mathcal{P})$:

$$\mathbb{A}(\pi, \pi') = \begin{cases} \mu(s') & \text{if } \pi' \text{ is of the form } \pi \xrightarrow{a,\mu} s' \text{ and} \\ & \mathcal{D}(\pi) = (a, \mu) \\ 0 & \text{otherwise.} \end{cases}$$

The probability $P^{\mathcal{D}}(\pi)$ for any finite path π where $\pi \in Paths_{fin}^{\mathcal{D}}(\mathcal{P})$ and $|\pi| = n$ is now defined as follows:

$$P^{\mathcal{D}}(\pi) = \begin{cases} 1 & \text{if } n = 0 \\ \mathbb{A}(\pi[0], \pi[1]) \cdot \ldots \cdot & \\ \mathbb{A}(\pi[n-1], \pi[n]) & \text{otherwise.} \end{cases}$$

The cylinder of a finite path π is defined as follows:

$$cyl^{\mathcal{D}}(\pi) = \{\pi' \in Paths_{inf}^{\mathcal{D}}(\mathcal{P}) \mid \pi \text{ is a prefix of } \pi'\},$$

and let $\mathcal{F}^{\mathcal{D}}$ is the smallest σ-field containing $\{cyl^{\mathcal{D}}(\pi) \mid \pi \in Paths^{\mathcal{D}}_{fin}(\mathcal{P})\}$. Next, we define $Prob^{\mathcal{D}}$ on $\mathcal{F}^{\mathcal{D}}$ as the unique measure such that $Prob^{\mathcal{D}}(cyl(\pi)) = P^{\mathcal{D}}(\pi)$ for all $\pi \in Paths^{\mathcal{D}}_{fin}(\mathcal{P})$.

Definition 2 PA $\mathcal{P} = (S, s_0, S_f, Act, V)$ is *deterministic*, if for every state s and action a we have: $|\{\mu \in Dist(S) \mid V(s, a, \mu) \neq \bot\}| \leq 1$.

In this paper, we focus on *maximum reachability probabilities* (resp. *minimum*) of reaching the set of final states $S_f \subset S$ in a PA, over all possible adversaries. Maximum reachability probabilities are defined as follows:

$$P^{max}_{\mathcal{P}}(S_f) = \sup_{\mathcal{D} \in Adv(\mathcal{P})} P^{\mathcal{D}}(S_f)$$

$$P^{\mathcal{D}}(S_f) = \sum_{\pi \in Paths^{\mathcal{D}, S_f}_{fin}(\mathcal{P})} P^{\mathcal{D}}(\pi)$$

Minimum reachability probabilities are defined in an analogous manner.

Example 1 Consider the PA $\mathcal{P}$ in Fig. 2 (right), where $S = \{s_0, s_1, s_2, s_f\}$, $Act = \{a, b, c\}$, s_0 is the initial state, and $S_f = \{s_f\}$. Here s_0 can move with action a to s_1 and s_2 with probability 0.3 and 0.7, respectively. An example finite path π is $s_0 a \mu_1 s_1 c \mu_2 s_1 c \mu_2 s_2 b \mu_3 s_f$, where $\mu_1(s_1) = 0.3, \mu_1(s_2) = 0.7, \mu_2(s_1) = 0.2, \mu_2(s_2) = 0.8$, and $\mu_3(s_f) = 1$. We have $|\pi| = 4$, and $\pi[2] = s_1$. It is easy to check that $\mathcal{P}$ is deterministic and cyclic.

2.2 Abstract Probabilistic Automata

Let φ be an arithmetic expression over variables whose values are in S. We call φ a constraint function. We require that for every variable in the arithmetic expression of φ, there exists a distribution μ that satisfies φ s.t. the value of the variable is nonzero. For example, we do not allow constraint function $\varphi_x = x_1 \geq 0.7 \wedge x_2 \leq 0.3 \wedge x_0 = 0 \wedge x_0 + x_1 + x_2 = 1$, since for every distribution μ that satisfies φ_x, the value of $x_0 = 0$. Let $C(S)$ be the set of all allowed constraint functions defined on S. Let $Sat(\varphi)$ denote the set of distributions that satisfy constraint function φ. Note that we do not restrict ourselves to linear constraint functions, as polynomial constraints are needed for defining layered and parallel composition.

Definition 3 [APA] An abstract PA (APA) is a tuple $\mathcal{N} = (S, s_0, S_f, Act, V)$ such that:

- S, s_0, S_f and Act are defined as before,
- $V : S \setminus S_f \times Act \times C(S) \to \mathbb{B}_3$ is a three-valued state-constraint function.

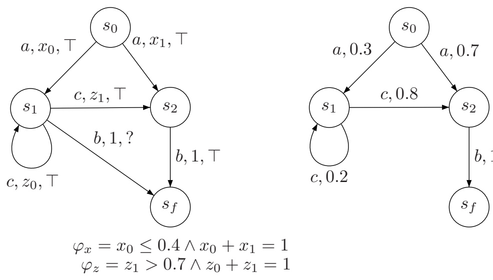

Figure 2. An APA $\mathcal{N}$ (left) and a PA $\mathcal{P}$ (right) that satisfies $\mathcal{N}$

Here $\mathbb{B}_3 = \{\bot, ?, \top\}$ denotes a complete lattice with the following ordering $\bot < ? < \top$ and meet $\sqcap$ and join $\sqcup$ operators. $V(s, a, \varphi)$ identifies the transition of the automaton: $\top$, ? and $\bot$ indicate a *must*, a *may* and absence of transition respectively. For simplicity we write $s \xrightarrow{a}_{\top} \varphi$ instead of $V(s, a, \varphi) = \top$. Similarly, we write $s \xrightarrow{a}_{?} \varphi$ instead of $V(s, a, \varphi) = ?$. Let $act(s)$ denote the set of enabled actions from state s, i.e., $act(s) = \{a \in Act \mid \exists \varphi : V(s, a, \varphi) \neq \bot\}$.

Note that a PA is an APA where every transition is a must-transition and for each constraint function φ, the number of distributions in $Sat(\varphi)$ equals one, i.e., $|Sat(\varphi)| = 1$. Thus, every PA is an APA. Similarly an APA where every may and must transition jump to the next state with probability one is a modal transition system (MTS) [21].

Definition 4 APA $\mathcal{N} = (S, s_0, S_f, Act, V)$ is *deterministic*, if for every state s and action a we have: $|\{\varphi \in C(S) \mid V(s, a, \varphi) \neq \bot\}| \leq 1$.

