Assume-Guarantee Verification for Probabilistic Systems

Marta Kwiatkowska[1], Gethin Norman[2], David Parker[1], and Hongyang Qu[1]

[1] Oxford University Computing Laboratory, Parks Road, Oxford, OX1 3QD, UK
[2] Department of Computing Science, University of Glasgow, Glasgow, G12 8RZ, UK

Abstract. We present a compositional verification technique for systems that exhibit both probabilistic and nondeterministic behaviour. We adopt an assume-guarantee approach to verification, where both the assumptions made about system components and the guarantees that they provide are regular safety properties, represented by finite automata. Unlike previous proposals for assume-guarantee reasoning about probabilistic systems, our approach does not require that components interact in a fully synchronous fashion. In addition, the compositional verification method is efficient and fully automated, based on a reduction to the problem of multi-objective probabilistic model checking. We present asymmetric and circular assume-guarantee rules, and show how they can be adapted to form quantitative queries, yielding lower and upper bounds on the actual probabilities that a property is satisfied. Our techniques have been implemented and applied to several large case studies, including instances where conventional probabilistic verification is infeasible.

1 Introduction

Many computerised systems exhibit probabilistic behaviour, for example due to the use of randomisation (e.g. in distributed communication or security protocols), or the presence of failures (e.g. in faulty devices or unreliable communication media). The prevalence of such systems in today's society makes techniques for their formal verification a necessity. This requires models and formalisms that incorporate both *probability* and *nondeterminism*. Although efficient algorithms for verifying such models are known [3,8] and mature tool support [11,7] exists, applying these techniques to large, real-life systems remains challenging, and hence techniques to improve scalability are essential.

In this paper, we focus on *compositional* verification techniques for probabilistic and nondeterministic systems, in which a system comprising multiple interacting components can be verified by analysing each component in isolation, rather than verifying the much larger model of the whole system. In the case of *non-probabilistic* models, a successful approach is the use of *assume-guarantee* reasoning. This is based on checking queries of the form $\langle A \rangle\, M\, \langle G \rangle$, with the meaning "whenever component M is part of a system satisfying the *assumption* A, then the system is *guaranteed* to satisfy property G". Proof rules can then

J. Esparza and R. Majumdar (Eds.): TACAS 2010, LNCS 6015, pp. 23–37, 2010.

be established that show, for example, that if $\langle true \rangle\, M_1\, \langle A \rangle$ (process M_1 satisfies assumption A in any environment) and $\langle A \rangle\, M_2\, \langle G \rangle$ hold, then the combined system $M_1 \| M_2$ satisfies G. For *probabilistic* systems, compositional approaches have also been studied, but a distinct lack of practical progress has been made. In this paper, we address this limitation, presenting the first fully-automated technique for compositional verification of systems exhibiting both probabilistic and nondeterministic behaviour, and illustrating its applicability and efficiency on several large case studies.

We use *probabilistic automata* [20,21], a well-studied formalism that is naturally suited to modelling multi-component probabilistic systems. Indeed, elegant proof techniques have been developed and used to manually prove correctness of large, complex randomised algorithms [18]. Several branching-time preorders (simulation and bisimulation) have been proposed for probabilistic automata and have been shown to be compositional (i.e. preserved under parallel composition) [21], but such branching-time equivalences are often too fine to give significant practical advantages for compositional verification.

A coarser linear-time preorder can be obtained through *trace distribution* (probability distributions over sequences of observable actions) inclusion [20]; however, it is well known that this relation is not preserved under parallel composition [19]. Various attempts have been made to characterise refinement relations that *are* preserved, e.g. [20,15]. An alternative direction is to restrict the forms of parallel composition that are allowed. One example is the formalism of switched probabilistic I/O automata [6], which places restrictions on the scheduling between parallel components. Another is [1] which uses a probabilistic extension of Reactive Modules, restricted to synchronous parallel composition. A limitation of all these approaches is that the relations used, such as trace distribution inclusion and weak probabilistic simulation, are not efficiently computable.

We propose an assume-guarantee verification technique for probabilistic automata, that has no restrictions on the parallel composition permitted between components, allowing greater flexibility to model complex systems. To achieve this, we represent both the assumptions made about system components and the guarantees that they provide as *safety properties*. In the context of probabilistic systems, safety properties capture a wide range of useful properties, e.g. "the maximum probability of an error occurring is at most 0.01" or "the minimum probability of terminating within k time-units is at least 0.75".

We represent safety properties using finite automata and show that verifying assume-guarantee queries reduces to the problem of *multi-objective model checking* for probabilistic automata [10], which can be implemented efficiently using linear programming. Another key benefit of using finite automata in this way is illustrated by the (non-probabilistic) assume-guarantee verification framework of [16]. There, not only is the verification of queries fully automated, but the assumptions themselves (represented as finite automata) are generated automatically using learning techniques. This opens the way for applying learning techniques to compositional verification in the probabilistic case.

We use our definitions of probabilistic assume guarantee reasoning to formulate and prove several assume-guarantee proof rules, representing commonly occurring patterns of processes. We also discuss how to employ *quantitative* reasoning, in particular obtaining lower and upper bounds on the actual probability that a system satisfies a safety property. The techniques have been implemented in a prototype tool and applied to several large case studies. We demonstrate significant speed-ups over traditional, non-compositional verification, and successfully verify models that cannot be analysed without compositional techniques.

A full version of this paper, including additional proofs, is available as [12].

Related work. In addition to the compositional techniques for probabilistic systems surveyed above [6,1,15,18,19,20,21], we mention several other related pieces of work. In particular, our approach was inspired by the large body of work by Giannakopoulou, Pasareanu et al. (see e.g. [16]) on *non-probabilistic* assume guarantee techniques. We also build upon ideas put forward in [10], which suggests using multi-objective verification to check probabilistic assume-guarantee queries. Also relevant are: [9], which presents an assume/guarantee framework using probabilistic contracts for non-probabilistic models; [4], which presents a theoretical framework for compositional verification of quantitative (but not probabilistic) properties; and [17], which uses probabilistic automata to model the environment of non-probabilistic components.

2 Background

We begin by briefly reviewing probabilistic automata and techniques for their verification. We also introduce safety properties, in the context of probabilistic systems, and discuss multi-objective model checking.

In the following, we use $Dist(S)$ to denote the set of all discrete probability distributions over a set S, η_s for the point distribution on $s \in S$, and $\mu_1 \times \mu_2 \in Dist(S_1 \times S_2)$ for the product distribution of $\mu_1 \in Dist(S_1)$ and $\mu_2 \in Dist(S_2)$.

