# Static Partial Order Reduction for Probabilistic Concurrent Systems

Álvaro Fernández-Díaz*, Christel Baier†, Clara Benac-Earle*, and Lars-Åke Fredlund*
*Universidad Politécnica de Madrid, Facultad de Informática, Boadilla del Monte, Spain,
*Email:{avalor,cbenac,fred}@babel.ls.fi.upm.es
† Technische Universität Dresden, Fakultät Informatik, Dresden, Germany
†Email:baier@tcs.inf.tu-dresden.de

***Abstract*—Sound criteria for partial order reduction for probabilistic concurrent systems have been presented in the literature. Their realization relies on a depth-first search-based approach for generating the reduced model. The drawback of this dynamic approach is that it can hardly be combined with other techniques to tackle the state explosion problem, e.g., symbolic probabilistic model checking with multi-terminal variants of binary decision diagrams. Following the approach presented by Kurshan et al. for non-probabilistic systems, we study partial order reduction techniques for probabilistic concurrent systems that can be realized by a static analysis. The idea is to inject the reduction criteria into the control flow graphs of the processes of the system to be analyzed. We provide the theoretical foundations of static partial order reduction for probabilistic concurrent systems and present algorithms to realize them. Finally, we report on some experimental results.**

## I. Introduction

The state space explosion problem is known to be one of the major limitations for the application of model checking as a formal verification technique for complex systems. Even for finite-state abstractions the exploration of the full state space of a system model $\mathcal{M}$ can be intractable due to computational and storage restrictions. Along the years, several approaches have been proposed to ease this problem, e.g. symbolic model checking [1], where the verification process does not deal with states individually but with groups of them. This symbolic approach is known to outperform enumerative approaches, also known as explicit model checking, in terms of the size of the state spaces that can be verified.

Other techniques that try to alleviate state space explosion are known as *partial order reductions* (POR), see e.g. [2], [3], [4], [5]. These techniques try to avoid the generation of the complete state space of a system model $\mathcal{M}$ by identifying redundant interleavings and generate a reduced system model with fewer transitions and states. The rough idea is to use the commutativity of independent actions and to ensure that the reduced model covers all possible behaviors of the full model up to permutations of independent actions. There are several desirable conditions for the reduced model $\hat{\mathcal{M}}$. First of all, the partial order reduction techniques should be property-preserving, which means that $\mathcal{M} \models \phi$ if and only if $\hat{\mathcal{M}} \models \phi$ where $\phi$ is the property to be checked for $\mathcal{M}$. Second, the time and memory requirements for generating $\hat{\mathcal{M}}$ and checking $\phi$ for $\hat{\mathcal{M}}$ should be less than those for analyzing $\mathcal{M}$ against $\phi$ directly. The realization of partial order reduction techniques typically relies on a static analysis of the concurrent processes of $\mathcal{M}$ to derive information on the (in)dependency of actions and to apply several heuristics to ensure global conditions on the topology of the underlying directed graphs of $\mathcal{M}$ and $\hat{\mathcal{M}}$. The latter are needed to ensure the equivalence of $\mathcal{M}$ and $\hat{\mathcal{M}}$ for the given property $\phi$. The classical partial order reduction approach, as e.g. realized in the prominent model checker SPIN [6], uses an approach based on a depth-first search (DFS) for the explicit on-the-fly generation and verification of the reduced model. We will refer to this approach as *dynamic* partial order reduction. Static partial order reduction [7], [8] completely relies on a preprocessing step that operates on the control flow graphs of the processes running in parallel. The idea is to inject the reduction criteria in the control graphs. In this way, the realization of the partial order reduction is completely *static*. The clear separation of partial order reduction techniques and the generation and analysis of $\hat{\mathcal{M}}$ yields the advantage that other techniques to tackle the state explosion problem for $\hat{\mathcal{M}}$ are directly applicable. For instance, the modified control flow graphs can serve as input for a symbolic model checker.

This is of particular interest for systems where the explicit model checking approach is known to be less powerful than advanced symbolic verification techniques. Indeed this applies for *probabilistic concurrent systems* modeled by Markov decision processes where the quantitative analysis with model checking techniques relies on a combination of graph algorithms and linear programming techniques.

Partial order reduction criteria for probabilistic concurrent systems have been proposed for next-free linear temporal logic [9], [10] and for branching-time properties specified by next-free formulas of probabilistic computation tree logic [11]. The former have been realized in the model checker LiQuor using a dynamic DFS-based approach for an on-the-fly generation of the reduced model [12]. Although the criteria are stronger than in the non-probabilistic case, empirical results have shown that the reductions in states and transitions are in the same order as for non-probabilistic transition systems. However, the major bottleneck is the linear program which then has to be solved

*This work has been partially supported by the following projects: DESAFIOS10 (TIN2009-14599-C03-03) and PROMETIDOS (P2009/TIC-1465).
†This work has been partially supported by the DFG-project QuaOS and the collaborative research center HAEC (SFB 912)

for an explicit representation of the reduced system.

The goal of this paper is to overcome this limitation by proposing a *static partial order reduction* approach for probabilistic concurrent systems. Following the concept of static partial order reduction developed for non-probabilistic transition systems by Kurshan et al [7], [8], we first provide a generic framework for the static partial order reduction in probabilistic concurrent systems modeled by Markov decision processes. For this purpose, we present reduction criteria that guarantee the preservation of quantitative stutter-invariant linear and branching time properties. We then present an algorithm for realizing these criteria by analyzing and modifying probabilistic control flow graphs. For evaluating the achieved degree of reduction, we considered a randomized version of the dining philosophers [13] and a randomized mutual exclusion protocol [14] and used the symbolic model checker PRISM [15] for comparing the number of states, transitions and the MTBDD-size of the full and the reduced model.

**Outline.** Section II presents our notations for probabilistic concurrent systems and recalls the partial order reduction criteria of [9], [11]. Our main contribution is contained in Section III where we provide the foundations of static partial order reduction for probabilistic concurrent systems and present algorithms for the static realization of the reduction criteria. Section IV reports on our experiments, while Section V contains some concluding remarks.

## II. Preliminaries

If $S$ is a countable set then $Distr(S)$ denotes the set of *probabilistic distributions* on $S$, i.e., functions $\mu : S \to [0,1]$ such that $\sum_{s\in S} \mu(s) = 1$. The *support* of $\mu$, denoted $Supp(\mu)$, consists of all elements $s \in S$ such that $\mu(s) > 0$.

**Markov decision processes (MDP).** *Markov decision processes* (MDP) [16], [17] and variants thereof are widely used as operational model for probabilistic concurrent systems. The rough idea is that in each state $s$ several actions (possibly of different processes) can be enabled. Each action can have a probabilistic effect on the program variables. The selection of an action for the current state is supposed to be nondeterministic. In this paper, we deal with finite-state MDPs with action labels and state labels for formalizing the properties to be verified. Formally, an MDP $\mathcal{M}$ is a tuple

$$\mathcal{M} = (S, Act, P, s_{init}, AP, L) \text{ where}$$

- $S$ is a finite set of *states*,
- $Act$ is a finite set of *actions*,
- $P : S \times Act \to Distr(S)$ is a partial function, called the *transition probability function*,
- $s_{init} \in S$ is the *initial state*,
- $AP$ is a finite set of *atomic propositions*,
- $L : S \to 2^{AP}$ is the *labeling function*.

We write $P(s,\alpha) = \bot$ to denote that $P(s,\alpha)$ is undefined. Action $\alpha \in Act$ is called *enabled* in state $s$ if $P(s,\alpha)$ is defined. In this case, the distribution $P(s,\alpha)$ specifies the probabilities for the successor states of $s$ when executing $\alpha$. $Act(s)$ denotes the set of actions that are enabled in state $s$:

$$Act(s) \;=\; \big\{\alpha \in Act : P(s,\alpha) \neq \bot\big\}.$$

**Transitions, successors.** We write $P(s,\alpha,t)$ rather than $P(s,\alpha)(t)$. If $P(s,\alpha,t) > 0$ then state $t$ is called an $\alpha$-*successor* of $s$. In this case, the triple $(s,\alpha,t)$ is called a *transition* of $\mathcal{M}$ and often written in the form $s \xrightarrow{\alpha} t$. If $\alpha \in Act(s)$ then $Post[\alpha](s) = Supp(P(s,\alpha))$ denotes the set of $\alpha$-successors of state $s$.

**Stutter and visible actions.** Action $\alpha$ is called a *stutter action* or *invisible* if its execution does not affect the truth value of the atomic propositions, i.e., if $L(s) = L(t)$ for all $s,t \in S$ such that $\alpha \in Act(s)$ and $t \in Post[\alpha](s)$. Otherwise $\alpha$ is called *visible*. Let $Vis$ denote the set of visible actions.

