Partial Order Reduction for Probabilistic Systems

Christel Baier, Marcus Größer*, Frank Ciesinski**
Institut für Informatik I, Universität Bonn
Römerstrasse 164, D-53117 Bonn, Germany
baier|groesser|ciesinsk@cs.uni-bonn.de

Abstract

In the past, several model checking algorithms have been proposed to verify probabilistic reactive systems. The techniques to combat the state-explosion problem have mainly concentrated on symbolic methods with variants of decision diagrams or abstraction methods. In this paper, we show how partial order reduction with a variant of Peled's ample set method can be applied in the context of LTL model checking for probabilistic systems modelled by Markov decision processes.

1 Introduction

Probabilistic phenomena occur naturally when modelling systems that rely on a (distributed) randomized algorithm or systems with unreliable modules and stochastic assumptions about their faulty behavior.[1] In the past, many of the specification formalisms and verification techniques that have been established in the non-probabilistic setting have been adapted to reason about quantitative aspects of probabilistic systems, e.g. with the help of process equivalences and model checking against temporal logical specifications. In the context of specifiying probabilistic systems against formulas of Linear Temporal Logic (LTL) [39] a specification consists of a LTL-formula φ that expresses a certain path property and a probability bound, e.g. "$=1$", "$\geq p$" or "$< p$" for some $p \in [0,1]$. Hence, LTL can serve to formulate qualitative properties such as *"with probability 1 any request will eventually be answered"* or quantitative properties such as *"there is a 98 % chance to reach a goal state"* or *"the probability that a waiting process is never allowed to enter its critical section is less than 0.005"*. Verification algorithms for qualitative or quantitative LTL specifications rely on modifications of the model checking techniques for non-probabilistic systems (such as graph algorithms to explore the state space and algorithms that construct an automaton for the given formula) and their combination with numerical methods to solve linear equation systems or linear programming problems [45, 46, 9, 40, 10, 13, 11, 6]. Thus, the state-explosion problem is as least as relevant (or even more) than in the non-probabilistic setting.

To reason about non-probabilistic systems, a variety of methods have been developed to avoid the state-explosion problem, including symbolic model checking with binary decision diagrams, partial order reduction, abstraction techniques and reasoning with symmetries, see e.g. [8] for an overview. In the probabilistic setting, research on methods that combat the state-explosion problem has mainly concentrated on symbolic techniques with variants of decision diagrams [20, 2, 5, 22, 34, 31]. For instance, the PCTL-model checkers PRISM [30], ProbVERUS [21] and RAPTURE [28] are based on a symbolic representation of the system by a multiterminal BDD. Beside the symbolic DD-based methods there is a variety of results about the state-aggregation with (formula-independent) bisimulation-like equivalences [27, 3, 38, 7] and property-driven abstraction techniques [12, 25, 26].

As far as we know this paper is the first that investigates *partial order reduction* for probabilistic systems. The starting point is a description of an asynchronous parallel system by a representation of the subsystems that run in parallel, e.g., as in the (non-probabilistic) model checker SPIN [23]. The rough idea behind partial order reduction in non-probabilistic systems [24, 42, 18, 19, 37] is to construct a reduced state graph by abolishing redundancies in the transition system that origin from the *interleaving* of independent activities that are executed in parallel. For independent actions α and β, the interleaving semantics represents their parallel execution by the nondeterministic choice between the action-sequences $\alpha\beta$ and $\beta\alpha$. As $\alpha\beta$ and $\beta\alpha$ have the same effect to the control and program variables, and thus, lead to the same state, the investigation of one order ($\alpha\beta$ or $\beta\alpha$) as a representative for both suffices under certain side conditions. More general, instead of constructing the full system M, the goal is to generate a sub-system $\hat{M}$ such that any path ς in M is "represented" in $\hat{M}$ by a path $\hat{\varsigma}$ where ς and $\hat{\varsigma}$ agree up to permutations of independent sequent actions. Of course, further conditions have to be taken into account. First, M and $\hat{M}$ have to be "equivalent" according to the property to be verified. Second, the algorithmic construction and analysis of $\hat{M}$ should be more efficient than model checking the full system M.

* Supported by the DFG-Project "VERIAM" and the DFG-NWO-Project "VOSS".
** Supported by the DFG-NWO-Project "VOSS".

[1] We emphasize here the discrete (time-abstract) case and do not consider stochastic models that specify the delay of transitions as it is the case e.g. in continuous-time Markov chains.

To adapt these techniques for probabilistic systems, we have to reason about the equivalence of schedulers stating that the probability distributions for stutter equivalence classes of paths agree. Although this leads to some technical difficulties in the correctness proof, we show in this paper that with only slight modifications *Peled's ample set method* [35] can be adapted for the probabilistic setting and the verification of $\mathrm{LTL}_{\setminus X}$ properties (or even stutter insensitive ω-regular properties). Compared to Peled's approach, we need an additional condition for the ample sets that has no non-probabilistic counterpart. Nevertheless, our criteria applied to an ordinary transition system, viewed as a probabilistic system with Dirac distributions only, reduce *exactly* to Peled's conditions for verifying linear time properties.

The heuristics to generate "small" ample sets that were developed for non-probabilistic systems can also be used for the probabilistic setting. One possibility is to use *exactly the same* techniques as they are known for non-probabilistic systems and *branching-time* properties [16, 36] as these cover our auxiliary condition. A possibly better reduction can be achieved by reformulating the algorithms for non-probabilistic systems and linear time properties (see e.g. [36, 8, 29]) in such a way that the auxiliary condition is taken into account.

Organization of the paper. Section 2 briefly summarizes the preliminaries concerning our model (Markov decision processes). The ample-set method for Markov decision processes is explained in Section 3. Heuristics for the construction of the reduced systems are discussed in Section 4. Section 5 is devoted to the correctness proof. Section 6 concludes the paper.

The submitted version contains an appendix which provides some technical details of the correctness proof. To meet the length restriction, the appendix will be removed for the final version, but will be made available through a technical report and our webpage.

2 Preliminaries

We use Markov decision processes (MDP for short) as operational model for asynchronous probabilistic systems. In contrast to Markov chains, non-determinism and probabilism coexists in MDPs. The non-determinism is essential to describe the parallel execution of independent action by *interleaving* which is the crucial ingredient for partial order reduction. E.g., given a description of the processes that run in parallel and communicate via shared variables and/or channels by a guarded command language with a probabilistic choice operator, an operational semantics can be provided by means of an MDP [1].

In an MDP, any state s might have several outgoing action-labeled transitions, each of them is associated with a probability distribution which yields the probabilities for the successor states. As in [41, 33, 13] we assume here that for any state s, the outgoing transitions of s have different action labels. (This corresponds to the so-called reactive model in the classification of [44].) In addition, we assume here a labelling function that attaches to any state s a set of atomic propositions that are assumed to hold in state s. The atomic propositions will serve as atoms to formulate the desired properties by formulas (or, more general, by automata specifications). E.g., the atomic propositions can assert that a certain subprocess is at a certain control point or that a program variable has a certain value.

Definition 1. [MDP, e.g. [41]] *A MDP is a tuple* $M = (S, \mathsf{Act}, \mathsf{P}, s_{init}, \mathsf{AP}, \mathsf{L})$, *where S is a finite set of states,* Act *a finite set of actions,* $\mathsf{P} : (S \times \mathsf{Act} \times S) \to [0,1]$ *is the (three-dimensional) probability matrix,* $s_{init} \in S$ *the initial state,* AP *a finite set of atomic propositions, and* $\mathsf{L} : S \to 2^{\mathsf{AP}}$ *is a labeling function.* $\mathsf{Act}(s)$ *denotes the set of actions that are enabled in state s, i.e. the set of actions* $\alpha \in \mathsf{Act}$ *such that* $\mathsf{P}(s,\alpha,t) > 0$ *for some state* $t \in S$. *For any state* $s \in S$, *we require that* $\mathsf{Act}(s) \neq \emptyset$ *and* $\sum_{s' \in S} \mathsf{P}(s,\alpha,s') = 1$ *for any action* $\alpha \in \mathsf{Act}(s)$. *(In particular, we assume that M does not have terminal states.)* □

The intuitive operational behavior of a MDP is as follows. If s is the current state then at first one of the actions $\alpha \in \mathsf{Act}(s)$ is chosen non-deterministically. Afterwards, α is executed leading to state t with probability $\mathsf{P}(s,\alpha,t)$. If $\mathsf{P}(s,\alpha,t) > 0$ and $\mathsf{P}(s,\alpha,u) > 0$ for some state s and for at least two distinct states t and u then α is called a *probabilistic action* as it has a random effect. Of course, when modelling realistic systems, most actions α will be non-probabilistic in the sense that they yield unique successor states (i.e., for some state t, $\mathsf{P}(s,\alpha,t) = 1$ and $\mathsf{P}(s,\alpha,u) = 0$ for all states $u \in S \setminus \{t\}$).

Example 2. We consider a mutual exclusion problem where two processes P_1 and P_2 share a common resource that requires exclusive access. Fig. 1 shows the control flow graphs. For simplicity, we abstract away from the activities during the noncritical phase, where P_i does not use the common resource, and the critical sections, where access to the common resource is required. If the computation of P_i requires access to the resource then P_i moves to a waiting location where P_i tries to synchronize with the resource manager by sending a request message.[2] For simplicity we consider here a data-abstract version, but we may imagine that the request message sent by P_i contains some additional informations about the intended use of the resource. Given this information, the resource manager performs a

[2] We use here CSP- or PROMELA-like notations $req_i!$ and $req?$ to denote the send resp. read-operation along the synchronous channel connecting P_i with the resource manager.

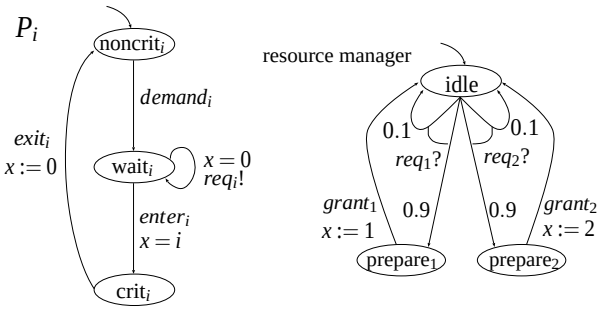


Fig. 1. **Processes P_i and the resource mananger**

certain preprocessing according to P_i's purposes (location prepare$_i$). We assume here an *unreliable synchronous* channel that might *corrupt* the sent request messages with probability 0.1. If the resource manager obtains a corrupted message then it ignores the request as it lacks the information about the necessary preprocessing. To guarantee exclusive access to the resource, shared variable x is used which is initially 0 and which the resource manager sets to i if P_i gets the grant to use the resource. Fig. 2 shows the MDP M that results from the parallel composition of P_1, P_2 and the resource manager.[3] Note that here only the synchronization actions *req* are probabilistic. The set AP of atomic propositions can be any subset of $\{\text{noncrit}_i, \text{wait}_i, \text{crit}_i : i = 1,2\} \cup \{\text{idle}, \text{prepare}_1, \text{prepare}_2, x = 0, x = 1, x = 2\}$ with the obvious labeling function. □

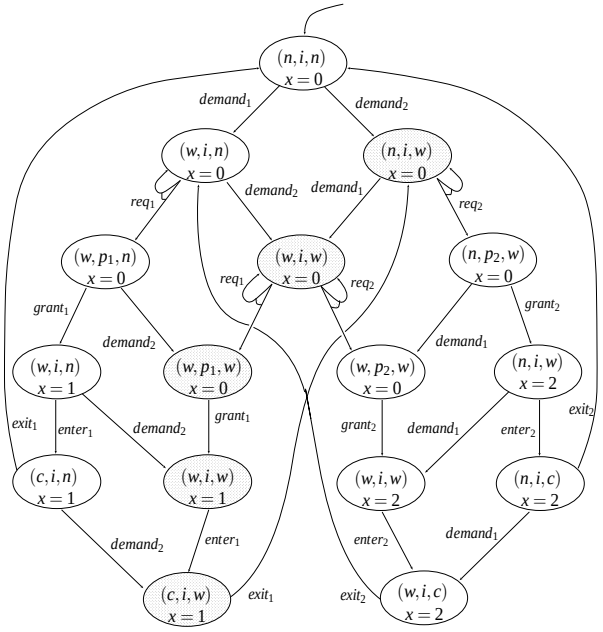


Fig. 2.

[3] We simply write "n" for "noncrit$_i$", "w" for " wait$_i$", etc. and *req$_i$* for the synchronized execution of *req$_i$*! and *req$_i$*?.