2.1 Probabilistic Automata

Probabilistic automata [20,21] are a modelling formalism for systems that exhibit both probabilistic and nondeterministic behaviour.

Definition 1. *A* probabilistic automaton *(PA) is a tuple* $M = (S, \overline{s}, \alpha_M, \delta_M, L)$ *where* S *is a set of states,* $\overline{s} \in S$ *is an initial state,* α_M *is an alphabet,* $\delta_M \subseteq S \times (\alpha_M \cup \{\tau\}) \times Dist(S)$ *is a probabilistic transition relation and* $L : S \to 2^{AP}$ *is a labelling function, assigning atomic propositions from a set* AP *to each state.*

In any state s of a PA M, a *transition*, denoted $s \xrightarrow{a} \mu$, where a is an *action* label and μ is a discrete probability distribution over states, is available[1] if $(s, a, \mu) \in \delta_M$. In an execution of the model, the choice between the available

[1] Markov decision processes, another commonly used model, are PAs with the restriction that action labels are unique amongst the available transitions for each state.

transitions in each state is nondeterministic; the choice of successor state is then made randomly according to the distribution μ. A *path* through M is a (finite or infinite) sequence $s_0 \xrightarrow{a_0,\mu_0} s_1 \xrightarrow{a_1,\mu_1} \cdots$ where $s_0 = \overline{s}$ and, for each $i \geqslant 0$, $s_i \xrightarrow{a_i} \mu_i$ is a transition and $\mu_i(s_{i+1}) > 0$. The sequence of actions $a_0, a_1, \ldots$, after removal of any "internal actions" τ, from a path π is called a *trace* and is denoted $tr(\pi)$.

To reason about PAs, we use the notion of *adversaries* (also called schedulers or strategies), which resolve the nondeterministic choices in a model, based on its execution history. Formally an adversary σ maps any finite path to a sub-distribution over the available transitions in the last state of the path. Adversaries are defined in terms of sub-distributions because they can opt to (with some probability) take none of the available choices and remain in the current state. For this reason, they are are sometimes called *partial* adversaries. Occasionally, we will distinguish between these and *complete* adversaries, in which all the distributions are total.

We denote by $Path^{\sigma}_{M}$ the set of all paths through M when controlled by adversary σ, and by Adv_M the set of all possible adversaries for M. Under an adversary σ, we define a probability space Pr^{σ}_{M} over the set of paths $Path^{\sigma}_{M}$, which captures the (purely probabilistic) behaviour of M under σ.

To reason about probabilistic systems comprising multiple components, we will need the notions of *parallel composition* and *alphabet extension*:

Definition 2 (Parallel composition of PAs). *If $M_1 = (S_1, \overline{s}_1, \alpha_{M_1}, \delta_{M_1}, L_1)$ and $M_2 = (S_2, \overline{s}_2, \alpha_{M_2}, \delta_{M_2}, L_2)$ are PAs, then their parallel composition, denoted $M_1 \| M_2$, is given by the PA $(S_1 \times S_2, (\overline{s}_1, \overline{s}_2), \alpha_{M_1} \cup \alpha_{M_2}, \delta_{M_1 \| M_2}, L)$ where $\delta_{M_1 \| M_2}$ is defined such that $(s_1, s_2) \xrightarrow{a} \mu_1 \times \mu_2$ if and only if one of the following holds:*

- $s_1 \xrightarrow{a} \mu_1$, $s_2 \xrightarrow{a} \mu_2$ *and* $a \in \alpha_{M_1} \cap \alpha_{M_2}$
- $s_1 \xrightarrow{a} \mu_1$, $\mu_2 = \eta_{s_2}$ *and* $a \in (\alpha_{M_1} \backslash \alpha_{M_2}) \cup \{\tau\}$
- $s_2 \xrightarrow{a} \mu_2$, $\mu_1 = \eta_{s_1}$ *and* $a \in (\alpha_{M_2} \backslash \alpha_{M_1}) \cup \{\tau\}$

and $L(s_1, s_2) = L_1(s_1) \cup L_2(s_2)$.

Definition 3 (Alphabet extension). *For any PA $M = (S, \overline{s}, \alpha_M, \delta_M, L)$ and set of actions Σ, we* extend the alphabet *of M to Σ, denoted $M[\Sigma]$, as follows: $M[\Sigma] = (S, \overline{s}, \alpha_M \cup \Sigma, \delta_{M[\Sigma]}, L)$ where $\delta_{M[\Sigma]} = \delta_M \cup \{(s, a, \eta_s) \mid s \in S \wedge a \in \Sigma \backslash \alpha_M\}$.*

We also require the notion of *projections*. First, for any state $s = (s_1, s_2)$ of $M_1 \| M_2$, the projection of s onto M_i, denoted by $s \lceil_{M_i}$, is s_i. We extend this notation to distributions over the state space $S_1 \times S_2$ of $M_1 \| M_2$ in the standard manner. Next, for any path π of $M_1 \| M_2$, the projection of π onto M_i, denoted $\pi \lceil_{M_i}$, is the path obtained from π by projecting each state of π onto M_i and removing all the actions not in α_{M_i} together with the subsequent states.

Definition 4 (Projections of adversaries). *Let M_1 and M_2 be PAs and σ an adversary of $M_1 \| M_2$. The* projection *of σ onto M_i, denoted $\sigma \lceil_{M_i}$, is the adversary on M_i where, for any finite path π of M_i:*

$$\sigma \lceil_{M_i}(\pi)(a, \mu) = \sum \{\!| Pr(\pi') \cdot \sigma(\pi')(a, \mu') \mid \pi' \in Path^{\sigma}_{M_1 \| M_2} \wedge \pi' \lceil_{M_i} = \pi \wedge \mu' \lceil_{M_i} = \mu |\!\} .$$

Compositional reasoning about PAs, and in particular adversary projections, necessitates the use of partial, rather than complete, adversaries. In particular, even if an adversary σ of $M_1 \| M_2$ is complete, the projection $\sigma{\restriction_{M_i}}$ onto one component may be partial.

2.2 Model Checking for PAs

The verification of PAs against properties specified either in temporal logic or as automata has been well studied. In this paper, both the states and transitions of PAs are labelled (with sets of atomic propositions and actions, respectively) and we formulate properties that refer to both types of labels. For the former, we will express properties in linear temporal logic (LTL), and for the latter, we will use safety properties represented by deterministic finite automata.