**Probabilistic/nonprobabilistic actions.** Action $\alpha$ is called *probabilistic* if there exists a state $s$ with $\alpha \in Act(s)$ and $|Post[\alpha](s)| \geqslant 2$. Otherwise $\alpha$ is called *non-probabilistic*. In this case, the effect of taking action $\alpha$ is deterministic, i.e., each state $s$ where $\alpha$ is enabled has a unique $\alpha$-successor.

**Paths, cycles.** Notions like paths or cycles of $\mathcal{M}$ refer to the underlying labeled graph of $\mathcal{M}$. For example, a *path* in $\mathcal{M}$ is a sequence of consecutive transitions, i.e., it has the form

$$\pi \;=\; s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots$$

A path is called *maximal* if it is either infinite or finite ending up in a terminal state, i.e., a state $s$ with $Act(s) = \varnothing$.

**End components.** An *end component* of an MDP $\mathcal{M}$ is a tuple $\mathcal{E} = (T, A)$ where $T$ is a nonempty subset of $S$ and $A : T \to 2^{Act}$ a function such that

- $\varnothing \neq A(t) \subseteq Act(t)$ and $Supp(P(t,\alpha)) \subseteq T$ for all states $t \in T$ and actions $\alpha \in A(t)$
- the underlying digraph is strongly connected, i.e., whenever $t,u \in T$ then there exists a finite path
$$s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_m} s_m$$
with $s_0 = t$, $s_m = u$ and $\alpha_i \in A(s_{i-1})$ for $1 \leqslant i \leqslant m$.

**Independence of actions.** Two different actions $\alpha,\beta \in Act$ are called *independent* in $\mathcal{M}$ if for all states $s \in S$ where $\alpha,\beta \in Act(s)$:

1. for all states $t$: $P(s,\alpha,t) > 0$ implies $\beta \in Act(t)$
2. for all states $u$: $P(s,\beta,u) > 0$ implies $\alpha \in Act(u)$
3. for all states $w$:

$$\sum_{t\in S} P(s,\alpha,t) \cdot P(t,\beta,w) \;=\; \sum_{u\in S} P(s,\beta,u) \cdot P(u,\alpha,w)$$

Action $\alpha$ is said to *depend* on some action set $A$ if (1) $\alpha \notin A$ and (2) there exists $\beta \in A$ such that actions $\alpha$ and $\beta$ are dependent, i.e., not independent.

**Probabilistic control graphs (PCG).** As specification formalism for probabilistic concurrent systems consisting of $n$ concurrent processes, we use *probabilistic control graphs* (PCG for short) that specify the operational behavior of the processes. The PCG for the processes can then be put in

parallel and unfolded into an MDP that models the stepwise interleaving behavior of the composite system. To ease the notations, we consider here only communication via shared variables. Communication over synchronous or asynchronous channels could be added, but is omitted here.

Formally, we fix a finite set $V$ of typed *variables*. The types, also called *domains*, of the variables (Boolean, integers of a bounded interval, and so on) are irrelevant for the purposes of the paper. The only requirement is that all variables have a finite domain. Let $Eval(V)$ denote the set of *evaluations* $\eta$ that assign to each variable $v \in V$ an element of its domain. $Cond(V)$ denotes the set of Boolean *conditions* for the variables, i.e., propositional formula over atoms that refer to the values of the variables in $V$. For instance, $(x > 5-y) \wedge \neg z \in Cond(V)$ if $x, y$ are integer variables and $z$ is a Boolean variable. For $\eta \in Eval(V)$ and $g \in Cond(V)$, we write $\eta \models g$ to denote that $\eta$ satisfies condition $g$. An *event* $e$ on $V$ is a list of assignments

$$v_1 := expr_1;\ \ldots\ ;\ v_k := expr_k$$

where $v_1, \ldots, v_k$ are pairwise distinct variables in $V$ and $expr_1, \ldots expr_k$ type-consistent expressions built by constants and variables and operators (e.g., arithmetic operators for integers and Boolean connectives for Boolean variables). Let $written(e) = \{v_1, \ldots, v_k\}$ be the set of variables for which $e$ contains an assignment, and $read(e)$ the set of variables that appear in one of the expressions $expr_i$ for $1 \leqslant i \leqslant k$. For instance, if $x, y$ are integer variables and $z, v, w$ Boolean variables then an event $e$ might consist of the three assignments $x := 2x+y$; $y := 5$; $z := \neg w \vee v$. In this example, $written(e) = \{x, y, z\}$ and $read(e) = \{x, y, w, v\}$. The assignments of an event are executed simultaneously in one atomic step. The *effect* of events is formalized by a function

$$Effect : Events(V) \times Eval(V) \to Eval(V).$$

where $Events(V)$ denotes the set of events over $V$ (assuming fixed sets of constants and operators for the domains of the variables). If $e$ is as above and $\eta \in Eval(V)$, then we have $Effect(e, \eta)(x) = \eta(x)$ if $x \in V \setminus \{v_1, \ldots, v_k\}$ and $Effect(e, \eta)(v_i) = [\![expr_i]\!]_\eta$ where $[\![expr]\!]_\eta$ denotes the value that is obtained by evaluating $expr$ under variable valuation $\eta$. A *probabilistic control graph* (PCG) over $V$ is a tuple

$$\mathcal{P} = (Loc, Edges, \ell_{init}) \text{ where}$$

- $Loc$ is a finite set of *locations* (i.e., control states),
- $Edges$ is a finite subset of $Loc \times Cond(V) \times Distr(Events(V) \times Loc)$, called the *edge relation*,
- $\ell_{init} \in Loc$ is the *initial location*.

The second component $g \in Cond(V)$ of an edge $(\ell, g, d) \in Edges$ is called the *guard*. For the last component $d \in Distr(Events(V) \times Loc)$ we require that the support of $d$ is finite. The edge relation specifies conditional transitions. An edge $(\ell, g, d)$ can be taken provided that $\ell$ is the current location of $\mathcal{P}$ and the guard $g \in Cond(V)$ holds for the current variable evaluation. Taking edge $(\ell, g, d)$ has a randomized effect according to distribution $d$, i.e., with probability $d(e, \ell')$ event $e$ will fire and the target location is $\ell'$.

**Probabilistic concurrent system (PCS).** A *probabilistic concurrent system* is a tuple

$$\mathcal{S} = (V, \mathcal{P}_1, \ldots, \mathcal{P}_n, \eta_{init}) \quad \text{where}$$

- $V$ is a finite set of (global) variables,
- $\mathcal{P}_1, \ldots, \mathcal{P}_n$ are probabilistic control graphs over $V$, say $\mathcal{P}_i = (Loc_i, Edges_i, \ell^i_{init})$,
- $\eta_{init} \in Eval(V)$ is an initial evaluation for the variables.

The MDP $\mathcal{M} = \mathcal{M}_\mathcal{S} = (S, Act, P, s_{init}, AP, L)$ modeling the stepwise interleaving behavior of $\mathcal{S}$ has the state space

$$S = Loc_1 \times \ldots \times Loc_n \times Eval(V).$$

That is, the states in $\mathcal{M}$ are tuples $\langle \ell_1, \ldots, \ell_n, \eta \rangle$ where $\ell_i$ is the current location in the PCG $\mathcal{P}_i$ and $\eta$ the current evaluation for the program variables. The action set $Act$ of $\mathcal{M}$ is the disjoint union of the set of the edges in the PCG $\mathcal{P}_1, \ldots, \mathcal{P}_n$. The transition probability function $P$ of $\mathcal{M}$ is defined as follows. Suppose $s \in \langle \ell_1, \ldots, \ell_i, \ldots, \ell_n, \eta \rangle$ is a state in $\mathcal{M}$. Action $\alpha = (\ell, g, d) \in Edges_i$ is enabled in state $s$ if and only if $\ell = \ell_i$ and $\eta \models g$. In this case, if $t$ is a state of the form $t = \langle \ell_1, \ldots, \ell', \ldots, \ell_n, \eta' \rangle$ that differs from $s$ at most in the location $\ell'$ of $\mathcal{P}_i$ and the variable evaluation $\eta'$ then

$$P(s, \alpha, t) = \sum_e d(e, \ell')$$

where $e$ ranges over all events such that $\eta' = Effect(e, \eta)$, otherwise $P(s, \alpha, t) = 0$ The initial state of $\mathcal{M}$ is $s_{init} = \langle \ell^1_{init}, \ldots, \ell^n_{init}, \eta_{init} \rangle$. The set $AP$ of atomic propositions in $\mathcal{M}$ can be an arbitrary finite subset of $Loc_1 \cup \ldots \cup Loc_n \cup Cond(V)$ with the obvious labeling function. One might assume that $AP$ consists of the atomic propositions that appear in the formula to be checked.

