**Project no.:** PIRSES-GA-2011-295261

**Project full title:** Mobility between Europe and Argentina applying Logics to Systems

**Project Acronym:** MEALS

# Deliverable no.: 1.7 / 1

# Title of Deliverable: Refinement and Difference for Probabilistic Automata

| | |
|---|---|
| **Contractual Date of Delivery to the CEC:** | **30-Sep-2013** |
| **Actual Date of Delivery to the CEC:** | **30-Sep-2013** |
| **Organisation name of lead contractor for this deliverable:** | **UNC** |
| **Author(s):** | **Benoît Delahaye, Uli Fahrenberg Kim G. Larsen, and Axel Legay** |
| **Participants(s):** | **SAU, RWT, TUD, TUE, UNC, UBA, INR** |
| **Work package contributing to the deliverable:** | **WP1** |
| **Nature:** | **R** |
| **Dissemination Level:** | **Public** |
| **Total number of pages:** | **19** |
| **Start date of project:** | 1 Oct. 2011 **Duration:** 48 month |

Abstract:

This paper studies a difference operator for stochastic systems whose specifications are represented by Abstract Probabilistic Automata (APAs). In the case refinement fails between two specifications, the target of this operator is to produce a specification APA that represents all witness PAs of this failure. Our contribution is an algorithm that allows to approximate the difference of two deterministic APAs with arbitrary precision. Our technique relies on new quantitative notions of distances between APAs used to assess convergence of the approximations as well as on an in-depth inspection of the refinement relation for APAs. The procedure is effective and not more complex than refinement checking.

Note:

This deliverable is based on material that has been published in K. Joshi et al. (Eds.), Proceedings of QEST 2013. LNCS 8054, pp. 22–38. Springer, 2013

*This project has received funding from the European Union Seventh Framework Programme (FP7 2007-2013) under Grant Agreement Nr. 295261.*

# Contents

# 1 Introduction

Probabilistic automata as promoted by Segala and Lynch [37] are a widely-used formalism for modeling systems with probabilistic behavior. These include randomized security and communication protocols, distributed systems, biological processes and many other applications. Probabilistic model checking [23, 5, 41] is then used to analyze and verify the behavior of such systems. Given the prevalence of applications of such systems, probabilistic model checking is a field of great interest. However, and similarly to the situation for non-probabilistic model checking, probabilistic model checking suffers from *state space explosion*, which hinders its applicability considerably.

One generally successful technique for combating state space explosion is the use of *compositional* techniques, where a (probabilistic) system is model checked by verifying its components one by one. This compositionality can be obtained by *decomposition*, that is, to check whether a given system satisfies a property, the system is automatically decomposed into components which are then verified. Several attempts at such automatic decomposition techniques have been made [10, 28], but in general, this approach has not been very successful [9].

As an alternative to the standard model checking approaches using logical specifications, such as e.g. LTL, MITL or PCTL [33, 3, 20], automata-based specification theories have been proposed, such as Input/Output Automata [31], Interface Automata [11], and Modal Specifications [29, 34, 6]. These support composition *at specification level*; hence a model which naturally consists of a composition of several components can be verified by model checking each component on its own, against its own specification. The overall model will then automatically satisfy the composition of the component specifications. Remark that this solves the decomposition problem mentioned above: instead of trying to automatically decompose a system for verification, specification theories make it possible to verify the system without constructing it in the first place.

Moreover, specification theories naturally support *stepwise refinement* of specifications, i.e. iterative implementation of specifications, and *quotient*, i.e. the synthesis of missing component specifications given an overall specification and a partial implementation. Hence they allow both logical and compositional reasoning at the same time, which makes them well-suited for compositional verification.

For probabilistic systems, such automata-based specification theories have been first introduced in [25], in the form of Interval Markov Chains. The focus there is only on refinement however; to be able to consider also composition and conjunction, we have in [7] proposed Constraint Markov Chains as a natural generalization which uses general constraints instead of intervals for next-state probabilities.

In [14], we have extended this specification theory to probabilistic automata, which combine stochastic and non-deterministic behaviors. These *Abstract Probabilistic Automata* (APA) combine modal specifications and constraint Markov chains. Our specification theory using APA should be viewed as an alternative to classical PCTL [20], probabilistic I/O automata [32] and stochastic extensions of CSP [21]. Like these, its purpose is model checking of probabilistic properties, but unlike the alternatives, APA support compositionality at specification level.

In the context of refinement of specifications, it is important that informative debugging information is given in case refinement fails. We hence need to be able to compare APA at the

semantic level, i.e. to capture the *difference between their sets of implementations*. This is, then, what we attempt in this paper: given two APAs $N_1$ and $N_2$, to generate another APA $N$ for which $[\![N]\!] = [\![N_1]\!] \setminus [\![N_2]\!]$ (where $[\![N]\!]$ denotes the set of implementations of $N$).

As a second contribution, we introduce a notion of *distance* between APAs which measures how far away one APA is from refining a second one. This distance, adapted from our work in [39, 6], is *accumulating* and *discounted*, so that differences between APAs accumulate along executions, but in a way so that differences further in the future are discounted, i.e. have less influence on the result than had they occurred earlier.

Both difference and distances are important tools to compare APAs which are not in refinement. During an iterative development process, one usually wishes to successively replace specifications by more refined ones, but due to external circumstances such as e.g. cost of implementation, it may happen that a specification needs to be replaced by one which is not a refinement of the old one. This is especially important when models incorporate quantitative information, such as for APAs; the reason for the failed refinement might simply be some changes in probability constraints due to e.g. measurement updates. In this case, it is important to assess precisely *how much* the new specification differs from the old one. Both the distance between the new and old specifications, as well as their precise difference, can aid in this assessment.

Unfortunately, because APAs are finite-state structures, the difference between two APAs cannot always itself be represented by an APA. Instead of extending the formalism, we propose to *approximate* the difference for a subclass of APAs. We introduce both over- and under-approximations of the difference of two *deterministic* APAs. We construct a sequence of under-approximations which converges to the exact difference, hence eventually capturing all PAs in $[\![N_1]\!] \setminus [\![N_2]\!]$, and a fixed over-approximation which may capture also PAs which are not in the exact difference, but whose distance to the exact difference is zero: hence any superfluous PAs which are captured by the over-approximation are infinitesimally close to the real difference. Taken together, these approximations hence solve the problem of assessing the precise difference between deterministic APAs in case of failing refinement.

We restrict ourselves to the subclass of deterministic APAs, as it allows syntactic reasoning to decide and compute refinement. Indeed, for deterministic APAs, syntactic refinement coincides with semantic refinement, hence allowing for efficient procedures. Note that although the class of APAs we consider is called "deterministic", it still offers non-determinism in the sense that one can choose between different actions in a given state. For space reasons, detailed proofs and additional comments are given in [13].

**Related work.** This paper embeds into a series of articles on APA as a specification theory [14, 15, 16]. In [14] we introduce deterministic APA, generalizing earlier work on interval-based abstractions of probabilistic systems [18, 25, 26], and define notions of refinement, logical composition, and structural composition for them. We also introduce a notion of *compositional abstraction* for APA. In [15] we extend this setting to non-deterministic APA and give a notion of (lossy) determinization, and in [16] we introduce the tool APAC. The distance and difference we introduce in the present paper complement the refinement and abstraction from [14].

Compositional abstraction of APA is also considered in [38], but using a different refinement relation. Differences between specifications are developed in [35] for the formalism of modal tran-

sition systems, and distances between specifications, in the variant of weighted modal automata, have been considered in [6]. Distances between probabilistic systems have been introduced in [12, 17, 40].

The originality of our present work is, then, the ability to measure how far away one probabilistic specification is from being a refinement of another, using distances and our new difference operator. Both are important in assessing precisely how much one APA differs from another.