From now on, we only consider deterministic APAs, as they are sufficient for modeling the behavior of typical randomized distributed algorithms [15]. Let $(a, \varphi)(s)$ denote the set of states in a deterministic APA $\mathcal{N}$ that can be reached from state s in one step by performing action a with constraint φ. Formally, $(a, \varphi)(s) = \{s' \in S \mid V(s, a, \varphi) \neq \bot \wedge \exists \mu \in Sat(\varphi) : \mu(s') > 0\}$. An abstract execution ρ of an APA $\mathcal{N}$ is a (possibly infinite) sequence of the form $\rho = s_0 a_1 \varphi_1 s_1 a_2 \varphi_2 s_2 a_3 \varphi_3 \ldots$, where $\forall n : s_n \xrightarrow{a_{n+1}}_{\top} \varphi_{n+1}$ or $s_n \xrightarrow{a_{n+1}}_{?} \varphi_{n+1}$, and $s_{n+1} \in (a_{n+1}, \varphi_{n+1})(s_n)$. Let $Exec_{fin}(\mathcal{N})$ be the set of all finite abstract executions, and $Exec_{inf}(\mathcal{N})$ the set of all infinite abstract executions of $\mathcal{N}$ that start in state s_0. Let $Exec^{S_f}_{fin}(\mathcal{N})$ be the set of all finite abstract executions of $\mathcal{N}$ that start in state s_0, and end in some state $s \in S_f$. Let $Exec^{s}_{fin}(\mathcal{N})$ be the set of all finite abstract executions that start in state s, and $|\rho|$ the length (number of actions) of a finite abstract execution ρ. For infinite abstract execution ρ and any $i \in \mathbb{N}$, let $\rho[i] = s_i$, the $(i+1)$-st state of ρ. For finite abstract execution ρ of length n, $\rho[i]$ is only defined for $i \leq n$ and defined as in the case of infinite abstract executions. Let $last(\rho)$ denote the last state of ρ (if ρ is finite).

Example 2 Consider the APA $\mathcal{N}$ in Fig. 2 (left), where $S = \{s_0, s_1, s_2, s_f\}$, $Act = \{a, b, c\}$, s_0 is the initial state, and $S_f = \{s_f\}$. Here s_0 has one outgoing transition: a must a-transition (s_0, a, φ_x). Similarly, s_1 has two outgoing transitions: a must c-transition (s_1, c, φ_z), and a may b-transition $(s_1, b, 1)$. Note that $\mathcal{N}$ is deterministic and cyclic. An example finite abstract execution ρ is $s_0 a \varphi_x s_1 c \varphi_z s_1 c \varphi_z s_2$ with $|\rho| = 3$, and $\rho[2] = s_1$.

3 Satisfaction and Refinement

This section presents the notions of *satisfaction* and *refinement* originally introduced in [8]. A satisfaction relation allows to relate a PA (implementation) with an APA (specification). A refinement relation is used to compare APAs w.r.t. their sets of implementations. We also define the notions of realisation and maximum (resp. minimum) probabilities of reaching the set of final states computed over all the implementations of an APA $\mathcal{N}$.

Definition 5 [Simulation relation [8]] Let S and S' be non-empty sets of states. Given $\mu \in Dist(S)$, $\mu' \in Dist(S')$, a function $\delta : S \to (S' \to [0,1])$, and a binary relation $R \subseteq S \times S'$, μ is *simulated* by μ' with respect to R and δ, denoted as $\mu \Subset_R^\delta \mu'$, iff

- for all $s \in S$, if $\mu(s) > 0$, then $\delta(s) \in Dist(S')$,
- for all $s' \in S'$, $\sum_{s \in S} \mu(s) \cdot \delta(s)(s') = \mu'(s')$, and
- if $\delta(s)(s') > 0$, then $(s, s') \in R$.

We write $\mu \Subset_R \mu'$ iff there exists a function δ such that $\mu \Subset_R^\delta \mu'$. Such δ is called a *correspondence* function.

Definition 6 [Satisfaction] Let $\mathcal{P} = (S, s_0, S_f, Act, V)$ be a PA and $\mathcal{N} = (S', s'_0, S'_f, Act, V')$ be an APA. $R \subseteq S \times S'$ is a *satisfaction relation* iff, for any $(s, s') \in R$, the following conditions hold:

- $\forall a \in Act, \forall \varphi' \in C(S') : V'(s', a, \varphi') = \top \Rightarrow \exists \mu \in Dist(S) : V(s, a, \mu) = \top$ and $\exists \mu' \in Sat(\varphi')$ such that $\mu \Subset_R \mu'$,
- $\forall a \in Act, \forall \mu \in Dist(S) : V(s, a, \mu) = \top \Rightarrow \exists \varphi' \in C(S') : V'(s', a, \varphi') \neq \bot$ and $\exists \mu' \in Sat(\varphi')$ such that $\mu \Subset_R \mu'$,
- $s \in S_f \Leftrightarrow s' \in S'_f$.

We say that $\mathcal{P}$ *satisfies* $\mathcal{N}$, denoted $\mathcal{P} \models \mathcal{N}$, iff there exists a satisfaction relation relating s_0 and s'_0. If $\mathcal{P} \models \mathcal{N}$, $\mathcal{P}$ is called an *implementation* of $\mathcal{N}$.

Intuitively, a state s in PA $\mathcal{P}$ satisfies state s' in APA $\mathcal{N}$ iff any must transition of s' is matched by a transition of s agreeing on the probability distributions specified by the constraint, and s does not contain any transitions without a corresponding transition (may or must) in s'. In addition, final states in $\mathcal{P}$ can only be related to final states in $\mathcal{N}$ and vice versa. Let $[\![\mathcal{N}]\!] = \{\mathcal{P} \mid \mathcal{P} \models \mathcal{N}\}$, the set of implementations of $\mathcal{N}$. Let $\mathcal{P} \in [\![\mathcal{N}]\!]$, and $\pi = s_0 a_1 \mu_1 s_1 a_2 \mu_2 s_2 \ldots s_n$ be a finite path of $\mathcal{P}$, i.e., $\pi \in Paths_{fin}(\mathcal{P})$. π is said to be a *realisation* of $\rho = s_0 a_1 \varphi_1 s_1 a_2 \varphi_2 s_2 \ldots s_n$ where $\rho \in Exec_{fin}(\mathcal{N})$, denoted $\pi \models \rho$, if $\forall i < n : \mu_{i+1} \in Sat(\varphi_{i+1})$. An implementation $\mathcal{P} \in [\![\mathcal{N}]\!]$ is said to be *realisable*, if $\mathcal{P}$ is deterministic and $\forall \pi \in Paths_{fin}(\mathcal{P}) \exists \rho \in Exec_{fin}(\mathcal{N}) : \pi \models \rho$.