**Quantitative properties.** For reasoning about measurable sets of maximal paths and their worst- or best-case probabilities we use standard concepts, such as maximal or minimal probabilities for stutter-invariant measurable path events formalized by linear temporal formulas that do not use the operator next ($\mathrm{LTL}_{\setminus\bigcirc}$-formulas) or branching-time formulas formalized in the fragment of PCTL or PCTL$^*$ without next operator. Details can be found e.g. in [17], [18], [19], [20].

**Partial order reduction.** Partial order reduction [3], [4], [2] has proven to be a successful technique in order to cope with the state explosion problem for concurrent systems. In this paper, we deal with the *ample-set* method that has been originally proposed by Peled for non-probabilistic systems and $\mathrm{LTL}_{\setminus\bigcirc}$-formulas [3]. Extensions for dealing with branching-time properties have been presented in [22]. For verifying quantitative properties of probabilistic concurrent systems, the ample-set approach has been studied independently in [9] and [10] for next-free LTL-formulas and later for branching-time properties specified by $\mathrm{PCTL}^*_{\setminus\bigcirc}$-formulas [11]. The idea of the ample-set method is to identify for state $s$ a small subset $ample(s)$ of $Act(s)$ and to perform the analysis with the

**(A0)** **Emptiness condition:** $ample(s) = \varnothing$ iff $Act(s) = \varnothing$

**(A1)** **Dependence condition:** For each finite path $s \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_m} s_m \xrightarrow{\beta} t$ in the complete model $\mathcal{M}$ such that $m \geqslant 1$ and $\beta$ depends on $ample(s)$ then there exists an index $i \in \{1, \ldots, m\}$ with $\alpha_i \in ample(s)$.

**(A2)** **Stutter condition:** If state $s$ in $\hat{\mathcal{M}}$ is not fully expanded then all actions in $ample(s)$ are invisible.

**(A3)** **End component condition:** For each end component $(T, A)$ in $\hat{\mathcal{M}}$: $\bigcap_{t \in T} Act(t) \subseteq \bigcup_{t \in T} ample(t)$

**(A4)** **Probabilistic condition:** If $s \in \hat{S}$ and there exists a finite path $s \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_m} s_m \xrightarrow{\beta} t$ in the complete model $\mathcal{M}$ such that $m \geqslant 0$ and $\{\alpha_1, \ldots, \alpha_m, \beta\} \cap ample(s) = \varnothing$ and $\beta$ is probabilistic then $|ample(s) = 1$.

Fig. 1. Conditions for the ample sets presented in [9], [21] to preserve quantitative $\mathrm{LTL}_{\setminus\bigcirc}$-properties

submodel $\hat{\mathcal{M}}$ of the original model $\mathcal{M}$ obtained by expanding $s$ only via the actions $\alpha \in ample(s)$ (but ignoring the actions $\alpha \in Act(s) \setminus ample(s)$). Soundness of the reduction is relative to the type of properties to be verified and requires that $\mathcal{M}$ and $\hat{\mathcal{M}}$ satisfy the same properties of the chosen type.

Suppose that the system model is an MDP, say $\mathcal{M} = (S, Act, P, s_{init}, AP, L)$. Given a function $ample : S \to 2^{Act}$ with $ample(s) \subseteq Act(s)$ for all s ∈ S then the reduced model is the following sub-MDP of $\mathcal{M}$:

$$\hat{\mathcal{M}} = (\hat{S}, Act, \hat{P}, s_{init}, AP, \hat{L})$$

The state space $\hat{S}$ of $\hat{\mathcal{M}}$ is the smallest subset of $S$ such that (1) $s_{init} \in \hat{S}$ and (2) $Post[\alpha](s) \subseteq \hat{S}$ for each state $s \in \hat{S}$ and $\alpha \in ample(s)$. The transition probability function of $\hat{\mathcal{M}}$ is given by $\hat{P}(s, \alpha, t) = P(s, \alpha, t)$ if $\alpha \in ample(s)$ and $\hat{P}(\cdot) = 0$ otherwise. The labeling function of $\hat{\mathcal{M}}$ is the restriction of $\mathcal{M}$'s labeling function to the states in $\hat{S}$, i.e., $\hat{L}(s) = L(s)$ for all $s \in \hat{S}$. If $ample(s) = Act(s)$ then state $s$ is said to be *fully expanded*.

Soundness of $\hat{\mathcal{M}}$ for quantitative $\mathrm{LTL}_{\setminus\bigcirc}$-properties means that $\mathcal{M}$ and $\hat{\mathcal{M}}$ have the same extremal (i.e., maximal or minimal) probabilities for all $\mathrm{LTL}_{\setminus\bigcirc}$-formulas. Soundness for $\mathrm{PCTL}^*_{\setminus\bigcirc}$ means that for all $\mathrm{PCTL}^*_{\setminus\bigcirc}$-state formulas $\Phi$:

$$s_{init} \models_{\mathcal{M}} \Phi \text{ iff } s_{init} \models_{\hat{\mathcal{M}}} \Phi.$$

In [9], [21] it has been shown that if the ample sets satisfy the five conditions (A0), (A1), (A2), (A3) and (A4) shown in Figure 1 then $\hat{\mathcal{M}}$ is sound for quantitative $\mathrm{LTL}_{\setminus\bigcirc}$-properties. The emptiness condition (A0), the dependence condition (A1) and the stutter condition (A2) are the same as in the ample-set approach proposed by Peled for non-probabilistic systems [3]. Condition (A3) can be understood as the probabilistic counterpart to the *cycle condition* requiring that each action that is enabled in all states of a cycle in the reduced model is contained in the ample set of some state of that cycle. The cycle condition in combination with (A0), (A1), (A2) preserves $\mathrm{LTL}_{\setminus\bigcirc}$-properties for non-probabilistic transition systems. The implementations of the ample-set method in SPIN or the probabilistic model checker LiQuor deal with a stronger condition than (A3), namely:

**(A3')** **Strong cycle condition:** On each cycle in the reduced model there is a fully expanded state.

Since all end components are cyclic, (A3') is stronger than (A3) for MDPs. The probabilistic condition (A4) is specific to the probabilistic setting. (A4) cannot be dropped since the independence of a probabilistic action $\beta$ from two actions $\alpha$ and $\gamma$ does not imply the commutativity of action $\beta$ and the nondeterministic choice between $\alpha$ and $\gamma$. (A4) is a global condition on $\mathcal{M}$ and can be replaced with the following stronger, local condition (A4'):

**(A4') Branching condition:** For each state $s \in \hat{S}$:
If $ample(s) \neq Act(s)$ then $|ample(s)| = 1$.

The implementation of the partial order reduction for MDPs in LiQuor [12] relies on conditions (A0), (A1), (A2), (A3') and (A4'). Similar to the techniques that have been realized in the SPIN model checker [6], [5], [23], LiQuor first performs a static analysis to derive an underapproximation of the independence relation and uses a dynamic DFS-based approach to generate the reduced MDP. In each expanded state $s = \langle \ell_1, \ldots, \ell_n, \eta \rangle$, the set of enabled actions of each process (i.e., edges in the PCG emanating from location $\ell_i$ where the guard holds for $\eta$) serves as candidate for the ample set. Instead of the dependence condition (A1), LiQuor checks a stronger condition by analyzing the PCG [12]. Stutter condition (A2) and branching condition (A4') are obvious to check. Cycle condition (A3') is treated on-the-fly using the standard cycle detection algorithm that searches for DFS-backward edges. As soon as a DFS-backward edge pointing to state $s$ is encountered, state $s$ will be fully expanded.

Although conditions (A0), (A1), (A2), (A3') and (A4') are sound for non-probabilistic systems and next-free CTL$^*$-formulas [22], (A0), (A1), (A2), (A3') and (A4') are not sufficient to ensure the equivalence of $\mathcal{M}$ and $\hat{\mathcal{M}}$ for $\mathrm{PCTL}_{\setminus\bigcirc}$-formulas [9], [10]. However, soundness for $\mathrm{PCTL}^*_{\setminus\bigcirc}$-formulas is guaranteed when replacing (A4') with the following condition (A4") [11]:

**(A4")** **Strong branching condition:** For each state $s \in \hat{S}$:
If $ample(s) \neq Act(s)$ then $ample(s) = \{\beta\}$
for some non-probabilistic action $\beta$.

## III. Static partial order reduction

As sketched in the end of Section II, the DFS-based on-the-fly generation of the reduced model $\hat{\mathcal{M}}$ as in SPIN or LiQuor fits very well with the cycle condition (A3') that imposes a condition on the cycles in $\hat{\mathcal{M}}$. However, this dynamic approach for ensuring the partial order reduction criteria can hardly be combined with other techniques that aim to combat the state-explosion problem. To use, e.g., the efficient MDP-engine of the symbolic probabilistic model checker PRISM [15], a translation from the explicit representation of $\hat{\mathcal{M}}$ obtained by the dynamic DFS-based approach into PRISM's guarded command input language would be required.