**Acknowledgement.** The authors wish to thank Joost-Pieter Katoen for interesting discussions and insightful comments on the subject of this work.

# 2 Background

Let $Dist(S)$ denote the set of all discrete probability distributions over a finite set $S$ and $\mathbb{B}_2 = \{\top, \bot\}$.

**Definition 1.** A probabilistic automaton (PA) [37] is a tuple $(S, A, L, AP, V, s_0)$, where $S$ is a finite set of states with the initial state $s_0 \in S$, $A$ is a finite set of actions, $L$: $S \times A \times Dist(S) \to \mathbb{B}_2$ is a (two-valued) transition function, $AP$ is a finite set of atomic propositions and $V$: $S \to 2^{AP}$ is a state-labeling function.

Consider a state $s$, an action $a$, and a probability distribution $\mu$. The value of $L(s, a, \mu)$ is set to $\top$ in case there exists a transition from $s$ under action $a$ to a distribution $\mu$ on successor states. In other cases, we have $L(s, a, \mu) = \bot$. We now introduce Abstract Probabilistic Automata (APA) [14], that is a specification theory for PAs. For a finite set $S$, we let $C(S)$ denote the set of constraints over discrete probability distributions on $S$. Each element $\varphi \in C(S)$ describes a set of distributions: $Sat(\varphi) \subseteq Dist(S)$. Let $\mathbb{B}_3 = \{\top, ?, \bot\}$. APAs are formally defined as follows.

**Definition 2.** An APA [14] is a tuple $(S, A, L, AP, V, S_0)$, where $S$ is a finite set of states, $S_0 \subseteq S$ is a set of initial states, $A$ is a finite set of actions, and $AP$ is a finite set of atomic propositions. $L : S \times A \times C(S) \to \mathbb{B}_3$ is a *three*-valued distribution-constraint function, and $V : S \to 2^{2^{AP}}$ maps each state in $S$ to a set of admissible labelings.

APAs play the role of specifications in our framework. An APA transition abstracts transitions of a certain unknown PA, called its implementation. Given a state $s$, an action $a$, and a constraint $\varphi$, the value of $L(s, a, \varphi)$ gives the modality of the transition. More precisely, the value $\top$ means that transitions under $a$ must exist in the PA to some distribution in $Sat(\varphi)$; ? means that these transitions are allowed to exist; $\bot$ means that such transitions must not exist. We will sometimes view $L$ as a *partial* function, with the convention that a lack of value for a given argument is equivalent to the $\bot$ value. The function $V$ labels each state with a subset of the powerset of $AP$, which models a disjunctive choice of possible combinations of atomic propositions. We say that an APA $N = (S, A, L, AP, V, S_0)$ is in *Single Valuation Normal Form* (SVNF) if the valuation function $V$ assigns at most one valuation to all states, i.e. $\forall s \in S, |V(s)| \leq 1$. From [14], we know that every APA can be turned into an APA in SVNF with the same set of implementations. An APA is *deterministic* [14] if (1) there is at most one outgoing transition for each action in all states,

(2) two states with overlapping atomic propositions can never be reached with the same transition, and (3) there is only one initial state.

Note that every PA is an APA in SVNF where all constraints represent single-point distributions. As a consequence, all the definitions we present for APAs in the following can be directly extended to PAs.

Let $N = (S, A, L, AP, V, \{s_0\})$ be an APA in SVNF and let $v \subseteq AP$. Given a state $s \in S$ and an action $a \in A$, we will use the notation $\mathsf{succ}_{s,a}(v)$ to represent the set of potential $a$-successors of $s$ that have $v$ as their valuation. Formally, $\mathsf{succ}_{s,a}(v) = \{s' \in S \mid V(s') = \{v\}, \exists \varphi \in C(S), \mu \in Sat(\varphi) : L(s, a, \varphi) \neq \bot, \mu(s') > 0\}$. When clear from the context, we may use $\mathsf{succ}_{s,a}(s')$ instead of $\mathsf{succ}_{s,a}(V(s'))$. Remark that when $N$ is deterministic, we have $|\mathsf{succ}_{s,a}(v)| \leq 1$ for all $s, a, v$.

# 3 Refinement and Distances between APAs

We introduce the notion of refinement between APAs. Roughly speaking, refinement guarantees that if $A_1$ refines $A_2$, then the set of implementations of $A_1$ is included in the one of $A_2$. We first recall the notion of simulation $\Subset_{\mathcal{R}}$ between two given distributions.

**Definition 3** ([14])**.** Let $S$ and $S'$ be non-empty sets, and $\mu, \mu'$ be distributions; $\mu \in Dist(S)$ and $\mu' \in Dist(S')$. We say that $\mu$ is *simulated* by $\mu'$ with respect to a relation $\mathcal{R} \subseteq S \times S'$ and a *correspondence function* $\delta : S \to (S' \to [0, 1])$ iff

1. for all $s \in S$ with $\mu(s) > 0$, $\delta(s)$ is a distribution on $S'$,
2. for all $s' \in S'$, $\sum_{s \in S} \mu(s) \cdot \delta(s)(s') = \mu'(s')$, and
3. whenever $\delta(s)(s') > 0$, then $(s, s') \in \mathcal{R}$.

We write $\mu \Subset^{\delta}_{\mathcal{R}} \mu'$ if $\mu$ is simulated by $\mu'$ w.r.t $\mathcal{R}$ and $\delta$, and $\mu \Subset_{\mathcal{R}} \mu'$ if there exists $\delta$ with $\mu \Subset^{\delta}_{\mathcal{R}} \mu'$.

We will also need distribution simulations without the requirement of a relation $\mathcal{R} \subseteq S \times S'$ (hence also without claim 3 above); these we denote by $\mu \Subset^{\delta} \mu'$.

**Definition 4** ([14])**.** Let $N_1 = (S_1, A, L_1, AP, V_1, S^1_0)$ and $N_2 = (S_2, A, L_2, AP, V_2, S^2_0)$ be APAs. A relation $\mathcal{R} \subseteq S_1 \times S_2$ is a *refinement* relation if and only if, for all $(s_1, s_2) \in \mathcal{R}$, we have $V_1(s_1) \subseteq V_2(s_2)$ and

1. $\forall a \in A$, $\forall \varphi_2 \in C(S_2)$, if $L_2(s_2, a, \varphi_2) = \top$, then $\exists \varphi_1 \in C(S_1) : L_1(s_1, a, \varphi_1) = \top$ and $\forall \mu_1 \in Sat(\varphi_1)$, $\exists \mu_2 \in Sat(\varphi_2)$ such that $\mu_1 \Subset_{\mathcal{R}} \mu_2$,
2. $\forall a \in A$, $\forall \varphi_1 \in C(S_1)$, if $L_1(s_1, a, \varphi_1) \neq \bot$, then $\exists \varphi_2 \in C(S_2)$ such that $L_2(s_2, a, \varphi_2) \neq \bot$ and $\forall \mu_1 \in Sat(\varphi_1)$, $\exists \mu_2 \in Sat(\varphi_2)$ such that $\mu_1 \Subset_{\mathcal{R}} \mu_2$.

We say that $N_1$ refines $N_2$, denoted $N_1 \preceq N_2$, iff there exists a refinement relation such that $\forall s^1_0 \in S^1_0, \exists s^2_0 \in S^2_0 : (s^1_0, s^2_0) \in \mathcal{R}$. Since any PA $P$ is also an APA, we say that $P$ *satisfies* $N$ (or equivalently $P$ *implements* $N$), denoted $P \models N$, iff $P \preceq N$. In [14], it is shown that for deterministic APAs $N_1, N_2$, we have $N_1 \preceq N_2 \iff [\![N_1]\!] \subseteq [\![N_2]\!]$, where $[\![N_i]\!]$ denotes the set of implementations of APA $N_i$. Hence for deterministic APAs, the difference $[\![N_1]\!] \setminus [\![N_2]\!]$ is non-empty iff $N_1 \not\preceq N_2$. This equivalence breaks for non-deterministic APAs [14], whence we develop our theory only for deterministic APAs.