Example 3 The PA $\mathcal{P}$ in Fig. 2 (right) is an implementation of the APA $\mathcal{N}$ in Fig. 2 (left). It is easy to check that there exists a satisfaction relation relating initial states of $\mathcal{P}$ and $\mathcal{N}$. Note that in this example, for every implementation $\mathcal{P}$ of $\mathcal{N}$, $S_f \neq \varnothing$ (since the value of x_1 is always greater than 0 and s_2 has a must-b transition moving to s_f with probability 1). Finite path $\pi = s_0 a \mu_1 s_1 c \mu_2 s_1 c \mu_2 s_2$ of PA $\mathcal{P}$ (Fig. 2 (right)) is a realisation of finite abstract execution $\rho = s_0 a \varphi_1 s_1 c \varphi_2 s_1 c \varphi_2 s_2$ of $\mathcal{N}$ (Fig. 2 (left)), where μ_1, μ_2 and μ_3 are defined as in Example 1. It can be checked that $\mathcal{P}$ is a realisable implementation of $\mathcal{N}$ (since $\mathcal{P}$ is deterministic and every finite path π of $\mathcal{P}$ is a realisation of some finite abstract execution ρ in $\mathcal{N}$).

Note that for a deterministic APA $\mathcal{N}$, if a path $\pi \in Paths_{fin}(\mathcal{P})$ where $\mathcal{P} \in [\![\mathcal{N}]\!]$ is a realisation of some finite abstract execution $\rho \in Exec_{fin}(\mathcal{N})$, then it cannot be a realisation of another finite abstract execution of $\mathcal{N}$.

Definition 7 [Refinement] Let $\mathcal{N} = (S, s_0, S_f, Act, V)$ and $\mathcal{N}' = (S', s'_0, S'_f, Act, V')$ be APAs. $R \subseteq S \times S'$ is a *strong refinement relation* iff, for all $(s, s') \in R$, the following conditions hold:

- $\forall a \in Act. \forall \varphi' \in C(S'). V'(s', a, \varphi') = \top \Rightarrow \exists \varphi \in C(S). V(s, a, \varphi) = \top$ and there exists a correspondence function $\delta : S \to (S' \to [0,1])$ such that $\forall \mu \in Sat(\varphi). \exists \mu' \in Sat(\varphi')$ with $\mu \Subset_R^\delta \mu'$,
- $\forall a \in Act. \forall \varphi \in C(S). V(s, a, \varphi) \neq \bot \Rightarrow \exists \varphi' \in C(S'). V'(s', a, \varphi') \neq \bot$ and there exists a correspondence function $\delta : S \to (S' \to [0,1])$ such that $\forall \mu \in Sat(\varphi). \exists \mu' \in Sat(\varphi')$ with $\mu \Subset_R^\delta \mu'$.
- $s \in S_f \Leftrightarrow s' \in S'_f$.

$\mathcal{N}$ strongly refines $\mathcal{N}'$, denoted $\mathcal{N} \preceq_S \mathcal{N}'$, iff there exists a strong refinement relation relating s_0 and s'_0.

Intuitively, a state s in APA $\mathcal{N}$ strongly refines state s' in APA $\mathcal{N}'$ iff any must transition of s' is matched by a transition of s agreeing on the probability distributions specified by the constraint, and s does not contain any transitions (may or must) without a corresponding transition (may or must) in s'. In addition, final states in $\mathcal{N}$ can only be related to final states in $\mathcal{N}'$ and vice versa.

Definition 8 [Refinement equivalence] APAs $\mathcal{N}$ and $\mathcal{N}'$ are *refinement equivalent*, denoted $\mathcal{N} \equiv \mathcal{N}'$, iff $\mathcal{N} \preceq_S \mathcal{N}'$ and $\mathcal{N}' \preceq_S \mathcal{N}$.

Since strong refinement implies inclusion of sets of implementations, if $\mathcal{N}$ and $\mathcal{N}'$ are refinement equivalent, they have the same set of implementations, i.e., $[\![\mathcal{N}]\!] = [\![\mathcal{N}']\!]$ [8].

Remark 1 *A satisfaction relation can be seen as a special case of refinement relation. In simple words, if $\mathcal{P}$ satisfies $\mathcal{N}$, then $\mathcal{P}$ also strongly refines $\mathcal{N}$ (since every PA is an APA and all the three conditions of strong refinement are satisfied).*

Assumptions. For the rest of the paper, we assume:

1. Every APA is acyclic and consistent. An APA is consistent iff it admits at least one implementation [8].
2. Every APA has a single final state, i.e., $|S_f| = 1$, and all its states (except the final state) have at least one outgoing transition.
3. Dependencies between actions of different components are known in advance.

The acyclicity assumption is required to establish an equivalence between the layered operator ($\bullet$) and sequential operator (;). This restriction is less limiting than may appear at first sight, since in many distributed systems the sub-components/phases are acyclic [15]. As mentioned before, we do allow the algorithm to be executed multiple times, i.e., top-level recursion is allowed. The second assumption ensures that deadlock states (which are usually undesirable) are absent. The dependency relation between actions of sub-components/phases can be either explicitly stated or derived from the operations performed on data variables that are updated during an action execution [15]. For instance, a read access to a shared variable depends on a write access of that variable.

Next, we define the maximum reachability probabilities (resp. minimum) of reaching the set of final states, determined over all the implementations of an APA $\mathcal{N}$. Intuitively, this means that for each implementation $\mathcal{P} \in [\![\mathcal{N}]\!]$, we are interested in computing the maximum (resp. minimum) probability to reach the set of states S_f. Finally, we would compute the maximum (resp. minimum) probability over all the implementations. More formally it is defined as follows:

$$P^{max}_{\mathcal{N}}(S_f) = \sup_{\mathcal{P} \in [\![\mathcal{N}]\!]} P^{max}_{\mathcal{P}}(S_f)$$

Minimum reachability probabilities are defined in an analogous manner.