Paths. An infinite path in a MDP is a sequence $\varsigma = s_0, \alpha_1, s_1, \alpha_2, \ldots \in (S \times \mathsf{Act})^\omega$ such that $\alpha_i \in \mathsf{Act}(s_{i-1})$ and $\mathsf{P}(s_{i-1}, \alpha_i, s_i) > 0$ for any $i \geq 1$. We write paths in the form

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

$\varsigma\downarrow^i = s_0 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_i} s_i$ denotes the i-th prefix of ς, $\mathsf{first}(\varsigma) = s_0$ the starting state of ς and $\mathsf{trace}(\varsigma) = \mathsf{L}(s_0), \mathsf{L}(s_1), \mathsf{L}(s_2), \ldots$ the word over the alphabet 2^{AP} obtained by the projection of ς to the state labels. $\mathsf{Paths}(s)$ is the set of infinite paths starting in state s. Finite paths (denoted by the greek letter σ) are finite prefixes of infinite paths that end in a state. We use the notations $\mathsf{first}(\sigma)$, $\mathsf{trace}(\sigma)$ and $\sigma\downarrow^i$ as for infinite paths and $\mathsf{last}(\sigma)$ for the last state of σ and $|\sigma|$ for the length (number of actions).

Schedulers. A scheduler denotes an instance that resolves the nondeterminism in the states, and thus, yields a Markov chain and a probability measure on the paths. We consider here history dependent, deterministic schedulers (briefly called schedulers) which are given by a function D that assigns to any finite path σ an action $D(\sigma) \in \mathsf{Act}(\mathsf{last}(\sigma))$. By a D-path, we mean a path σ that can be generated by D, i.e., $D(\sigma\downarrow^i)$ is the $(i+1)$-st action in σ for all indices $i < |\sigma|$. (For the definition of a scheduler it suffices to define $D(\sigma)$ for the D-paths.) Given a state s and a scheduler D, the behavior of M under D can be formalized by a (possibly infinite-state) Markov chain. We write $\mathsf{Pr}^{D,s}$ or simply Pr^D to denote the standard probability measure on the Borel field of the infinite paths ς with $\mathsf{first}(\varsigma) = s$.

Linear Time Logic. In the following section, we will present criteria for a partial order reduction technique that preserves the maximal or minimal probabilities for path properties specified in LTL$_{\setminus X}$, the fragment of LTL without the next step operator which is given by the grammar:

$$\varphi \;::=\; \mathit{true} \;\Big|\; a \;\Big|\; \neg\varphi \;\Big|\; \varphi_1 \wedge \varphi_2 \;\Big|\; \varphi_1 \mathsf{U} \varphi_2$$

where $a \in \mathsf{AP}$ is an atomic proposition. Intuitively, $\varphi_1 \mathsf{U} \varphi_2$ holds for an infinite path σ if eventually φ_2 holds while before continuously φ_1 is satisfied. For a precise definition of the semantics of LTL see e.g. [8]. Operators for modelling eventually $\Diamond$ or always $\Box$ can be derived by $\Diamond\varphi = \mathit{true}\mathsf{U}\varphi$ and $\Box\varphi = \neg\Diamond\neg\varphi$. Using LTL$_{\setminus X}$ as a specification formalism for MDPs under qualitative aspects one requires that a given formula φ holds with probability 1 or with positive probability under all schedulers. E.g., the MDP M in Fig. 2 fulfills the formulas $\Box(\neg\text{crit}_1 \vee \neg\text{crit}_2)$, $\Box((x = 0) \rightarrow (\neg\text{crit}_1 \wedge \neg\text{crit}_2))$, $\Box((x = i) \rightarrow \Diamond\text{crit}_i)$, $i = 1,2$, and $\Box\Diamond\text{crit}_1 \vee \Box\Diamond\text{crit}_2$ with probability 1.[4]

In the quantitative setting, one asks for the maximal or minimal probabilities for φ where maximum and minimum are

[4] The property $\Box(\text{wait}_1 \rightarrow \Diamond\text{crit}_1)$ does not hold almost surely, unless fairness for the resource manager is required.

taken over all schedulers. To reason about quantitative aspects in Example 2, we may use additional variables y_1 and y_2 that serve as counters for the requests sent by P_1 and P_2 respectively. If the demand-action resets y_i to 0 while *req*! increments y_i then any global state in the full MDP where the resource manager is in location idle and $x = y_1 = y_2 = 0$ satisfies the formula $\mathsf{idle}\mathsf{U}((\mathrm{crit}_1 \vee \mathrm{crit}_2) \wedge (y_1 + y_2 \leq 5))$ with probability 0.9999 under all schedulers.

End Components. The rough idea of a qualitative or quantitative analysis of a MDP M, a $\mathrm{LTL}_{\setminus X}$ specification and an upper probability bound relies on a transformation of the given formula into a deterministic ω-automaton A, a graph-analysis that calculates the so-called end components of the product MDP $M \times A$ and finally the calculation of the maximal probabilities to reach an accepting end component [10, 13–15]. The problem of lower probability bounds, say "$\geq p$", for an $\mathrm{LTL}_{\setminus X}$ formula φ is dual as we may consider the problem whether $\neg\varphi$ holds with probability $\leq 1 - p$.
End components [13, 14] can be viewed as the MDP-counterpart to terminal strongly connected components in Markov chains. They consists of a state-set T and action-sets $A(t)$ for each state $t \in T$ such that, once T is entered and only actions in $A(t)$ are chosen, T will not be left and any state of T is visited infinitely often almost surely. Formally, an *end component* of M is a pair (T, A) consisting of a state-set $T \subseteq S$ and a function $A : T \to 2^{\mathsf{Act}}$ such that:

(1) $\emptyset \neq A(t) \subseteq \mathsf{Act}(t)$ for all states $t \in T$
(2) $\sum_{t' \in T} \mathsf{P}(t, \alpha, t') = 1$ for all $\alpha \in A(t)$ and $t \in T$
(3) The directed graph $(T, \longrightarrow_A)$ is strongly connected.

Here, $\longrightarrow_A$ denotes the edge-relation induced by A, i.e., $t \longrightarrow_A t'$ iff $\mathsf{P}(t, \alpha, t') > 0$ for some action $\alpha \in A(t)$. E.g., in Fig. 2 the grey states together with the obvious singleton action-sets yield an end component. The observation that under each scheduler D, almost all paths "end" in an end component can be formalized as follows:

$$\Pr^D\{\varsigma \in \mathsf{Paths}(s) : \mathsf{Limit}(\varsigma) \text{ is an end component } \} = 1$$

Here, for an infinite path $\varsigma = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots$, $\mathsf{Limit}(\varsigma)$ is the pair (T, A) where T is the set of states t that occur infinitely often in ς and $A(t)$ the set of actions $\alpha \in \mathsf{Act}(t)$ such that $s_i = t$ and $\alpha_{i+1} = \alpha$ for infinitely many indices i.

Stutter equivalence. The correctness of partial order reduction criteria and linear time properties (e.g., specified by $\mathrm{LTL}_{\setminus X}$ formulas) is typically formulated by means of an equivalence that identifies those paths whose traces agree up to stuttering. In this context, stuttering refers to the repetition of the same state-labels. Formally, two infinite words θ_1 and θ_2 over the alphabet 2^{AP} are called *stutter equivalent* iff there is an infinite word $\ell_1, \ell_2, \ldots$ over the alphabet 2^{AP} such that $\theta_1 = \ell_1^{k_1}, \ell_2^{k_2}, \ldots$ and $\theta_2 = \ell_1^{n_1}, \ell_2^{n_2}, \ldots$ where k_i, $n_i \geq 1$. Two infinite paths ς_1 and ς_2 in a MDP are called stutter equivalent iff the induced words $\mathsf{trace}(\varsigma_1)$ and $\mathsf{trace}(\varsigma_2)$ over 2^{AP} are stutter equivalent. It is well-known that stutter equivalent paths fulfill the same $\mathrm{LTL}_{\setminus X}$ formulas [32].
For the partial order reduction we shall need the concept of *stutter actions*, i.e., actions that have no effect on the state-labels, no matter in which state they are taken. Formally, action α of a MDP M is called a stutter action iff for all states $s, t \in S$ we have: $\mathsf{P}(s, \alpha, t) > 0$ implies $\mathsf{L}(s) = \mathsf{L}(t)$. E.g., using $\mathsf{AP} = \{\mathrm{crit}_1, \mathrm{crit}_2\}$ in the MDP in Fig. 2, the demand- and the grant-actions are stutter actions and all paths where only P_1 passes infinitely often its three locations while P_2 never enters its critical section are stutter equivalent as they all induce traces of the form $(\emptyset^+\{\mathrm{crit}_1\}^+)^\omega$.

3 The ample set method

We now discuss which modifications on the criteria for Peled's ample sets [35, 24, 36] are needed to ensure the correctness for MDPs and properties specified by $\mathrm{LTL}_{\setminus X}$ formulas. The starting point is a MDP $M = (S, \mathsf{Act}, \mathsf{P}, s_{\mathrm{init}}, \mathsf{AP}, \mathsf{L})$ that we want to analyze against an $\mathrm{LTL}_{\setminus X}$ formula. The rough idea is to assign to any (reachable) state s a set $\mathsf{ample}(s) \subseteq \mathsf{Act}(s)$ and to construct a reduced MDP $\hat{M}$ that results by using the action-sets $\mathsf{ample}(s)$ instead of $\mathsf{Act}(s)$. Formally, given a function $\mathsf{ample} : S \to 2^{\mathsf{Act}}$ with $\mathsf{ample}(s) \subseteq \mathsf{Act}(s)$ for all states s, the state space of the reduced MDP

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

induced by ample is the smallest set $\hat{S} \subseteq S$ that contains s_{init} and any state t where $\mathsf{P}(s, \alpha, t) > 0$ for some $s \in \hat{S}$ and $\alpha \in \mathsf{ample}(s)$. The labeling function $\hat{\mathsf{L}} : \hat{S} \to 2^{\mathsf{AP}}$ is the restriction of the original labeling function L to the state-set $\hat{S}$.[5] The transition probability matrix of $\hat{M}$ is given by: $\hat{\mathsf{P}}(s, \alpha, t) = \mathsf{P}(s, \alpha, t)$ if $\alpha \in \mathsf{ample}(s)$ and 0 otherwise.
If the ample sets are "small" then we might expect that the state space of $\hat{M}$ is a proper subset of the state space of M, and thus, "easier" to analyze. Moreover, the linear programs to be solved for $\hat{M}$ contain less inequations for any reachable state s that is not fully expanded (i.e., $\mathsf{ample}(s) \neq \mathsf{Act}(s)$).
To formulate our conditions for the ample-sets that will ensure the correctness of the reduction, in the sense that the full MDP M and the reduced MDP $\hat{M}$ satisfy the same $\mathrm{LTL}_{\setminus X}$ formulas, we shall need the notion of independent actions. In non-probabilistic systems, independence of two actions α and β means that for any state s where both α

[5] We may assume that all atomic propositions that do not occur in the given formula are removed from AP. This simple step can identify more paths as stutter equivalent, and thus, can improve the reduction.

A1 For all states $s \in S$, $\emptyset \neq \mathsf{ample}(s) \subseteq \mathsf{Act}(s)$.

A2 If $s \in \hat{S}$ and $\mathsf{ample}(s) \neq \mathsf{Act}(s)$ then all actions $\alpha \in \mathsf{ample}(s)$ are stutter actions.

A3 For each path $\sigma = s \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} s_n \xrightarrow{\gamma} \ldots$ in M where $s \in \hat{S}$ and γ is dependent on $\mathsf{ample}(s)$ there exists an index $i \in \{1,\ldots,n\}$ such that $\alpha_i \in \mathsf{ample}(s)$.

A4 For each end component (T,A) in $\hat{M}$: $\alpha \in \bigcap_{t \in T} A(t)$ implies $\alpha \in \bigcup_{t \in T} \mathsf{ample}(t)$

A5 If $\sigma = s \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} s_n \xrightarrow{\gamma} \ldots$ is a path in M where $s \in \hat{S}$, $\alpha_1,\ldots,\alpha_n,\gamma \notin \mathsf{ample}(s)$ and γ is probabilistic then $|\mathsf{ample}(s)| = 1$.