To show a convergence theorem about our difference construction in Sect. 4.2 below, we need a relaxed notion of refinement which takes into account that APAs are a *quantitative* formalism. Indeed, refinement as of Def. 4 is a purely qualitative relation; if both $N_2 \not\preceq N_1$ and $N_3 \not\preceq N_1$, then there are no criteria to compare $N_2$ and $N_3$ with respect to $N_1$, saying which one is the closest to $N_1$. We provide such a relaxed notion by generalizing refinement to a *discounted distance* which provides precisely such criteria. In Sect. 4.2, we will show how those distances can be used to prove that increasingly precise difference approximations between APAs converge to the real difference. The next definition shows how a distance between states is lifted to a distance between constraints.

**Definition 5.** Let $d : S_1 \times S_2 \to \mathbb{R}^+$ and $\varphi_1 \in C(S_1)$, $\varphi_2 \in C(S_2)$ be constraints in $N_1$ and $N_2$. Define the distance $D_{N_1,N_2}$ between $\varphi_1$ and $\varphi_2$ as follows:

$$D_{N_1,N_2}(\varphi_1, \varphi_2, d) = \sup_{\mu_1 \in Sat(\varphi_1)} \left[ \inf_{\mu_2 \in Sat(\varphi_2)} \left( \inf_{\delta : \mu_1 \Subset^{\delta} \mu_2} \sum_{(s_1,s_2) \in S_1 \times S_2} \mu_1(s_1)\delta(s_1)(s_2)d(s_1, s_2) \right) \right]$$

For the definition of $d$ below, we say that states $s_1 \in S_1$, $s_2 \in S_2$ are *not compatible* if either (1) $V_1(s_1) \neq V_2(s_2)$, (2) there exists $a \in A$ and $\varphi_1 \in C(S_1)$ such that $L_1(s_1, a, \varphi_1) \neq \bot$ and for all $\varphi_2 \in C(S_2)$, $L_2(s_2, a, \varphi_2) = \bot$, or (3) there exists $a \in A$ and $\varphi_2 \in C(S_2)$ such that $L_2(s_2, a, \varphi_2) = \top$ and for all $\varphi_1 \in C(S_1)$, $L_1(s_1, a, \varphi_1) \neq \top$. For compatible states, their distance is similar to the accumulating branching distance on modal transition systems as introduced in [6, 39], adapted to our formalism. In the rest of the paper, the real constant $0 < \lambda < 1$ represents a discount factor. Formally, $d : S_1 \times S_2 \to [0, 1]$ is the least fixpoint to the following system of equations:

$$d(s_1, s_2) = \tag{1}$$
$$\begin{cases} 1 \text{ if } s_1 \text{ is not compatible with } s_2 \\ \max \begin{cases} \max\limits_{\{a,\varphi_1 : L_1(s_1,a,\varphi_1) \neq \bot\}} \min\limits_{\{\varphi_2 : L_2(s_2,a,\varphi_2) \neq \bot\}} \lambda D_{N_1,N_2}(\varphi_1, \varphi_2, d) \\ \max\limits_{\{a,\varphi_2 : L_2(s_2,a,\varphi_2) = \top\}} \min\limits_{\{\varphi_1 : L_1(s_1,a,\varphi_1) = \top\}} \lambda D_{N_1,N_2}(\varphi_1, \varphi_2, d) \end{cases} & \text{otherwise} \end{cases}$$

Since the above system of linear equations defines a *contraction*, the existence and uniqueness of its least fixpoint is ensured, cf. [30]. This definition intuitively extends to PAs, which allows us to propose the two following notions of distance:

**Definition 6.** Let $N_1 = (S_1, A, L_1, AP, V_1, S_0^1)$ and $N_2 = (S_2, A, L_2, AP, V_2, S_0^2)$ be APAs in SVNF. The *syntactic* distance and *thorough* distances between $N_1$ and $N_2$ are defined as follows:

- **Syntactic distance.** $d(N_1, N_2) = \max_{s_0^1 \in S_0^1} (\min_{s_0^2 \in S_0^2} d(s_0^1, s_0^2))$.
- **Thorough distance.** $d_t(N_1, N_2) = \sup_{P_1 \in [\![N_1]\!]} (\inf_{P_2 \in [\![N_2]\!]} d(P_1, P_2))$.

Note that the notion of thorough distance defined above intuitively extends to sets of PAs: given two sets of PAs $\mathbb{S}_1$, $\mathbb{S}_2$, we have $d_t(\mathbb{S}_1, \mathbb{S}_2) = \sup_{P_1 \in \mathbb{S}_1} (\inf_{P_2 \in \mathbb{S}_2} d(P_1, P_2))$.

The intuition here is that $d(s_1, s_2)$ compares not only the probability distributions at $s_1$ and $s_2$, but also (recursively) the distributions at all states reachable from $s_1$ and $s_2$, weighted by their

probability. Each step is discounted by $\lambda$, hence steps further in the future contribute less to the distance. We also remark that $N_1 \preceq N_2$ implies $d(N_1, N_2) = 0$. It can easily be shown, cf. [39], that both $d$ and $d_t$ are *asymmetric pseudometrics* (or *hemimetrics*), i.e. satisfying $d(N_1, N_1) = 0$ and $d(N_1, N_2) + d(N_2, N_3) \geq d(N_1, N_3)$ for all APAs $N_1, N_2, N_3$ (and similarly for $d_t$). The fact that they are only pseudometrics, i.e. that $d(N_1, N_2) = 0$ does not imply $N_1 = N_2$, will play a role in our convergence arguments later. The following proposition shows that the thorough distance is bounded above by the syntactic distance. Hence we can bound distances between (sets of) implementations by the syntactic distance between their specifications.

**Proposition 1.** *For all APAs $N_1$ and $N_2$ in SVNF, it holds that $d_t(N_1, N_2) \leq d(N_1, N_2)$.*

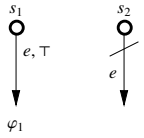
# 4 Difference Operators for Deterministic APAs

The difference $N_1 \setminus N_2$ of two APAs $N_1$, $N_2$ is meant to be a syntactic representation of *all counterexamples*, i.e. all PAs $P$ for which $P \in [\![N_1]\!]$ but $P \notin [\![N_2]\!]$. We will see later that such difference cannot be an APA itself; instead we will *approximate* it using APAs.

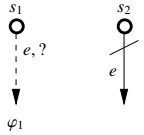
Because $N_1$ and $N_2$ are deterministic, we know that the difference $[\![N_1]\!] \setminus [\![N_2]\!]$ is non-empty if and only if $N_1 \not\preceq N_2$. So let us assume that $N_1 \not\preceq N_2$, and let $\mathcal{R}$ be a maximal refinement relation between $N_1$ and $N_2$. Since $N_1 \not\preceq N_2$, we know that $(s_0^1, s_0^2) \notin \mathcal{R}$. Given $(s_1, s_2) \in S_1 \times S_2$, we can distinguish between the following cases:

1. $(s_1, s_2) \in \mathcal{R}$
2. $V_1(s_1) \neq V_2(s_2)$,
3. $(s_1, s_2) \notin \mathcal{R}$ and $V_1(s_1) = V_2(s_2)$, and

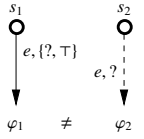
   (a) there exists $e \in A$ and $\varphi_1 \in C(S_1)$ such that $L_1(s_1, e, \varphi_1) = \top$ and $\forall \varphi_2 \in C(S_2) : L_2(s_2, e, \varphi_2) = \bot$,