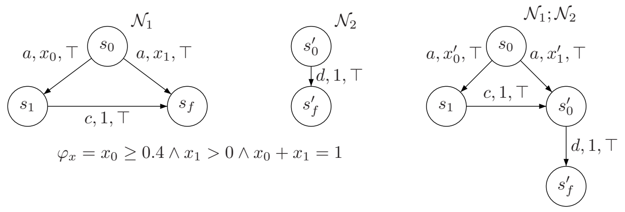

Figure 3. APAs $\mathcal{N}_1$ and $\mathcal{N}_2$ (left) and their sequential composition (right)

4 Composition and CCL Laws

In this section we define composition operators that enable to combine two APAs. We propose sequential, and layered composition operators, and recall parallel composition from [8]. The framework of CCL laws is also formulated, which is required for carrying out the layered transformations.

Definition 9 [Sequential composition] Given APAs $\mathcal{N}_i = (S_i, s_{0i}, \{s_{fi}\}, Act_i, V_i)$, where $i \in \{1, 2\}$ with $S_1 \cap S_2 = \varnothing$. Their *sequential composition*, denoted $\mathcal{N}_1;\mathcal{N}_2$, is the APA $(S, s_{01}, \{s_{f2}\}, Act_1 \cup Act_2, V)$, where $S = S_1 \setminus \{s_{f1}\} \cup S_2$ and $V = V'_1 \cup V_2$. Here $V'_1 = V_1[s_{02} \leftarrow s_{f1}]$ is defined by $V'_1(s, a, \varphi') = V_1(s, a, \varphi)$ with φ' the new constraint in $C(S)$ s.t. for any μ, $\mu \in Sat(\varphi)$ iff there exists $\mu' \in Sat(\varphi')$ with

$$\mu'(s') = \mu(s') \text{ if } s' \neq s_{f1}, \text{ and}$$
$$\mu'(s_{02}) = \mu(s_{f1}) \text{ otherwise.}$$

Intuitively, the sequential composition of two APAs $\mathcal{N}_1$ and $\mathcal{N}_2$ requires executing the actions of $\mathcal{N}_1$ followed by actions of $\mathcal{N}_2$. Note that all the incoming transitions to state s_{f1} are redirected to s_{02}. Here, s_{01}, s_{f2} are the new initial and final states in the resulting APA $\mathcal{N}_1;\mathcal{N}_2$, respectively.

Definition 10 [Parallel composition [8]] Given APAs $\mathcal{N}_i = (S_i, s_{0i}, \{s_{fi}\}, Act_i, V_i)$, where $i \in \{1, 2\}$ with $S_1 \cap S_2 = \varnothing$. Their *parallel composition*, denoted $\mathcal{N}_1||\mathcal{N}_2$, is the APA $(S_1 \times S_2, (s_{01}, s_{02}), \{(s_{f1}, s_{f2})\}, Act_1 \cup Act_2, V)$ where V is defined by:

- For all $(s, s') \in S_1 \times S_2$, $a \in Act_1 \cap Act_2$, if there exists $\varphi \in C(S_1)$ and $\varphi' \in C(S_2)$, such that $V_1(s, a, \varphi) \neq \bot$ and $V_2(s', a, \varphi') \neq \bot$, define $V((s, s'), a, \tilde{\varphi}) = V_1(s, a, \varphi) \sqcap V_2(s', a, \varphi')$ with $\tilde{\varphi}$ the new constraint in $C(S_1 \times S_2)$ s.t. $\tilde{\mu} \in Sat(\tilde{\varphi})$ iff there exists $\mu \in Sat(\varphi)$ and $\mu' \in Sat(\varphi')$ such that $\tilde{\mu}(u, v) = \mu(u) \cdot \mu'(v)$ for all $u \in S_1$ and $v \in S_2$. If either for all $\varphi \in C(S_1)$, we have $V_1(s, a, \varphi) = \bot$, or $\forall \varphi' \in C(S_2)$, we have $V_2(s', a, \varphi') = \bot$ then for all $\tilde{\varphi} \in C(S_1 \times S_2)$, $V((s, s'), a, \tilde{\varphi}) = \bot$.

- For all $(s, s') \in S_1 \times S_2$, $a \in Act_1 \setminus Act_2$, and for all $\varphi \in C(S_1)$, define $V((s, s'), a, \tilde{\varphi}) = V_1(s, a, \varphi)$ with $\tilde{\varphi}$ the new constraint in $C(S_1 \times S_2)$ such that $\tilde{\mu} \in Sat(\tilde{\varphi})$ iff for all $u \in S_1$ and $v \neq s'$, $\tilde{\mu}(u, v) = 0$ and the distribution $\mu : t \mapsto \tilde{\mu}(t, s')$ is in $Sat(\varphi)$.

- For all $(s, s') \in S_1 \times S_2$, $a' \in Act_2 \setminus Act_1$ and for all $\varphi' \in C(S_2)$, define $V((s, s'), a', \tilde{\varphi}') = V_2(s', a', \varphi')$ with $\tilde{\varphi}'$ the new constraint in $C(S_1 \times S_2)$ such that $\tilde{\mu}' \in Sat(\tilde{\varphi}')$ iff for all $u \neq s$ and $v \in S_2$, $\tilde{\mu}'(u, v) = 0$ and the distribution $\mu' : t' \mapsto \tilde{\mu}'(s, t')$ is in $Sat(\varphi')$.

Parallel composition forces synchronization on all common actions and interleaving on disjoint actions. Note that the synchronization of two must transitions results in a must transition, and composing may-must, must-may and may-may transitions results in a may transition.

Next, we introduce the notion of action independence which is subsequently used to define layered composition. The dependency between actions a and b is denoted $a \dagger b$. Two additional requirements for the dependency relation are that it is reflexive and symmetric. Two distinct actions that are not dependent are said to be independent, where independence is defined as follows:

Definition 11 [Action independence] For an APA $\mathcal{N} = (S, s_0, S_f, Act, V)$, actions $a, b \in Act$ are said to be *independent*, denoted $a \ddagger b$, iff for all states $s \in S$ with $a, b \in act(s)$ we have:

- For any $s' \in S$: $s' \in (a, \varphi)(s) \Rightarrow b \in act(s')$,

- For any $s' \in S$: $s' \in (b, \varphi')(s) \Rightarrow a \in act(s')$, and

- For any $s'' \in S$:

  - $\forall \rho \in Exec^s_{fin}(\mathcal{N}) : \rho = sa\varphi s' b\varphi' s'' \Rightarrow$
    $\exists \rho' \in Exec^s_{fin}(\mathcal{N}) : \rho' = sb\varphi' s' a\varphi s''$.
  - $\forall \rho \in Exec^s_{fin}(\mathcal{N}) : \rho = sb\varphi' s' a\varphi s'' \Rightarrow$
    $\exists \rho' \in Exec^s_{fin}(\mathcal{N}) : \rho' = sa\varphi s' b\varphi' s''$.