Fig. 3. Five conditions for the ample-sets

and β are enabled the execution of α does not affect the enabledness of β (i.e., the α-successor of s has an outgoing β-transition), and vice versa, and in addition the action sequences $\alpha\beta$ and $\beta\alpha$ lead to same state. Typically, actions of different subprocesses that only refer to local variables are independent. We make here the additional requirement that $\alpha\beta$ and $\beta\alpha$ have the same probabilistic effect:

Definition 3 (Independence of actions). *Two actions α, β with $\alpha \neq \beta$ are called independent (in M) iff for all states $s \in S$ with $\{\alpha,\beta\} \subseteq \mathsf{Act}(s)$ we have: (1) $\mathsf{P}(s,\alpha,t) > 0$ implies $\beta \in \mathsf{Act}(t)$, (2) $\mathsf{P}(s,\beta,u) > 0$ implies $\alpha \in \mathsf{Act}(u)$ and (3) for all states $w \in S$:*

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

Two different actions α and β are called dependent iff α and β are not independent. If $\mathsf{A} \subseteq \mathsf{Act}$ and $\alpha \in \mathsf{Act} \setminus \mathsf{A}$ then α is called independent from A iff for all actions $\beta \in \mathsf{A}$, α and β are independent. Otherwise, α is called dependent on A. □

Applying the above definition to non-probabilistic actions α and β (i.e., where $\mathsf{P}(s,\alpha,t)$, $\mathsf{P}(s,\beta,t) \in \{0,1\}$ for all states s, t) yields the standard definition of independence of actions in ordinary transition systems.

Example 4 (Independent actions). Fig. 4 shows a fragment of a MDP M_1 representing the parallel execution of independent actions α and β. For example, α might stand for the experiment "roll a 6-sided dice" and assign value 1 or 0 to boolean variable *x*, depending on whether the outcome is 1 or in $\{2,3,4,5,6\}$, while β stands for the experiment "roll a 6-sided dice" and assign value 1 or 0 to boolean variable *y*, depending on whether the outcome is in $\{1,2\}$ or in $\{3,4,5,6\}$. In general, whenever α and β stand for stochastic experiments that are independent in the classical sense then α and β viewed as actions of a MDP are independent. However, there are also other situations where two actions can be independent. For instance, actions α and β in the MDP M_2 in Fig. 4 are independent. To see this, we first notice that only in state *s* both α and β are enabled. The α-successors t,s of *s* have a β-transition to state *u*, while the β-successor *u* has a α-transition to itself. The effect under the action sequences $\alpha\beta$ and $\beta\alpha$ is the same as in either case we reach state *u* with probability 1. In Fig. 2, only the actions req_1 and req_2 are dependent. □

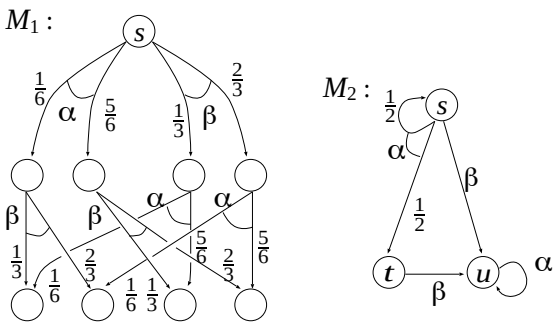


Fig. 4. Examples for independent actions

The conditions that we require for the ample-sets are summarized in Fig. 3. Condition (A1) states that the reduced MDP does not contain terminal states. (Recall that we we assume that all states in M are non-terminal.) Conditions (A2) and (A3) are exactly the same as in the non-probabilistic setting. (A4) can be viewed as the probabilistic counterpart to Peled's *cycle-condition* which requires that any action that is enabled in all states of a cycle in the reduced system belongs to the ample set of some of the states in the cycle. Our condition (A4) applied to an ordinary transition system viewed as a MDP (i.e., where all actions are non-probabilistic and yield unique successor states) reduces to the cycle-condition. The reason is that for such "non-probabilistic MDPs" the end components correspond to the cycles. However, our condition (A4) also allows for certain cycles violating the cycle-condition. For instance, for the MDP M_2 in Fig. 4, (A4) allows to choose $\mathsf{ample}(s) = \{\alpha\}$ (provided that α is a stutter-action), as state *s* is not contained in an end component. (A5) is an additional condition which has no non-probabilistic counterpart.

Non-probabilistic case. To understand the above conditions, we recall the argument in the non-probabilistic case how under conditions (A1)-(A3) and the non-probabilistic

cycle-condition any path in M can be replaced by a stutter-equivalent path in $\hat{M}$ [36, 8]. Given an infinite path

$$\varsigma = s \xrightarrow{\alpha_1} t_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} t_n \xrightarrow{\beta} w \xrightarrow{\gamma} \ldots$$

in M where $\alpha_1, \ldots, \alpha_n \notin \mathsf{ample}(s)$ and $\beta \in \mathsf{ample}(s)$, we can conclude from (A3) that α_i and β are independent, $i = 1, \ldots, n$. Hence, we may replace the action sequence $\alpha_1 \ldots \alpha_n \beta$ by the action sequence $\beta\alpha_1 \ldots \alpha_n$ to obtain a path

$$\varsigma_1 = s \xrightarrow{\beta} u_1 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_{n-1}} u_n \xrightarrow{\alpha_n} w \xrightarrow{\gamma} \ldots$$

which is stutter equivalent to ς. Condition (A2) guarantees that β is a stutter action. Thus, ς and ς_1 are stutter equivalent. Similarly, any infinite path ς with the action sequence $\alpha_1\alpha_2\ldots$ in M where none of the actions α_i belongs to $\mathsf{ample}(\mathsf{first}(\varsigma))$ can be replaced by a stutter equivalent path with the same starting state $\mathsf{first}(\varsigma)$ and the action sequence $\beta\alpha_1\alpha_2\ldots$ where β is an arbitrary action in $\mathsf{ample}(\mathsf{first}(\varsigma))$. In either case we obtain a path ς_1 that starts with a transition in the reduced system $\hat{M}$. We now may apply the same technique to the paths ς_1 (more precisely, to the suffix of ς_1 that starts in the second state) to obtain a stutter equivalent path ς_2 whose first two transitions are transitions in $\hat{M}$. We continue in this way until any path in M is "transformed" into a path $\hat{\varsigma}$ in $\hat{M}$. Although conditions (A1)-(A3) are sufficient to guarantee the stutter equivalence of ς and the paths $\varsigma_1, \varsigma_2, \ldots$, the cycle condition is needed to ensure the stutter equivalence of ς and $\hat{\varsigma}$.

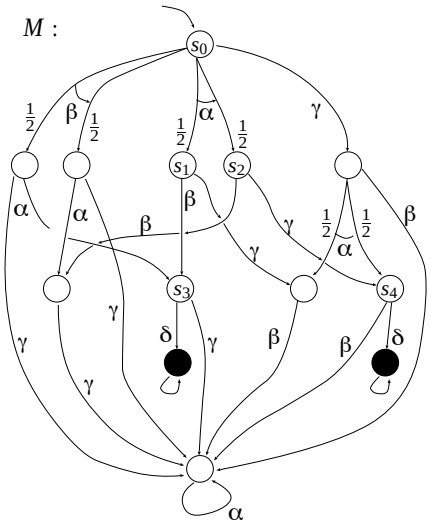


Fig. 5. (A1)-(A4) are not sufficient

Probabilistic case. The intuitive motivation for conditions (A1)-(A4) is as in the non-probabilistic case. Unlike the situation in the non-probabilistic case, conditions (A1)-(A4) are not sufficient to guarantee the equivalence of the MDPs M and $\hat{M}$ for $\mathrm{LTL}_{\setminus X}$ specifications as the counterexample in Fig. 5 shows. We assume $\mathsf{AP} = \{\mathrm{white}, \mathrm{black}\}$ with the obvious labelling function. Thus, α, β and γ are stutter actions, while δ is not. Consider a scheduler D with $D(s_0) = \alpha$,

$$D(s_0 \xrightarrow{\alpha} s_1) = \beta, \qquad D(s_0 \xrightarrow{\alpha} s_1 \xrightarrow{\beta} s_3) = \delta,$$
$$D(s_0 \xrightarrow{\alpha} s_2) = \gamma, \qquad D(s_0 \xrightarrow{\alpha} s_2 \xrightarrow{\gamma} s_4) = \delta.$$

The probability for the $\mathrm{LTL}_{\setminus X}$ formula $\Diamond$black under scheduler D is 1. If we choose $\mathsf{ample}(s_0) = \{\beta, \gamma\} \neq \mathsf{Act}(s_0)$ and $\mathsf{ample}(t) = \mathsf{Act}(t)$ for all other states t then conditions (A1)-(A4) are fulfilled, but there is no scheduler for the reduced MDP that reaches a black state with probability 1. This is because no matter whether a scheduler of $\hat{M}$ chooses β or γ in the state s_0, the maximal probability to reach a black state is at most $\frac{1}{2}$. The problem in the above example is that M can branch probabilistically by choosing α in s_0. In one α-successor (s_1), action β leads to eventually reaching a black state, while in the other α-successor (s_2), action γ leads to eventually reaching a black state.

In Fig. 5, actions β and γ are independent. Hence, also the requirement that – in addition to (A1)-(A4) – for any state s which is not fully expanded, all actions in $\mathsf{ample}(s)$ are pairwise independent, is still not sufficient.

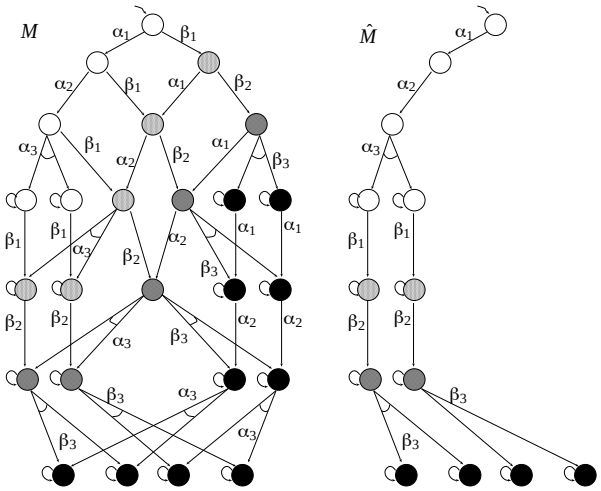


Fig. 6. Interleaving of $\alpha_1\alpha_2\alpha_3$ and $\beta_1\beta_2\beta_3$

One possibly remedy for the situation is to add the requirement that for all paths σ in M that do not contain an action $\alpha \in \mathsf{ample}(\mathsf{first}(\sigma))$ all actions in σ are non-probabilistic. But this condition is too strong since hardly any reduction could be achieved. E.g., Fig. 6 shows on its left an MDP M that results from the interleaving of two independent processes P_α and P_β (without shared variables and no communication) that perform the action sequences $\alpha_1\alpha_2\alpha_3$ and $\beta_1\beta_2\beta_3$ respectively and afterwards certain internal actions that have no effect on the state. We assume here that the property of interest does not refer to the local states or variables of P_α. Thus, the α's are stutter actions, while the β's

are not. As α_3, β_3 are probabilistic, the above requirement would *not* allow for any reduction in Fig. 6.
For this reason, we added condition (A5) in Fig. 3 which requires the ample set of state s to be a singleton if probabilistic branches on paths in M starting in s can occur before an ample action is executed. Note that (A5) is irrelevant if all actions are non-probabilistic, i.e., if the given MDP is an ordinary transition system. Thus, in this case, our conditions (A1)-(A5) agree *exactly* with Peled's conditions.

Example 5. For the MDP M in Fig. 6, (A1)-(A5) allow to built the reduced MDP by the action sequence $\alpha_1\alpha_2\alpha_3\beta_1\beta_2\beta_3$. For the MDP M in Example 2 (Fig. 2) and $\mathsf{AP} = \{\text{crit}_1, \text{crit}_2\}$, (A1)-(A5) allow to use the reduced MDP as shown in Fig. 7. (Note that only the synchronization actions req_1 and req_2 are probabilistic and dependent.) The MDP in Fig. 5 can be reduced by putting $\mathsf{ample}(s_0) = \{\alpha\}$ and $\mathsf{ample}(t) = \mathsf{Act}(t)$ for all states $t \neq s_0$. □

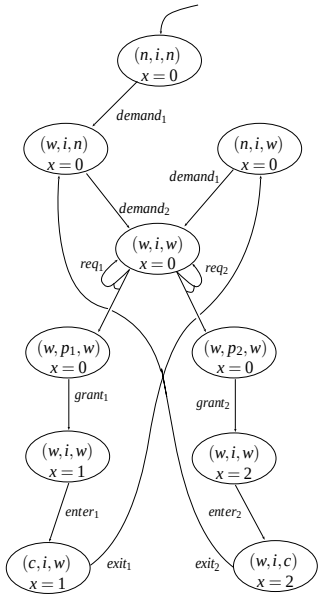


Fig. 7.