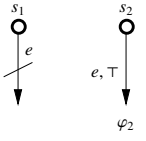
   (b) there exists $e \in A$ and $\varphi_1 \in C(S_1)$ such that $L_1(s_1, e, \varphi_1) =?$ and $\forall \varphi_2 \in C(S_2) : L_2(s_2, e, \varphi_2) = \bot$,

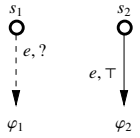
   (c) there exists $e \in A$ and $\varphi_1 \in C(S_1)$ such that $L_1(s_1, e, \varphi_1) \geq?$ and $\exists \varphi_2 \in C(S_2) : L_2(s_2, e, \varphi_2) =?, \exists \mu \in Sat(\varphi_1)$ such that $\forall \mu' \in Sat(\varphi_2) : \mu \not\in_{\mathcal{R}} \mu'$,

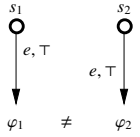
   (d) there exists $e \in A$ and $\varphi_2 \in C(S_2)$ such that $L_2(s_2, e, \varphi_2) = \top$ and $\forall \varphi_1 \in C(S_1) : L_1(s_1, e, \varphi_1) = \bot$,

(e) there exists $e \in A$ and $\varphi_2 \in C(S_2)$ such that $L_2(s_2, e, \varphi_2) = \top$ and $\exists \varphi_1 \in C(S_1) : L_1(s_1, e, \varphi_1) =?$,




(f) there exists $e \in A$ and $\varphi_2 \in C(S_2)$ such that $L_2(s_2, e, \varphi_2) = \top$, $\exists \varphi_1 \in C(S_1) : L_1(s_1, e, \varphi_1) = \top$ and $\exists \mu \in Sat(\varphi_1)$ such that $\forall \mu' \in Sat(\varphi_2) : \mu \not\in_{\mathcal{R}} \mu'$.

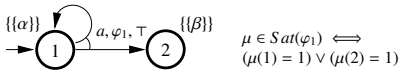



Remark that because of the determinism and SVNF of APAs $N_1$ and $N_2$, cases 1, 2 and 3 cannot happen at the same time. Moreover, although the cases in 3 can happen simultaneously, they cannot be "triggered" by the same action. In order to keep track of these "concurrent" situations, we define the following sets.

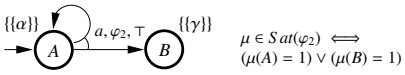
Given a pair of states $(s_1, s_2)$, let us define $B_a(s_1, s_2)$ to be the set of actions in $A$ such that case 3.$a$ above holds. If there is no such action, then $B_a(s_1, s_2) = \emptyset$. Similarly, we define $B_b(s_1, s_2), B_c(s_1, s_2), B_d(s_1, s_2), B_e(s_1, s_2)$ and $B_f(s_1, s_2)$ to be the sets of actions such that cases 3.$b, c, d, e$ and 3.$f$ holds respectively. Given a set $X \subseteq \{a, b, c, d, e, f\}$, let $B_X(s_1, s_2) = \cup_{x \in X} B_x(s_1, s_2)$. In addition, let $B(s_1, s_2) = B_{\{a,b,c,d,e,f\}}(s_1, s_2)$.

## 4.1 Over-Approximating Difference

We now try to compute an APA that represents the difference between the sets of implementations of two APAs. We first observe that such a set may not be representable by an APA, then we will propose over- and under-approximations. Consider the APAs $N_1$ and $N_2$ given in Figures 1a and 1b, where $\alpha \neq \beta \neq \gamma$. Consider the difference of their sets of implementations. It is easy to see that this set contains all the PAs that can finitely loop on valuation $\alpha$ and then move into a state with valuation $\beta$. Since there is no bound on the time spent in the loop, there is no finite-state APA that can represent this set of implementations.




(a) APA $N_1$




(b) APA $N_2$

Figure 1: APAs $N_1$ and $N_2$ such that $[\![N_1]\!] \setminus [\![N_2]\!]$ cannot be represented using a finite-state APA.

Now we propose a construction $\setminus^*$ that over-approximates the difference between APAs in the following sense: given two deterministic APAs $N_1 = (S_1, A, L_1, AP, V_1, \{s_0^1\})$ and $N_2 = (S_2, A, L_2, AP, V_2, \{s_0^2\})$ in SVNF, such that $N_1 \not\leq N_2$, we have $[\![N_1]\!] \setminus [\![N_2]\!] \subseteq [\![N_1 \setminus^* N_2]\!]$. We first observe that if $V_1(s_0^1) \neq V_2(s_0^2)$, i.e. $(s_0^1, s_0^2)$ in case 2, then $[\![N_1]\!] \cap [\![N_2]\!] = \emptyset$. In such case, we define $N_1 \setminus^* N_2$ as $N_1$. Otherwise, we build on the reasons for which refinement fails between $N_1$ and $N_2$. Note that the assumption $N_1 \not\leq N_2$ implies that the pair $(s_0^1, s_0^2)$ can never be in any

refinement relation, hence in case 1. We first give an informal intuition of how the construction works and then define it formally.

In our construction, states in $N_1 \setminus^* N_2$ will be elements of $S_1 \times (S_2 \cup \{\bot\}) \times (A \cup \{\varepsilon\})$. Our objective is to ensure that any implementation of our constructed APA will satisfy $N_1$ and not $N_2$. In $(s_1, s_2, e)$, states $s_1$ and $s_2$ keep track of executions of $N_1$ and $N_2$. Action $e$ is the action of $N_1$ that will be used to break satisfaction with respect to $N_2$, i.e. the action that will be the cause for which any implementation of $(s_1, s_2, e)$ cannot satisfy $N_2$. Since satisfaction is defined recursively, the breaking is not necessarily immediate and can be postponed to successors. $\bot$ is used to represent states that can only be reached after breaking the satisfaction relation to $N_2$. In these states, we do not need to keep track of the corresponding execution in $N_2$, thus only focus on satisfying $N_1$. States of the form $(s_1, s_2, \varepsilon)$ with $s_2 \neq \bot$ are states where the satisfaction is broken by a distribution that does not match constraints in $N_2$ (cases 3.c and 3.f). In order to invalidate these constraints, we still need to keep track of the corresponding execution in $N_2$, hence the use of $\varepsilon$ instead of $\bot$.

The transitions in our construction will match the different cases shown in the previous section, ensuring that in each state, either the relation is broken immediately or reported to at least one successor. Since there can be several ways of breaking the relation in state $(s_0^1, s_0^2)$, each corresponding to an action $e \in B(s_0^1, s_0^2)$, the APA $N_1 \setminus^* N_2$ will have one initial state for each of them. Formally, if $(s_0^1, s_0^2)$ is in case 3, we define the over-approximation of the difference of $N_1$ and $N_2$ as follows.

**Definition 7.** Let $N_1 \setminus^* N_2 = (S, A, L, AP, V, S_0)$, where $S = S_1 \times (S_2 \cup \{\bot\}) \times (A \cup \{\varepsilon\})$, $V(s_1, s_2, a) = V(s_1)$ for all $s_2$ and $a$, $S_0 = \{(s_0^1, s_0^2, f) \mid f \in B(s_0^1, s_0^2)\}$, and $L$ is defined by:

- If $s_2 = \bot$ or $e = \varepsilon$ or $(s_1, s_2)$ in case 1 or 2, then for all $a \in A$ and $\varphi \in C(S_1)$ such that $L_1(s_1, a, \varphi) \neq \bot$, let $L((s_1, s_2, e), a, \varphi^\bot) = L_1(s_1, a, \varphi)$, with $\varphi^\bot$ defined below. For all other $b \in A$ and $\varphi \in C(S)$, let $L((s_1, s_2, e), b, \varphi) = \bot$.
- Else, we have $(s_1, s_2)$ in case 3 and $B(s_1, s_2) \neq \emptyset$ by construction. The definition of $L$ is given in Table 1, with the constraints $\varphi^\bot$ and $\varphi^B_{12}$ defined hereafter.