The first two conditions assert that a and b should not disable each other. The last condition asserts that the same state should be reached from s in two steps by either performing a followed by b, or by performing b followed by a. Let $act(s) \ddagger a$ denote $\forall b \in act(s) : b \ddagger a$. Note that this notion of action independence is a purely syntactic notion, and —in contrast to action independence in partial-order reduction [1]— does not take the transition probabilities into account.

Definition 12 APAs $\mathcal{N}_1$ and $\mathcal{N}_2$ are independent, denoted $\mathcal{N}_1 \ddagger \mathcal{N}_2$, iff every action of $\mathcal{N}_1$ is independent of every action of $\mathcal{N}_2$ in $\mathcal{N}_1 || \mathcal{N}_2$.

Let $s \xrightarrow{*} s'$ denote that state s' is reachable from s through an arbitrary finite sequence of transitions in APA $\mathcal{N}$. In other words, $s \xrightarrow{*} s'$ denote that there exists a finite abstract execution $\rho \in Exec^s_{fin}(\mathcal{N})$ with $last(\rho) = s'$.

Definition 13 [Layered composition] Given APAs $\mathcal{N}_i = (S_i, s_{0i}, \{s_{fi}\}, Act_i, V_i)$, where $i \in \{1, 2\}$ with $S_1 \cap S_2 = \varnothing$. Their *layered composition*, denoted $\mathcal{N}_1 \bullet \mathcal{N}_2$, is the APA $(S_1 \times S_2, (s_{01}, s_{02}), \{(s_{f1}, s_{f2})\}, Act_1 \cup Act_2, V)$ where V is defined by:

- The first two clauses are the same as for Def. 10, i.e., parallel composition.

- For all $(s, s') \in S_1 \times S_2$, $a' \in Act_2 \setminus Act_1$ and for all $\varphi' \in C(S_2)$:

  1. If $\forall s^* : s \xrightarrow{*} s^* : act(s^*) \ddagger a'$ in $\mathcal{N}_1 || \mathcal{N}_2$, then we define $V((s, s'), a', \tilde{\varphi}') = V_2(s', a', \varphi')$ with $\tilde{\varphi}'$ the new constraint in $C(S_1 \times S_2)$ such that $\tilde{\mu}' \in Sat(\tilde{\varphi}')$ iff for all $u \neq s$, $v \in S_2$, $\tilde{\mu}'(u, v) = 0$, the distribution $\mu' : t' \mapsto \tilde{\mu}'(s, t')$ is in $Sat(\varphi')$.
  2. If $\exists s^* : s \xrightarrow{*} s^* : act(s^*) \dagger a'$ in $\mathcal{N}_1 || \mathcal{N}_2$, then $\forall \tilde{\varphi}' \in C(S_1 \times S_2)$ let $V((s, s'), a', \tilde{\varphi}') = \bot$.

The crux of the definition of layered composition lies in the last clause. The first part of this clause defines V in the same way as for parallel composition, in case the action in $\mathcal{N}_2$ is independent of all actions enabled in states reachable from the current state in $\mathcal{N}_1$. The second part of this clause ensures that in case an action a' in $\mathcal{N}_2$ depends on one or more actions in $\mathcal{N}_1$, then it cannot be executed before all the actions in $\mathcal{N}_1$ (on which it depends) have taken place. In other words, layered composition restricts the asynchronous interleaving (for the right component), to only those actions of the right component, which are independent to the actions of the left component.

Remark 2 *Note that the parallel and layered composition of two APAs with linear constraint functions may lead to an APA whose constraints are polynomial.*

Example 4 The sequential composition of two APAs $\mathcal{N}_1, \mathcal{N}_2$ (Fig. 3 (left and middle)) is shown in Fig. 3 (right). The parallel composition of $\mathcal{N}_1, \mathcal{N}_2$ is shown in Fig. 4 (left). If we assume that actions a, d are dependent in $\mathcal{N}_1 || \mathcal{N}_2$, then layered composition of $\mathcal{N}_1, \mathcal{N}_2$ is illustrated in Fig. 4 (right). Note that d cannot be executed before a in $\mathcal{N}_1 \bullet \mathcal{N}_2$. To keep the figures simple, we do not show constraint functions obtained after composing the two systems.

Next, we use the above mentioned composition operators for formulating the communication closed layer (CCL) laws as follows:

Theorem 1 [CCL laws] For APAs $\mathcal{N}_1, \mathcal{N}_2, \mathcal{M}_1$, and $\mathcal{M}_2$, with $\mathcal{N}_1 \ddagger \mathcal{M}_2$ and $\mathcal{M}_1 \ddagger \mathcal{N}_2$, the following communication closed layer (CCL) equivalences hold:

$\mathcal{N}_1 \bullet \mathcal{M}_2 \equiv \mathcal{N}_1 || \mathcal{M}_2$ (IND)
$(\mathcal{N}_1 \bullet \mathcal{N}_2) || \mathcal{M}_2 \equiv \mathcal{N}_1 \bullet (\mathcal{N}_2 || \mathcal{M}_2)$ (CCL-L)
$(\mathcal{N}_1 \bullet \mathcal{N}_2) || \mathcal{M}_1 \equiv (\mathcal{N}_1 || \mathcal{M}_1) \bullet \mathcal{N}_2$ (CCL-R)
$(\mathcal{N}_1 \bullet \mathcal{N}_2) || (\mathcal{M}_1 \bullet \mathcal{M}_2) \equiv (\mathcal{N}_1 || \mathcal{M}_1) \bullet (\mathcal{N}_2 || \mathcal{M}_2)$ (CCL)

5 Partial Order Equivalence and Property Preservation

This section defines the notion of partial order equivalence ($\equiv^*_{po}$) between APAs which is used to prove that if two APAs are po-equivalent then their maximum (resp. minimum) probabilities to reach the set of final states coincide. We start this section by showing that to obtain the maximum (resp. minimum) reachability probabilities for an APA $\mathcal{N}$, it suffices to consider only those implementations that are realisable, i.e., deterministic and satisfy the following condition: every finite path of the implementation is a realisation of some finite abstract execution of $\mathcal{N}$.

