Compositional Probabilistic Verification through Multi-Objective Model Checking

Marta Kwiatkowska[a], Gethin Norman[b], David Parker[c,1,*], Hongyang Qu[a,2]

[a] *Department of Computer Science, University of Oxford, Oxford, OX1 3QD, UK*
[b] *School of Computing Science, University of Glasgow, Glasgow, G12 8RZ, UK*
[c] *School of Computer Science, University of Birmingham, Birmingham, B15 2TT, UK*

Abstract

Compositional approaches to verification offer a powerful means to address the challenge of scalability. In this paper, we develop techniques for compositional verification of probabilistic systems based on the assume-guarantee paradigm. We target systems that exhibit both nondeterministic and stochastic behaviour, modelled as probabilistic automata, and augment these models with costs or rewards to reason about, for example, energy usage or performance metrics. Despite significant theoretical advances in compositional reasoning for probabilistic automata, there has been a distinct lack of practical progress regarding automated verification. We propose a new assume-guarantee framework based on multi-objective probabilistic model checking which supports compositional verification for a range of quantitative properties, including probabilistic ω-regular specifications and expected total cost or reward measures. We present a wide selection of assume-guarantee proof rules, including asymmetric, circular and asynchronous variants, and also show how to obtain numerical results in a compositional fashion. Given appropriate assumptions to be used in the proof rules, our compositional verification methods are, in contrast to previously proposed approaches, efficient and fully automated. Experimental results demonstrate their practical applicability on several large case studies, including instances where conventional probabilistic verification is infeasible.

Keywords: probabilistic verification, compositional verification, assume-guarantee reasoning, probabilistic automata

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

*Corresponding author
Email addresses: marta.kwiatkowska@cs.ox.ac.uk (Marta Kwiatkowska), gethin.norman@glasgow.ac.uk (Gethin Norman), d.a.parker@cs.bham.ac.uk (David Parker), h.qu@sheffield.ac.uk (Hongyang Qu)
[1] Contact details for David Parker: Telephone: +44 (0) 121 4143711
[2] Hongyang Qu is now with the Department of Automatic Control and Systems Engineering, University of Sheffield

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 [1, 2, 3] and mature tool support exists [4, 5], 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 models, 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 [6, 7]. This is based on checking queries of the form $\langle \Psi_A \rangle \, \mathcal{M} \, \langle \Psi_G \rangle$, with the meaning "whenever component $\mathcal{M}$ is part of a system satisfying the *assumption* Ψ_A, then the system is *guaranteed* to satisfy property Ψ_G". Proof rules can then be established to show, for example, that, if a component $\mathcal{M}_1$ satisfies assumption Ψ_A and $\langle \Psi_A \rangle \, \mathcal{M}_2 \, \langle \Psi_G \rangle$ holds for a second component $\mathcal{M}_2$, then the combined system $\mathcal{M}_1 \| \mathcal{M}_2$ satisfies Ψ_G.

For *probabilistic* systems, compositional approaches have also been studied, but a distinct lack of practical progress has been made. In this paper, we present novel assume-guarantee techniques for compositional verification of systems exhibiting both probabilistic and nondeterministic behaviour, and illustrate their applicability and efficiency on several large case studies. This is the first approach that, given appropriate assumptions about components, can perform compositional verification in an efficient and fully-automated manner.

We use *probabilistic automata* (PAs) [8, 9], a well-studied formalism that is naturally suited to modelling multi-component probabilistic systems. We also augment PAs with *rewards* (or, dually, *costs*), which can be used to model a variety of quantitative measures of system behaviour, such as execution time or power consumption. We present compositional techniques for verification of a range of *quantitative* properties, including probabilistic ω-regular properties (which subsume, for example, probabilistic LTL and probabilistic safety properties) and expected total reward/cost properties (which can also encode the expected reward/cost to reach a target and time-bounded reward measures).