Given $\varphi \in C(S_1)$, $\varphi^\bot \in C(S)$ is defined as follows: $\mu \in Sat(\varphi^\bot)$ iff $\forall s_1 \in S_1, \forall s_2 \neq \bot, \forall b \neq \varepsilon, \mu(s_1, s_2, b) = 0$ and the distribution $(\mu \downarrow_1 : s_1 \mapsto \mu(s_1, \bot, \varepsilon))$ is in $Sat(\varphi)$. Given a state $(s_1, s_2, e) \in S$ with $s_2 \neq \bot$ and $e \neq \varepsilon$ and two constraints $\varphi_1 \in C(S_1)$, $\varphi_2 \in C(S_2)$ such that $L_1(s_1, e, \varphi_1) \neq \bot$ and $L_2(s_2, e, \varphi_2) \neq \bot$, the constraint $\varphi^B_{12} \in C(S)$ is defined as follows: $\mu \in Sat(\varphi^B_{12})$ iff (1) for all $(s_1', s_2', c) \in S$, we have $\mu(s_1', s_2', c) > 0 \Rightarrow s_2' = \bot$ if $\mathsf{succ}_{s_2,e}(s_1') = \emptyset$ and $\{s_2'\} = \mathsf{succ}_{s_2,e}(s_1')$ otherwise, and $c \in B(s_1', s_2') \cup \{\varepsilon\}$, (2) the distribution $\mu_1 : s_1' \mapsto \sum_{c \in A \cup \{\varepsilon\}, s_2' \in S_2 \cup \{\bot\}} \mu(s_1', s_2', c)$ satisfies $\varphi_1$, and (3) either (a) there exists $(s_1', \bot, c)$ such that $\mu(s_1', \bot, c) > 0$ or (b) the distribution $\mu_2 : s_2' \mapsto \sum_{c \in A \cup \{\varepsilon\}, s_1' \in S_1} \mu(s_1', s_2', c)$ does not satisfy $\varphi_2$, or (c) there exists $s_1' \in S_1$, $s_2' \in S_2$ and $c \neq \varepsilon$ such that $\mu(s_1', s_2', c) > 0$. Informally, distributions in $\varphi^B_{12}$ must (1) follow the corresponding execution is $N_1$ and $N_2$ if possible, (2) satisfy $\varphi_1$ and (3) either (a) reach a state in $N_1$ that cannot be matched in $N_2$ or (b) break the constraint $\varphi_2$, or (c) report breaking the relation to at least one successor state.

The following theorem shows that $N_1 \setminus^* N_2$ is an over-approximation of the difference of $N_1$ and $N_2$ in terms of sets of implementations.

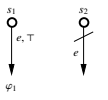
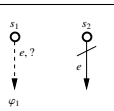
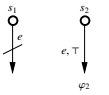
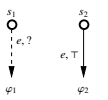
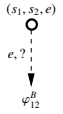
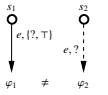
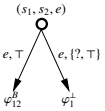
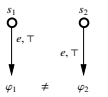
| $e \in$ | $N_1, N_2$ | $N_1 \setminus^* N_2$ | Formal Definition of $L$ |
|---|---|---|---|
| $B_a(s_1, s_2)$ |  |  | For all $a \neq e \in A$ and $\varphi \in C(S_1)$ such that $L_1(s_1, a, \varphi) \neq \bot$, let $L((s_1, s_2, e), a, \varphi^\bot) = L_1(s_1, a, \varphi)$. In addition, let $L((s_1, s_2, e), e, \varphi_1^\bot) = \top$. For all other $b \in A$ and $\varphi \in C(S)$, let $L((s_1, s_2, e), b, \varphi) = \bot$. |
| $B_b(s_1, s_2)$ |  | | |
| $B_d(s_1, s_2)$ |  |  | For all $a \in A$ and $\varphi \in C(S_1)$ such that $L_1(s_1, a, \varphi) \neq \bot$, let $L((s_1, s_2, e), a, \varphi^\bot) = L_1(s_1, a, \varphi)$. For all other $b \in A$ and $\varphi \in C(S)$, let $L((s_1, s_2, e), b, \varphi) = \bot$. |
| $B_e(s_1, s_2)$ |  |  | For all $a \neq e \in A$ and $\varphi \in C(S_1)$ such that $L_1(s_1, a, \varphi) \neq \bot$, let $L((s_1, s_2, e), a, \varphi^\bot) = L_1(s_1, a, \varphi)$. In addition, let $L((s_1, s_2, e), e, \varphi_{12}^B) =?$. For all other $b \in A$ and $\varphi \in C(S)$, let $L((s_1, s_2, e), b, \varphi) = \bot$. |
| $B_c(s_1, s_2)$ |  |  | For all $a \in A$ and $\varphi \in C(S_1)$ such that $L_1(s_1, a, \varphi) \neq \bot$ (including $e$ and $\varphi_1$), let $L((s_1, s_2, e), a, \varphi^\bot) = L_1(s_1, a, \varphi)$. In addition, let $L((s_1, s_2, e), e, \varphi_{12}^B) = \top$. For all other $b \in A$ and $\varphi \in C(S)$, let $L((s_1, s_2, e), b, \varphi) = \bot$. |
| $B_f(s_1, s_2)$ |  | | |

Table 1: Definition of the transition function $L$ in $N_1 \setminus^* N_2$.

**Theorem 1.** *For all deterministic APAs $N_1$ and $N_2$ in SVNF such that $N_1 \not\leq N_2$, we have $[\![N_1]\!] \setminus [\![N_2]\!] \subseteq [\![N_1 \setminus^* N_2]\!]$.*

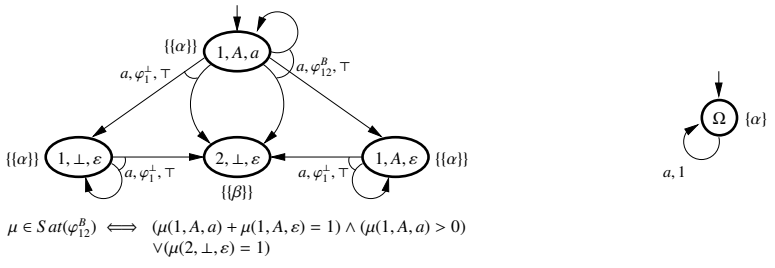
The reverse inclusion unfortunately does not hold. Intuitively, as explained in the construction of the constraint $\varphi_{12}^B$ above, one can postpone the breaking of the satisfaction relation for $N_2$ to the next state (condition (3.c)). This assumption is necessary in order to produce an APA representing *all* counterexamples. However, when there are cycles in the execution of $N_1 \setminus^* N_2$, this assumption allows to postpone forever, thus allowing for implementations that will ultimately satisfy $N_2$. This is illustrated in the following example.

*Example* 1. Consider the APAs $N_1$ and $N_2$ given in Fig. 1. Their over-approximating difference $N_1 \setminus^* N_2$ is given in Fig. 2a. One can see that the PA $P$ in Fig. 2b satisfies both $N_1 \setminus^* N_2$ and $N_2$.

We will later see in Corollary 1 that even though $N_1 \setminus^* N_2$ may be capturing too many counterexamples, the *distance* between $N_1 \setminus^* N_2$ and the real set of counterexamples $[\![N_1]\!] \setminus [\![N_2]\!]$ is zero. This means that the two sets are infinitesimally close to each other, so in this sense, $N_1 \setminus^* N_2$ is the *best possible* over-approximation.