For non-probabilistic systems, Kurshan et al [7], [8] presented alternatives for the global condition (A3') that can be realized by analyzing and modifying the control graphs. The modified control graphs constitute a symbolic representation of the reduced system and provide a good starting point for the application of symbolic or other advanced model checking techniques to the reduced model. The approach proposed by [7], [8] assumes an appropriate choice of an action set $Sticky$ consisting of so-called *sticky action*. For the set of sticky actions, it is required that all visible actions are contained in $Sticky$ and that each cycle in the reduced model contains at least one sticky action. See conditions (S1) and (S2) in Figure 2. [7], [8] use these conditions together with a condition that imposes a constraint on both the sticky actions and the ample sets as basis for static partial order reduction:

**Theorem 1 (see [7], [8]).** *Conditions (S1), (S2) and (AS1) in Figure 2 imply the stutter condition (A2) and the strong cycle condition (A3').*

In the probabilistic setting where the goal is to preserve quantitative properties, conditions (S1), (S2), (A0), (A1) and (AS1) are not sufficient because of the additional constraints imposed by (A4), (A4') or (A4"). However, (AS1) can be replaced with slightly stronger conditions (AS2) or (AS3) in Figure 2 to ensure soundness for quantitative properties:

**Theorem 2 (Basic SPOR for MDP).** *Let $\mathcal{M}$ be an MDP with state space $S$ and action set $Act$, Sticky a subset of $Act$ such that (S1) and (S2) are satisfied and $ample : S \to 2^{Act}$ a function that assigns to each state $s$ a subset of $Act(s)$.*

- *(a) If (A0), (A1) and (AS2) hold then $\mathcal{M}$ and $\hat{\mathcal{M}}$ have the same extremal probabilities for all $\mathrm{LTL}_{\setminus\bigcirc}$-formulas.*
- *(b) If (A0), (A1) and (AS3) hold then $\mathcal{M}$ and $\hat{\mathcal{M}}$ satisfy the same $\mathrm{PCTL}_{\setminus\bigcirc}$-formulas.*

*Proof:* Obviously, (AS2) and (AS3) are stronger than (AS1). Hence, in part (a) and (b) conditions (A2) and (A3') hold by Theorem 1. In part (a), (A4) and even branching condition (A4') is a consequence of (AS2). In part (b), (AS3) implies the strong branching condition (A4"). Thus, the sufficient conditions are fulfilled for each case and the results in [9], [11] yield the claim. ∎

### A. Realization of the basic SPOR

We now address the task how to implement the reduction criteria of the basic SPOR in part (a) or (b) of Theorem 2. Our starting point is a probabilistic concurrent system $\mathcal{S} = (V, \mathcal{P}_1, \ldots, \mathcal{P}_n, \eta_{init})$ consisting of probabilistic control graphs $\mathcal{P}_1, \ldots, \mathcal{P}_n$ over some finite variable set $V$. As before, $\mathcal{M} = (S, Act, P, s_{init}, AP, L)$ denotes the MDP associated with $\mathcal{S}$. The goal is to transform $\mathcal{P}_1, \ldots, \mathcal{P}_n$ into probabilistic control graphs $\hat{\mathcal{P}}_1, \ldots, \hat{\mathcal{P}}_n$ over some finite extension $\hat{V}$ of $V$ such that the MDP $\hat{\mathcal{M}}$ of the composite probabilistic concurrent system

$$\hat{\mathcal{S}} = (\hat{V}, \hat{\mathcal{P}}_1, \ldots, \hat{\mathcal{P}}_n, \hat{\eta}_{init})$$

can be viewed as a submodel of $\mathcal{M}$ obtained by an ample function such that part (a) or (b) or Theorem 2 holds for some action set $Sticky$.

**Static preprocessing.** We first explain how to derive the formal ingredients (set of visible actions, dependence relation, and so on) from the syntax of the probabilistic control graphs. Actually, it suffices to deal with safe approximations (i.e., overapproximations) of the set of visible actions or the dependence relations. In what follows, let $\mathcal{P}_i = (Loc_i, Edges_i, \ell^i_{init})$. Recall that the last component $d$ of an edge $(\ell, g, d)$ in $\mathcal{P}_i$ is a distribution over pairs $(e, \ell')$ consisting of an event $e$, i.e., a list of assignments that is executed atomically, and a location $\ell' \in Loc_i$. We write $Events(\alpha)$ for the set of events $e$ such that $(e, \ell') \in Supp(d)$ for some $\ell' \in Loc_i$ and $\alpha = (l, g, d) \in Act$. Say $\alpha = (\ell, g, d) \in Edges_i$, let

$$written(\alpha) \;=\; \bigcup_{e \in Events(\alpha)} written(e)$$

and $read(\alpha)$ the set of variables $v \in V \setminus written(\alpha)$ such that $v$ appears in an atom of the guard $g$ of $\alpha$ or $v \in read(e)$ for some $e \in Events(\alpha)$. Moreover, we define

$$accessed(\alpha) \;=\; written(\alpha) \cup read(\alpha).$$

For the preservation result in Theorem 2 we deal with formulas where the atomic propositions are taken from $AP$. We write $V|_{AP}$ for the subset of $V$ consisting of variables that appear in the atomic propositions contained in $AP$. The set $Vis$ of visible actions is then defined as the set consisting of all actions $\alpha \in Act$ that might modify some variable that appears in the atomic propositions of the formulas under consideration:

$$Vis \;=\; \big\{\alpha \in Act : written(\alpha) \cap V|_{AP} \neq \varnothing\big\}$$

Here, $Vis$ is an overapproximation that can be computed only having information obtained statically. Let $s = \langle \ell_1, \ldots, \ell_n, \eta \rangle$ be a state in the MDP $\mathcal{M}$. We write $Act_i(s)$ for the set of actions of process $i$ that are enabled in state $s$, i.e., the edges $\alpha = (\ell_i, g, d)$ in $\mathcal{P}_i$ where the guard of $\alpha$ holds for the variable valuation $\eta$ of $s$:

$$Act_i(s) \;=\; \big\{(\ell_i, g, d) \in Edges_i : \eta \models g\big\}$$

The sets $written(\alpha)$ and $read(\alpha)$ can be used to derive an underapproximation of the independence relation over $Act$ as

**(S1)** **Visibility condition:** $Vis \subseteq Sticky$

**(S2)** **Cycle-breaking condition:** For each cycle $s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_m} s_m = s_0$ in $\hat{\mathcal{M}}$:
$$Sticky \cap \{\alpha_1, \alpha_2, \ldots, \alpha_m\} \neq \varnothing$$

**(AS1)** **Sticky condition:** If $ample(s) \neq Act(s)$ then $ample(s) \cap Sticky = \varnothing$.

**(AS2)** **Combined sticky and branching condition:**
If $ample(s) \neq Act(s)$ then $ample(s) = \{\alpha\}$ for some action $\alpha \in Act(s) \setminus Sticky$.

**(AS3)** **Combined sticky and strong branching condition:**
If $ample(s) \neq Act(s)$ then $ample(s) = \{\alpha\}$ for some non-probabilistic action $\alpha \in Act(s) \setminus Sticky$.

Fig. 2. Conditions (S1), (S2) for the set of sticky actions and combined conditions (AS1), (AS2), (AS3) for sticky actions and ample sets

the complement of the following binary relation $D$ on $Act$. Relation $D \subseteq Act \times Act$ is defined as the smallest reflexive, symmetric relation such that for all actions $\alpha \in Act_i$, $\beta \in Act_j$:

$$\text{If } written(\alpha) \cap accessed(\beta) \neq \varnothing \text{ then } (\alpha, \beta) \in D.$$

Then, $(\alpha, \beta) \in Act^2 \setminus D$, $\alpha \neq \beta$ and $i \neq j$ implies the independence of $\alpha$ and $\beta$. Relation $D$ is central to SPOR, as establishes the dependency between actions appearing in (A1).

**Computation of the set of sticky actions.** To compute a set $Sticky$ satisfying the visibility condition (S1) and the cycle-breaking condition (S2) we can apply a similar technique as proposed in [7], [8]. The idea is to analyze the probabilistic control graphs $\mathcal{P}_1, \ldots, \mathcal{P}_n$. For this purpose, we run a depth-first-search (DFS) in the labeled directed graph $\mathcal{G}_i$ where the node-set is $Loc_i \cup Edges_i$ and the edges in $\mathcal{G}_i$ are derived from the edges in $\mathcal{P}_i$ as follows. If $\alpha = (\ell, g, d)$ is an edge in $\mathcal{P}_i$ then $\mathcal{G}_i$ contains edges from $\ell$ to $\alpha$ and from $\alpha$ to each location $\ell'$ where $d(e, \ell') > 0$ for some $e \in Events$. Let $\mathcal{B}_i$ be the set of actions $\alpha$ that belong to some backward edge (either the edge starts in $\alpha$ and points to some previously visited location in the DFS-stack or the edge points to $\alpha$) detected by the DFS in $\mathcal{G}_i$ . Since each cycle in $\mathcal{M}$ induces a cycle in each of the graphs $\mathcal{G}_i$ that perform at least one action along that cycle in $\mathcal{M}$, the action set

$$Sticky = Vis \cup \mathcal{B}_1 \cup \ldots \cup \mathcal{B}_n$$

satisfies conditions (S1) and (S2) in Figure 2.