Probabilistic automata were developed as a formalism for the modelling and analysis of distributed, randomised systems [8], and a rich underlying theory has been developed, in particular for models in which PAs are combined through parallel composition. A variety of elegant proof techniques have been created and used to manually prove the correctness of large, complex randomised algorithms [10]. Key ingredients of the underlying theory of PAs include probabilistic versions of strong and weak (bi)simulation [9] and trace distribution inclusion [8]. The branching-time preorders (simulation and bisimulation) have been shown to be compositional [9] (i.e., preserved under parallel composition), but are often too fine to give significant practical advantages for compositional verification. Trace distribution inclusion, which is defined in terms of probability distributions over sequences of observable actions, is a natural generalisation of the (non-probabilistic) notion of trace inclusion but is known *not* to be preserved under parallel composition [11]. Thus, other proposals for compositional verification frameworks based on PAs tend to restrict the forms of parallel composition that are allowed [12, 13]. By contrast, the approach we present in this paper does not impose restrictions on the parallel composition permitted between components, allowing greater flexibility to model complex systems.

Our assume-guarantee framework uses *multi-objective* probabilistic model checking [14, 15], which is a technique for verifying multiple, possibly conflicting properties of a probabilistic automaton. Conventional verification techniques for PAs quantify over its *adversaries*, which represent the various different ways in which nondeterminism in the model can be resolved. A

typical property to be verified states, for example, "the probability of a system failure is at most 0.01, *for any possible adversary*". Multi-objective model checking, on the other hand, permits reasoning about the existence of an adversary satisfying two or more distinct properties, for example, "is there an adversary under which the probability of a system failure is at most 0.005 and the expected battery lifetime remains below 2 hours?".

Our compositional approach to verification is based on queries of the form $\langle\Psi_A\rangle\, \mathcal{M}\, \langle\Psi_G\rangle$, with the meaning "under any adversary of PA $\mathcal{M}$ for which assumption Ψ_A is satisfied, Ψ_G is guaranteed to hold". The assumptions Ψ_A and guarantees Ψ_G are *quantitative multi-objective properties* [15], which are conjunctions of predicates, each of which imposes a bound on either the probability of an ω-regular property or the expected total value of some reward structure. A simple example of an assumption is "with probability 1, component $\mathcal{M}_1$ eventually sends a request, and the expected time before this occurs is at most 5 seconds". We show that checking these assume-guarantee queries can be reduced to existing multi-objective model checking techniques [14, 15], which can be implemented efficiently using linear programming.

Building upon this notion of probabilistic assume-guarantee reasoning, we formulate and prove several compositional proof rules, which can be used to decompose the process of verifying a multi-component probabilistic system into several smaller sub-tasks. One important class of such proof rules is those that restrict assumptions and guarantees to be *probabilistic safety properties*, which impose a bound on the probability of satisfying a regular safety property. These are slightly cheaper to verify than the other properties we consider, but still represent a useful set of system properties. In order to present proof rules for the more general class of quantitative properties (probabilistic ω-regular and expected total reward), we incorporate a notion of *fairness*, restricting our analysis to cases where each component in a system executes a step infinitely often.

For both of these classes of properties, we present several different assume-guarantee proof rules, including variants that are asymmetric (using assumptions only about one component) and circular (assumptions about all components). We also give proof rules for systems with components that are asynchronous and methods to decompose the analysis of reward-based properties. Finally, we describe how to obtain *numerical* results from compositional verification, in particular, obtaining lower and upper bounds on the actual probability that a system satisfies a property and constructing Pareto curves to investigate trade-offs between multiple system properties in a compositional fashion.

We have implemented our assume-guarantee verification techniques by extending the PRISM model checker [4], and present experimental results from its application to several large case studies. We demonstrate significant speed-ups over conventional, non-compositional verification, and also successfully verify models that are too large to be analysed without compositional techniques.

1.1. Related Work

As mentioned above, there is a significant body of work which develops the underlying theory, built upon in this paper, for the compositional modelling and analysis of probabilistic systems. Segala and Lynch [8, 9] proposed the model of probabilistic automata, and defined many key accompanying notions such as parallel composition, projections and various compositional preorders, including strong and weak variants of probabilistic simulation and bisimulation. A number of compositional proof techniques were also developed and, in [10], applied to the manual verification of Aspnes and Herlihy's randomised consensus algorithm.