LTL Verification. For an LTL formula ψ, PA M and adversary $\sigma \in Adv_M$:

$$Pr_M^\sigma(\psi) \stackrel{\text{def}}{=} Pr_M^\sigma\{\pi \in Path_M^\sigma \mid \pi \models \psi\}$$

where $\pi \models \psi$ denotes satisfaction according to the standard semantics of LTL. Verifying an LTL specification ψ against M typically involves checking that the probability of satisfying ψ meets a probability bound for all adversaries. This reduces to computing the minimum or maximum probability of satisfying ψ:

$$Pr_M^{\min}(\psi) \stackrel{\text{def}}{=} \inf_{\sigma \in Adv_M} Pr_M^\sigma(\psi) \quad \text{and} \quad Pr_M^{\max}(\psi) \stackrel{\text{def}}{=} \sup_{\sigma \in Adv_M} Pr_M^\sigma(\psi)\,.$$

The complexity of this computation is polynomial in the size of M and doubly exponential in the size of ψ [8]. In practice, the LTL formula ψ is small and, for simple, commonly used cases such as $\Diamond ap$ ("eventually ap") or $\Box ap$ ("globally ap"), model checking is polynomial [3]. Furthermore, efficient implementations of LTL verification exist in tools such as PRISM [11] and LiQuor [7].

Safety Properties. A *regular safety property* A represents a set of infinite words, denoted $\mathcal{L}(A)$, that is characterised by a regular language of *bad prefixes*, finite words of which any extension is not in $\mathcal{L}(A)$. More precisely, we will define a regular safety property A by a (complete) deterministic finite automaton (DFA) $A^{err} = (Q, \overline{q}, \alpha_A, \delta_A, F)$, comprising states Q, initial state $\overline{q} \in Q$, alphabet α_A, transition function $\delta_A : Q \times \alpha_A \to Q$ and accepting states $F \subseteq Q$. The DFA A^{err} defines, in standard fashion, a regular language $\mathcal{L}(A^{err}) \subseteq (\alpha_A)^*$. The language $\mathcal{L}(A)$ is then defined as $\mathcal{L}(A) = \{w \in (\alpha_A)^\omega \mid \text{no prefix of } w \text{ is in } \mathcal{L}(A^{err})\}$.

Given a PA M, adversary $\sigma \in Adv_M$ and regular safety property A with $\alpha_A \subseteq \alpha_M$, we define the probability of M under σ satisfying A as:

$$Pr_M^\sigma(A) \stackrel{\text{def}}{=} Pr_M^\sigma\{\pi \in Path_M^\sigma \mid tr(\pi){\restriction_{\alpha_A}} \in \mathcal{L}(A)\}$$

where $w{\restriction_\alpha}$ is the projection of word w onto a subset α of its alphabet. We then define $Pr_M^{\min}(A)$ and $Pr_M^{\max}(A)$ as for LTL above.

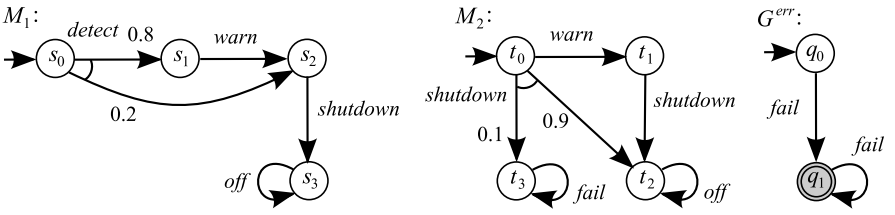

Fig. 1. Two probabilistic automata M_1, M_2 and the DFA for a safety property G

Definition 5 (Probabilistic safety properties). *A* probabilistic safety property $\langle A \rangle_{\geqslant p}$ *comprises a regular safety property* A *and a rational probability bound* p*. We say that a PA* M *satisfies the property, denoted* $M \models \langle A \rangle_{\geqslant p}$*, if the probability of satisfying* A *is at least* p *for any adversary:*

$$M \models \langle A \rangle_{\geqslant p} \quad \Leftrightarrow \quad \forall \sigma \in Adv_M \; . \; Pr^{\sigma}_M(A) \geqslant p \quad \Leftrightarrow \quad Pr^{\min}_M(A) \geqslant p \, .$$

Safety properties can be used to represent a wide range of useful properties of probabilistic automata. Examples include:

- "the probability of an error occurring is at most 0.01"
- "event A always occurs before event B with probability at least 0.98"
- "the probability of terminating within k time-units is at least 0.75"

The last of these represents a very useful class of properties for *timed* probabilistic systems, perhaps not typically considered as safety properties. Using the *digital clocks* approach of [13], verifying real-time probabilistic systems can often be reduced to analysis of a PA with time steps encoded as a special action type. Such requirements are then naturally encoded as safety properties.

Example 1. Figure 1 shows two PAs M_1 and M_2. Component M_1 represents a controller that powers down devices. Upon receipt of the *detect* signal, it first issues the *warn* signal followed by *shutdown*; however, with probability 0.2 it will fail to issue the *warn* signal. M_2 represents a device which, given the *shutdown* signal, powers down correctly if it first receives the *warn* signal and otherwise will only power down correctly 90% of the time. We consider a simple safety property G "action *fail* never occurs", represented by the DFA G^{err} also shown in Figure 1. Composing the two PAs in parallel and applying model checking, we have that $Pr^{\min}_{M_1 \| M_2}(G) = 0.98$. Thus, $M_1 \| M_2 \models \langle G \rangle_{\geqslant 0.98}$.

Safety Verification. Using standard automata-based techniques for model checking PAs [8], verifying correctness of probabilistic safety properties reduces to model checking the product of a PA and a DFA:

Definition 6 (PA-DFA product). *The* product *of a PA* $M{=}(S, \overline{s}, \alpha_M, \delta_M, L)$ *and DFA* $A^{err}{=}(Q, \overline{q}, \alpha_A, \delta_A, F)$ *with* $\alpha_A \subseteq \alpha_M$ *is given by the PA* $M \otimes A^{err} = (S{\times}Q, (\overline{s}, \overline{q}), \alpha_M, \delta', L')$ *where:*

- $(s, q) \xrightarrow{a} \mu{\times}\eta_{q'}$ *if* $s \xrightarrow{a} \mu$ *and* $q' = \delta_A(q, a)$ *if* $a \in \alpha_A$ *and* $q' = q$ *otherwise;*
- $L'(s, q) = L(s) \cup \{ err_A \}$ *if* $q \in F$ *and* $L'(s, q) = L(s)$ *otherwise.*

Proposition 1. *For PA M and regular safety property A, we have:*

$$Pr^{\min}_M(A) = 1 - Pr^{\max}_{M \otimes A^{err}}(\Diamond err_A)\,.$$

Thus, using [3], satisfaction of the probabilistic safety property $\langle A \rangle_{\geqslant p}$ can be checked in time polynomial in the size of $M \otimes A^{err}$. Note that maximum reachability probabilities, and therefore satisfaction of probabilistic safety properties, are independent of whether complete or partial adversaries are considered.