4 Construction of the reduced MDP

The simplest possibility for the construction of $\hat{M}$ is to replace the end component condition (A4) with the (stronger) cycle-condition and (A5) with the stronger condition

A6 $|\mathsf{ample}(s)| = 1$ or $\mathsf{ample}(s) = \mathsf{Act}(s)$ for any state $s \in \hat{S}$.

For non-probabilistic systems, (A6) together with (A1)-(A3) and the cycle-condition guarantees that the reduced system is equivalent to the full system for all $\text{CTL}_{\setminus X}$-formulas [16, 36]. Hence, the reduced MDP can be generated using *exactly* the same techniques as for non-probabilistic systems and branching-time properties.

Another possibility for the construction of the reduced MDP that might yield a better reduction (as it also allows for non fully expanded states with a ample set consisting of the action sets of two or more processes that run in parallel) is to use the non-probabilistic techniques to ensure (A1)-(A3) and the cycle condition, see e.g. [36, 8]. For (A5), we may use the same idea as suggested in [36] to treat (A3) for communicating processes. That is, we try to choose the ample set of state s to be the action sets of some processes $P_1, \ldots, P_j$ such that no probabilistic action of another process is enabled in s or can be enabled by certain transitions of the processes $P_k \notin \{P_1, \ldots, P_j\}$.
In the context of $\text{LTL}_{\setminus X}$ model checking, the above mentioned techniques for the construction of the reduced MDP can be applied in a preprocessing phase. The reduced MDP (rather than the full MDP) will then serve as input for the model checking algorithms. Unfortunately, an *on-the-fly* technique that combines the construction of the reduced MDP with the verification algorithm, as e.g. realized in SPIN, seems to be difficult for a quantitative analysis, as the latter requires solving linear programs rather than a cycle-search. In fact, a further alternative for the construction of the reduced MDP relies on an adaption of the *static* approach suggested in [29] to our setting. The idea is to generate the reduced system during the compilation phase and then to apply the standard verification algorithms. In our setting, we may replace conditions (A2) and (A4) with the following requirement:

A7 There is an action-set $\mathsf{Act}' \subseteq \mathsf{Act}$ such that (i) Act' contains all actions $\alpha \in \mathsf{Act}$ that are not stutter-actions, (ii) for all states s in $\hat{M}$ we have $\mathsf{ample}(s) \cap \mathsf{Act}' = \emptyset$ or $\mathsf{ample}(s) = \mathsf{Act}(s)$ and (iii) for each end component (T, A) of $\hat{M}$ we have $(\bigcup_{t \in T} A(t)) \cap \mathsf{Act}' \neq \emptyset$.

It can easily be checked that (A7) implies (A2) and (A4). Subcondition (iii) is weaker than the requirement that any cycle in the reduced MDP contains an action in Act'. Thus, the techniques of finding such an action-set Act' suggested in [29] can be applied here as well.

5 Correctness of the reduction

We now state the correctness result for the ample-set method in MDPs in the sense that the original MDP and the reduced MDP yield the same extremal probabilities for all $\text{LTL}_{\setminus X}$ formulas. In fact, we can prove a stronger result stating the equivalence of both MDPs with respect to all stutter insensitive, ω-regular properties.

Notation 6. By a *stutter-invariant language* over AP, we mean a set $\mathbf{L} \subseteq (2^{\mathsf{AP}})^\omega$ of infinite words over 2^{AP} such that with any word $\theta_1 \in \mathbf{L}$ all words θ_2 that are stutter equivalent to θ_1 are also contained in $\mathbf{L}$. To reason about the

probabilities for paths that generate a trace in $\mathbf{L}$ we need the additional requirement that the path-sets $\{\varsigma \in \mathsf{Paths}(s) : \mathsf{trace}(\varsigma) \in \mathbf{L}\}$ are *measurable*. For this, we require that $\mathbf{L}$ is an element of the sigma-field induced by the *trace-cylinders* $Cyl(\ell_1^+, \ldots, \ell_n^+)$ consisting of all infinite words over 2^{AP} that have a prefix of the form $\ell_1^{k_1}, \ldots, \ell_n^{k_n}$ where $k_1, \ldots, k_n \geq 1$. Here, $\ell_1, \ldots, \ell_n$ are pairwise distinct subsets of AP. Given a scheduler D for M, $\Pr^{D,s}(\mathbf{L}) = \Pr^D\{\varsigma \in \mathsf{Paths}(s) : \mathsf{trace}(\varsigma) \in \mathbf{L}\}$ denotes the probability for the D-paths that start in state s and induce a trace in $\mathbf{L}$. □

M and $\hat{M}$ are said to be *stutter equivalent* iff for any scheduler D for M there exists a scheduler $\hat{D}$ for $\hat{M}$ with

$$\Pr^{D,s_{\text{init}}}(\mathbf{L}) = \Pr^{\hat{D},s_{\text{init}}}(\mathbf{L}) \qquad \textbf{(+)}$$

for all stutter-invariant languages $\mathbf{L}$, and vice versa. (As $\hat{M}$ is a sub-MDP of M it is obvious that for any scheduler $\hat{D}$ for $\hat{M}$ there is a scheduler D for M such that (+) holds. We just may put $D = \hat{D}$.)

Theorem 7 (Correctness of conditions A1-A5). *If conditions (A1)-(A5) hold then M and $\hat{M}$ are stutter equivalent.*

As $\mathrm{LTL}_{\setminus X}$-formulas induce stutter-invariant, measurable languages [45], we obtain

$$\sup_D \Pr^D(s_{\text{init}}, \varphi) = \sup_{\hat{D}} \Pr^{\hat{D}}(s_{\text{init}}, \varphi)$$

where D and $\hat{D}$ range over all schedulers for M and $\hat{M}$ respectively. The same holds for "inf" instead of "sup". Hence, under conditions (A1)-(A5) M and $\hat{M}$ are fully equivalent for $\mathrm{LTL}_{\setminus X}$ specifications, in both the qualitative and quantitative setting. This even holds for stutter insensitive ω-regular languages, i.e., stutter insensitive languages that are generated by a nondeterministic Büchi automaton.

Proof sketch for Theorem 7. In the remainder of this section, we assume that conditions (A1)-(A5) are fulfilled and show that M and $\hat{M}$ are stutter equivalent. For every scheduler D for M, we have to construct a scheduler $\hat{D}$ for $\hat{M}$ such that (+) holds for all stutter insensitive, measurable languages $\mathbf{L}$.

Starting with the given scheduler D for M, we construct an infinite sequence of schedulers $D_1, D_2, D_3, \ldots$ for M, such that condition (+) holds for D and D_i (for each state s) and $D_i(\sigma) \in \mathsf{ample}(\mathsf{last}(\sigma))$ for all finite D_i-paths σ of length $< i$. I.e.,

$$\Pr^{D,s}(\mathbf{L}) = \Pr^{D_i,s}(\mathbf{L}) \qquad \textbf{(++)}$$

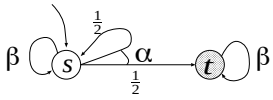
for each stutter-invariant language $\mathbf{L}$ and each state s and for each D_i-path σ the i-th prefix of σ is a path in $\hat{M}$. For the construction of the schedulers D_i and the proof of (++) see the appendix.

Finally, a scheduler $\hat{D}$ for $\hat{M}$ is derived from the schedulers D_i as follows: We define

$$\hat{D}(\hat{\sigma}) = D_{i+1}(\hat{\sigma}).$$

if $\hat{\sigma}$ is a finite $\hat{D}$-path of length i.

The remaining argumentation is similar to the non-probabilistic case. We cannot immediately conclude that D and $\hat{D}$ yield the same probabilities for the trace-cylinders because the generated $\hat{D}$-paths might "delay" a certain action of an D-path ad infinity as in the following example.



The colors "white" and "grey" indicate the labeling of the states. Thus, β is a stutter action, while α is not. For $\mathsf{ample}(s) = \{\beta\}$ and scheduler D where $D(\sigma) = \alpha$ for all paths σ with $\mathsf{last}(\sigma) = s$, the construction explained in the appendix yields

$$D_i(\underbrace{s \xrightarrow{\beta} s \ldots \xrightarrow{\beta} s}_{\text{length } j}) = \begin{cases} \beta : \text{for } j \leq i-1, \\ \alpha : \text{for } j = i. \end{cases}$$

Thus, scheduler $\hat{D}$ always schedules β in the state s. (In fact, $\hat{D}$ is the only scheduler for $\hat{M}$ as $\hat{M}$ consists of state s with the β-loop.) Under D and each of the schedulers D_i, we obtain probability 1 for the $\mathrm{LTL}_{\setminus X}$ formula $\Diamond$grey, while the probability for $\Diamond$grey under $\hat{D}$ is 0. However, in this example, conditions (A1), (A2),(A3) and (A5) hold, but the end component $(s, \{\beta\})$ violates condition (A4).

We now show that the end component condition (A4) ensures that any action of a D-path will be "consumed" by $\hat{D}$ almost surely. To simplify the notations we consider here only the case where $D(s) = \alpha_1 \notin \mathsf{ample}(s)$ and show that $\sum_{\hat{\sigma}} \Pr^{\hat{D}}(\hat{\sigma}) = 1$ where $\hat{\sigma}$ ranges over all $\hat{D}$-paths with $\mathsf{first}(\hat{\sigma}) = s$, $\alpha_1 \notin \mathsf{Act}(\hat{\sigma})$ and $\hat{D}(\hat{\sigma}) = \alpha_1$. Let us assume that the sum is strictly less than 1. Then,

$$\Pr^{\hat{D},s}\{\hat{\varsigma} \in \mathsf{Paths}(s) : \alpha_1 \notin \mathsf{Act}(\hat{\varsigma})\} > 0 \qquad \textbf{(VII)}$$

All infinite $\hat{D}$-paths $\hat{\varsigma}$ with $\mathsf{first}(\hat{\varsigma}) = s = s_0$ and $\alpha_1 \notin \mathsf{Act}(\hat{\varsigma})$ have the form

$$\hat{\varsigma} = s \xrightarrow{\beta_1} s_1 \xrightarrow{\beta_2} s_2 \xrightarrow{\beta_3} \ldots$$

where $\alpha_1 \in \mathsf{Act}(s_i) \setminus \mathsf{ample}(s_i)$ for all indices i. This follows from condition (A3) which guarantees that α_1 is independent from the β's, and hence, enabled in all states s_i. Moreover, D_i (and $\hat{D}$) would have chosen α_1 for $\hat{\varsigma}\downarrow^i$ if $\alpha_1 \in \mathsf{ample}(s_i)$. Because of (VII) and the result of [13] we may choose such an $\hat{D}$-path $\hat{\varsigma}$ where $\mathsf{Limit}(\hat{\varsigma})$ is an end component.[6] Thus, we have $\alpha_1 \in \mathsf{Act}(t) \setminus \mathsf{ample}(t)$ for all states t of an end component in $\hat{M}$. This contradicts (A4).

[6] Recall that $\mathsf{Limit}(\hat{\varsigma})$ consists of all states that occur infinitely often in $\hat{\varsigma}$ and their actions that are chosen infinitely often in $\hat{\varsigma}$.

This observation together with the fact that D and the intermediate schedulers D_i yield the same probabilities for all stutter insensitive, measurable languages allows us to conclude that for all $\overline{\ell} = (\ell_1, \ldots, \ell_n) \in (2^{\mathsf{AP}})^*$ and all action sequences $\overline{\alpha} = \alpha_1 \ldots \alpha_n$, $\Pr^{D,s}(\Delta(\overline{\ell}, \overline{\alpha})) = \Pr^{\hat{D},s}(\Delta(\overline{\ell}, \overline{\alpha}))$. Here, $\Delta(\overline{\ell}, \overline{\alpha})$ denotes the set of all finite paths σ of minimal length that induce a trace of the form $\ell_1^+ \ldots \ell_n^+$ and that have an action sequence which results from $\overline{\alpha}$ by exchanging the order of independent actions and possibly adding stutter actions. From this, we may derive that $\Pr^{D,s}(\mathbf{L}) = \Pr^{\hat{D},s}(\mathbf{L})$ for all stutter insensitive, measurable languages $\mathbf{L}$.

6 Conclusion and future work

We presented conditions for a reduction technique in MDPs that allow to analyze a fragment of the state space rather than the full state space. Soundness of our criteria was shown for stutter insensitive measurable properties in the sense that for any scheduler for the original MDP there is a scheduler for the reduced MDP that yields the same probabilities for the given property, and vice versa. In particular, the original and reduced MDP are fully equivalent for $\mathrm{LTL}_{\setminus X}$ specifications.