There are several proposals for compositional verification frameworks based on PAs. For example, de Alfaro et al. define a probabilistic extension of Reactive Modules [16], which is restricted to *synchronous* parallel composition [12]. This allows the formulation of assume-guarantee rules based on the use of trace distribution inclusion [8] which, in general, is not compositional [11]. However, there is no known algorithm to check trace distribution inclusion and an implementation of the techniques in [12] is not considered. An alternative approach is the formalism of switched probabilistic I/O automata by Cheung et al. [13]. This distinguishes between *local* resolutions of nondeterminism by individual components, and the *global* resolution of nondeterminism by a scheduler. The result is a framework which does allow compositional, traced-based relations to be established between models. Again, though, practical implementations of the techniques are not developed.

This paper is an extended version of [17], which presented the first fully-automated compositional verification techniques for probabilistic automata. That work focused specifically on the use of probabilistic safety properties; here, we give methods to verify a much wider class of properties, some of which were originally introduced in [15]. This paper also draws inspiration from several other sources. In particular, the fragment of our framework that deals with safety properties was inspired by the large body of work by Giannakopoulou, Pasareanu et al. (see, e.g., [7]) on *non-probabilistic* assume-guarantee techniques. We also build upon ideas put forward in [18], which suggests using multi-objective verification for compositional probabilistic verification, but does not give a concrete proposal of how to achieve this.

The assume-guarantee techniques of [17] have also been extended with algorithmic learning methods [19, 20], which are used to automatically generate assumptions for performing compositional verification. In recent work, an alternative way to generate assumptions has been proposed in [21], based on the use of an abstraction-refinement loop. In that work, components, assumptions and properties are all modelled as probabilistic automata, with the relationships between them captured by strong simulation [9]. Our work, by comparison, permits verification for a more expressive class of quantitative properties, including for example reward-based measures, and supports a wider range of compositional proof rules.

We also mention the development of compositional verification techniques for other classes of models. One example is the fully probabilistic setting, i.e., where models cannot exhibit nondeterministic behaviour. In [22], compositional techniques are presented for probabilistic model checking of hardware systems modelled as discrete-time Markov chains. This is based on a decomposition of the property to be checked and the use of conditional probabilities. In [23], a notion of contracts is proposed for probabilistic systems with a limited degree of nondeterminism, i.e., a model that is less expressive than PAs. The authors consider additional operations over contracts, such as refinement, but do not consider a practical implementation of their approach. Moving away from probabilistic systems, [24] presents a theoretical framework for compositional verification of quantitative properties of labelled transition systems and [25] presents tool support for compositional analysis of timed systems.

1.2. Paper Structure

The remainder of the paper is structured as follows. Section 2 introduces the necessary background material regarding probabilistic automata, including notions required for compositional methods such as parallel composition and projection. Section 3 summarises the classes of properties of PAs that we use in this paper and techniques for their verification. It also covers the topics of multi-objective model checking and verification of PAs under fairness constraints. Section 4

introduces our assume-guarantee framework, defining the basic underlying ideas and presenting instances of the two main classes of compositional proof rules that we consider. Section 5 gives several additional proof rules and then Section 6 discusses how our techniques can be adapted to produce numerical results. In Section 7, we describe an implementation of our techniques and show results from its application to several large case studies. Section 8 concludes the paper.

2. Probabilistic Automata

We begin with some background material on probabilistic automata. In the following, we use $Dist(S)$ to denote the set of all discrete probability distributions over a set S, i.e. functions $\mu : S \to [0,1]$ satisfying $\sum_{s \in S} \mu(s) = 1$. We use 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 $\mu_2 \in Dist(S_2)$, defined by $\mu_1 \times \mu_2((s_1, s_2)) \stackrel{\text{def}}{=} \mu_1(s_1) \cdot \mu_2(s_2)$. We also denote by $SubDist(S)$ the set of sub-distributions over S, i.e. functions $\mu : S \to [0,1]$ satisfying $\sum_{s \in S} \mu(s) \leq 1$.

2.1. Probabilistic Automata (PAs)

Probabilistic automata [8, 9] are commonly used for modelling systems that exhibit both probabilistic and nondeterministic behaviour. They are a slight generalisation of Markov decision processes.[3] For the purposes of applying standard probabilistic verification techniques, the two models can often be treated identically; however, probabilistic automata are particularly well suited to compositional modelling and analysis of probabilistic systems.