**Reduced probabilistic concurrent system $\hat{\mathcal{S}}$.** To construct the reduced system $\hat{\mathcal{S}}$ satisfying (A0), (A1) and (AS2) we adapt the procedure to modify the probabilistic control graphs proposed in [7], [8]. (Condition (AS3) will be addressed in the end of Section III-A.)

**Definition 3 (Ample location).** *Location $\ell$ of $\mathcal{P}_i$ is called* ample *if the following conditions (i), (ii) and (iii) hold:*

*(i) For each edge $\alpha = (\ell, g, d)$ in $\mathcal{P}_i$ emanating from $\ell$:*
- *$\alpha \notin Sticky$*
- *for all edges $\beta$ in $\mathcal{P}_1, \ldots, \mathcal{P}_{i-1}, \mathcal{P}_{i+1}, \ldots, \mathcal{P}_n$ we have: $(\alpha, \beta) \notin D$.*

*(ii) If $\alpha_1 = (\ell, g_1, d_1)$ and $\alpha_2 = (\ell, g_2, d_2)$ are different edges from location $\ell$ in $\mathcal{P}_i$ then the guards are disjoint, i.e., $g_1 \wedge g_2 \equiv false$.*

*(iii) The disjunction of all guards of all edges emanating from $\ell$ is a tautology.*

The purpose of the three conditions is as follows. Suppose $s = \langle \ldots, \ell_i, \ldots, \eta \rangle$ is a state of the original MDP $\mathcal{M} = \mathcal{M}_{\mathcal{S}}$ where $\ell_i$ is ample. Then, the action set of process $i$ is viewed as a candidate for the ample set of state $s$. By the first part of condition (i), $Act_i(s)$ does not contain a sticky action (as required in (S1)). The second part of condition (i) ensures that all actions of other processes, i.e., edges in PCG $\mathcal{P}_j$ for some $j \neq i$, are independent from the actions in $Act_i(s)$. This is a crucial prequisite for the cycle-breaking (S2) and the dependence condition (A2). Condition (ii) guarantees that $Act_i(s)$ is a singleton as required in (AS2). The purpose of condition (iii) is to ensure the emptiness condition (A0).

The variable set $\hat{V}$ of $\hat{\mathcal{S}}$ extends the variable set $V$ of the original system $\mathcal{S}$ by $n$ Boolean variables $\mathsf{a}_1, \ldots, \mathsf{a}_n$. Variable $\mathsf{a}_i$ will be used to indicate whether the current location of $\mathcal{P}_i$ is ample. Let $\mathsf{A}_1 = \mathsf{a}_1$ and for $2 \leqslant i \leqslant n$:

$$\mathsf{A}_i \;=\; \mathsf{a}_i \wedge \bigwedge_{1 \leqslant j < i} \neg \mathsf{a}_j \qquad \text{and} \qquad \overline{\mathsf{A}} \;=\; \bigwedge_{1 \leqslant j \leqslant n} \neg \mathsf{a}_j$$

The probabilistic control graph $\hat{\mathcal{P}}_i$ results from $\mathcal{P}_i$ by modifying the edges in $\mathcal{P}_i$ as follows. Edge $\alpha = (\ell_i, g, d)$ in $\mathcal{P}_i$ is replaced with $\hat{\alpha} = (\ell_i, \hat{g}, \hat{d})$ where guard $\hat{g}$ and distribution $\hat{d}$ are defined as follows.

- $\hat{g} = g \wedge guard(\ell_i)$ where $guard(\ell_i) = \mathsf{A}_i$ if $\ell_i$ is ample and $guard(\ell_i) = \overline{\mathsf{A}}$ if $\ell_i$ is not ample.
- Distribution $\hat{d}$ results from $d$ by adding an assignment for variable $\mathsf{a}_i$ to the events in the support of $d$. The purpose of these additional assignments is to indicate whether or not the target location is ample. Formally:
  - if $d(e, \ell') > 0$ and $\ell'$ is ample then
    $$\hat{d}(e; \mathsf{a}_i := true, \ell') = d(e, \ell').$$
  - if $d(e, \ell') > 0$ and $\ell'$ is not ample then
    $$\hat{d}(e; \mathsf{a}_i := false, \ell') = d(e, \ell').$$

  In all other cases, $\hat{d}(\cdot) = 0$.

The initial variable evaluation $\hat{\eta}_{init}$ of $\hat{\mathcal{S}}$ extends $\mathcal{S}$'s initial variable evaluation $\eta_{init}$ by $\hat{\eta}_{init}(\mathsf{a}_i) = true$ if the initial

location $\ell^i_{init}$ of $\mathcal{P}_i$ is ample. Otherwise $\hat{\eta}_{init}(\mathsf{a}_i) = \mathit{false}$.

**Lemma 4.** *For each reachable state $\hat{s} = \langle \ell_1, \ldots, \ell_n, \hat{\eta} \rangle$ in the MDP $\hat{\mathcal{M}}$ for the modified system $\hat{\mathcal{S}}$, we have:*

$$\hat{s} \models \mathsf{a}_i \;\; \textit{iff} \;\; \hat{\eta}(\mathsf{a}_i) = \textit{true} \;\; \textit{iff} \;\; \ell_i \textit{ is ample}$$

As the formulas $\mathsf{A}_1, \ldots, \mathsf{A}_n$ are pairwise disjoint and $\overline{\mathsf{A}}$ is their complement, for each state $s$ of $\mathcal{M}$ we have:

- if some location of $s$ is ample then $s \models \mathsf{A}_i$ for exactly one index $i$
- if none of the locations of $s$ is ample then $s \models \overline{\mathsf{A}}$.

For each action $\hat{\alpha} = (\ell_i, \hat{g}, \hat{d})$ that is enabled in state $\hat{s} = \langle \ell_1, \ldots, \ell_n, \hat{\eta} \rangle$ of $\hat{\mathcal{M}}$, the original action $\alpha = (\ell_i, g, d)$ belongs to $Act_i(s)$ where $s = \langle \ell_1, \ldots, \ell_n, \hat{\eta}\big|_V \rangle$. Thus, the reachable fragment of the MDP $\hat{\mathcal{M}}$ associated with $\hat{\mathcal{S}}$ can be viewed as a sub-MDP of the MDP $\mathcal{M}$ associated with the original system $\mathcal{S}$. Up to isomorphism, the reachable fragment of $\hat{\mathcal{M}}$ results from $\mathcal{M}$ by using as ample sets the set of enabled actions in $\hat{\mathcal{M}}$, i.e., $\mathit{ample}(s)$ consists of the edges $\alpha = (\ell, g, d)$ such that $\hat{\alpha} = (\ell, \hat{g}, \hat{d})$ is enabled in state $\hat{s}$ of $\hat{\mathcal{M}}$.

**Theorem 5 (Soundness of the reduction).** *$\hat{\mathcal{S}}$ provides a sound specification of a reduced MDP where soundness is understood with respect to quantitative $\mathrm{LTL}_{\setminus\bigcirc}$-properties.*

*Proof:* By part (a) of Theorem 2 it suffices to show that conditions (A0), (A1) and (AS2) hold. In what follows, we consider the states and actions (i.e., edges in the PCGs) in the reachable fragment of $\hat{\mathcal{S}}$ as states and actions of $\mathcal{S}$. As before, $Act_i(s)$ denotes the set of actions of process $\mathcal{P}_i$ that are enabled in state $s$ of the full MDP $\mathcal{M}$, while $\mathit{ample}(s)$ is the set of actions that are enabled in state $s$ of the reduced MDP. For each reachable state $s = \langle \ell_1, \ldots, \ell_n, \eta \rangle$ in $\hat{\mathcal{M}}$, exactly one of the following cases applies:

(1) If $\ell_i$ is ample, while $\ell_1, \ldots, \ell_{i-1}$ are not ample, then $s \models \mathsf{A}_i$ and $s \not\models \overline{\mathsf{A}}$. In this case, the guard of each edge $(\ell_j, \hat{g}, \hat{d})$ in $\hat{\mathcal{P}}_j$ for $j \neq i$ is violated, while the edges emanating from $\ell_i$ in $\hat{\mathcal{P}}_i$ that are enabled in state $s$ of $\hat{\mathcal{M}}$ are precisely the edges that emanating from $\ell_i$ in $\mathcal{P}_i$ that are enabled in state $s$ of $\mathcal{M}$. Hence, $\mathit{ample}(s) = Act_i(s)$. Since $\ell_i$ is ample, conditions (i), (ii) and (iii) in Definition 3 hold for $\ell_i$. But then $Act_i(s)$ is a singleton, say $Act_i(s) = \{\alpha\}$. By the first part of condition (i), we get that $\alpha \notin \mathit{Sticky}$.