Proposition 1 Let $\mathcal{N}$ be an APA. Then we have:

$$\sup_{\mathcal{P} \in [\![\mathcal{N}]\!]} P^{max}_{\mathcal{P}}(S_f) \quad = \quad \sup_{\mathcal{Q} \in [\![\mathcal{N}]\!]} P^{max}_{\mathcal{Q}}(S_f)$$

where $\mathcal{Q}$ is realisable.

The above mentioned result can be easily extended to minimum reachability probabilities. From now on, we use $[\![\mathcal{N}]\!]$ to denote the set of implementations of $\mathcal{N}$ that are realisable. For APAs $\mathcal{N}_1$ and $\mathcal{N}_2$, let $\mathcal{N} = \mathcal{N}_1 \bullet \mathcal{N}_2$. Then we define $\mathcal{N}^{\setminus sync}$ as the APA obtained from $\mathcal{N}$ s.t. it does not have any synchronized transitions (which are present in $\mathcal{N}$ as a result of synchronization over the common actions). Intuitively, this means that abstract executions in $\mathcal{N}$ with synchronized transitions can be rewritten in $\mathcal{N}^{\setminus sync}$ such that for every synchronized transition there is a corresponding sequence of transitions in $\mathcal{N}^{\setminus sync}$ obtained by allowing interleaving on common actions. For example, let $\mathcal{N}$ have a may a-transition which is a result of synchronization of a must a-transition (from $\mathcal{N}_1$), and a may a-transition (from $\mathcal{N}_2$). In this case $\mathcal{N}^{\setminus sync}$ will have a corresponding[1] sequence of transitions, i.e., a must a-transition (from $\mathcal{N}_1$) followed by a may a-transition (from $\mathcal{N}_2$). This transformation is required as we want to establish the result that layered composition is po-equivalent to sequential composition. Note that sequential composition of two APAs does not have synchronized transitions. This means that for any $\mathcal{N} = \mathcal{N}_1 ; \mathcal{N}_2$, it holds $\mathcal{N}^{\setminus sync} = \mathcal{N}$.

[1]In fact, if we would not restrict ourselves to deterministic APAs, then two corresponding transition sequences making a diamond shape would be obtained in $\mathcal{N}^{\setminus sync}$.

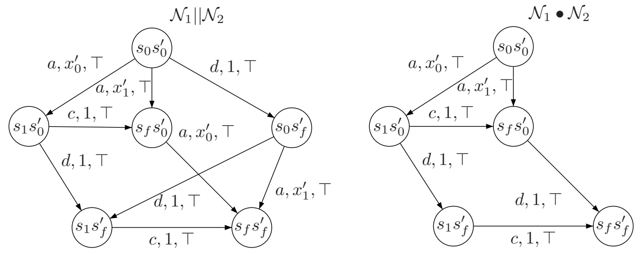

Figure 4. Parallel composition $\mathcal{N}_1 || \mathcal{N}_2$ (left) and layered composition $\mathcal{N}_1 \bullet \mathcal{N}_2$ (right) where $a \dagger d$

Theorem 2 For APAs $\mathcal{N}_1$ and $\mathcal{N}_2$, let $\mathcal{N} = \mathcal{N}_1 \bullet \mathcal{N}_2$. Then we have $\forall \mathcal{P} \in [\![\mathcal{N}]\!] \forall \mathcal{D} \in Adv(\mathcal{P}) \forall \pi \in Paths^{\mathcal{D}}_{fin}(\mathcal{P})$: $\exists \mathcal{P}' \in [\![\mathcal{N}^{\setminus sync}]\!] \exists \mathcal{D}' \in Adv(\mathcal{P}') \exists \pi' \in Paths^{\mathcal{D}'}_{fin}(\mathcal{P}')$: $P^{\mathcal{D}}(\pi) = P^{\mathcal{D}'}(\pi')$.

In stated words, this theorem says that reasoning about $\mathcal{N}^{\setminus sync}$ instead of $\mathcal{N}$ is not a restriction, as the probabilistic behaviour of $\mathcal{N}$ (w.r.t. its implementations) is completely mimicked by $\mathcal{N}^{\setminus sync}$.

Definition 14 [po-equivalence] Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two APAs with transition functions V_1 and V_2 respectively. Let $\rho_1 \in Exec_{fin}(\mathcal{N}_1^{\setminus sync})$ and $\rho_2 \in Exec_{fin}(\mathcal{N}_2^{\setminus sync})$. Then $\rho_1 \equiv_{po} \rho_2$ iff there exist finite abstract executions ρ', ρ'' and $\exists c_1, d_1$ with $c_1 \neq d_1$ s.t. the following holds:

- $\rho_1 = \rho' c_1 \varphi_1 s d_1 \varphi_2 \rho'' \wedge \rho_2 = \rho' d_1 \varphi_2 s' c_1 \varphi_1 \rho''$, where $c_1 \ddagger d_1$.
- $V_1(last(\rho'), c_1, \varphi_1) = V_2(s', c_1, \varphi_1) \wedge V_1(s, d_1, \varphi_2) = V_2(last(\rho'), d_1, \varphi_2)$.
- $\forall i < |\rho'| : V_1(s_i, a_{i+1}, \varphi_{i+1}) = V_2(s_i, a_{i+1}, \varphi_{i+1})$.
- $\forall (|\rho'| + 2) \leq i < (|\rho'| + |\rho''| + 2) : V_1(s_i, a_{i+1}, \varphi_{i+1}) = V_2(s_i, a_{i+1}, \varphi_{i+1})$.

Let $\equiv^*_{po}$, called *po-equivalence*, denote the reflexive, transitive closure of $\equiv_{po}$.

Stated in words, if two finite abstract executions ρ_1, ρ_2 are po-equivalent, then ρ_1 can be obtained from ρ_2 by repeated permutation of adjacent independent actions. Note that the last two conditions of Def. 14 are required to ensure that if for example $\rho' = s_0 c_1 \varphi_1 s_1 d_1 \varphi_2 s_2$ where c_1 is a must transition and d_1 is a may transition in $\mathcal{N}_1^{\setminus sync}$, then c_1, d_1 are also must and may transitions in $\mathcal{N}_2^{\setminus sync}$. This notion of po-equivalence is similar to Mazurkiewicz trace theory [22].