Definition 1 (PA). *A* probabilistic automaton *(PA) is a tuple* $\mathcal{M} = (S, \overline{s}, \alpha_{\mathcal{M}}, \delta_{\mathcal{M}}, L)$ *where:*

- S *is a set of states and* $\overline{s} \in S$ *is an initial state;*
- $\alpha_{\mathcal{M}}$ *is an* alphabet *of action labels;*
- $\delta_{\mathcal{M}} \subseteq S \times \alpha_{\mathcal{M}} \times Dist(S)$ *is a probabilistic transition relation;*
- $L : S \to 2^{AP}$ *is a labelling function mapping states to sets of atomic propositions taken from a set* AP.

In any state s of a PA $\mathcal{M}$, a *transition*, denoted $s \xrightarrow{a} \mu$, where a is an *action* label and μ is a discrete probability distribution over states, is available if $(s, a, \mu) \in \delta_{\mathcal{M}}$. In an execution of the model, the choice between the available transitions in each state is nondeterministic; the choice of successor state is then made randomly according to the distribution μ. For presentational convenience elsewhere in the paper, we do not identify a special "silent" action τ, but this can easily be added if required.

A *path* through $\mathcal{M}$ is a (finite or infinite) sequence $\pi = s_0 \xrightarrow{a_0,\mu_0} s_1 \xrightarrow{a_1,\mu_1} \cdots$ where $s_0 = \overline{s}$ and, for each $i \geq 0$, $s_i \xrightarrow{a_i} \mu_i$ is a transition and $\mu_i(s_{i+1}) > 0$. We denote by $\pi(i)$ the $(i+1)$th state s_i of π and by $IPaths_{\mathcal{M}}$ ($FPaths_{\mathcal{M}}$) the set of all infinite (finite) paths in $\mathcal{M}$. If π is finite, $|\pi|$ denotes its length and $last(\pi)$ its last state. The *trace*, $tr(\pi)$, of π is the sequence of actions $a_0 a_1 \ldots$ and we use $tr(\pi)\restriction_\alpha$ to indicate the projection of such a trace onto an alphabet $\alpha \subseteq \alpha_{\mathcal{M}}$.

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.

[3] For Markov decision processes, the probabilistic transition relation of Definition 1 becomes a partial function $\delta_{\mathcal{M}} : (S \times \alpha_{\mathcal{M}}) \to Dist(S)$.

Definition 2 (Adversary). *An* adversary *of a PA* $\mathcal{M} = (S, \overline{s}, \alpha_{\mathcal{M}}, \delta_{\mathcal{M}}, L)$ *is given by a function* $\sigma : FPaths_{\mathcal{M}} \to Dist(\alpha_{\mathcal{M}} \times Dist(S))$ *such that, for any finite path* π *of* $\mathcal{M}$*,* $\sigma(\pi)$ *only assigns non-zero probabilities to action-distribution pairs* (a, μ) *for which* $(last(\pi), a, \mu) \in \delta_{\mathcal{M}}$*.*

Employing standard techniques [26], an adversary σ of a PA $\mathcal{M}$ induces a probability measure $Pr^{\sigma}_{\mathcal{M}}$ over $IPaths_{\mathcal{M}}$, which captures the (purely probabilistic) behaviour of $\mathcal{M}$ when under the control of σ. We also use $IPaths^{\sigma}_{\mathcal{M}}$ ($FPaths^{\sigma}_{\mathcal{M}}$) for the set of infinite (finite) paths of $\mathcal{M}$ under σ.

The set of all adversaries for PA $\mathcal{M}$ is denoted by $Adv_{\mathcal{M}}$. Amongst these, we distinguish several important classes. An adversary σ is *deterministic* if $\sigma(\pi)$ is a point distribution for all π, and *randomised* otherwise; σ is *memoryless* if $\sigma(\pi)$ depends only on $last(\pi)$, and *finite-memory* if there are a finite number of memory configurations such that $\sigma(\pi)$ depends only on $last(\pi)$ and the current memory configuration, which is updated (possibly stochastically) when an action is performed. We will also sometimes need to distinguish between *partial* and *complete* adversaries, which are discussed in the next section.