Multi-objective Model Checking. In addition to traditional probabilistic model checking techniques, the approach presented in this paper requires the use of *multi-objective* model checking [10]. The conventional approach described above allows us to check whether, for all adversaries (or, dually, for at least one adversary), the probability of some property is above (or below) a given bound. Multi-objective queries allow us to check the existence of an adversary satisfying *multiple* properties of this form. In particular, consider k predicates of the form $Pr^{\sigma}_M(\psi_i) \sim_i p_i$ where ψ_i is an LTL formula, $p_i \in [0,1]$ is a rational probability bound and $\sim_i \in \{\geqslant, >\}$. Using the techniques in [10], we can verify whether:

$$\exists \sigma \in Adv_M \;.\; \wedge_{i=1}^{k} \left(Pr^{\sigma}_M(\psi_i) \sim_i p_i\right)$$

by a reduction to a linear programming (LP) problem. Like for (single-objective) LTL verification, this can be done in time polynomial in the size of M (and doubly exponential in the sizes of ψ_i). In fact, [10] also shows that this technique generalises to checking existential or universal queries over a Boolean combination of predicates for which $\sim_i \in \{\geqslant, >, \leqslant, <\}$. In all cases, if an adversary which satisfies the predicates exists, then it can also easily be obtained.

Finally, through a trivial extension of this approach (and without increasing the complexity), we can formulate *quantitative* multi-objective queries. For example, given a conjunction of the above predicates $\Psi = \wedge_{i=1}^{k} Pr^{\sigma}_M(\psi_i) \sim_i p_i$, and an additional LTL formula ψ_0, we can compute the maximum probability of ψ_0 that is achievable whilst also satisfying Ψ:

$$Pr^{\max}_M(\psi_0 \,|\, \Psi) \stackrel{\text{def}}{=} \sup\{Pr^{\sigma}_M(\psi_0) \,|\, \sigma \in Adv_M \wedge \Psi)\}\,.$$

3 Compositional Verification for PAs

We now describe our approach for compositional verification of probabilistic automata. We first define the basic underlying ideas and then present several different proof rules. For clarity, we present the simplest of these rules in some detail and then discuss some generalisations and extensions.

We extend the notion of *assume-guarantee* reasoning to PAs using *probabilistic assume-guarantee triples* of the form $\langle A \rangle_{\geqslant p_A} M \langle G \rangle_{\geqslant p_G}$, where $\langle A \rangle_{\geqslant p_A}$ and $\langle G \rangle_{\geqslant p_G}$ are probabilistic safety properties and M is a PA. Informally, the

meaning of this is "whenever M is part of a system satisfying A with probability at least p_A, then the system will satisfy G with probability at least p_G". Formally:

Definition 7 (Assume-guarantee semantics). *If $\langle A\rangle_{\geqslant p_A}$ and $\langle G\rangle_{\geqslant p_G}$ are probabilistic safety properties, M is a PA and $\alpha_G \subseteq \alpha_A \cup \alpha_M$, then*

$$\langle A\rangle_{\geqslant p_A}\, M\, \langle G\rangle_{\geqslant p_G} \;\Leftrightarrow\; \forall \sigma \in Adv_{M[\alpha_A]}\,.\,\left(Pr^{\sigma}_{M[\alpha_A]}(A) \geqslant p_A \rightarrow Pr^{\sigma}_{M[\alpha_A]}(G) \geqslant p_G\right).$$

The use of $M[\alpha_A]$, i.e. M extended to the alphabet of A, in this definition is required for the case where the property G includes actions that are not in M.

We write $\langle true\rangle\, M\, \langle G\rangle_{\geqslant p_G}$ to denote the absence of any assumption, i.e. the query $\langle true\rangle\, M\, \langle G\rangle_{\geqslant p_G}$ is equivalent to $M \models \langle G\rangle_{\geqslant p_G}$ which, as described above, is standard model checking [3]. In the general case, we check the satisfaction of a probabilistic assume-guarantee triple using multi-objective PA model checking:

Proposition 2 (Assume-guarantee model checking). *Let M be a PA, $\langle A\rangle_{\geqslant p_A}$, $\langle G\rangle_{\geqslant p_G}$ be probabilistic safety properties and $M' = M[\alpha_A] \otimes A^{err} \otimes G^{err}$. The probabilistic assume-guarantee triple $\langle A\rangle_{\geqslant p_A}\, M\, \langle G\rangle_{\geqslant p_G}$ holds if and only if:*

$$\neg\exists \sigma' \in Adv_{M'}\,.\,\left(Pr^{\sigma'}_{M'}(\Box \neg err_A) \geqslant p_A \wedge Pr^{\sigma'}_{M'}(\Diamond err_G) > 1 - p_G\right)$$

which can be checked in time polynomial in $|M'|$ by solving an LP problem [10].

We now present, using the definitions above, several assume-guarantee proof rules to allow compositional verification.

An asymmetric proof rule. The first rule we consider is *asymmetric*, in the sense that we require only a single assumption about one component. Experience in the non-probabilistic setting [16] indicates that, despite its simplicity, rules of this form are widely applicable.

Theorem 1. *If M_1, M_2 are probabilistic automata and $\langle A\rangle_{\geqslant p_A}, \langle G\rangle_{\geqslant p_G}$ probabilistic safety properties such that $\alpha_A \subseteq \alpha_{M_1}$ and $\alpha_G \subseteq \alpha_{M_2} \cup \alpha_A$, then the following proof rule holds:*

$$\frac{\begin{array}{c}\langle true\rangle\, M_1\, \langle A\rangle_{\geqslant p_A} \\ \langle A\rangle_{\geqslant p_A}\, M_2\, \langle G\rangle_{\geqslant p_G}\end{array}}{\langle true\rangle\, M_1 \parallel M_2\, \langle G\rangle_{\geqslant p_G}} \qquad (\textsc{Asym})$$

Theorem 1 means that, given an appropriate assumption $\langle A\rangle_{\geqslant p_A}$, we can check the correctness of a probabilistic safety property $\langle G\rangle_{\geqslant p_G}$ on $M_1 \| M_2$, without constructing and model checking the full model. Instead, we perform one instance of (standard) model checking on M_1 (to check the first condition of rule (Asym)) and one instance of multi-objective model checking on $M_2[\alpha_A] \otimes A^{err}$ (to check the second). If A^{err} is much smaller than M_1, we can expect significant gains in terms of the verification performance.