## 4.2 Under-Approximating Difference

We now propose a construction that instead *under-estimates* the difference between APAs. This construction resembles the over-approximation presented in the previous section, the main differ-




(a) $N_1 \setminus^* N_2$ (b) $P$

Figure 2: Over-approximating difference $N_1 \setminus^* N_2$ of APAs $N_1$ and $N_2$ from Figure 1 and PA $P$ such that $P \models N_1 \setminus^* N_2$ and $P \models N_2$.

ence being that in the under-approximation, states are indexed with an integer that represents the maximal depth of the unfolding of counterexamples. The construction is as follows.

Let $N_1 = (S_1, A, L_1, AP, V_1, \{s_0^1\})$ and $N_2 = (S_2, A, L_2, AP, V_2, \{s_0^2\})$ be two deterministic APAs in SVNF such that $N_1 \not\leq N_2$. Let $K \in \mathbb{N}$ be the parameter of our construction. As in Section 4.1, if $V_1(s_0^1) \neq V_2(s_0^2)$, i.e. $(s_0^1, s_0^2)$ in case 2, then $[\![N_1]\!] \cap [\![N_2]\!] = \emptyset$. In this case, we define $N_1 \setminus^K N_2$ as $N_1$. Otherwise, the under-approximation is defined as follows.

**Definition 8.** Let $N_1 \setminus^K N_2 = (S, A, L, AP, V, S_0^K)$, where $S = S_1 \times (S_2 \cup \{\bot\}) \times (A \cup \{\varepsilon\}) \times \{1, \ldots, K\}$, $V(s_1, s_2, a, k) = V(s_1)$ for all $s_2$, $a$, $k < K$, $S_0^K = \{(s_0^1, s_0^2, f, K) \mid f \in B(s_0^1, s_0^2)\}$, and $L$ is defined by:

- If $s_2 = \bot$ or $e = \varepsilon$ or $(s_1, s_2)$ in case 1 or 2, then for all $a \in A$ and $\varphi \in C(S_1)$ such that $L_1(s_1, a, \varphi) \neq \bot$, let $L((s_1, s_2, e, k), a, \varphi^\bot) = L_1(s_1, a, \varphi)$, with $\varphi^\bot$ defined below. For all other $b \in A$ and $\varphi \in C(S)$, let $L((s_1, s_2, e, k), b, \varphi) = \bot$.
- Else we have $(s_1, s_2)$ in case 3 and $B(s_1, s_2) \neq \emptyset$ by construction. The definition of $L$ is given in Table 2. The constraints $\varphi^\bot$ and $\varphi_{12}^{B,k}$ are defined hereafter.

Given a constraint $\varphi \in C(S_1)$, the constraint $\varphi^\bot \in C(S)$ is defined as follows: $\mu \in Sat(\varphi^\bot)$ iff $\forall s_1 \in S_1, \forall s_2 \neq \bot, \forall b \neq \varepsilon, \forall k \neq 1, \mu(s_1, s_2, b, k) = 0$ and the distribution $(\mu \downarrow_1 : s_1 \mapsto \mu(s_1, \bot, \varepsilon, 1))$ is in $Sat(\varphi)$. Given a state $(s_1, s_2, e, k) \in S$ with $s_2 \neq \bot$ and $e \neq \varepsilon$ and two constraints $\varphi_1 \in C(S_1)$ and $\varphi_2 \in C(S_2)$ such that $L_1(s_1, e, \varphi_1) \neq \bot$ and $L_2(s_2, e, \varphi_2) \neq \bot$, the constraint $\varphi_{12}^{B,k} \in C(S)$ is defined as follows: $\mu \in Sat(\varphi_{12}^{B,k})$ iff (1) for all $(s_1', s_2', c, k') \in S$, if $\mu(s_1', s_2', c, k') > 0$, then $c \in B(s_1', s_2') \cup \{\varepsilon\}$ and either $\mathsf{succ}_{s_2,e}(s_1') = \emptyset$, $s_2' = \bot$ and $k' = 1$, or $\{s_2'\} = \mathsf{succ}_{s_2,e}(s_1')$, (2) the distribution $\mu_1 : s_1' \mapsto \sum_{c \in A \cup \{\varepsilon\}, s_2' \in S_2 \cup \{\bot\}, k' \geq 1} \mu(s_1', s_2', c, k')$ satisfies $\varphi_1$, and (3) either (a) there exists $(s_1', \bot, c, 1)$ such that $\mu(s_1', \bot, c, 1) > 0$ , or (b) the distribution $\mu_2 : s_2' \mapsto \sum_{c \in A \cup \{\varepsilon\}, s_1' \in S_1, k' \geq 1} \mu(s_1', s_2', c, k')$ does not satisfy $\varphi_2$, or (c) $k \neq 1$ and there exists $s_1' \in S_1$, $s_2' \in S_2$, $c \neq \varepsilon$ and $k' < k$ such that $\mu(s_1', s_2', c, k') > 0$. The construction is illustrated in Figure 3.

## 4.3 Properties

We already saw in Theorem 1 that $N_1 \setminus^* N_2$ is a correct over-approximation of the difference of $N_1$ by $N_2$ in terms of sets of implementations. The next theorem shows that, similarly, all $N_1 \setminus^K N_2$ are correct under-approximations. Moreover, for increasing $K$ the approximation is

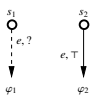
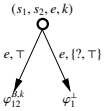
| $e \in$ | $N_1, N_2$ | $N_1 \setminus^K N_2$ | Formal Definition of $L$ |
|---|---|---|---|
| $B_a(s_1, s_2)$ |  |  | For all $a \neq e \in A$ and $\varphi \in C(S_1)$ such that $L_1(s_1, a, \varphi) \neq \bot$, let $L((s_1, s_2, e, k), a, \varphi^\bot) = L_1(s_1, a, \varphi)$. In addition, let $L((s_1, s_2, e, k), e, \varphi_1^\bot) = \top$. For all other $b \in A$ and $\varphi \in C(S)$, let $L((s_1, s_2, e, k), b, \varphi) = \bot$. |
| $B_b(s_1, s_2)$ |  | | |
| $B_d(s_1, s_2)$ |  |  | For all $a \in A$ and $\varphi \in C(S_1)$ such that $L_1(s_1, a, \varphi) \neq \bot$, let $L((s_1, s_2, e, k), a, \varphi^\bot) = L_1(s_1, a, \varphi)$. For all other $b \in A$ and $\varphi \in C(S)$, let $L((s_1, s_2, e, k), b, \varphi) = \bot$. |
| $B_e(s_1, s_2)$ |  |  | For all $a \neq e \in A$ and $\varphi \in C(S_1)$ such that $L_1(s_1, a, \varphi) \neq \bot$, let $L((s_1, s_2, e, k), a, \varphi^\bot) = L_1(s_1, a, \varphi)$. In addition, let $L((s_1, s_2, e, k), e, \varphi_{12}^{B,k}) = ?$. For all other $b \in A$ and $\varphi \in C(S)$, let $L((s_1, s_2, e, k), b, \varphi) = \bot$. |
| $B_c(s_1, s_2)$ |  |  | For all $a \in A$ and $\varphi \in C(S_1)$ such that $L_1(s_1, a, \varphi) \neq \bot$ (including $e$ and $\varphi_1$), let $L((s_1, s_2, e, k), a, \varphi^\bot) = L_1(s_1, a, \varphi)$. In addition, let $L((s_1, s_2, e, k), e, \varphi_{12}^{B,k}) = \top$. For all other $b \in A$ and $\varphi \in C(S)$, let $L((s_1, s_2, e, k), b, \varphi) = \bot$. |
| $B_f(s_1, s_2)$ |  | | |

Table 2: Definition of the transition function $L$ in $N_1 \setminus^K N_2$.

improving, and eventually all PAs in $[\![N_1]\!] \setminus [\![N_2]\!]$ are getting caught. (Hence in a set-theoretic sense, $\lim_{K\to\infty} [\![N_1 \setminus^K N_2]\!] = [\![N_1]\!] \setminus [\![N_2]\!]$.)