Definition 15 [Layered normal form] Let S_f be the set of final states in $(\mathcal{N}_1 \bullet \mathcal{N}_2)^{\backslash sync}$. Then $\rho \in Exec^{S_f}_{fin}((\mathcal{N}_1 \bullet \mathcal{N}_2)^{\backslash sync})$ is in *layered normal form* (LNF) iff it first consecutively executes actions of $\mathcal{N}_1$ (until $\mathcal{N}_1$ has terminated in S_{f1}), and thereafter consecutively executes actions of $\mathcal{N}_2$.

Let $Exec^{LNF}_{fin}((\mathcal{N}_1 \bullet \mathcal{N}_2)^{\backslash sync})$ be the set of all finite abstract executions in $Exec^{S_f}_{fin}((\mathcal{N}_1 \bullet \mathcal{N}_2)^{\backslash sync})$ that are in LNF. Next we show that for each finite abstract execution of $Exec^{S_f}_{fin}((\mathcal{N}_1 \bullet \mathcal{N}_2)^{\backslash sync})$, a po-equivalent abstract exceution in LNF does exist.

Lemma 1 [LNF existence] Let $\mathcal{N}_1, \mathcal{N}_2$ be two APAs. Then we have $\forall \rho \in Exec^{S_f}_{fin}((\mathcal{N}_1 \bullet \mathcal{N}_2)^{\backslash sync}) \exists \rho' \in Exec^{LNF}_{fin}((\mathcal{N}_1 \bullet \mathcal{N}_2)^{\backslash sync})$ such that $\rho \equiv^*_{po} \rho'$.

Next, we define partial order equivalence between APAs.

Definition 16 [po-equivalence for APAs] Two APAs $\mathcal{N}_1, \mathcal{N}_2$ are said to be *po-equivalent* denoted, $\mathcal{N}_1 \equiv^*_{po} \mathcal{N}_2$, iff for $i \in \{1,2\}$: $\forall \rho_i \in Exec^{S_{fi}}_{fin}(\mathcal{N}_i^{\backslash sync}) \exists \rho_{3-i} \in Exec^{S_{f3-i}}_{fin}(\mathcal{N}_{3-i}^{\backslash sync})$ such that $\rho_i \equiv^*_{po} \rho_{3-i}$.

Theorem 3 For any two APAs $\mathcal{N}_1, \mathcal{N}_2$, it holds: $\mathcal{N}_1 \bullet \mathcal{N}_2 \equiv^*_{po} \mathcal{N}_1 ; \mathcal{N}_2$.

Proposition 2 Let $\mathcal{N}_1, \mathcal{N}_2$ and $\mathcal{M}_1$ be three APAs such that $\mathcal{N}_1 \equiv^*_{po} \mathcal{N}_2$. Then we have:

$$\mathcal{N}_1 || \mathcal{M}_1 \equiv^*_{po} \mathcal{N}_2 || \mathcal{M}_1$$

Theorem 4 [Property preservation] Let $\mathcal{N}_1, \mathcal{N}_2$ be two APAs with set of final states S_{f1} and S_{f2} respectively. If $\mathcal{N}_1 \equiv^*_{po} \mathcal{N}_2$ then we have:

$$\begin{aligned} P^{max}_{\mathcal{N}_1}(S_{f1}) &= P^{max}_{\mathcal{N}_2}(S_{f2}) \\ P^{min}_{\mathcal{N}_1}(S_{f1}) &= P^{min}_{\mathcal{N}_2}(S_{f2}) \end{aligned}$$

In simple words this theorem states that if two APAs are po-equivalent, then their maximum (resp. minimum) reachability probabilities computed over all the implementations coincide. The results of Theorem 3, Proposition 2 and Theorem 4 enable us to replace ; with $\bullet$ and vice versa. This replacement along with CCL laws (Theorem 1) can be used for state space reduction as follows:

State space reduction. Let $\mathcal{N}_1, \mathcal{N}_2$ and $\mathcal{M}_1$ be three APAs, and $\mathcal{N} = (\mathcal{N}_1 ; \mathcal{N}_2) || \mathcal{M}_1$. Let us say we want to compute $P^{max}_{\mathcal{N}}(S_f)$ or $P^{min}_{\mathcal{N}}(S_f)$. Here, S_f is the set of final states in $\mathcal{N}$. Assume $\mathcal{M}_1 \ddagger \mathcal{N}_2$, and $\mathcal{N}_1, \mathcal{N}_2, \mathcal{M}_1$ each consist of 20 states. In this case $\mathcal{N}_1 ; \mathcal{N}_2$ has 39 states which combined with the 20 states of $\mathcal{M}_1$ gives 780 states. We can transform $\mathcal{N}$ in the following way:

$$\begin{array}{cl} (\mathcal{N}_1 ; \mathcal{N}_2) || \mathcal{M}_1 & \\ \equiv^*_{po} & \text{Theorem 3, Proposition 2} \\ (\mathcal{N}_1 \bullet \mathcal{N}_2) || \mathcal{M}_1 & \\ \equiv & \text{CCL-R} \\ (\mathcal{N}_1 || \mathcal{M}_1) \bullet \mathcal{N}_2 & \\ \equiv^*_{po} & \text{Theorem 3} \\ (\mathcal{N}_1 || \mathcal{M}_1) ; \mathcal{N}_2 & \end{array}$$

Note that the transformed system, i.e., $(\mathcal{N}_1 || \mathcal{M}_1) ; \mathcal{N}_2$ has 419 states.

6 Possible Extensions

In this section we briefly discuss the extension of our results to an APA equipped with data variables and rewards.

APA with data. An APA $\mathcal{N}$ can be extended with data variables such that whenever an action is executed its associated data variables are updated according to an assignment. These data variables can take values in some finite range D. The definitions of satisfaction, and refinement can be slightly modified by imposing an extra condition that ensures that related states have the same data valuations. For an APA with data, two actions are said to be dependent if one of the two writes a variable that is read or written by the other action. More formally, two actions a and b are dependent, denoted $a \dagger b$, iff $Write(b) \cap Read(a) \neq \varnothing$ or $Write(a) \cap Read(b) \neq \varnothing$ or $Write(a) \cap Write(b) \neq \varnothing$. Here, $Write(a)$ denotes the set of data variables written by the action a. Similarly, $Read(a)$ denotes the set of data variables read by the action a. Using this dependency relation, our theory can be applied to APAs with data. We do not go into details on these matters here, however, refer the interested reader to [3, 15, 32].