Example 2. We illustrate the rule (Asym) on the PAs M_1, M_2 and property $\langle G\rangle_{\geqslant 0.98}$ from Example 1. Figure 2 (left) shows a DFA A^{err} representing the

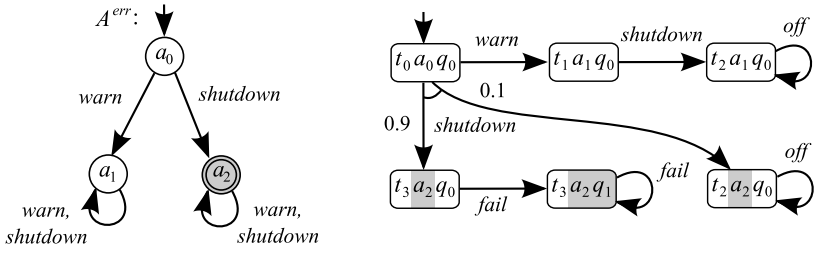

Fig. 2. DFA for safety property A and the product PA $M_2 \otimes A^{err} \otimes G^{err}$ (see Figure 1)

safety property A "*warn* occurs before *shutdown*". We will use the probabilistic safety property $\langle A \rangle_{\geqslant 0.8}$ as the assumption about M_1 in (ASYM).

Checking the first condition of (ASYM) amounts to verifying $M_1 \models \langle A \rangle_{\geqslant 0.8}$, which can be done with standard probabilistic model checking. To complete the verification, we need to check the second condition $\langle A \rangle_{\geqslant 0.8}\, M_2\, \langle G \rangle_{\geqslant 0.98}$, which, from Proposition 2, is achieved though multi-objective model checking on the product[2] $M_2 \otimes A^{err} \otimes G^{err}$. More precisely, we check there is no adversary under which the probability of remaining within states not satisfying err_A is at least 0.8 *and* the probability of reaching an err_G state is above $1-0.98 = 0.02$. The product is shown in Figure 2 (right), where we indicate states satisfying err_A and err_G by highlighting the accepting states a_2 and q_1 of DFAs A^{err} and G^{err}.

By inspection, we see that no such adversary exists, so we can conclude that $M_1 \| M_2 \models \langle G \rangle_{\geqslant 0.98}$. Consider, however, the adversary σ which, in the initial state, chooses *warn* with probability 0.8 and *shutdown* with probability 0.2. This satisfies $\Box \neg err_A$ with probability 0.8 and $\Diamond err_G$ with probability 0.02. Hence, $\langle A \rangle_{\geqslant 0.8}\, M_2\, \langle G \rangle_{\geqslant p_G}$ does *not* hold for any value of $p_G > 1-0.02 = 0.98$.

Proof of Theorem 1. We give below the proof of Theorem 1. This requires the following lemma, which is a simple extension of [20, Lemma 7.2.6, page 141].

Lemma 1. *Let M_1, M_2 be PAs, $\sigma \in Adv_{M_1 \| M_2}$, $\Sigma \subseteq \alpha_{M_1 \| M_2}$ and $i = 1, 2$. If A and B are regular safety properties such that $\alpha_A \subseteq \alpha_{M_i}$ and $\alpha_B \subseteq \alpha_{M_i[\Sigma]}$, then*

$$\text{(a) } Pr^{\sigma}_{M_1 \| M_2}(A) = Pr^{\sigma\restriction_{M_i}}_{M_i}(A) \quad \text{and} \quad \text{(b) } Pr^{\sigma}_{M_1 \| M_2}(B) = Pr^{\sigma\restriction_{M_i[\Sigma]}}_{M_i[\Sigma]}(B)\,.$$

Note that the projections onto $M_i[\Sigma]$ in the above are well defined since the condition $\Sigma \subseteq \alpha_{M_1 \| M_2}$ implies that $M_1 \| M_2 = M_1[\Sigma] \| M_2 = M_1 \| M_2[\Sigma]$.

Proof (of Theorem 1). The proof is by contradiction. Assume that there exist PAs M_1 and M_2 and probabilistic safety properties $\langle A \rangle_{\geqslant p_A}$ and $\langle G \rangle_{\geqslant p_G}$ such that $\langle true \rangle\, M_1\, \langle A \rangle_{\geqslant p_A}$ and $\langle A \rangle_{\geqslant p_A}\, M_2\, \langle G \rangle_{\geqslant p_G}$ hold, while $\langle true \rangle\, M_1 \| M_2\, \langle G \rangle_{\geqslant p_G}$ does not. From the latter, it follows that there exists an adversary $\sigma \in Adv_{M_1 \| M_2}$

[2] In this example, $\alpha_A = \{warn, shutdown\} \subseteq \alpha_{M_2}$ so $M_2[\alpha_A] = M_2$.

such that $Pr^{\sigma}_{M_1 \| M_2}(G) < p_G$. Now, since $\langle true \rangle\, M_1\, \langle A \rangle_{\geqslant p_A}$ and $\sigma{\restriction_{M_1}} \in Adv_{M_1}$, it follows that:

$$\begin{aligned}
Pr^{\sigma{\restriction_{M_1}}}_{M_1}(A) \geqslant p_A \Rightarrow\ & Pr^{\sigma}_{M_1 \| M_2}(A) \geqslant p_A && \text{by Lemma 1(a) since } \alpha_A \subseteq \alpha_{M_1} \\
\Rightarrow\ & Pr^{\sigma{\restriction_{M_2[\alpha_A]}}}_{M_2[\alpha_A]}(A) \geqslant p_A && \text{by Lemma 1(b) since } \alpha_A \subseteq \alpha_{M_2[\alpha_A]} \\
\Rightarrow\ & Pr^{\sigma{\restriction_{M_2[\alpha_A]}}}_{M_2[\alpha_A]}(G) \geqslant p_G && \text{since } \langle A \rangle_{\geqslant p_A}\, M_2\, \langle G \rangle_{\geqslant p_G} \\
\Rightarrow\ & Pr^{\sigma}_{M_1 \| M_2}(G) \geqslant p_G && \text{by Lemma 1(b) since } \alpha_G \subseteq \alpha_{M_2[\alpha_A]}
\end{aligned}$$

which contradicts the assumption that $Pr^{\sigma}_{M_1 \| M_2}(G) < p_G$. □

Generalising the proof rule. Next, we state two useful generalisations of the above proof rule. First, using $\langle A_1, \ldots, A_k \rangle_{\geqslant p_1, \ldots, p_k}$ to denote the conjunction of probabilistic safety properties $\langle A_i \rangle_{\geqslant p_i}$ for $i = 1, \ldots, k$, we have:

$$\frac{\langle true \rangle\, M_1\, \langle A_1, \ldots, A_k \rangle_{\geqslant p_1, \ldots, p_k} \quad \langle A_1, \ldots, A_k \rangle_{\geqslant p_1, \ldots, p_k}\, M_2\, \langle G \rangle_{\geqslant p_G}}{\langle true \rangle\, M_1 \parallel M_2\, \langle G \rangle_{\geqslant p_G}} \quad \text{(ASym-Mult)}$$