We augment PAs with *rewards*, which will be used to capture a variety of quantitative properties of the systems that we model. In this paper, we attach rewards to the transitions of a PA, according to the actions that label them.

Definition 3 (Reward structure). *A* reward structure *for a PA* $\mathcal{M}$ *is a mapping* $\rho : \alpha_\rho \to \mathbb{R}_{>0}$ *from some alphabet* $\alpha_\rho \subseteq \alpha_{\mathcal{M}}$ *to the positive reals.*

For an infinite path $\pi = s_0 \xrightarrow{a_0,\mu_0} s_1 \xrightarrow{a_1,\mu_1} \cdots$, the *total reward* for π over reward structure ρ is given by $\rho(\pi) \stackrel{\text{def}}{=} \sum_{i \in \mathbb{N} \wedge a_i \in \alpha_\rho} \rho(a_i)$.

2.2. Parallel Composition of PAs

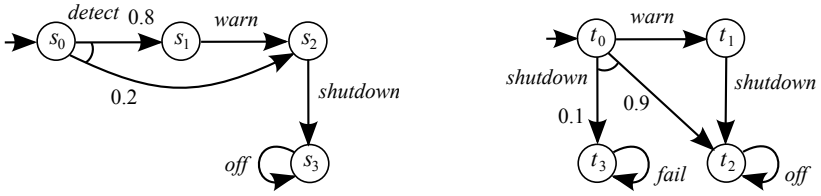
To model and analyse probabilistic systems comprising multiple components, we need *parallel composition* of PAs. We use the definition of [8, 9], which is based on multi-way synchronisation over transitions with identical action labels, in the style of the process algebra CSP.

Definition 4 (Parallel composition). *Let* $\mathcal{M}_1, \mathcal{M}_2$ *be PAs such that* $\mathcal{M}_i = (S_i, \overline{s}_i, \alpha_{\mathcal{M}_i}, \delta_{\mathcal{M}_i}, L_i)$ *for* i=1, 2*. Their* parallel composition *is the PA* $\mathcal{M}_1 \| \mathcal{M}_2 = (S_1 \times S_2, (\overline{s}_1, \overline{s}_2), \alpha_{\mathcal{M}_1} \cup \alpha_{\mathcal{M}_2}, \delta_{\mathcal{M}_1 \| \mathcal{M}_2}, L)$ *where* $\delta_{\mathcal{M}_1 \| \mathcal{M}_2}$ *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* $a \in \alpha_{\mathcal{M}_1} \cap \alpha_{\mathcal{M}_2}$*;*
- $s_1 \xrightarrow{a} \mu_1$, $\mu_2 = \eta_{s_2}$ *and* $a \in (\alpha_{\mathcal{M}_1} \backslash \alpha_{\mathcal{M}_2})$*;*
- $\mu_1 = \eta_{s_1}$, $s_2 \xrightarrow{a} \mu_2$ *and* $a \in (\alpha_{\mathcal{M}_2} \backslash \alpha_{\mathcal{M}_1})$*;*

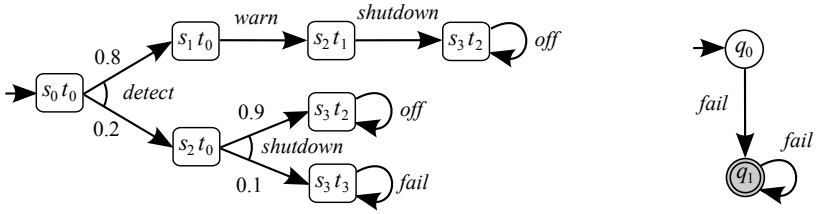
and $L(s_1, s_2) = L_1(s_1) \cup L_2(s_2)$*. The atomic propositions used by the labelling functions* (L_1, L_2 *and* L*) are assumed to be from some global set AP. In practice, the atomic propositions labelling the states of each individual PA are usually disjoint.*

Example 1. Figure 1 shows a pair of PAs, $\mathcal{M}_s$ and $\mathcal{M}_d$, which we will use as one of our running examples throughout the paper. We draw each transition of a PA as a group of arrows, joined by an arc and annotated with its action. We label individual arrows with their corresponding



(a) PA $\mathcal{M}_s$ (b) PA $\mathcal{M}_d$

Figure 1: Two probabilistic automata $\mathcal{M}_s$, $\mathcal{M}_d$, representing a sensor and a device (see Example 1)



(a) PA $\mathcal{M}_s \| \mathcal{M}_d$ (b) DFA $\mathcal{A}^{err}_{\phi_G}$

Figure 2: Parallel composition of PAs $\mathcal{M}_s$, $\mathcal{M}_d$ from Figure 1 and a DFA $\mathcal{A}^{err}_{\phi_G}$ for a safety property ϕ_G

probabilities, but omit this information if the probability is 1. A short incoming arrow denotes the initial state of the PA.