(2) If none of the locations $\ell_1, \ldots, \ell_n$ is ample, then $s \models \overline{\mathsf{A}}$. In this case, $\mathit{ample}(s) = Act(s)$.

In particular, if $\mathit{ample}(s) \neq Act(s)$ then case (1) applies and we obtain $\mathit{ample}(s) = \{\alpha\}$ for some action $\alpha \in Act \setminus \mathit{Sticky}$ as required in condition (AS2) in Figure 2. The emptiness condition (A0) is obvious by requirement (iii) of Definition 3 for ample locations. The remaining task is to establish the dependence condition (A1). Suppose

$$\pi = s \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_m} s_m \xrightarrow{\beta} t$$

is a path in $\mathcal{M}$ such that $m \geqslant 1$ and $\beta$ depends on $\mathit{ample}(s)$. Assume by contradiction that $\{\alpha_1, \ldots, \alpha_m\} \cap \mathit{ample}(s) = \varnothing$. This is only possible if $s$ is not fully expanded, i.e., case (1) applies for state $s$. Without loss of generality, $m$ is minimal, i.e., $\alpha_1, \ldots, \alpha_m$ are independent from each action in $\mathit{ample}(s)$.

Let $s = \langle \ell_1, \ldots, \ell_n, \eta \rangle$ and $\mathit{ample}(s) = Act_i(s)$. Then, location $\ell_i$ is ample. In particular, condition (i) in Definition 3 holds for location $\ell_i$. By the second part of (i), $(\alpha, \gamma) \notin D$ for each action $\gamma$ of processes $\mathcal{P}_1, \ldots, \mathcal{P}_{i-1}, \mathcal{P}_{i+1}, \ldots, \mathcal{P}_n$ and each action $\alpha \in \mathit{ample}(s)$. Since $D$ is an overapproximation of all pairs of dependent actions, $\beta$ cannot be an action of some process $\mathcal{P}_j$ for $j \neq i$. Thus, $\beta$ is an action of process $\mathcal{P}_i$. Furthermore, the second part of condition (i) in Definition 3 ensures that the guard of any edge emanating from $\ell_i$ in $\mathcal{P}_i$ is not affected by the actions of $\mathcal{P}_j$ for $j \neq i$. But then $\alpha_1, \ldots, \alpha_m$ must be edges in $\mathcal{P}_1, \ldots, \mathcal{P}_{i-1}, \mathcal{P}_{i+1}, \ldots, \mathcal{P}_n$ and $\beta$ must be enabled in state $s$. As a consequence, we obtain $\beta \in Act_i(s) = \mathit{ample}(s)$, contradicting the assumption that $\beta$ depends on $\mathit{ample}(s)$. ∎

**Basic SPOR for branching-time properties.** For the generation of probabilistic control graphs of a reduced system that satisfies the same $\mathrm{PCTL}^*_{\setminus\bigcirc}$-formulas, we can apply the same procedure, except that we have to deal with a stronger notion of ample locations. In this context a location $\ell$ is said to be ample if conditions (i), (ii), (iii) in Definition 3 hold for $\ell$ and the following condition (iv):

(iv) For each edge $\alpha = (\ell, g, d)$ in $\mathcal{P}_i$, the support of $d$ is a singleton, i.e., action $\alpha$ is non-probabilistic.

Then, (AS3) holds. Hence, $\hat{\mathcal{S}}$ and $\mathcal{S}$ are equivalent for $\mathrm{PCTL}^*_{\setminus\bigcirc}$-properties by part (b) of Theorem 2.

### *B. Reachability-aware SPOR*

We now present an alternative technique for modifying the probabilistic control graphs $\mathcal{P}_i$ such that the (reachable fragment of the) reduced model $\hat{\mathcal{M}}$ yields a better (i.e., smaller) abstraction. The idea is to switch back from the branching condition (A4') resp. its strong variant (A4") to the probabilistic condition (A4) in Figure 1. Actually, (A4) leaves more freedom since it permits ample sets with two or more actions for states that are not fully expanded. However, the reduced systems obtained only maintain extremal probabilities for linear time properties. The new algorithm, called *reachability-aware SPOR*, attempts to improve the reduction obtained by the algorithm presented in the previous section when no location of some state $s$ is ample. The idea is that then the union of the action sets $Act_i(s)$ for those processes $\mathcal{P}_i$ where a probabilistic action is reachable along a control path is a candidate for $\mathit{ample}(s)$.

**Static preprocessing.** The computation of (an overapproximation of) the set $\mathit{Vis}$ of visible actions and the dependence relation $D$ as well as the set $\mathit{Sticky}$ is as before (Section III-A). Then, (S1) and (S2) are satisfied. Additionally, we perform a reachability analysis in the underlying directed graphs $\mathcal{G}_i$ associated with the PCG $\mathcal{P}_i$ to identify all potentially probabilistic locations, i.e., locations $\ell \in Loc_i$ that can reach

a location $\ell' \in Loc_i$ via some control path such that $\ell'$ has an outgoing edge representing a probabilistic action. Formally:

**Definition 6 (Potentially probabilistic location).** *Location $\ell$ of $\mathcal{P}_i$ is called* potentially probabilistic *if there is a sequence $(k_0, g_0, d_0)\,(k_1, g_1, d_1)\,\ldots\,(k_m, g_m, d_m)$ of edges in $\mathcal{P}_i$ such that $m \geqslant 0$, $k_0 = \ell$ and there are events $e_0, e_1, \ldots, e_{m-1}$ with $d_j(e_j, k_{j+1}) > 0$ for $0 \leqslant j < m$ and $|Supp(d_m)| \geqslant 2$.*

**Lemma 7.** *If $s \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_m} s_m$ is a path in $\mathcal{M}$ such that $s = \langle \ell_1, \ldots, \ell_n, \eta \rangle$ and $Act_i(s_m)$ contains some probabilistic action, then location $\ell_i$ is potentially probabilistic.*

**Definition 8 (Weakly ample locations).** *Location $\ell$ of PCG $\mathcal{P}_i$ is said to be* weakly ample *if it satifies conditions (i) and (iii) in Definition 3.*

**Reduced probabilistic concurrent system $\hat{\mathcal{S}}$.** To ensure that the constructed probabilistic concurrent system $\hat{\mathcal{S}}$ satisfies conditions (A0), (AS1) and (A4) we deal with a variant of the technique presented in the previous section. The variable set $\hat{V}$ of $\hat{\mathcal{S}}$ extends the original variable set by three Boolean variables $\mathsf{a}_i, \mathsf{w}_i, \mathsf{p}_i$ for each probabilistic control graph $\mathcal{P}_i$. The role of $\mathsf{a}_i$ is as before (Section III-A). Variable $\mathsf{w}_i$ indicates whether the current location of $\mathcal{P}_i$ is weakly ample, while variable $\mathsf{p}_i$ states whether the current location of $\mathcal{P}_i$ is potentially probabilistic. The initial variable evaluation $\hat{\eta}_{init}$ of $\hat{\mathcal{S}}$ extends $\eta_{init}$ by:

$$\begin{array}{lcll} \hat{\eta}_{init}(\mathsf{a}_i) & = & true & \text{iff} \quad \ell^i_{init} \text{ is ample} \\ \hat{\eta}_{init}(\mathsf{w}_i) & = & true & \text{iff} \quad \ell^i_{init} \text{ is weakly ample} \\ \hat{\eta}_{init}(\mathsf{p}_i) & = & true & \text{iff} \quad \ell^i_{init} \text{ is potentially probabilistic} \end{array}$$

Suppose now that $s = \langle \ell_1, \ldots, \ell_n, \eta \rangle$ is a state in $\mathcal{M}$ where none of the locations is ample. To ensure (A4), the reachability-aware SPOR includes all actions in $Act_i(s)$ for those processes $\mathcal{P}_i$ where the current location $\ell_i$ in state $s$ is potentially probabilistic. The actions of all other processes $\mathcal{P}_j$ where location $\ell_j$ is not potentially probabilistic can be ignored in state $s$, provided that all potentially probabilistic locations $\ell_i$ are weakly ample (this ensures (AS1)) and at least one such index $i$ exists (this ensures (A0)).

We first explain a simplified version where in the above case the ample set of $s$ contains the actions of all processes where the current location is weakly ample. In the end of this section, we present refinements of this approach.