Definition 7 extends naturally to k assumptions, replacing α_A with $\cup_{i=1}^{k} \alpha_{A_i}$ and the single probabilistic safety property on the left-hand side of the implication with the conjunction. In similar fashion, by adapting Proposition 2, model checking of the query $\langle A_1, \ldots, A_k \rangle_{\geqslant p_1, \ldots, p_k}\, M\, \langle G \rangle_{\geqslant p_G}$ reduces to multi-objective model checking on the product $M[\cup_{i=1}^{k} \alpha_{A_i}] \otimes A_1^{err} \otimes \cdots \otimes A_k^{err} \otimes G^{err}$.

Secondly, we observe that, through repeated application of (ASym), we obtain a rule of the following form for n components:

$$\frac{\begin{array}{c} \langle true \rangle\, M_1\, \langle A_1 \rangle_{\geqslant p_1} \\ \langle A_1 \rangle_{\geqslant p_1}\, M_2\, \langle A_2 \rangle_{\geqslant p_2} \\ \cdots \\ \langle A_{n-1} \rangle_{\geqslant p_{n-1}}\, M_n\, \langle G \rangle_{\geqslant p_G} \end{array}}{\langle true \rangle\, M_1 \parallel \cdots \parallel M_n\, \langle G \rangle_{\geqslant p_G}} \quad \text{(ASym-N)}$$

A circular proof rule. One potential limitation of the rule (Asym) is that we may not be able to show that the assumption A_1 about M_1 holds without making additional assumptions about M_2. This can be overcome by using the following *circular* proof rule:

Theorem 2. *If M_1, M_2 are PAs and $\langle A_1 \rangle_{\geqslant p_1}$, $\langle A_2 \rangle_{\geqslant p_2}$ and $\langle G \rangle_{\geqslant p_G}$ probabilistic safety properties such that $\alpha_{A_2} \subseteq \alpha_{M_2}$, $\alpha_{A_1} \subseteq \alpha_{M_1} \cup \alpha_{A_2}$ and $\alpha_G \subseteq \alpha_{M_2} \cup \alpha_{A_1}$, then the following circular assume-guarantee proof rule holds:*

$$\frac{\begin{array}{c} \langle true \rangle\, M_2\, \langle A_2 \rangle_{\geqslant p_2} \\ \langle A_2 \rangle_{\geqslant p_2}\, M_1\, \langle A_1 \rangle_{\geqslant p_1} \\ \langle A_1 \rangle_{\geqslant p_1}\, M_2\, \langle G \rangle_{\geqslant p_G} \end{array}}{\langle true \rangle\, M_1 \parallel M_2\, \langle G \rangle_{\geqslant p_G}} \quad \text{(Circ)}$$

An asynchronous proof rule. This rule is motivated by the fact that, often, part of a system comprises several *asynchronous* components, that is, components with disjoint alphabets. In such cases, it can be difficult to establish useful probability bounds on the combined system if the fact that the components act *independently* is ignored. For example, consider the case of n independent coin flips; in isolation, we have that the probability of any coin not returning a tail is $1/2$. Now, ignoring the independence of the coins, all we can say is that the probability of any of them not returning a tail is at least $1/2$. However, using their independence, we have that this probability is at least $1-1/2^n$.

Theorem 3. *For any PAs M_1, M_2 and probabilistic safety properties $\langle A_1 \rangle_{\geqslant p_{A_2}}$, $\langle A_2 \rangle_{\geqslant p_{A_1}}$, $\langle G_1 \rangle_{\geqslant p_{G_1}}$ and $\langle G_2 \rangle_{\geqslant p_{G_2}}$ such that $\alpha_{M_1} \cap \alpha_{M_2} = \emptyset$, $\alpha_{G_1} \subseteq \alpha_{M_1} \cup \alpha_{A_1}$ and $\alpha_{G_2} \subseteq \alpha_{M_2} \cup \alpha_{A_2}$, we have the following asynchronous assume-guarantee proof rule:*

$$\frac{\begin{array}{c}\langle A_1 \rangle_{\geqslant p_{A_1}} \; M_1 \; \langle G_1 \rangle_{\geqslant p_{G_1}} \\ \langle A_2 \rangle_{\geqslant p_{A_2}} \; M_2 \; \langle G_2 \rangle_{\geqslant p_{G_2}}\end{array}}{\langle A_1, A_2 \rangle_{\geqslant p_{A_1} \cdot p_{A_2}} \; M_1 \| M_2 \; \langle G_1 \vee G_2 \rangle_{\geqslant p_{G_1} + p_{G_2} - p_{G_1} \cdot p_{G_2}}} \quad (\text{ASYNC})$$

where the disjunction of safety properties G_1 and G_2 is obtained by taking the intersection of the DFAs G_1^{err} and G_2^{err}.

4 Quantitative Assume-Guarantee Queries

Practical experience with probabilistic verification suggests that it is often more useful to adopt a *quantitative* approach. For example, rather than checking the correctness of a probabilistic safety property $\langle G \rangle_{\geqslant p_G}$, it may be preferable to just compute the actual worst-case (minimum) probability $Pr_M^{\min}(G)$ that G is satisfied. In this section we consider how to formulate such quantitative queries in the context of assume-guarantee reasoning. For simplicity, we restrict our attention here to the rule (ASYM) for fixed PAs M_1 and M_2, and property G. Similar reasoning applies to the other rules presented above.