**Theorem 2.** *For all deterministic APAs $N_1$ and $N_2$ in SVNF such that $N_1 \not\preceq N_2$:*

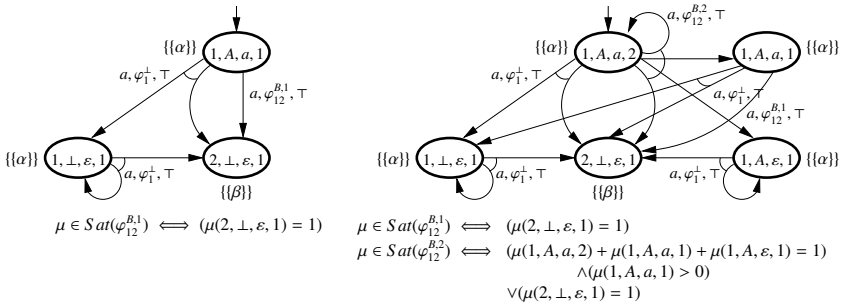
1. *for all $K \in \mathbb{N}$, we have $N_1 \setminus^K N_2 \preceq N_1 \setminus^{K+1} N_2$,*
2. *for all $K \in \mathbb{N}$, $[\![N_1 \setminus^K N_2]\!] \subseteq [\![N_1]\!] \setminus [\![N_2]\!]$, and*
3. *for all PA $P \in [\![N_1]\!] \setminus [\![N_2]\!]$, there exists $K \in \mathbb{N}$ such that $P \in [\![N_1 \setminus^K N_2]\!]$.*

Note that item 3 implies that for all PA $P \in [\![N_1]\!] \setminus [\![N_2]\!]$, there is a finite specification capturing $[\![N_1]\!] \setminus [\![N_2]\!]$ "up to" $P$.

Using our distance defined in Section 3, we can make the above convergence result more precise. The next proposition shows that the speed of convergence is exponential in $K$; hence in practice, $K$ will typically not need to be very large.

**Proposition 2.** *Let $N_1$ and $N_2$ be two deterministic APAs in SVNF such that $N_1 \not\preceq N_2$, and let $K \in \mathbb{N}$. Then $d_t([\![N_1]\!] \setminus [\![N_2]\!], [\![N_1 \setminus^K N_2]\!]) \leq \lambda^K (1-\lambda)^{-1}$.*

For the actual application at hand however, the particular accumulating distance $d$ we have introduced in Section 3 may have limited interest, especially considering that one has to choose a discounting factor for actually calculating it.




(a) $N_1 \setminus^1 N_2$ (b) $N_1 \setminus^2 N_2$

Figure 3: Under-approximations at level 1 and 2 of the difference of APAs $N_1$ and $N_2$ from Figure 1.

What is more interesting are results of a *topological* nature which abstract away from the particular distance used and apply to all distances which are *topologically equivalent* to $d$. The results we present below are of this nature.

It can be shown, c.f. [39], that accumulating distances for different choices of $\lambda$ are topologically equivalent (indeed, even Lipschitz equivalent), hence the particular choice of discounting factor is not important. Also some other system distances are Lipschitz equivalent to the accumulating one, in particular the so-called *point-wise* and *maximum-lead* ones, see again [39].

**Theorem 3.** *Let $N_1$ and $N_2$ be two deterministic APAs in SVNF such that $N_1 \not\leq N_2$.*

1. *The sequence $(N_1 \setminus^K N_2)_{K\in\mathbb{N}}$ converges in the distance $d$, and $\lim_{K\to\infty} d(N_1 \setminus^* N_2, N_1 \setminus^K N_2) = 0$.*
2. *The sequence $([\![N_1 \setminus^K N_2]\!])_{K\in\mathbb{N}}$ converges in the distance $d_t$, and $\lim_{K\to\infty} d_t([\![N_1]\!] \setminus [\![N_2]\!], [\![N_1 \setminus^K N_2]\!]) = 0$.*

Recall that as $d$ and $d_t$ are not metrics, but only (asymmetric) pseudometrics (i.e. hemimetrics), the above sequences may have more than one limit; hence the particular formulation. The theorem's statements are topological as they only allure to convergence of sequences and distance 0; topologically equivalent distances obey precisely the property of having the same convergence behaviour and the same kernel, c.f. [1].

The next corollary, which is easily proven from the above theorem by noticing that its first part implies that also $\lim_{K\to\infty} d_t([\![N_1 \setminus^* N_2]\!], [\![N_1 \setminus^K N_2]\!]) = 0$, shows what we mentioned already at the end of Section 4.1: $N_1 \setminus^* N_2$ is the best possible over-approximation of $[\![N_1]\!] \setminus [\![N_2]\!]$.

**Corollary 1.** *Let $N_1$ and $N_2$ be two deterministic APAs in SVNF such that $N_1 \not\leq N_2$. Then $d_t([\![N_1 \setminus^* N_2]\!], [\![N_1]\!] \setminus [\![N_2]\!]) = 0$.*

Again, as $d_t$ is not a metric, the distance being zero does not imply that the sets $[\![N_1 \setminus^* N_2]\!]$ and $[\![N_1]\!] \setminus [\![N_2]\!]$ are equal; it merely means that they are *indistinguishable* by the distance $d_t$, or infinitesimally close to each other.

# 5 Conclusion

We have in this paper added an important aspect to the specification theory of Abstract Probabilistic Automata, in that we have shown how to exhaustively characterize the *difference* between two deterministic specifications. In a stepwise refinement methodology, difference is an important tool to gauge refinement failures.

We have also introduced a notion of *discounted distance* between specifications which can be used as another measure for how far one specification is from being a refinement of another. Using this distance, we were able to show that our sequence of under-approximations converges, semantically, to the real difference of sets of implementations, and that our over-approximation is infinitesimally close to the real difference.

There are many different ways to measure distances between implementations and specifications, allowing to put the focus on either transient or steady-state behavior. In this paper we have chosen one specific discounted distance, placing the focus on transient behavior. Apart from the fact that this can indeed be a useful distance in practice, we remark that the convergence results about our under- and over-approximations are topological in nature and hence apply with respect to all distances which are topologically equivalent to the specific one used here, typically discounted distances. Although the results presented in the paper do not hold in general for the accumulating (undiscounted) distance, there are other notions of distances that are more relevant for steady-state behavior, e.g. limit-average. Whether our results hold in this setting remains future work.

We also remark that we have shown that it is not more difficult to compute the difference of two APAs than to check for their refinement. Hence if a refinement failure is detected (using e.g. the methods presented in our APAC tool), it is not difficult to also compute the difference for information about the reason for refinement failure.

One limitation of our approach is the use of *deterministic* APAs. Even though deterministic specifications are generally considered to suffice from a modeling point of view [29], non-determinism may be introduced e.g. when composing specifications. Indeed, our constructions themselves introduce non-determinism: for deterministic APAs $N_1$, $N_2$, both $N_1 \setminus^* N_2$ and $N_1 \setminus^K N_2$ may be non-deterministic. Hence it is of interest to extend our approach to non-deterministic specifications. The problem here is, however, that for non-deterministic specifications, the relation between refinement and inclusion of sets of implementations $N_1 \preceq N_2 \iff [\![N_1]\!] \subseteq [\![N_2]\!]$ breaks: we may well have $N_1 \not\preceq N_2$ but $[\![N_1]\!] \subseteq [\![N_2]\!]$, cf. [14]. So the technique we have used in this paper to compute differences will not work for non-deterministic APAs, and techniques based on *thorough refinement* will have to be used.