Formulas $\mathsf{A}_i$ and $\overline{\mathsf{A}}$ are defined as before (Section III-A). The following formula $\mathsf{B}$ characterizes those states $s$ where all potentially probabilistic locations are weakly ample, while $\mathsf{W}$ asserts the existence of some weakly ample location.

$$\mathsf{B} = \bigwedge_{1 \leqslant i \leqslant n} (\neg \mathsf{p}_i \vee \mathsf{w}_i) \qquad \mathsf{W} = \bigvee_{1 \leqslant i \leqslant n} \mathsf{w}_i \qquad \mathsf{C} = \overline{\mathsf{B}} \vee \overline{\mathsf{W}}$$

where $\overline{\mathsf{W}} = \neg\mathsf{W}$ and $\overline{\mathsf{B}} = \neg\mathsf{B}$. For $\ell \in Loc_i$, we define

$$guard(\ell_i) \;=\; \begin{cases} \mathsf{A}_i & : \text{ if } \ell_i \text{ is ample,} \\ \overline{\mathsf{A}} \wedge \big((\mathsf{B} \wedge \mathsf{w}_i) \vee \mathsf{C}\big) & : \text{ otherwise.} \end{cases}$$

Thanks to Lemma 9 below, $guard(\ell_i)$ could be simplified to $\overline{\mathsf{A}} \wedge \mathsf{C}$ if $\ell_i$ is not weakly ample.

To obtain $\hat{\mathcal{P}}_i$ from $\mathcal{P}_i$ we replace each edge $(\ell_i, g, d)$ in $\mathcal{P}_i$ with the edge $(\ell_i, \hat{g}, \hat{d})$ where the new guard is defined by

$$\hat{g} = g \wedge guard(\ell_i).$$

Distribution $\hat{d}$ results from $d$ by replacing each pair $(e, \ell')$ in the support of $d$ with $(\hat{e}, \ell')$ where $\hat{e}$ extends $e$ by assignments for the variable $\mathsf{a}_i, \mathsf{w}_i, \mathsf{p}_i$ according to the mode of the target location $\ell'$:

$$\begin{array}{rcll} \mathsf{a}_i & := & \begin{cases} true & \text{if } \ell' \text{ is ample} \\ false & \text{otherwise} \end{cases} \\ \mathsf{w}_i & := & \begin{cases} true & \text{if } \ell' \text{ is weakly ample} \\ false & \text{otherwise} \end{cases} \\ \mathsf{p}_i & := & \begin{cases} true & \text{if } \ell' \text{ is potentially probabilistic} \\ false & \text{otherwise} \end{cases} \end{array}$$

The choice of the initial variable evaluation and these additional assignments in the events of $\hat{\mathcal{P}}_1, \ldots, \hat{\mathcal{P}}_n$ ensure:

**Lemma 9.** *For each reachable state $\hat{s} = \langle \ell_1, \ldots, \ell_n, \hat{\eta} \rangle$ in the MDP $\hat{\mathcal{M}}$ for the modified system $\hat{\mathcal{S}}$, we have:*

$$\begin{array}{l} \hat{s} \models \mathsf{a}_i \;\; \textit{iff} \;\; \hat{\eta}(\mathsf{a}_i) = true \;\; \textit{iff} \;\; \ell_i \textit{ is ample} \\ \hat{s} \models \mathsf{w}_i \;\; \textit{iff} \;\; \hat{\eta}(\mathsf{w}_i) = true \;\; \textit{iff} \;\; \ell_i \textit{ is weakly ample} \\ \hat{s} \models \mathsf{p}_i \;\; \textit{iff} \;\; \hat{\eta}(\mathsf{p}_i) = true \;\; \textit{iff} \;\; \ell_i \textit{ is potentially probabilistic} \end{array}$$

As in the previous section, this observation permits to consider $\hat{\mathcal{S}}$ as a specification for a reduced MDP $\hat{\mathcal{M}}$ obtained by the ample function that assigns to each state $s$ the set $ample(s)$ of actions that are enabled in $s$ viewed as a state of the MDP associated with $\hat{\mathcal{S}}$. (As before, $Act_i(s)$ is the set of actions that are enabled in state $s$ of the (full) MDP $\mathcal{M}$ for $\mathcal{S}$.)

**Lemma 10.** *For each $i \in \{1, \ldots, n\}$ and each state $s$ of $\hat{\mathcal{M}}$:*

$$Act_i(s) \subseteq ample(s) \textit{ iff } s \models \mathsf{A}_i \vee \big(\overline{\mathsf{A}} \wedge \mathsf{B} \wedge \mathsf{w}_i\big) \vee \big(\overline{\mathsf{A}} \wedge \mathsf{C}\big)$$

In particular, we can identify three cases for the ample set of state $s = \langle \ell_1, \ldots, \ell_n, \eta \rangle$:

(1) If some location of $s$ is ample then $s \models \mathsf{A}_i$ and $ample(s) = Act_i(s)$ where $i$ is the smallest index such that $\ell_i$ is ample. In this case,

(2) If no locations of $s$ is ample, but all potentially probabilistic locations $\ell_i$ are weakly ample and at least one location $\ell_i$ is weakly ample then $s \models \overline{\mathsf{A}} \wedge \mathsf{B} \wedge \mathsf{W}$ and

$$ample(s) \;=\; \bigcup_{\substack{1 \leqslant i \leqslant n \\ s \models \mathsf{w}_i}} Act_i(s)$$

(3) If neither (1) nor (2) applies then $s \models \overline{\mathsf{A}} \wedge \mathsf{C}$ and state $s$ is fully expanded.

Because of case (2), condition (AS2) cannot be guaranteed. However, the constructed system $\hat{\mathcal{S}}$ is equivalent to the original system $\mathcal{S}$ with respect to quantitative $\mathrm{LTL}_{\setminus\bigcirc}$-formulas as the original conditions of [9] in Figure 1 hold:

**Theorem 11 (Soundness of reachability-aware SPOR).** *Conditions (A0), (A1), (A2), (A3') and (A4) hold for the ample sets induced by the constructed system $\hat{\mathcal{S}}$.*

Several improvements of the presented version of reachability-aware SPOR are possible. In the presented construction, if $s \models \overline{\mathsf{A}} \wedge \mathsf{B} \wedge \mathsf{W}$ then $ample(s)$ has been defined as the set of all actions in $Act_i(s)$ where $i$ ranges over all indices such that $\ell_i$ is weakly ample. However, it suffices to define $ample(s)$ as the union of the action sets $Act_i(s)$ where $\ell_i$ is potentially probabilistic (in which case $\ell_i$ is weakly ample as $s \models \mathsf{B}$), provided there is at least one such index $i$. If no location of $s$ is potentially probabilistic, then we might choose the smallest index $i$ such that $\ell_i$ is weakly ample and define $ample(s) = Act_i(s)$. The latter corresponds to the non-probabilistic approach of [7], [8].

These improvements can be achieved by redefining $guard(\ell_i)$ accordingly. We put

$$\mathsf{W}_i = \mathsf{w}_i \wedge \bigwedge_{1 \leqslant j < i} \neg \mathsf{w}_j \qquad \mathsf{P} = \bigvee_{1 \leqslant j < n} \mathsf{p}_j \qquad \overline{\mathsf{P}} = \neg \mathsf{P}$$

and redefine $guard(\ell_i)$ for the case where $\ell_i$ is not ample by the condition:

$$\overline{\mathsf{A}} \wedge \big( \ (\overline{\mathsf{P}} \wedge \mathsf{W}_i) \vee (\mathsf{P} \wedge \mathsf{B} \wedge \mathsf{p}_i) \vee \mathsf{C} \ \big)$$

This condition can be simplified according to the mode of $\ell_i$. For instance, the constraint $\overline{\mathsf{P}} \wedge \mathsf{W}_i$ could be dropped if $\ell_i$ is not weakly ample or $\ell_i$ is potentially probabilistic.

## IV. Experiments

To obtain some empirical feedback regarding the potential for reduction of the algorithms, we extended the modeling engine of the tool LiQuor [24], [12] and applied the symbolic model checker PRISM [15] to compare the time and memory requirements for analyzing the original and the reduced model. LiQuor has been used to generate the reduced models using the proposed algorithms of Sections III-A and III-B for realizing the reduction criteria purely statically on the level the probabilistic control graphs. Our extension of LiQuor supports the generation of the modified probabilistic control graphs by computing overapproximations of the dependence relation, the action sets *Vis* and *Sticky* and by reporting on the new location labelings (ample, weakly ample and potentially probabilistic). This information provided by LiQuor has been used to generate descriptions of the probabilistic control graphs of the reduced system in PRISM's input language. The translation of LiQuor's output into PRISM's input has been done manually.