We model a system with two components, each corresponding to one of the PAs. Component $\mathcal{M}_s$ represents a *sensor* which, upon detection of a system failure, issues instructions to other devices causing them to power down. 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. $\mathcal{M}_d$ represents a *device* which, given the *shutdown* signal, powers down correctly if it first receives the *warn* signal and otherwise only powers down correctly 90% of the time. The combined system, i.e. the parallel composition $\mathcal{M}_s \| \mathcal{M}_d$ of the two PAs, is shown in Figure 2(a).

For compositional reasoning about PAs, we also require the notion of *projections* [8], used to decompose models that have been constructed through parallel composition. First, for any state $s{=}(s_1, s_2)$ of $\mathcal{M}_1 \| \mathcal{M}_2$, the projection of s onto $\mathcal{M}_i$, denoted by $s{\restriction_{\mathcal{M}_i}}$, equals s_i. We extend this notation to distributions over the state space $S_1 \times S_2$ of $\mathcal{M}_1 \| \mathcal{M}_2$ in the standard manner. Next, for any (finite or infinite) path π of $\mathcal{M}_1 \| \mathcal{M}_2$, the projection of π onto $\mathcal{M}_i$, denoted $\pi{\restriction_{\mathcal{M}_i}}$, is the path obtained from π by projecting each state of π onto $\mathcal{M}_i$ and removing all the actions not in $\alpha_{\mathcal{M}_i}$ together with the subsequent states. Notice that the projection of an infinite path may be finite.

To define projections of adversaries, we first need the notion of *partial* adversaries, which are functions that map finite paths to sub-distributions over available transitions in the final state of the path, rather than distributions. The interpretation is that such an adversary can opt to (with some probability) take none of the available transitions and remain in the current state. This generalises the normal definition of an adversary, which we will sometimes refer to as a *complete* adversary.

Definition 5 (Partial adversary). *A* partial adversary *of a PA* $\mathcal{M} = (S, \overline{s}, \alpha_{\mathcal{M}}, \delta_{\mathcal{M}}, L)$ *is a function* $\sigma : FPaths_{\mathcal{M}} \to SubDist(\alpha_{\mathcal{M}} \times Dist(S))$ *such that, for any finite path* π *of* $\mathcal{M}$*,* $\sigma(\pi)$ *only assigns non-zero probabilities to action-distribution pairs* (a, μ) *for which* $(last(\pi), a, \mu) \in \delta_{\mathcal{M}}$.

The probability space $Pr^{\sigma}_{\mathcal{M}}$ for a partial adversary σ is defined in a similar manner as for a complete adversary. We use $Adv^{\text{part}}_{\mathcal{M}}$ to denote the set of all partial adversaries of $\mathcal{M}$ and sometimes, for consistency, use $Adv^{\text{comp}}_{\mathcal{M}}$ to refer to the set all complete adversaries (i.e. $Adv_{\mathcal{M}}$).

Compositional reasoning about PAs may require partial adversaries since, even if an adversary of $\mathcal{M}_1 \| \mathcal{M}_2$ is complete, its projection onto one component may be partial. Later, in Section 3.5, we will show that, by restricting our attention to *fair* adversaries, we can ensure that the projection of a complete adversary remains complete.