As a last note, we wish to compare our approach of difference between APA specifications with the use of *counterexamples* in probabilistic model checking. Counterexample generation is studied in a number of papers [2, 19, 42, 4, 24, 36, 22, 43, 8, 27], typically with the purpose of embedding it into a procedure of counterexample guided abstraction refinement (CEGAR). The focus typically is on generation of *one* particular counterexample to refinement, which can then be used to adapt the abstraction accordingly.

In contrast, our approach at computing APA difference generates a representation of *all counterexamples*. Our focus is not on refinement of abstractions at *system* level, using counterex-

amples, but on assessment of *specifications*. This is, then, the reason why we want to compute all counterexamples instead of only one. We remark, however, that our approach also can be used, in a quite simplified version, to generate only one counterexample; details of this are given in [13]. Our work is hence supplementary and orthogonal to the CEGAR-type use of counterexamples: CEGAR procedures can be used also to refine APA specifications, but only our difference can assess the precise distinction between specifications.

# Bibliography

[1] C. D. Aliprantis and K. C. Border. *Infinite Dimensional Analysis: A Hitchhiker's Guide*. Springer, 3rd edition, 2007.

[2] H. Aljazzar and S. Leue. Directed explicit state-space search in the generation of counterexamples for stochastic model checking. *IEEE Trans. Software Eng.*, 2010.

[3] R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996.

[4] M. E. Andrés, P. R. D'Argenio, and P. van Rossum. Significant diagnostic counterexamples in probabilistic model checking. In *HVC*, vol. 5394 of *LNCS*, pp. 129–148. Springer, 2008.

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

[6] S. S. Bauer, U. Fahrenberg, A. Legay, and C. Thrane. General quantitative specification theories with modalities. In *CSR*, vol. 7999 of *LNCS*. Springer, 2012.

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

[8] R. Chadha and M. Viswanathan. A counterexample-guided abstraction-refinement framework for Markov decision processes. *ACM Trans. Comput. Log.*, 12(1):1, 2010.

[9] J. M. Cobleigh, G. S. Avrunin, and L. A. Clarke. Breaking up is hard to do: An evaluation of automated assume-guarantee reasoning. *ACM Trans. Softw. Eng. Methodol.*, 17(2), 2008.

[10] J. M. Cobleigh, D. Giannakopoulou, and C. S. Pasareanu. Learning assumptions for compositional verification. In *TACAS*, vol. 2619 of *LNCS*, pp. 331–346. Springer, 2003.

[11] L. de Alfaro and T. A. Henzinger. Interface automata. In *FSE*, pp. 109–120. ACM, 2001.

[12] L. de Alfaro, R. Majumdar, V. Raman, and M. Stoelinga. Game relations and metrics. In *LICS*, pp. 99–108. IEEE Computer Society, 2007.

[13] B. Delahaye, U. Fahrenberg, K. G. Larsen, and A. Legay. Refinement and difference for probabilistic automata - long version, 2013. Available at http://delahaye.benoit.free.fr/rapports/QEST13-long.pdf.

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

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

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

[17] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for labelled Markov processes. *TCS*, 318(3):323–354, 2004.

[18] H. Fecher, M. Leucker, and V. Wolf. Don't know in probabilistic systems. In *SPIN*, vol. 3925 of *LNCS*, pp. 71–88. Springer, 2006.

[19] T. Han, J.-P. Katoen, and B. Damman. Counterexample generation in probabilistic model checking. *IEEE Trans. Software Eng.*, 35(2):241–257, 2009.

[20] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Asp. Comput.*, 6(5):512–535, 1994.

[21] H. Hermanns, U. Herzog, and J. Katoen. Process algebra for performance evaluation. *TCS*, 274(1-2):43–87, 2002.

[22] H. Hermanns, B. Wachter, and L. Zhang. Probabilistic CEGAR. In *CAV*, vol. 5123 of *LNCS*, pp. 162–175. Springer, 2008.

[23] A. Hinton, M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *TACAS*, LNCS. Springer, 2006.

[24] N. Jansen, E. Ábrahám, J. Katelaan, R. Wimmer, J.-P. Katoen, and B. Becker. Hierarchical counterexamples for discrete-time Markov chains. In *ATVA*, LNCS. Springer, 2011.

[25] B. Jonsson and K. G. Larsen. Specification and refinement of probabilistic processes. In *LICS*, pp. 266–277. IEEE, 1991.

[26] J.-P. Katoen, D. Klink, M. Leucker, and V. Wolf. Three-valued abstraction for continuous-time Markov chains. In *CAV*, vol. 4590 of *LNCS*, pp. 311–324. Springer, 2007.

[27] A. Komuravelli, C. S. Pasareanu, and E. M. Clarke. Assume-guarantee abstraction refinement for probabilistic systems. In *CAV*, vol. 7358 of *LNCS*, pp. 310–326. Springer, 2012.

[28] M. Z. Kwiatkowska, G. Norman, D. Parker, and H. Qu. Assume-guarantee verification for probabilistic systems. In *TACAS*, vol. 6015 of *LNCS*. Springer, 2010.

[29] K. G. Larsen. Modal specifications. In *AVMS*, vol. 407 of *LNCS*, pp. 232–246, 1989.

[30] K. G. Larsen, U. Fahrenberg, and C. Thrane. Metrics for weighted transition systems: Axiomatization and complexity. *TCS*, 412(28):3358–3369, 2011.

[31] N. Lynch and M. R. Tuttle. An introduction to Input/Output automata. *CWI*, 2(3), 1989.

[32] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[33] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer, 1992.

[34] J.-B. Raclet. *Quotient de spécifications pour la réutilisation de composants*. PhD thesis, Université de Rennes I, Dec. 2007. (In French).

[35] M. Sassolas, M. Chechik, and S. Uchitel. Exploring inconsistencies between modal transition systems. *Software and System Modeling*, 10(1):117–142, 2011.

[36] M. Schmalz, D. Varacca, and H. Völzer. Counterexamples in probabilistic LTL model checking for Markov chains. In *CONCUR*, vol. 5710 of *LNCS*, 2009.

[37] R. Segala and N. A. Lynch. Probabilistic simulations for probabilistic processes. In *CONCUR*, vol. 836 of *LNCS*, pp. 481–496. Springer, 1994.

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

[39] C. Thrane, U. Fahrenberg, and K. G. Larsen. Quantitative analysis of weighted transition systems. *JLAP*, 79(7):689–703, 2010.

[40] F. van Breugel, M. W. Mislove, J. Ouaknine, and J. Worrell. An intrinsic characterization of approximate probabilistic bisimilarity. In *FoSSaCS*, vol. 2620 of *LNCS*, pp. 200–215. Springer, 2003.

[41] M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *FOCS*, pp. 327–338. IEEE, 1985.

[42] R. Wimmer, B. Braitling, and B. Becker. Counterexample generation for discrete-time Markov chains using bounded model checking. In *VMCAI*, vol. 5403 of *LNCS*. Springer, 2009.

[43] R. Wimmer, N. Jansen, E. Ábrahám, B. Becker, and J.-P. Katoen. Minimal critical subsystems for discrete-time Markov models. In *TACAS*, vol. 7214 of *LNCS*. Springer, 2012.

# MEALS Partner Abbreviations

**SAU:** Saarland University, D

**RWT:** RWTH Aachen University, D

**TUD:** Technische Universität Dresden, D

**INR:** Institut National de Recherche en Informatique et en Automatique, FR

**IMP:** Imperial College of Science, Technology and Medicine, UK

**ULEIC:** University of Leicester, UK

**TUE:** Technische Universiteit Eindhoven, NL

**UNC:** Universidad Nacional de Córdoba, AR

**UBA:** Universidad de Buenos Aires, AR

**UNR:** Universidad Nacional de Río Cuarto, AR

**ITBA:** Instituto Técnológico Buenos Aires, AR