We followed here the approach of the ample sets, but we expect that also the concepts of persistent [18] or stubborn [43, 42] sets could be adapted for the probabilistic case. Compared to the non-probabilistic case, we needed an auxiliary condition (A5) which requires singleton ample sets to avoid probabilistic branches in the full MDP that cannot be "simulated" in the reduced MDP. The need of an auxiliary condition has a simple intuitive explanation: E.g., for the MDP in Fig. 5, the experiment "(1) toss a coin (action α) and (2) choose one of the actions β or γ" may allow for better strategies (schedulers) to achieve a certain goal (e.g. the $\mathrm{LTL}_{\setminus X}$ formula $\Diamond$ black) then first choosing an action β or γ and then tossing a coin via action α as in the former case the decision between β or γ may depend on the outcome of the coin tossing experiment. In fact, our condition (A5) makes sure that whenever an experiment like the above appears in M, but not in $\hat{M}$, then in step (2) there is only one ample action that can be chosen. Thus, the order of the coin-tossing- and choosing-an-action-event does not matter when (A5) is required.

However, (A5) is mostly irrelevant when each state is either fully expanded or uses the action set of one of the processes that run in parallel and when the processes only have a few locations with nondeterministic branches. For this reason, we expect that the saving of states with a reduction relying on our conditions (A1)-(A5) is "in average" similar to the non-probabilistic case where partial order reduction has been proven to be very useful in many applications, see e.g. [17, 19]. In fact, in the three toy-examples in Example 5, the savings of states is ca. 50%, 30% and 24% respectively. We obtained similar results for the randomized dining philosophers where the saving of states was about 25%–30%.[7]

# philosophers	3	4	5
full MDP	12.565	252.834	470.145
reduced MDP	9.585	171.927	310.295
ratio	0.76	0.68	0.66

We are currently implementing a model checker for MDPs and automata- and LTL-specifications, but cannot yet report on experimental results with the partial order reduction in combination with $\mathrm{LTL}_{\setminus X}$ model checking. Having in mind a probabilistic counterpart to the well-known model checker SPIN [23], we use a probabilistic variant of PROMELA as input language [1]. The similarities to PROMELA allow us to use roughly the same techniques as in SPIN for the construction of the reduced system.

Although the construction of the reduced MDP can be performed using the same techniques as for non-probabilistic systems, the question arises whether special algorithms can be developed that generate better (i.e., smaller) ample sets using the topological characteristics of end components (rather than cycles). E.g. it is worth looking for heuristics to realize subcondition (iii) in (A7) rather than the stronger condition that requires an Act'-action on each cycle in the reduced MDP. This as well as the question whether an on-the-fly technique that interleaves the construction of the reduced MDP and the graph analysis which is required for a *qualitative analysis* (which checks whether an $\mathrm{LTL}_{\setminus X}$ formula holds almost surely) will be investigated in future work.

Another future direction is the development of reduction criteria that preserve branching time properties. In this context it is worth noting that – in contrast to the non-probabilistic case – conditions (A1)-(A4) and (A6) are not sufficient for verifying branching-time properties specified in PCTL [4]. The counterexample, given in Fig. 8, that illustrates this observation is just a probabilistic variant of the example presented in [16, 36] to demonstrate that (A1)-(A3) and the cycle condition cannot guarantee that a non-probabilistic system and the reduced system are $\mathrm{CTL}_{\setminus X}$-equivalent. In Fig. 8, a and b are atomic propositions. As β is a stutter action which is independent from α, conditions (A1)-(A4) and (A6) are fulfilled when choosing the singleton ample set $\{\beta\}$ in the initial state. But then, the $\mathrm{PCTL}_{\setminus X}$-formula

$$\mathbf{P}_{=1}\big[\Box\big((a \wedge \neg b) \rightarrow (\mathbf{P}_{=1}[\Diamond b] \vee \mathbf{P}_{=1}[\Diamond \neg a])\big)\big]$$

holds for $\hat{M}$, but not for M. An intuitive explanation for this phenomenon is the fact that (A6) still allows for prob-

[7] For this experimental study, we used an automated state space generator (see below), but calculated and encoded the ample sets "by hand".

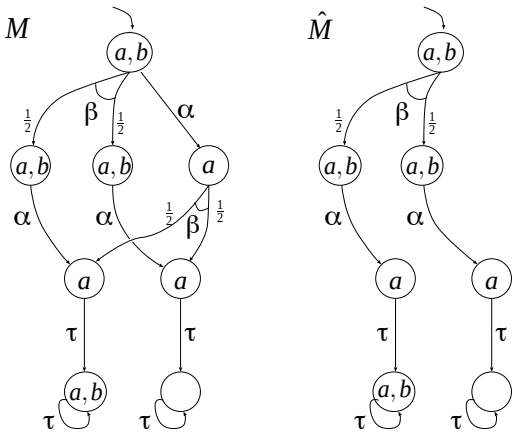


Fig. 8. **(A1)-(A4), (A6) are not sufficient for PCTL$_{\setminus X}$**

abilistic branches in non-fully expanded states leading to states that are not PCTL$_{\setminus X}$-equivalent. However, we claim that conditions (A1)-(A4) and

A8 If $\mathsf{ample}(s) \neq \mathsf{Act}(s)$ then $\mathsf{ample}(s)$ is a singleton consisting of a non-probabilistic action.

guarantee the equivalence of M and $\hat{M}$ for PCTL$_{\setminus X}$-formulas (or even PCTL$^*_{\setminus X}$-formulas).

References

1. C. Baier, F. Ciesinski, M. Größer. Probmela: a modeling language for communicating probabilistic systems. In *Proc. MEMOCODE*. To appear. 2004.
2. C. Baier, E. Clarke, V. Hartonas-Garmhausen, M. Kwiatkowska, M. Ryan. Symbolic model checking for probabilistic processes. In *Proc. ICALP*, LNCS 1256:430–440, 1997.
3. C. Baier, B. Engelen, M. Majster-Cederbaum. Deciding bisimularity anbd similarity for probabilistic processes. *Journal of Computer and System Sciences*, 60:187–231, 2000.
4. A. Bianco, L. de Alfaro. Model Checking of Probabilistic and Nondeterministic Systems, In *Proc. FST & TCS*, LNCS 1026: 499–513, 1995.
5. M. Bozga, O. Maler. On the Representation of Probabilities over Structured Domains. In *Proc. CAV*, LNCS 1633:261–273, 1999.
6. D. Bustan, S. Rubin, M. Vardi. Verifying ω-regular properties of markov chains, 2004. in *Proc. CAV*, 2004. To appear.
7. S. Cattani, R. Segala. Decision algorithms for probabilistic bisimulation. In *Proc. CONCUR*, LNCS 2421:371–385, 2002.
8. E. Clarke, O. Grumberg, D. Peled. *Model Checking*. MIT Press, 1999.
9. C. Courcoubetis, M. Yannakakis. Markov decision processes and regular events. *Proc. ICALP*, LNCS 443:336–349, 1990.
10. C. Courcoubetis, M. Yannakakis. The complexity of probabilistic verification. *J. of the ACM*, 42(4):857–907, 1995.
11. J.-M. Couvreur, N. Saheb, G. Sutre. An optimal automata approach to LTL model checking of probabilistic systems. In *Proc. LPAR*, LNAI 2850:361–375, 2003.
12. P. d' Argenio, B. Jeannet, H. Jensen, K. Larsen. Reachability analysis of probabilistic systems by successive refinements. In *Proc. PAPM/PROBMIV*, LNCS 2165:57–76, 2001.
13. L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997.
14. L. de Alfaro. Stochastic transition systems. In *Proc. CONCUR*, LNCS 1466:423–438, 1998.
15. L. de Alfaro. Computing minimum and maximum reachability times in probabilistic systems. In *Proc. CONCUR*, LNCS 1664:66–81, 1999.
16. R. Gerth, R. Kuiper, D. Peled, W. Penczek. A Partial Order Approach to Branching Time Logic Model Checking. In Proc. 3rd Israel Symposium on the Theory of Computing Systems (ISTCS'95), IEEE Press, pages 130-139, 1995.
17. P. Godefroid. On the costs and benefits of using partial-order methods for the verification of concurrent systems. In [37]. pages 289–303, 1996.
18. P. Godefroid. *Partial Order Methods for the Verification of Concurrent Systems: An Approach to the State Explosion Problem*, LNCS 1032, 1996.
19. P. Godefroid, D. Peled, M. Staskauskas. Using partial-order methods in the formal validation of industrial concurrent programs. In *Proc. International Symposium on Software Testing and Analysis*, pages 261–269. ACM Press, 1996.
20. G. Hachtel, E. Macii, A. Pardo, F. Somenzi. Probabilistic Analysis of Large Finite State Machines. In *Proc. DAC*, San Diego Convention Center, 1994.
21. V. Hartonas-Garmhausen, S. Campos, E. Clarke. Probverus: Probabilistic symbolic model checking. In *Proc. ARTS'99*, LNCS 1601:96–110, 1999.
22. H. Hermanns, M. Kwiatkowska, G. Norman, D. Parker, M. Siegle. On the use of mtbdds for performability analysis and verification of stochastic systems. *Journal of Logic and Algebraic Programming*, Special Issue on Prob. Techniques for the Design and Analysis of Systems, 56:23–67, 2003.
23. G. Holzmann. *The SPIN Model Checker, Primer and Reference Manual*. Addison Wesley, 2003.
24. G. Holzmann, D. Peled. An improvement in formal verification. In *Proc. Formal Description Techniques, FORTE94*, pages 197–211. Chapman & Hall. 1994.
25. M. Huth. Possibilistic and probabilistic abstraction-based model checking. In *Proc. PAPM/PROBMIV*, LNCS 2399:115–134, 2002.
26. M. Huth. Abstraction and probabilities for hybrid logics. In *Proc. 2nd workshop on Quantitative Aspects of Programming Languages*, to appear, 2004.
27. T. Huynh, L. Tian. On some equivalence relations for probabilistic processes. *Fundamenta Informaticae*, 17:211–234, 1992.
28. B. Jeannet, P.R. d' Argenio, K.G. Larsen. RAPTURE: A tool for verifying Markov decision processes. In *Proc. Tools Day'02*, Technical Report. Masaryk University Brno, 2002.
29. R. Kurshan, V. Levin, M. Minea, D. Peled, H. Yenign. Static partial order reduction. In *Proc. TACAS*, LNCS 1384:345–357, 1998.
30. M. Kwiatkowska, G. Norman, D. Parker. PRISM: Probabilistic symbolic model checker. In *Proc. Comp. Performance Eval. TOOLS*, pages 200–204, 2002.

31. M. Kwiatkowska, G. Norman, D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. *International Journal on Software Tools for Technology Transfer (STTT)*. To appear. 2004.
32. L. Lamport. Specifying concurrent program modules. *ACM TOPLAS*, 5(2):190–222, 1983.
33. K. Larsen, A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.
34. A. Miner, D. Parker. Symbolic representations and analysis of large probabilistic systems. In *Validation of Stochastic Systems*, LNCS 2925, 2003. To appear.
35. D. Peled. All from one, one for all: On model checking using representatives. In *Proc. CAV*, LNCS 697:409–423, 1993.
36. D. Peled. Partial order reduction: Linear and branching time logics and process algebras. In [37], pages 79–88, 1996.
37. D. Peled, V. Pratt, G. Holzmann (editors). *Partial Order Methods in Verification*, DIMACS 29(10), Am. Math. Soc., 1997.
38. A. Philippou, I. Lee, O. Sokolsky. Weak bisimulation for probabilistic systems. In *Proc. CONCUR*, LNCS 1877:334–349, 2000.
39. A. Pnueli. The temporal logic of programs. In *Proc. FOCS*, pages 46–57. IEEE Comp. Soc. Press, 1977.
40. A. Pnueli, L. Zuck. Probabilistic verification. *Information and Computation*, 103(1):1–29, 1993.
41. M. Puterman. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.
42. A. Valmari. State of the art report: Stubborn sets. *Petri-Net Newsletters*, 46:6–14, 1994.
43. A. Valmari. A stubborn attack on state explosion. In *Proc. CAV*, LNCS 531:156–165, 1991.
44. R. van Glabbeek, S. Smolka, B. Steffen, C. Tofts. Reactive, generative, and stratified models of probabilistic processes. In *Proc. LICS*, pages 130–141. IEEE Comp. Soc. Press, 1990.
45. M. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proc. FOCS*, pages 327–338, 1985.
46. M. Vardi, P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proc. LICS*, pages 332–344. IEEE Comp. Soc. Press, 1986.