Definition 6 (Adversary projection). *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be PAs and σ an adversary of $\mathcal{M}_1 \| \mathcal{M}_2$. The* projection *of σ onto $\mathcal{M}_i$, denoted $\sigma{\restriction_{\mathcal{M}_i}}$, is the (partial) adversary on $\mathcal{M}_i$ where, for any finite path π_i of $\mathcal{M}_i$ and $(last(\pi_i), a, \mu_i) \in \delta_{\mathcal{M}_i}$:*

$$\sigma{\restriction_{\mathcal{M}_i}}(\pi_i)(a,\mu_i) = \frac{\sum \left\{ Pr^{\sigma}_{\mathcal{M}_1\|\mathcal{M}_2}(\pi) \cdot \sigma(\pi)(a,\mu) \;\middle|\; \pi \in FPaths^{\sigma}_{\mathcal{M}_1\|\mathcal{M}_2} \wedge \pi{\restriction_{\mathcal{M}_i}} = \pi_i \wedge \mu{\restriction_{\mathcal{M}_i}} = \mu_i \right\}}{Pr^{\sigma{\restriction_{\mathcal{M}_i}}}_{\mathcal{M}_i}(\pi_i)}$$

and, for a finite path π of $\mathcal{M}$, $Pr^{\sigma}_{\mathcal{M}}(\pi)$ denotes $Pr^{\sigma}_{\mathcal{M}}(\{\pi' \in IPaths^{\sigma}_{\mathcal{M}} \mid \pi \text{ is a prefix of } \pi'\})$.

For any partial adversary σ of a PA $\mathcal{M}$, a complete adversary σ' of $\mathcal{M}$ is called a *completion* of σ if $\sigma(\pi)(a,\mu) \geq \sigma'(\pi)(a,\mu)$ for all paths π and action-distribution pairs (a,μ) of $\mathcal{M}$. Any partial adversary always has at least one completion.

Finally, in this section, we define the notion of *alphabet extension* for a PA $\mathcal{M}$. This operation ensures that all of the actions from a set α are included in the alphabet of $\mathcal{M}$ and, for any new action a, adds an a-labelled self-loop to each state of $\mathcal{M}$.

Definition 7 (Alphabet extension). *For any PA $\mathcal{M} = (S, \overline{s}, \alpha_{\mathcal{M}}, \delta_{\mathcal{M}}, L)$ and set of actions α, we* extend the alphabet *of $\mathcal{M}$ to α, denoted $\mathcal{M}[\alpha]$, as follows: $\mathcal{M}[\alpha] = (S, \overline{s}, \alpha_{\mathcal{M}} \cup \alpha, \delta_{\mathcal{M}[\alpha]}, L)$ where $\delta_{\mathcal{M}[\alpha]} = \delta_{\mathcal{M}} \cup \{(s, a, \eta_s) \mid s \in S \wedge a \in \alpha \backslash \alpha_{\mathcal{M}}\}$.*

3. Quantitative Verification of Probabilistic Automata

In this section, we describe how to specify and verify a variety of quantitative properties of probabilistic automata. We also discuss multi-objective probabilistic model checking and verification under both partial and fair adversaries.

3.1. Specifying Properties of PAs

There are various different ways of formally specifying properties of PAs for the purposes of verification. In this paper, we focus primarily on linear-time, action-based properties (i.e those defined in terms of the action labels attached to PA transitions). More precisely, we will use probabilistic ω-regular properties (which subsume, for example, probabilistic LTL) and expected total reward properties. For the former, we will make particular use of the subclass of probabilistic safety properties. The latter, as mentioned earlier, also allows a variety of other reward-based properties to be encoded, including the expected reward to reach a target and time-bounded reward measures; see below for details.

In this section, we introduce the required properties; in the following section, we outline the corresponding techniques to perform model checking. Throughout, we will assume a fixed PA $\mathcal{M} = (S, \overline{s}, \alpha_{\mathcal{M}}, \delta_{\mathcal{M}}, L)$. We begin by defining *probabilistic predicates* and *reward predicates*.