APA with rewards. Rewards are useful for computing, for instance, the resource consumption in a PA under an adversary $\mathcal{D}$. For example in a communication system where a sender and a receiver can transfer messages via an unreliable channel, in this case an interesting measure of interest is the maximum (resp. minimum) expected number of attempts to send a message until correct delivery. An APA $\mathcal{N}$ can be extended with rewards by augmenting state and action pairs, i.e., (s, a) with rewards, which are non-negative real-valued numbers. This intuitively means that in every implementation of $\mathcal{N}$, the state and action pairs corresponding to some state and action pair in $\mathcal{N}$ also have the same reward associated with them. The reward associated with a state and action pair, i.e., (s, a) is earned only when the state s is left by executing an action a. In this setting, the expected reward earned along a finite path π of PA

$\mathcal{P}$ (under some adversary $\mathcal{D}$) before reaching the final state is obtained by taking the product of $P^{\mathcal{D}}(\pi)$ and the total reward earned along π. The preservation results presented in this paper can be extended to APA with rewards, i.e., if two APAs are po-equivalent, then their maximum (resp. minimum) expected rewards earned before reaching the set of final states computed over all the implementations coincide.

7 Conclusion

This paper presented a state-space reduction technique for a network of acyclic APAs, based on the notion of layering. We proposed a layered composition operator, and formulated communication closed layer (CCL) laws. Next, we defined the partial order (po) equivalence between acyclic APAs, and proved that if two APAs are po-equivalent, then their probabilities to reach the set of final states computed over all the implementations coincide. Finally, we discussed the possible extensions of our results.

Acknowledgements. The authors thank Ian Larson for modeling the randomized mutual exclusion algorithm case study in PRISM and conducting the experiments.

References

[1] C. Baier, P. R. D'Argenio, and M. Größer. Partial order reduction for probabilistic branching time. *Electr. Notes Theor. Comput. Sci.*, 153(2):97–116, 2006.

[2] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

[3] S. S. Bauer, K. G. Larsen, A. Legay, U. Nyman, and A. Wasowski. A modal specification theory for components with data. In *FACS*, LNCS 7253, pages 61–78. Springer, 2011.

[4] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1986.

[5] B. Caillaud, B. Delahaye, K. G. Larsen, A. Legay, M. L. Pedersen, and A. Wasowski. Constraint Markov chains. *Theor. Comput. Sci.*, 412(34):4373–4404, 2011.

[6] L. Cheung, N. A. Lynch, R. Segala, and F. W. Vaandrager. Switched PIOA: Parallel composition via distributed scheduling. *Theor. Comput. Sci.*, 365(1-2):83–108, 2006.

[7] B. Delahaye, J.-P. Katoen, K. G. Larsen, A. Legay, M. Pedersen, F. Sher, and A. Wasowski. Abstract probabilistic automata. *Inf. Comput.*, 232:66–116, 2013.

[8] B. Delahaye, J.-P. Katoen, K. G. Larsen, A. Legay, M. L. Pedersen, F. Sher, and A. Wasowski. Abstract probabilistic automata. In *VMCAI*, LNCS 6538, pages 324–339. Springer, 2011.

[9] B. Delahaye, J.-P. Katoen, K. G. Larsen, A. Legay, M. L. Pedersen, F. Sher, and A. Wasowski. New results on abstract probabilistic automata. In *ACSD*, pages 118–127. IEEE, 2011.

[10] B. Delahaye, K. G. Larsen, A. Legay, M. L. Pedersen, and A. Wasowski. APAC: A tool for reasoning about abstract probabilistic automata. In *QEST*, pages 151–152. IEEE, 2011.

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

[12] R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Trans. Program. Lang. Syst.*, 5(1):66–77, 1983.

[13] F. D. Garcia, P. van Rossum, and A. Sokolova. Probabilistic anonymity and admissible schedulers. *CoRR*, abs/0706.1019, 2007.

[14] T. A. Henzinger and J. Sifakis. The embedded systems design challenge. In *FM*, LNCS 4085, pages 1–15. Springer, 2006.

[15] W. Janssen. *Layered Design of Parallel Systems*. PhD dissertation, Universiteit Twente, 1994.

[16] W. Janssen, M. Poel, Q. Xu, and J. Zwiers. Layering of real-time distributed processes. In *FTRTFT*, LNCS 863, pages 393–417. Springer, 1994.

[17] W. Janssen, M. Poel, and J. Zwiers. Action systems and action refinement in the development of parallel systems - an algebraic approach. In *CONCUR*, LNCS 527, pages 298–316. Springer, 1991.

[18] W. Janssen and J. Zwiers. From sequential layers to distributed processes: Deriving a distributed minimum weight spanning tree algorithm (extended anstract). In *PODC*, pages 215–227. ACM, 1992.

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

[20] M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, LNCS 6806, pages 585–591. Springer, 2011.

[21] K. G. Larsen and B. Thomsen. A modal process logic. In *LICS*, pages 203–210, 1988.

[22] A. W. Mazurkiewicz. Basic notions of trace theory. In *REX Workshop*, LNCS 354, pages 285–363, 1988.

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

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

[25] E.-R. Olderog and M. Swaminathan. Layered composition for timed automata. In *FORMATS*, LNCS 6246, pages 228–242. Springer, 2010.

[26] E.-R. Olderog and M. Swaminathan. Structural transformations for data-enriched real-time systems. *Formal Asp. Comput.*, 2014 (to appear).

[27] A. Parma and R. Segala. Axiomatization of trace semantics for stochastic nondeterministic processes. In *QEST*, pages 294–303. IEEE, 2004.

[28] R. Segala. *Modelling and Verification of Randomized Distributed Real Time Systems*. PhD thesis, MIT, 1995.

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

[30] A. Sharma and J.-P. Katoen. Layered reduction for modal specification theories. In *FACS*, LNCS 8348. Springer, 2014 (to appear).

[31] F. Sher and J.-P. Katoen. Compositional abstraction techniques for probabilistic automata. In *IFIP TCS*, LNCS 7604, pages 325–341. Springer, 2012.

[32] M. Swaminathan, J.-P. Katoen, and E.-R. Olderog. Layered reasoning for randomized distributed algorithms. *Formal Asp. Comput.*, 24(4-6):477–496, 2012.