Appendix: Construction of the schedulers D_i

Let M be a MDP and D be a given scheduler for M. We assume that conditions (A1)-(A5) are fulfilled.
Recall that starting with the scheduler D for M, we want to construct an infinite sequence of schedulers $D_1, D_2, D_3, \ldots$ for M, such that

$$\mathsf{Pr}^{D,s}(\mathbf{L}) = \mathsf{Pr}^{D_i,s}(\mathbf{L}), \qquad i = 1,2,\ldots \qquad \textbf{(++)}$$

for each stutter-invariant language $\mathbf{L}$ and each state s and for each D_i-path σ, the i-th prefix of σ is a path in $\hat{M}$. Using standard arguments of measure theory, to prove (++) for all stutter insensitive, measurable languages, it suffices to establish condition (++) for the trace-cylinders $\mathbf{L} = \mathit{Cyl}(\ell_1^+, \ldots, \ell_k^+)$.

Randomized schedulers. So far, we have only considered history dependent deterministic (HD) schedulers, but the schedulers D_i will be history dependent randomized (HR). However, the D_i's will make deterministic decision on paths of length less than i and the resulting scheduler $\hat{D}$ on $\hat{M}$ is deterministic.
Formally, a HR-scheduler E assigns to any finite path σ a distribution for Act, such that $E(\sigma)(\alpha) > 0$ implies $\alpha \in \mathsf{Act}(\mathsf{last}(\sigma))$.[8] We simply write $E(\sigma) = \alpha$ if $E(\sigma)$ assigns probability 1 to action α. Given a finite path σ, $\mathsf{Pr}^E(\sigma)$ denotes the probability under E for the basic cylinder spanned by σ, i.e., $\mathsf{Pr}^E(\sigma) = 1$ if $\sigma = s$ is a path of length 0 and

$$\mathsf{Pr}^E(\sigma \xrightarrow{\gamma} t) = \mathsf{Pr}^E(\sigma) \cdot E(\sigma)(\gamma) \cdot \mathsf{P}(\mathsf{last}(\sigma), \gamma, t).$$

In the sequel, we will consider the given deterministic scheduler D for M as an HR-scheduler.

Construction of D_1. We fix a state $s \in S$ and present the definition of $D_1(\sigma_1)$ for finite D_1-paths starting in s. If $D(s) = \beta$ for some $\beta \in \mathsf{ample}(s)$ then we put $D_1(\sigma) = D(\sigma)$ for all finite paths σ with $\mathsf{first}(\sigma) = s$.
In what follows, we assume that $D(s) = \alpha_1 \notin \mathsf{ample}(s)$. We first associate with s an action $\beta = \beta_s \in \mathsf{ample}(s)$ as follows. If there is a D-path starting in s with an action sequence $\alpha_1, \ldots, \alpha_n, \gamma$ such that $\alpha_i \notin \mathsf{ample}(s)$, $i = 1, \ldots, n$, and $\gamma \in \mathsf{ample}(s)$ then we put $\beta = \gamma$.[9] If no such path exists then we choose an arbitrary action $\beta \in \mathsf{ample}(s)$. As $\mathsf{ample}(s) \neq \mathsf{Act}(s)$, β is a stutter action (condition (A2)) and $D(s) = \alpha_1$ and β are independent (condition (A3)). In particular, $\alpha_1 \in \mathsf{Act}(t)$ for any β-successor t of s. We define $D_1(s) = \beta$ and for paths of length 1:

$$D_1(s \xrightarrow{\beta} t) = \alpha_1 \qquad \textbf{(*)}$$

To present the definition of $D_1(\sigma_1)$ for paths of length ≥ 2, we need some auxiliary notations for paths.

Notation 8. If ρ is a finite or infinite path then $\mathsf{act}(\rho)$ denotes the action sequence of ρ and $\mathsf{Act}(\rho)$ the set of actions occurring in ρ. For finite paths starting in s, we define $\sim$ to be the finest equivalence such that (1) $\sigma_1 \sim \sigma$ and $\sigma_1 \sim \sigma_1'$ for all finite paths σ_1, σ_1' and σ of the form

$$\begin{aligned}
\sigma_1 &= s \xrightarrow{\beta} t_0 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_{n-1}} t_{n-1} \xrightarrow{\alpha_n} t \\
\sigma_1' &= s \xrightarrow{\beta} u_0 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_{n-1}} u_{n-1} \xrightarrow{\alpha_n} t \\
\sigma &= s \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} s_n \xrightarrow{\beta} t
\end{aligned}$$

where $\alpha_1, \ldots, \alpha_n \notin \mathsf{ample}(s)$ are independent from β and there exist state-labels $\ell_0, \ldots, \ell_n \subseteq \mathsf{AP}$ with $\mathsf{trace}(\sigma_1) = \mathsf{trace}(\sigma_1') = \ell_0, \ell_0, \ell_1, \ldots, \ell_n$, and $\mathsf{trace}(\sigma) = \ell_0, \ell_1, \ldots, \ell_n, \ell_n$, and (2) $\sigma_1 \circ \rho \sim \sigma_1' \circ \rho \sim \sigma \circ \rho$ for all σ_1, σ_1', σ as in (1) and all finite paths ρ starting in t. $[\sigma_1]$ denotes the $\sim$-equivalence class of σ_1. □

The definition for $D_1(\sigma_1)$ is shown in Fig. 9, where σ_1 is a D_1-path of length ≥ 2 which starts in state s and ends in state t. Some explanations are in order. If σ_1 is a D_1-path as in Notation 8 then the D-paths $\sigma \in [\sigma_1]$ have the form

$$\sigma = s \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_i} s_i \xrightarrow{\beta} v_i \xrightarrow{\alpha_{i+1}} \ldots \xrightarrow{\alpha_n} v_n \text{ with } v_n = t$$

where $0 \leq i \leq n$, $\mathsf{L}(s_k) = \ell_k = \mathsf{L}(t_k)$, $k = 1 \ldots, i$, and $\mathsf{L}(v_j) = \ell_j = \mathsf{L}(t_j)$, $j = i, \ldots, n$. Note that the $(i+1)$-st prefix $\sigma\downarrow^{i+1}$ is $\sim$-equivalent to some path of the form

$$s \xrightarrow{\beta} u_0 \xrightarrow{\alpha_1} u_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_i} u_i \text{ where } u_i = v_i.$$

Thus, $\sigma \sim s \xrightarrow{\beta} u_0 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_i} v_i \xrightarrow{\alpha_{i+1}} \ldots \xrightarrow{\alpha_n} v_n \sim \sigma_1$. The paths π in the right sum in Fig. 9 have the form $\pi = s \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} s_n$ where $\alpha_1, \ldots, \alpha_n \notin \mathsf{ample}(s)$ and

$$\pi \xrightarrow{\beta} t \;=\; s \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} s_n \xrightarrow{\beta} t \;\in [\sigma_1].$$

If $\sigma_1 = s \xrightarrow{\beta} t_0 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} t_n \xrightarrow{\gamma_1} \ldots \xrightarrow{\gamma_m} t_{n+m}$ where $\alpha_1, \ldots, \alpha_n \notin \mathsf{ample}(s)$ are independent from β, $\gamma_1 \in \mathsf{ample}(s)$ or γ_1 and β are dependent then the sum on the right in Fig. 9 is 0 as no such path π exists. The D-paths $\sigma \in [\sigma_1]$ have an action sequence $\mathsf{act}(\sigma) = \alpha_1 \ldots \alpha_i \beta \alpha_{i+1} \ldots \alpha_n \gamma_1 \ldots \gamma_m$, coincide with σ_1 latest from the $(n+2)$nd state on and have a trace of the form $\ell_0, \ell_1, \ldots, \ell_{i-1}, \ell_i, \ell_i, \ell_{i+1}, \ldots, \ell_{n+m}$ where $\ell_0 = \mathsf{L}(s)$ and $\ell_j = \mathsf{L}(t_j)$, $j = 1, \ldots, n+m$.

To show that D_1 is a HR-scheduler, we have to prove (i) $D_1(\sigma_1)(\gamma) > 0$ implies $\gamma \in \mathsf{Act}(t)$ and (ii) $\sum_\gamma D_1(\sigma_1)(\gamma) = 1$. This is obvious if σ_1 has length 0 (i.e., $\sigma_1 = s$) or length 1 (see (*)). For $|\sigma_1| \geq 2$, (i) is an easy verification and its

[8] By a distribution on Act, we mean a function $\mu : \mathsf{Act} \to [0,1]$ such that $\sum_{\alpha \in \mathsf{Act}} \mu(\alpha) = 1$.

[9] Note that in this case, γ is uniquely defined, since if one of the α_i's is probabilistic, then $|\mathsf{ample}(s)| = 1$ (condition (A5)). If $\alpha_1, \ldots, \alpha_n$ are non-probabilistic then γ is uniquely defined, because D is a deterministic scheduler.

$$D_1(\sigma_1)(\gamma) = \frac{1}{\sum_{\sigma_1' \in [\sigma_1]} \Pr^{D_1}(\sigma_1')} \cdot \left(\sum_{\sigma \in [\sigma_1]} \Pr^{D}(\sigma) \cdot D(\sigma)(\gamma) \quad + \sum_{\substack{\pi \text{ s.t. } \pi \xrightarrow{\beta} t \in [\sigma_1] \\ \mathsf{ample}(s) \cap \mathsf{Act}(\pi) = \emptyset}} \Pr^{D}(\pi) \cdot D(\pi)(\gamma) \cdot \mathsf{P}(\mathsf{last}(\pi), \beta, t) \right) \quad \text{if } \gamma \neq \beta$$

$$D_1(\sigma_1)(\beta) = \frac{\sum_{\sigma \in [\sigma_1]} \Pr^{D}(\sigma) \cdot D(\sigma)(\beta)}{\sum_{\sigma_1' \in [\sigma_1]} \Pr^{D_1}(\sigma_1')}$$

Fig. 9. **Definition of $D_1(\sigma_1)(\gamma)$ if $\beta \neq \gamma$**

proof is omitted here. (ii) can be derived from the following observation (see page 15):

$$\sum_{\sigma_1' \in [\sigma_1]} \Pr^{D_1}(\sigma_1') = \sum_{\sigma \in [\sigma_1]} \Pr^{D}(\sigma) + \sum_{\substack{\pi \text{ s.t. } \pi \xrightarrow{\beta} t \in [\sigma_1] \\ \mathsf{ample}(s) \cap \mathsf{Act}(\pi) = \emptyset}} \Pr^{D}(\pi) \cdot (1 - D(\pi)(\beta)) \cdot \mathsf{P}(\mathsf{last}(\pi), \beta, t) \quad \textbf{(I)}$$

We now show that D and D_1 yield the same probabilities for all stutter insensitive, measurable languages.

Notation 9. If σ_1 is a finite D_1-path of length ≥ 2 starting in s (thus, the first action in σ_1 is β) then $[\sigma_1]^+$ denotes the union of all equivalence classes $[\sigma_1']$ where σ_1' is a finite path with $\mathsf{act}(\sigma_1) = \mathsf{act}(\sigma_1')$ and $\mathsf{trace}(\sigma_1) = \mathsf{trace}(\sigma_1')$. □

Note that the paths in $[\sigma_1]^+$ are pairwise stutter equivalent. From (I) we may derive:

$$\sum_{\sigma_1' \in [\sigma_1]^+} \Pr^{D_1}(\sigma_1') = \sum_{\sigma \in [\sigma_1]^+} \Pr^{D}(\sigma) + \sum_{\substack{\pi \text{ s.t. } \pi \xrightarrow{\beta} t \in [\sigma_1]^+ \\ \mathsf{ample}(s) \cap \mathsf{Act}(\pi) = \emptyset}} \Pr^{D}(\pi) \cdot (1 - D(\pi)(\beta)) \quad \textbf{(II)}$$

Notation 10. Given sequences $\overline{\ell} = \ell_0, \ell_1, \ldots, \ell_n \in (2^{\mathsf{AP}})^*$ and $\overline{\alpha} = \alpha_1, \ldots, \alpha_n \in \mathsf{Act}^*$, we define the path-set $\Pi(\overline{\ell}, \overline{\alpha})$ consisting of (1) all infinite paths

$$\varsigma = s \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_{i-1}} s_{i-1} \xrightarrow{\beta} t_{i-1} \xrightarrow{\alpha_i} t_i \xrightarrow{\alpha_{i+1}} \ldots \xrightarrow{\alpha_n} t_n \longrightarrow \ldots$$

where $i \in \{1, \ldots, n+1\}$, $\{\alpha_1, \ldots, \alpha_{i-1}\} \cap \mathsf{ample}(s) = \emptyset$ and $\mathsf{trace}(\varsigma)$ has the prefix $\ell_0, \ell_1, \ldots, \ell_{i-1}, \ell_{i-1}, \ell_i, \ell_{i+1}, \ldots, \ell_n$ and (2) all infinite paths

$$\varsigma = s \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} s_n \xrightarrow{\gamma} \ldots$$

where $\{\alpha_1, \ldots, \alpha_n, \gamma\} \cap \mathsf{ample}(s) = \emptyset$ and $\mathsf{trace}(\varsigma)$ has the prefix $\ell_0, \ell_1, \ldots, \ell_n$. □