We considered two standard examples for randomized protocols: the randomized dining philosophers [13] and the randomised mutual exclusion proposed by Pnueli and Zuck [14]. The results are presented in Figures 3 and 4. In both systems, all locations are potentially probabilistic. Hence, the reduction achieved by applying both proposed algorithms is the same. The tables show that the number of states and transitions is indeed reduced and that the reduction ratio increases proportionally to the number of processes. In the case of the randomized mutual exclusion protocol (Figure 4), the reduction is obtained in both the number of nodes and transitions. For the randomized dining philosophers (Figure 3), the reduction leads to a blowup of the size of the MTBDD for the transition probability function. Although the introduction of new variables $\mathsf{a}_i, \mathsf{w}_i, \mathsf{p}_i$ might be a possible explanation of this phenomenon, we expect that additional heuristics to find good variable orderings for the reduced systems might avoid the blowup for the symbolic model representation. In our experiments, it turned out to be beneficial to declare the Boolean variables $\mathsf{a}_i, \mathsf{w}_i, \mathsf{p}_i$ as (three groups of) global variables rather than using them as local variables in the PRISM modules.

The time required to generate the MTBDD-representations for the original model $\mathcal{M}$ and the reduced MDP $\hat{\mathcal{M}}$ was very similar. E.g., it took 10 seconds to build the unreduced model of the randomized mutual exclusion protocol with 10 processes, while it required 8.9 seconds to construct the reduced model. The time difference for model checking the same property in both systems is of tenths of milliseconds. Finally, we can also conclude that the algorithm is efficient, as the difference of time to perform model checking of the properties for both reduced and unreduced models was negligible, while the memory requirements decreased.

As stated above, the probabilistic control graphs of both examples have no locations that are not potentially probabilistic. To study the impact of reachability-aware SPOR against basic SPOR, we have made some experiments with synthetic examples of probabilistic concurrent systems containing locations that are not potentially probabilistic. As expected, the reachability-aware SPOR of Section III-B achieves a better reduction than the basic SPOR algorithm of Section III-A.

## V. Conclusion and future work

The main motivation for this paper was to provide the foundations for combining partial order reduction techniques with other advanced methods for the quantitative analysis of probabilistic concurrent systems. We proved that the techniques proposed by Kurshan et al. for systems modeled by ordinary nonprobabilistic transition systems can be adapted to the probabilistic setting. We introduced criteria that are sound for stutter-invariant linear and branching-time properties and presented algorithms to realize the reduction by modifying the probabilistic control graphs.

Our experiments illustrate that the quality of the reduction in terms of the MDP-size (number of states and transitions) is fairly good. For applying static partial order techniques as a preprocessing step for a symbolic MTBDD-based quantitative analysis, the example with the dining philosophers exposes the task to develop heuristics for improving the variable orderings in the MTBDD representing the reduced model or for finding good state encodings by variables used in the MTBDD-representation. This aspect as well as improvements of the static preprocessing to overapproximate the dependency relations or to find suitable sets of sticky actions will be addressed in future work. Furthermore, it would be interesting to investigate whether the cycle-breaking condition can be replaced with an analogous condition for end components.

| number of processes | number of states | number of transitions | nodes in the MTBDD | node reduction | transition reduction |
|---|---|---|---|---|---|
| unreduced dining philosophers | | | | | |
| 4 | 9440 | 48656 | 8833 | | |
| 5 | 93068 | 599600 | 24738 | | |
| 6 | 917424 | 7092696 | 64171 | | |
| 7 | 9043420 | 81568144 | 158390 | | |
| reduced dining philosophers | | | | | |
| 4 | 8215 | 28324 | 12297 | $-39.22\%$ | $41.79\%$ |
| 5 | 75082 | 291320 | 35441 | $-43.27\%$ | $51.41\%$ |
| 6 | 679228 | 2875866 | 93616 | $-45.89\%$ | $59.45\%$ |
| 7 | 6093934 | 27600790 | 234151 | $-47.83\%$ | $66.16\%$ |

Fig. 3. Results for the dining philosophers

| number of processes | number of states | number of transitions | nodes in the MTBDD | node reduction | transition reduction |
|---|---|---|---|---|---|
| unreduced mutual exclusion | | | | | |
| 4 | 27600 | 136992 | 12355 | | |
| 6 | 3377344 | 25470144 | 33297 | | |
| 8 | $3,9 \cdot 10^8$ | $3,9 \cdot 10^9$ | 62871 | | |
| 10 | $4,4 \cdot 10^{10}$ | $5,6 \cdot 10^{11}$ | 101077 | | |
| reduced mutual exclusion | | | | | |
| 4 | 21040 | 97360 | 10655 | $13.76\%$ | $28.93\%$ |
| 6 | 2482432 | 17598112 | 26310 | $20.98\%$ | $30.9\%$ |
| 8 | $2,8 \cdot 10^8$ | $2,7 \cdot 10^9$ | 47697 | $24.14\%$ | $30.77\%$ |
| 10 | $3,1 \cdot 10^{10}$ | $3,9 \cdot 10^{11}$ | 74816 | $25.98\%$ | $31.58\%$ |

Fig. 4. Randomised mutual exclusion protocol

## REFERENCES


[1] J. Burch, E. Clarke, K. McMillan, D. Dill, and L. Hwang, "Symbolic model checking: $10^{20}$ states and beyond," *Information and computation*, vol. 98, no. 2, pp. 142–170, 1992.

[2] A. Valmari, "Stubborn sets for reduced state space generation," *Advances in Petri Nets*, vol. 483, pp. 491–515, 1991.

[3] D. Peled, "All from one, one for all: on model checking using representatives," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, vol. 697, 1993, pp. 409–423.

[4] P. Godefroid, J. van Leeuwen, J. Hartmanis, G. Goos, and P. Wolper, *Partial-order methods for the verification of concurrent systems: an approach to the state-explosion problem*. Springer, 1996, vol. 1032.

[5] G. Holzmann and D. Peled, "An improvement in formal verification," in *Proceedings of the 7th IFIP WG6*, vol. 1, 1994, pp. 197–211.

[6] G. Holzmann, *The SPIN model checker: Primer and reference manual*. Addison-Wesley, 2004, vol. 1003.

[7] R. Kurshan, V. Levin, M. Minea, D. Peled, and H. Yenigün, "Static partial order reduction," in *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, ser. Lecture Notes in Computer Science, vol. 1384, 1998, pp. 345–357.

[8] ——, "Combining software and hardware verification techniques," *Formal Methods in System Design*, vol. 21, no. 3, pp. 251–280, 2002.

[9] C. Baier, M. Größer, and F. Ciesinski, "Partial order reduction for probabilistic systems," in *Quantitative Evaluation of Systems (QEST)*. IEEE Computer Society, 2004, pp. 230–239.

[10] P. D'Argenio and P. Niebert, "Partial order reduction on concurrent probabilistic programs," in *Quantitative Evaluation of Systems (QEST)*, vol. 4, 2004, pp. 240–249.

[11] C. Baier, P. D'Argenio, and M. Groesser, "Partial order reduction for probabilistic branching time," in *Quantitative Aspects of Programming Languages (QAPL)*, ser. Electronic Notes in Theoretical Computer Science, vol. 153, no. 2. Elsevier, 2006, pp. 97–116.

[12] F. Ciesinski, "High-level modelling and efficient analysis of randomized protocols," Ph.D. dissertation, Technische Universität Dresden, 2011.

[13] D. Lehmann and M. Rabin, "On the advantages of free choice: a symmetric and fully distributed solution to the dining philosophers problem," in *Principles of programming languages*. ACM, 1981, pp. 133–138.

[14] A. Pnueli and L. Zuck, "Verification of multiprocess probabilistic protocols," *Distributed Computing*, vol. 1, no. 1, pp. 53–72, 1986.

[15] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, vol. 6806, 2011, pp. 585–591.

[16] R. Bellman, "A markovian decision process," DTIC Document, Tech. Rep., 1957.

[17] M. Puterman, *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc., 1994.

[18] A. Bianco and L. De Alfaro, "Model checking of probabilistic and nondeterministic systems," in *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, ser. Lecture Notes in Computer Science, vol. 1026, 1995, pp. 499–513.

[19] C. Courcoubetis and M. Yannakakis, "The complexity of probabilistic verification," *Journal of the ACM (JACM)*, vol. 42, no. 4, pp. 857–907, 1995.

[20] C. Baier and J. Katoen, *Principles of model checking*. MIT press, 2008, vol. 950.

[21] M. Größer, "Reduction methods for probabilistic model checking," Ph.D. dissertation, Technische Universität Dresden, 2008.

[22] R. Gerth, R. Kuiper, D. Peled, and W. Penczek, "A partial order approach to branching time logic model checking," in *Third Israel Symposium on the Theory of Computing and Systems*. IEEE, 1995, pp. 130–139.

[23] D. Peled, "Partial order reduction: Linear and branching time logics and process algebras," in *Partial Order Methods in Verification*, vol. 29, no. 10. American Mathematical Society, 1997, pp. 79–88.

[24] F. Ciesinski and C. Baier, "Liquor: A tool for qualitative and quantitative linear time analysis of reactive systems," in *Quantitative Evaluation of Systems (QEST)*. IEEE Computer Society, 2006, pp. 131–132.