Maximal lower bounds. Rule (ASYM) allows us to establish *lower bounds* for the probability $Pr_{M_1 \| M_2}^{\min}(G)$, i.e. it can be used to prove, for certain values of p_G, that $Pr_{M_1 \| M_2}^{\min}(G) \geqslant p_G$. We consider now how to obtain the highest such lower bound, say $p_G^\star$. First, we note that, from Definition 7, it is clear that the highest value of p_G for which $\langle A \rangle_{\geqslant p_A} \; M_2 \; \langle G \rangle_{\geqslant p_G}$ holds will be obtained by using the maximum possible value of p_A. For rule (ASYM) to be applicable, this is equal to $Pr_{M_1}^{\min}(A)$, since for any higher value of p_A the first condition will fail to hold. Now, by Proposition 2, and letting $M' = M_2[\alpha_A] \otimes A^{err} \otimes G^{err}$, the value $p_G^\star$ can be obtained through multi-objective model checking as follows:

$$p_G^\star = 1 - Pr_{M'}^{\max}(\Diamond err_G \,|\, \Psi) \text{ where } \Psi = Pr_{M'}^{\sigma}(\Box \neg err_A) \geqslant p_A.$$

Parameterised queries. Let us assume that component M_1 is parameterised by a variable x in such a way that varying x changes the probability of M_1 satisfying the assumption A. For example, increasing the value of x might increase the probability $Pr_{M_1}(A)$, but simultaneously worsen some other performance measure or cost associated with M_1. In this situation, it is desirable to establish a trade-off between the probability of $M_1 \| M_2$ satisfying G and the secondary 'cost' of M_1. Our use of multi-objective model checking for compositional verification offers two choices here. Firstly, we can pick a suitable threshold for $Pr_{M_1 \| M_2}(G)$ and then compute the lowest value of $Pr_{M_1}(A)$ which guarantees this, allowing an appropriate value of x to be chosen. Alternatively, we can consider the so-called *Pareto curve*: the set of achievable combinations of $Pr_{M_1 \| M_2}(G)$ and $Pr_{M_1}(A)$, which will present a clear view of the trade-off. For the latter, we can use the techniques of [10] for approximate exploration of the Pareto curve.

Upper bounds. Since application of (ASym) gives lower bounds on $Pr^{\min}_{M_1 \| M_2}(G)$, it is desirable to also generate *upper* bounds on this probability. This can be done as follows. When checking condition 2 of (ASym), using multi-objective model checking, we also obtain an adversary $\sigma \in Adv_{M_2[\alpha_A] \otimes A^{err}}$ that satisfies $\langle A \rangle_{\geqslant p_A}$ and gives the minimum (i.e. worst-case) probability of satisfying G. This can then be projected onto M_2, giving an adversary σ_2 which achieves the worst-case behaviour of the single component M_2 with respect to G satisfying $\langle A \rangle_{\geqslant p_A}$. Furthermore, from σ_2, we can easily construct a PA $M_2^{\sigma_2}$ that represents the behaviour of M_2 under σ_2.

Finally, we compute the probability of satisfying G on $M_1 \| M_2^{\sigma_2}$. Because $M_2^{\sigma_2}$ is likely to be much smaller than M_2, there is scope for this to be efficient, even if model checking $M_1 \| M_2$ in full is not feasible. Since $M_1 \| M_2^{\sigma_2}$ represents only a subset of the behaviour of $M_1 \| M_2$, the probability computed is guaranteed to give an upper bound on $Pr^{\min}_{M_1 \| M_2}(G)$. We use σ_2 (which achieves the worst-case behaviour with respect to G), rather than an arbitrary adversary of M_2, in order to obtain a tighter upper bound.

5 Implementation and Case Studies

We have implemented our compositional verification approach in a prototype tool. Recall that, using the rules given in Section 3, verification requires both standard (automata-based) model checking and multi-objective model checking. Our tool is based on the probabilistic model checker PRISM [11], which already supports LTL model checking of probabilistic automata. Model checking of probabilistic safety properties, represented by DFAs, can be achieved with existing versions of PRISM, since DFAs can easily be encoded in PRISM's modelling language. For multi-objective model checking, we have extended PRISM with an implementation of the techniques in [10]. This requires the solution of Linear Programming (LP) problems, for which we use the ECLiPSe Constraint Logic Programming system with the COIN-OR CBC solver, implementing a branch-and-cut algorithm. All experiments were run on a 2GHz PC with 2GB RAM. Any run exceeding a time-limit of 24 hours was disregarded.

We demonstrate the application of our tool to two large case studies. The first is the randomised consensus algorithm of Aspnes & Herlihy [2]. The algorithm allows N processes in a distributed network to reach a consensus and employs, in each round, a shared coin protocol parameterised by K. The PA model is based on [14] and consists of an automaton for each process and for the shared coin protocol of each round. We analyse the minimum probability that the processes decide by round R. The compositional verification employs $R-2$ uses of the ASYNC rule to return a probabilistic safety property satisfied by the (asynchronous) composition of the shared coin protocols for the first $R-2$ rounds. This is then used as the assumption of an ASYM rule for the subsystem representing the processes.

The second case study is the Zeroconf network configuration protocol [5]. We use the PA model from [13] consisting of two components, one representing a new host joining the network (parameterised by K, the number of probes it sends before using an IP address), and the second representing the environment, i.e. the existing network. We consider two properties: the minimum probability that a host employs a fresh IP address and that a host is configured by time T. In each case the compositional verification uses one application of the CIRC rule.

Table 1 shows experimental results for these case studies. We present the total time required for both compositional verification, as described in this paper, and

Table 1. Experimental results, comparing with non-compositional verification

Case study [parameters]		Non-compositional			Compositional		
		States	Time (s)	Result†	LP size	Time (s)	Result†
consensus (2 processes) [*R K*]	3 2	5,158	1.6	0.108333	1,064	**0.9**	0.108333
	3 20	40,294	108.1	0.012500	1,064	**7.4**	0.012500
	4 2	20,886	3.6	0.011736	2,372	**1.2**	0.011736
	4 20	166,614	343.1	0.000156	2,372	**7.8**	0.000156
	5 2	83,798	7.7	0.001271	4,988	**2.2**	0.001271
	5 20	671,894	1,347	0.000002	4,988	**8.8**	0.000002
consensus (3 processes) [*R K*]	3 2	1,418,545	18,971	0.229092	40,542	**29.6**	0.229092
	3 12	16,674,145*	time-out	-	40,542	**49.7**	0.041643
	3 20	39,827,233*	time-out	-	40,542	**125.3**	0.024960
	4 2	150,487,585	78,955	0.052483	141,168	**376.1**	0.052483
	4 12	1,053,762,385*	mem-out	-	141,168	**396.3**	0.001734
	4 20	2,028,200,209*	mem-out	-	141,168	**471.9**	0.000623
zeroconf [*K*]	2	91,041	39.0	2.0e-5	6,910	**9.3**	3.1e-4
	4	313,541	103.9	7.3e-7	20,927	**21.9**	3.1e-4
	6	811,290	275.2	2.6e-8	40,258	**54.8**	2.5e-4
	8	1,892,952	592.2	9.5e-10	66,436	**107.6**	9.0e-6
zeroconf (time bounded) [*K T*]	2 10	665,567	**46.3**	5.9e-5	62,188	89.0	2.1e-4
	2 14	106,177	**63.1**	2.0e-8	101,313	170.8	8.1e-8
	4 10	976,247	**88.2**	3.3e+0	74,484	170.8	3.3e+0
	4 14	2,288,771	**128.3**	7.0e-5	166,203	430.6	3.1e-4

* These models can be constructed, but not model checked, in PRISM.