If σ_1 is a D_1-path with $\mathsf{first}(\sigma_1) = s$, $\mathsf{act}(\sigma_1) = \beta\alpha_1 \ldots \alpha_n$ and $\mathsf{trace}(\sigma_1) = \ell_0, \ell_0, \ell_1, \ldots, \ell_n$ then $\varsigma \in \Pi(\overline{\ell}, \overline{\alpha})$ iff either ς has a prefix in $[\sigma_1]^+$ or $\mathsf{act}(\varsigma) = \alpha_1 \ldots \alpha_n \gamma \ldots$ where $\{\alpha_1, \ldots, \alpha_n, \gamma\} \cap \mathsf{ample}(s) = \emptyset$ and $\ell_0, \ell_1, \ldots, \ell_n$ is a prefix of $\mathsf{trace}(\varsigma)$. (The latter case is only possible if $\alpha_1, \ldots, \alpha_n$ (and γ) are independent from β.) This observation yields:

$$\Pr^{D_1,s}(\Pi(\overline{\ell}, \overline{\alpha})) = \sum_{\sigma_1' \in [\sigma_1]^+} \Pr^{D_1}(\sigma_1') \quad \textbf{(III)}$$

$$\Pr^{D,s}(\Pi(\overline{\ell}, \overline{\alpha})) = \sum_{\sigma \in [\sigma_1]^+} \Pr^{D}(\sigma) + \sum_{\substack{\pi \text{ s.t. } \pi \xrightarrow{\beta} t \in [\sigma_1]^+ \\ \mathsf{ample}(s) \cap \mathsf{Act}(\pi) = \emptyset}} \Pr^{D}(\pi) \cdot (1 - D(\pi)(\beta)) \quad \textbf{(IV)}$$

Combining (II), (III) and (IV) yields:

$$\Pr^{D_1,s}(\Pi(\overline{\ell}, \overline{\alpha})) = \Pr^{D,s}(\Pi(\overline{\ell}, \overline{\alpha})) \quad \textbf{(V)}$$

We now abstract away from the action sequences and define $\Pi(\overline{\ell})$ as the union of all path-sets $\Pi(\overline{\ell}, \overline{\alpha})$. As the path-sets $\Pi(\overline{\ell}, \overline{\alpha})$ are pairwise disjoint, we conclude from (V):

$$\Pr^{D_1,s}(\Pi(\overline{\ell})) = \sum_{\alpha} \Pr^{D_1,s}(\Pi(\overline{\ell}, \overline{\alpha})) = \sum_{\alpha} \Pr^{D,s}(\Pi(\overline{\ell}, \overline{\alpha})) = \Pr^{D,s}(\Pi(\overline{\ell})) \quad \textbf{(VI)}$$

If we are given a trace-cylinder $\mathit{Cyl}(\ell_1^+, \ldots, \ell_n^+) \subseteq (2^{\mathsf{AP}})^\omega$ (where $\ell_1, \ldots, \ell_n$ are pairwise distinct) then the induced path-cylinder $\{\varsigma : \mathsf{trace}(\varsigma) \in \mathit{Cyl}(\ell_1^+, \ldots, \ell_n^+)\}$ agrees with

$$\bigcup_{k_1, \ldots, k_n \geq 1} \left(\bigcup_{\substack{\ell \in 2^{\mathsf{AP}} \\ \ell \neq \ell_n}} \Pi(\ell_1^{k_1}, \ldots, \ell_n^{k_n}, \ell) \cup \bigcap_{k \geq 1} \Pi(\ell_1^{k_1}, \ldots, \ell_n^{k_n}, \ell_n^k) \right)$$

The paths-sets $\Pi(\ell_1^{k_1}, \ldots, \ell_n^{k_n}, \ell)$ and $\bigcap_{k \geq 1} \Pi(\ell_1^{k_1}, \ldots, \ell_n^{k_n}, \ell_n^k)$ are pairwise disjoint. Thus, we obtain from (VI):

$$\Pr^{D_1,s}(\mathit{Cyl}(\ell_1^+, \ldots, \ell_n^+)) = \Pr^{D,s}(\mathit{Cyl}(\ell_1^+, \ldots, \ell_n^+)).$$

Hence, $\Pr^{D_1,s}(\mathbf{L}) = \Pr^{D,s}(\mathbf{L})$ for any stutter insensitive, measurable language $\mathbf{L}$.

Inductive definition of a scheduler sequence. In a similar fashion we can now define a sequence of HR-schedulers $D_2, D_3, \ldots$. Given a D_i-path $\hat{\sigma}$ of length $\leq i-1$, $D_i(\hat{\sigma})$ assigns probability 1 to some action in $\mathsf{ample}(\mathsf{last}(\hat{\sigma}))$. That is, for the first $i-1$ steps, D_i chooses deterministically ample actions, and thus, the $(i-1)$-st prefix of any D_i-path is a path in $\hat{M}$. D_{i+1} is constructed from D_i, mimicking its first $i-1$ steps (i.e., $D_{i+1}(\hat{\sigma}) = D_i(\hat{\sigma})$ if $|\hat{\sigma}| \leq i-1$) and then applying the same technique as in the construction D_1 from D.[10] Using induction on i, we obtain $\mathsf{Pr}^{D_i,s}(\mathbf{L}) = \mathsf{Pr}^{D,s}(\mathbf{L})$ for all stutter insensitive, measurable languages $\mathbf{L}$.

[10] Note that – because of the assumption that the original scheduler D is deterministic and (*) – for any D_i-path σ_i of length i there is at most one action $\beta \in \mathsf{ample}(\mathsf{last}(\sigma_i))$ such that a D_i-path of the form $\sigma_i \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_n} s_n \xrightarrow{\beta} t$ exists where $\alpha_1, \ldots, \alpha_n \notin \mathsf{ample}(\mathsf{last}(\sigma_i))$. Thus, for the definition of D_{i+1} from D_i we are exactly in the same situation as before.

Proof of

$$\textbf{(I)} \qquad \sum_{\sigma_1' \in [\sigma_1]} \mathsf{Pr}^{D_1}(\sigma_1') \;=\; \sum_{\sigma \in [\sigma_1]} \mathsf{Pr}^{D}(\sigma) \;+\; \sum_{\substack{\pi \text{ s.t. } \pi \xrightarrow{\beta} t \in [\sigma_1] \\ \mathsf{ample}(s) \cap \mathsf{Act}(\pi) = \emptyset}} \mathsf{Pr}^{D}(\pi) \cdot (1 - D(\pi)(\beta)) \cdot \mathsf{P}(\mathsf{last}(\pi), \beta, t)$$

Let $\rho_1 = s \xrightarrow{\beta} t_0 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_{n-1}} t_{n-1} \xrightarrow{\alpha_n} t_n$, where $\alpha_1, \ldots, \alpha_n \notin \mathsf{ample}(s)$ are independent from β and $\rho_2 = v_0 \xrightarrow{\gamma_1} v_1 \xrightarrow{\gamma_2} \ldots \xrightarrow{\gamma_{m-1}} v_{m-1} \xrightarrow{\gamma_m} v_m$ where γ_1 is dependent on $\mathsf{ample}(s)$ and $v_0 = t_n$. Let σ_1 be $\rho_1 \circ \rho_2$. Let $\ell_i = \{s : \mathsf{L}(s) = \mathsf{L}(t_i)\},\ 0 \leq i \leq n$. We split the proof of **(I)** in two parts, one dealing with $m \geq 1$ and one dealing with $m = 0$. Given a path σ of length ≥ 1 we will denote the prefix $\sigma \downarrow^{|\sigma|-1}$ by $\sigma \downarrow^{-1}$.

Case $m \geq 1$:

$$\begin{aligned}
&\sum_{\sigma_1' \in [\sigma_1]} \mathsf{Pr}^{D_1}(\sigma_1') && = \\
&\sum_{\substack{x_0, x_i \in \ell_i \\ i=1,\ldots,n-1}} \mathsf{Pr}^{D_1}(s \xrightarrow{\beta} x_0 \xrightarrow{\alpha_1} x_1 \ldots x_{n-1} \xrightarrow{\alpha_n} t_n \circ \rho_2) && = \\
&\sum_{\substack{x_0, x_i \in \ell_i \\ i=1,\ldots,n-1}} \mathsf{Pr}^{D_1}(\underbrace{s \xrightarrow{\beta} x_0 \xrightarrow{\alpha_1} x_1 \ldots x_{n-1} \xrightarrow{\alpha_n} t_n \circ \rho_2 \downarrow^{-1}}_{=\sigma}) \cdot D_1(\sigma)(\gamma_m) \cdot \mathsf{P}(v_{m-1}, \gamma_m, v_m) && \overset{\text{since } \beta \text{ and } \gamma_m \text{ are dependent}}{=} \\
&\sum_{\substack{x_0, x_i \in \ell_i \\ i=1,\ldots,n-1}} \mathsf{Pr}^{D_1}(s \xrightarrow{\beta} x_0 \xrightarrow{\alpha_1} x_1 \ldots x_{n-1} \xrightarrow{\alpha_n} t_n \circ \rho_2 \downarrow^{-1}) \cdot \mathsf{P}(v_{m-1}, \gamma_m, v_m) \cdot \frac{1}{\sum\limits_{\substack{z_0, z_i \in \ell_i \\ i=1,\ldots,n-1}} \mathsf{Pr}^{D_1}(s \xrightarrow{\beta} z_0 \xrightarrow{\alpha_1} z_1 \ldots z_{n-1} \xrightarrow{\alpha_n} t_n \circ \rho_2 \downarrow^{-1})} \cdot && \\
&\Big(\sum_{j=1}^{n} \sum_{\substack{w_0, w_i \in \ell_i \\ i=1,\ldots,n-1}} \mathsf{Pr}^{D}(\underbrace{s \xrightarrow{\alpha_1} w_1 \ldots \xrightarrow{\alpha_j} w_j \xrightarrow{\beta} w_0 \xrightarrow{\alpha_{j+1}} w_{j+1} \ldots \xrightarrow{\alpha_{n-1}} w_{n-1} \xrightarrow{\alpha_n} t_n \circ \rho_2 \downarrow^{-1}}_{=\sigma'}) \cdot D(\sigma')(\gamma_m) \;+\; 0 \Big) && = \\
&\sum_{j=1}^{n} \sum_{\substack{w_0, w_i \in \ell_i \\ i=1,\ldots,n-1}} \Big(\mathsf{Pr}^{D}(s \xrightarrow{\alpha_1} w_1 \ldots \xrightarrow{\alpha_j} w_j \xrightarrow{\beta} w_0 \xrightarrow{\alpha_{j+1}} w_{j+1} \ldots \xrightarrow{\alpha_{n-1}} w_{n-1} \xrightarrow{\alpha_n} t_n \circ \rho_2 \downarrow^{-1}) \cdot D(\sigma')(\gamma_m) \cdot \mathsf{P}(v_{m-1}, \gamma_m, v_m) \cdot && \\
&\frac{1}{\sum\limits_{\substack{z_0, z_i \in \ell_i \\ i=1,\ldots,n-1}} \mathsf{Pr}^{D_1}(s \xrightarrow{\beta} z_0 \xrightarrow{\alpha_1} z_1 \ldots z_{n-1} \xrightarrow{\alpha_n} t_n \circ \rho_2 \downarrow^{-1})} \cdot \sum_{\substack{x_0, x_i \in \ell_i \\ i=1,\ldots,n-1}} \mathsf{Pr}^{D_1}(s \xrightarrow{\beta} x_0 \xrightarrow{\alpha_1} x_1 \ldots x_{n-1} \xrightarrow{\alpha_n} t_n \circ \rho_2 \downarrow^{-1}) \Big) && = \\
&\sum_{j=1}^{n} \sum_{\substack{w_0, w_i \in \ell_i \\ i=1,\ldots,n-1}} \big(\mathsf{Pr}^{D}(s \xrightarrow{\alpha_1} w_1 \ldots \xrightarrow{\alpha_j} w_j \xrightarrow{\beta} w_0 \xrightarrow{\alpha_{j+1}} w_{j+1} \ldots \xrightarrow{\alpha_{n-1}} w_{n-1} \xrightarrow{\alpha_n} t_n \circ \rho_2) && = \\
&\sum_{\sigma \in [\sigma_1]} \mathsf{Pr}^{D}(\sigma), &&
\end{aligned}$$

which is what we wanted to show, since the second sum on the right hand side of equation **(I)** equals zero for the case $m \geq 1$.