† Results are maximum probabilities of error so actual values are these subtracted from 1.

non-compositional verification using PRISM (with the fastest available engine). Note that, in each case, we use the quantitative approach described in Section 4 and give actual (bounds on) probabilities computed. To give an indication of the size of the models considered, we give the number of states for the full (non-compositional) models and the number of variables in the LP problems used for multi-objective model checking in the compositional case.

In summary, we see that the compositional approach is faster in the majority of cases. Furthermore, it allows verification of several models for which it is infeasible with conventional techniques. For the cases where compositional verification is slower, this is due to the cost of solving a large LP problem, which is known to be more expensive than the highly optimised techniques used in PRISM. Furthermore, LP solution represents the limiting factor with respect to the scalability of the compositional approach. We expect that improvements to our technique can be made that will reduce LP problem sizes and improve performance. Finally, we note that the numerical values produced using compositional verification are generally good; in fact, for the consensus case study, the bounds obtained are precise.

6 Conclusions

We have presented a compositional verification technique, based on assume-guarantee rules, for probabilistic automata. Properties of these models are represented as probabilistic safety properties, and we show how verifying the resulting assume-guarantee queries reduces to the problem of multi-objective model checking. We also show how this can be leveraged to provide a *quantitative* approach to compositional verification. In contrast to existing work in this area, our techniques can be implemented efficiently and we demonstrate successful results on several large case studies.

There are several interesting directions for future work. In particular, we plan to experiment with the use of learning techniques to automatically produce the assumptions required for compositional reasoning. We also intend to further develop our compositional proof rules and investigate to what extent they are complete. Finally, we plan to expand the range of properties that can be verified, including for example reward-based specifications.

Acknowledgments. The authors are supported in part by EPSRC grants EP/D07956X and EP/D076625 and European Commission FP 7 project CONNECT (IST Project Number 231167). We also gratefully acknowledge several useful discussions with Kousha Etessami.

References

1. de Alfaro, L., Henzinger, T., Jhala, R.: Compositional methods for probabilistic systems. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 351–365. Springer, Heidelberg (2001)
2. Aspnes, J., Herlihy, M.: Fast randomized consensus using shared memory. Journal of Algorithms 15(1) (1990)

3. Bianco, A., de Alfaro, L.: Model checking of probabilistic and nondeterministic systems. In: Thiagarajan, P.S. (ed.) FSTTCS 1995. LNCS, vol. 1026. Springer, Heidelberg (1995)
4. Chatterjee, K., de Alfaro, L., Faella, M., Henzinger, T., Majumdar, R., Stoelinga, M.: Compositional quantitative reasoning. In: Proc. QEST 2006 (2006)
5. Cheshire, S., Adoba, B., Gutterman, E.: Dynamic configuration of IPv4 link local addresses, `http://www.ietf.org/rfc/rfc3927.txt`
6. Cheung, L., Lynch, N., Segala, R., Vaandrager, F.: Switched probabilistic I/O automata. In: Liu, Z., Araki, K. (eds.) ICTAC 2004. LNCS, vol. 3407, pp. 494–510. Springer, Heidelberg (2005)
7. Ciesinski, F., Baier, C.: Liquor: A tool for qualitative and quantitative linear time analysis of reactive systems. In: Proc. QEST 2006 (2006)
8. Courcoubetis, C., Yannakakis, M.: Markov decision processes and regular events. In: Paterson, M. (ed.) ICALP 1990. LNCS, vol. 443. Springer, Heidelberg (1990)
9. Delahaye, B., Caillaud, B.: A model for probabilistic reasoning on assume/guarantee contracts. Tech. Rep. 6719, INRIA (2008)
10. Etessami, K., Kwiatkowska, M., Vardi, M.Y., Yannakakis, M.: Multi-objective Model Checking of Markov Decision Processes. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 50–65. Springer, Heidelberg (2007)
11. Hinton, A., Kwiatkowska, M., Norman, G., Parker, D.: PRISM: A tool for automatic verification of probabilistic systems. In: Hermanns, H., Palsberg, J. (eds.) TACAS 2006. LNCS, vol. 3920, pp. 441–444. Springer, Heidelberg (2006)
12. Kwiatkowska, M., Norman, G., Parker, D., Qu, H.: Assume-guarantee verification for probabilistic systems. Tech. Rep. RR-09-17, Oxford University Computing Laboratory (December 2009)
13. Kwiatkowska, M., Norman, G., Parker, D., Sproston, J.: Performance analysis of probabilistic timed automata using digital clocks. Formal Methods in System Design 29 (2006)
14. Kwiatkowska, M., Norman, G., Segala, R.: Automated verification of a randomized distributed consensus protocol using Cadence SMV and PRISM. In: Berry, G., Comon, H., Finkel, A. (eds.) CAV 2001. LNCS, vol. 2102, pp. 194–206. Springer, Heidelberg (2001)
15. Lynch, N., Segala, R., Vaandrager, F.: Observing branching structure through probabilistic contexts. SIAM Journal on Computing 37(4), 977–1013 (2007)
16. Pasareanu, C., Giannakopoulou, D., Bobaru, M., Cobleigh, J., Barringer, H.: Learning to divide and conquer: Applying the L* algorithm to automate assume-guarantee reasoning. Formal Methods in System Design 32(3) (2008)
17. Pavese, E., Braberman, V., Uchitel, S.: Probabilistic environments in the quantitative analysis of (non-probabilistic) behaviour models. In: Proc. ESEC/FSE 2009 (2009)
18. Pogosyants, A., Segala, R., Lynch, N.: Verification of the randomized consensus algorithm of Aspnes and Herlihy: A case study. Dist. Comp. 13(4) (2000)
19. Segala, R.: A compositional trace-based semantics for probabilistic automata. In: Lee, I., Smolka, S.A. (eds.) CONCUR 1995. LNCS, vol. 962, pp. 234–248. Springer, Heidelberg (1995)
20. Segala, R.: Modelling and Verification of Randomized Distributed Real Time Systems. Ph.D. thesis, Massachusetts Institute of Technology (1995)
21. Segala, R., Lynch, N.: Probabilistic simulations for probabilistic processes. Nordic Journal of Computing 2(2) (1995)