Case $m = 0$:
Since $\rho_2 = t_n$, we get

$$\begin{aligned}
&\sum_{\sigma_1' \in [\sigma_1]} \mathsf{Pr}^{D_1}(\sigma_1') && = \\
&\sum_{\substack{x_0, x_i \in \ell_i \\ i=1,\ldots,n-1}} \mathsf{Pr}^{D_1}(s \xrightarrow{\beta} x_0 \xrightarrow{\alpha_1} x_1 \ldots x_{n-1} \xrightarrow{\alpha_n} t_n) && = \\
&\sum_{\substack{x_0, x_i \in \ell_i \\ i=1,\ldots,n-1}} \mathsf{Pr}^{D_1}(\underbrace{s \xrightarrow{\beta} x_0 \xrightarrow{\alpha_1} x_1 \ldots \xrightarrow{\alpha_{n-1}} x_{n-1}}_{=\sigma}) \cdot D_1(\sigma)(\alpha_n) \cdot \mathsf{P}(x_{n-1}, \alpha_n, t_n) && =
\end{aligned}$$

$$\sum_{\substack{x_0,x_i\in\ell_i\\ i=1,\dots,n-1}} \mathsf{Pr}^{D_1}(s \xrightarrow{\beta} x_0 \xrightarrow{\alpha_1} x_1 \dots \xrightarrow{\alpha_{n-1}} x_{n-1}) \cdot \mathsf{P}(x_{n-1},\alpha_n,t_n) \cdot \frac{1}{\sum\limits_{\substack{z_0,z_i\in\ell_i\\ i=1,\dots,n-2}} \mathsf{Pr}^{D_1}(s \xrightarrow{\beta} z_0 \xrightarrow{\alpha_1} z_1 \dots \xrightarrow{\alpha_{n-1}} x_{n-1})} \cdot$$

$$\Big(\sum_{j=1}^{n-1} \sum_{\substack{w_0,w_i\in\ell_i\\ i=1,\dots,n-2}} \mathsf{Pr}^{D}(\underbrace{s \xrightarrow{\alpha_1} w_1 \dots \xrightarrow{\alpha_j} w_j \xrightarrow{\beta} w_0 \xrightarrow{\alpha_{j+1}} w_{j+1} \dots \xrightarrow{\alpha_{n-2}} w_{n-2} \xrightarrow{\alpha_{n-1}} x_{n-1}}_{=\sigma'}) \cdot D(\sigma')(\alpha_n) \;+$$

$$\sum_{\substack{y_i\in\ell_i\\ i=1,\dots,n-1}} \mathsf{Pr}^{D}(\underbrace{s \xrightarrow{\alpha_1} y_1 \dots \xrightarrow{\alpha_{n-1}} y_{n-1}}_{=\sigma''}) \cdot D(\sigma'')(\alpha_n) \cdot \mathsf{P}(y_{n-1},\beta,x_{n-1})\Big) \;=$$

$$\sum_{x_{n-1}\in\ell_{n-1}} \sum_{j=1}^{n-1} \sum_{\substack{w_0,w_i\in\ell_i\\ i=1,\dots,n-2}} \mathsf{Pr}^{D}(s \xrightarrow{\alpha_1} w_1 \dots \xrightarrow{\alpha_j} w_j \xrightarrow{\beta} w_0 \xrightarrow{\alpha_{j+1}} w_{j+1} \dots \xrightarrow{\alpha_{n-2}} w_{n-2} \xrightarrow{\alpha_{n-1}} x_{n-1}) \cdot D(\sigma')(\alpha_n) \;\cdot$$

$$\mathsf{P}(x_{n-1},\alpha_n,t_n) \cdot \frac{1}{\sum\limits_{\substack{z_0,z_i\in\ell_i\\ i=1,\dots,n-2}} \mathsf{Pr}^{D_1}(s \xrightarrow{\beta} z_0 \xrightarrow{\alpha_1} z_1 \dots \xrightarrow{\alpha_{n-1}} x_{n-1})} \cdot \sum_{\substack{x_0,x_i\in\ell_i\\ i=1,\dots,n-2}} \mathsf{Pr}^{D_1}(s \xrightarrow{\beta} x_0 \xrightarrow{\alpha_1} x_1 \dots \xrightarrow{\alpha_{n-1}} x_{n-1}) \;+$$

$$\sum_{x_{n-1}\in\ell_{n-1}} \sum_{\substack{y_i\in\ell_i\\ i=1,\dots,n-1}} \mathsf{Pr}^{D}(s \xrightarrow{\alpha_1} y_1 \dots \xrightarrow{\alpha_{n-1}} y_{n-1}) \cdot D(\sigma'')(\alpha_n) \cdot \mathsf{P}(y_{n-1},\beta,x_{n-1}) \;\cdot$$

$$\mathsf{P}(x_{n-1},\alpha_n,t_n) \cdot \frac{1}{\sum\limits_{\substack{z_0,z_i\in\ell_i\\ i=1,\dots,n-2}} \mathsf{Pr}^{D_1}(s \xrightarrow{\beta} z_0 \xrightarrow{\alpha_1} z_1 \dots \xrightarrow{\alpha_{n-1}} x_{n-1})} \cdot \sum_{\substack{x_0,x_i\in\ell_i\\ i=1,\dots,n-2}} \mathsf{Pr}^{D_1}(s \xrightarrow{\beta} x_0 \xrightarrow{\alpha_1} x_1 \dots \xrightarrow{\alpha_{n-1}} x_{n-1}) \;=$$

$$\textbf{(1)} \quad \sum_{j=1}^{n-1} \sum_{\substack{w_0,w_i\in\ell_i\\ i=1,\dots,n-2\\ x_{n-1}\in\ell_{n-1}}} \mathsf{Pr}^{D}(s \xrightarrow{\alpha_1} w_1 \dots \xrightarrow{\alpha_j} w_j \xrightarrow{\beta} w_0 \xrightarrow{\alpha_{j+1}} w_{j+1} \dots \xrightarrow{\alpha_{n-2}} w_{n-2} \xrightarrow{\alpha_{n-1}} x_{n-1}) \cdot D(\sigma')(\alpha_n) \;\cdot \mathsf{P}(x_{n-1},\alpha_n,t_n) \;+$$

$$\textbf{(2)} \quad \sum_{\substack{y_i\in\ell_i\\ i=1,\dots,n-1}} \mathsf{Pr}^{D}(s \xrightarrow{\alpha_1} y_1 \dots \xrightarrow{\alpha_{n-1}} y_{n-1}) \cdot D(\sigma'')(\alpha_n) \cdot \sum_{x_{n-1}\in\ell_{n-1}} \big(\mathsf{P}(y_{n-1},\beta,x_{n-1}) \;\cdot \mathsf{P}(x_{n-1},\alpha_n,t_n)\big)$$

We compute the sum of **(1)** and **(2)** separately.

$$\textbf{(1)} = \sum_{j=1}^{n-1} \sum_{\substack{w_0,w_i\in\ell_i\\ i=1,\dots,n-1}} \mathsf{Pr}^{D}(s \xrightarrow{\alpha_1} w_1 \dots \xrightarrow{\alpha_j} w_j \xrightarrow{\beta} w_0 \xrightarrow{\alpha_{j+1}} w_{j+1} \dots \xrightarrow{\alpha_{n-2}} w_{n-2} \xrightarrow{\alpha_{n-1}} w_{n-1} \xrightarrow{\alpha_n} t_n)$$

$$\textbf{(2)} \overset{\text{since } \beta \text{ is a stutter action}}{=} \sum_{\substack{y_i\in\ell_i\\ i=1,\dots,n-1}} \mathsf{Pr}^{D}(s \xrightarrow{\alpha_1} y_1 \dots \xrightarrow{\alpha_{n-1}} y_{n-1}) \cdot D(\sigma'')(\alpha_n) \cdot \sum_{x_{n-1}} \big(\mathsf{P}(y_{n-1},\beta,x_{n-1}) \;\cdot \mathsf{P}(x_{n-1},\alpha_n,t_n)\big) \overset{\text{since } \beta \text{ and } \alpha_n \text{ are independent}}{=}$$

$$\sum_{\substack{y_i\in\ell_i\\ i=1,\dots,n-1}} \mathsf{Pr}^{D}(s \xrightarrow{\alpha_1} y_1 \dots \xrightarrow{\alpha_{n-1}} y_{n-1}) \cdot D(\sigma'')(\alpha_n) \cdot \sum_{x_{n-1}} \big(\mathsf{P}(y_{n-1},\alpha_n,x_{n-1}) \cdot \mathsf{P}(x_{n-1},\beta,t_n)\big) \;=$$

$$\sum_{x_{n-1}} \sum_{\substack{y_i\in\ell_i\\ i=1,\dots,n-1}} \mathsf{Pr}^{D}(\underbrace{s \xrightarrow{\alpha_1} y_1 \dots \xrightarrow{\alpha_{n-1}} y_{n-1} \xrightarrow{\alpha_n} x_{n-1}}_{=\tilde{\sigma}}) \cdot \mathsf{P}(x_{n-1},\beta,t_n) \cdot \big((1-D(\tilde{\sigma})(\beta)) + D(\tilde{\sigma})(\beta)\big) \overset{\text{since } \beta \text{ is a stutter action}}{=}$$

$$\sum_{\substack{y_i\in\ell_i\\ i=1,\dots,n}} \mathsf{Pr}^{D}(s \xrightarrow{\alpha_1} y_1 \dots \xrightarrow{\alpha_{n-1}} y_{n-1} \xrightarrow{\alpha_n} y_n) \cdot \mathsf{P}(y_n,\beta,t_n) \cdot (1-D(\tilde{\sigma})(\beta)) \;+$$

$$\sum_{\substack{y_i\in\ell_i\\ i=1,\dots,n}} \mathsf{Pr}^{D}(s \xrightarrow{\alpha_1} y_1 \dots \xrightarrow{\alpha_{n-1}} y_{n-1} \xrightarrow{\alpha_n} y_n) \cdot \mathsf{P}(y_n,\beta,t_n) \cdot D(\tilde{\sigma})(\beta) \;=$$

$$\sum_{\substack{\pi \text{ s.t. } \pi \xrightarrow{\beta} t \in [\sigma_1]\\ \mathsf{ample}(s)\cap\mathsf{Act}(\pi)=\emptyset}} \mathsf{Pr}^{D}(\pi) \cdot (1-D(\pi)(\beta)) \cdot \mathsf{P}(\mathsf{last}(\pi),\beta,t) \;+$$

$$\sum_{\substack{y_i\in\ell_i\\ i=1,\dots,n}} \mathsf{Pr}^{D}(s \xrightarrow{\alpha_1} y_1 \dots \xrightarrow{\alpha_{n-1}} y_{n-1} \xrightarrow{\alpha_n} y_n) \cdot \mathsf{P}(y_n,\beta,t_n) \cdot D(\tilde{\sigma})(\beta) \;=$$

$$\sum_{\substack{\pi \text{ s.t. } \pi \xrightarrow{\beta} t \in [\sigma_1] \\ \mathsf{ample}(s) \cap \mathsf{Act}(\pi) = \emptyset}} \mathsf{Pr}^D(\pi) \cdot (1 - D(\pi)(\beta)) \cdot \mathsf{P}(\mathsf{last}(\pi), \beta, t) \; + \;$$

$$\sum_{\substack{y_i \in \ell_i \\ i=1,\ldots,n}} \mathsf{Pr}^D(s \xrightarrow{\alpha_1} y_1 \ldots \xrightarrow{\alpha_{n-1}} y_{n-1} \xrightarrow{\alpha_n} y_n \xrightarrow{\beta} t_n) \qquad =$$

This concludes to

$$\sum_{\sigma_1' \in [\sigma_1]} \mathsf{Pr}^{D_1}(\sigma_1') \; = \; \mathbf{(1)} + \mathbf{(2)} \; = \; \sum_{\sigma' \in [\sigma_1]} \mathsf{Pr}^D(\sigma') \; + \; \sum_{\substack{\pi \text{ s.t. } \pi \xrightarrow{\beta} t \in [\sigma_1] \\ \mathsf{ample}(s) \cap \mathsf{Act}(\pi) = \emptyset}} \mathsf{Pr}^D(\pi) \cdot (1 - D(\pi)(\beta)) \cdot \mathsf{P}(\mathsf{last}(\pi), \beta, t),$$

which is equation **(I)** and completes the proof.