Definition 8 (Probabilistic predicate). *A* probabilistic predicate $[\phi]_{\sim p}$ *comprises an ω-regular property*[4] $\phi \subseteq (\alpha_\phi)^\omega$ *over some alphabet $\alpha_\phi \subseteq \alpha_\mathcal{M}$, a relational operator $\sim \in \{<, \leq, >, \geq\}$ and a rational probability bound p. The* probability of satisfying ϕ *under adversary σ is:*

$$Pr^\sigma_\mathcal{M}(\phi) \;=\; Pr^\sigma_\mathcal{M}(\{\, \pi \in IPaths_\mathcal{M} \mid tr(\pi)\restriction_{\alpha_\phi} \in \phi \,\})$$

and satisfaction of $[\phi]_{\sim p}$ by $\mathcal{M}$ under adversary σ, denoted $\mathcal{M}, \sigma \models [\phi]_{\sim p}$, is defined as follows:

$$\mathcal{M}, \sigma \models [\phi]_{\sim p} \;\Leftrightarrow\; Pr^\sigma_\mathcal{M}(\phi) \sim p\,.$$

We say that $\mathcal{M}$ satisfies $[\phi]_{\sim p}$, denoted $\mathcal{M} \models [\phi]_{\sim p}$, if it does so under all adversaries:

$$\mathcal{M} \models [\phi]_{\sim p} \;\Leftrightarrow\; \forall \sigma \in Adv_\mathcal{M}\,.\, \mathcal{M}, \sigma \models [\phi]_{\sim p}\,.$$

Definition 9 (Reward predicate). *A* reward predicate $[\rho]_{\sim r}$ *comprises a reward structure $\rho : \alpha_\rho \to \mathbb{R}_{>0}$ over some alphabet $\alpha_\rho \subseteq \alpha_\mathcal{M}$, a relational operator $\sim \in \{<, \leq, >, \geq\}$ and a rational reward bound r. The* expected total reward *for ρ under adversary σ is:*

$$ExpTot^\sigma_\mathcal{M}(\rho) \;=\; \int_{\pi \in IPaths_\mathcal{M}} \rho(\pi)\, dPr^\sigma_\mathcal{M}(\pi)\,,$$

satisfaction of $[\rho]_{\sim r}$ by $\mathcal{M}$ under adversary σ, denoted $\mathcal{M}, \sigma \models [\rho]_{\sim r}$, is defined as:

$$\mathcal{M}, \sigma \models [\rho]_{\sim r} \;\Leftrightarrow\; ExpTot^\sigma_\mathcal{M}(\rho) \sim r$$

and satisfaction of $[\rho]_{\sim r}$ by $\mathcal{M}$ is defined as:

$$\mathcal{M} \models [\rho]_{\sim r} \;\Leftrightarrow\; \forall \sigma \in Adv_\mathcal{M}\,.\, \mathcal{M}, \sigma \models [\rho]_{\sim r}\,.$$

It is worth noting that expected total reward properties (i.e. reward predicates) can also be used to specify the expected total reward accumulated until some set T of target states is reached, which is another commonly used class of properties. This is done by modifying the PA $\mathcal{M}$ to ensure that no rewards are accumulated after a state in T is reached for the first time. In the worst case, this requires adding a second copy of each state in the PA. Properties of this kind can, in turn, be used to encode other useful classes of properties, such as the expected amount of reward accumulated over a fixed time period (see [28] for details).

Probabilistic predicates and reward predicates will be collectively referred to as *quantitative predicates*. We typically use $[\phi]_{\sim p}$ and $[\rho]_{\sim r}$ to denote probabilistic and reward predicates, respectively, and $[\psi]_{\sim x}$ for an arbitrary quantitative predicate.

In this work, we will often use a particular class of probabilistic predicates called *probabilistic safety properties*, defined in terms of *regular safety properties*.

Definition 10 (Regular safety property). *A* regular safety property $\phi \subseteq (\alpha_\phi)^\omega$ *for PA $\mathcal{M}$ is a set of infinite words over alphabet $\alpha_\phi \subseteq \alpha_\mathcal{M}$ that is characterised by a regular language $\overline{\phi}$ of* bad prefixes*. These are finite words, any extension of which is not in ϕ, i.e.:*

$$\phi = \{\, w \in (\alpha_\phi)^\omega \mid \textit{no prefix of } w \textit{ is in } \overline{\phi} \,\}\,.$$

We represent ϕ by its error automaton *$\mathcal{A}^{err}_\phi$, a (complete) deterministic finite automaton (DFA) $(Q, \overline{q}, \alpha_\phi, \delta_\phi, F)$, comprising states Q, initial state $\overline{q} \in Q$, alphabet α_ϕ, transition function $\delta_\phi : Q \times \alpha_\phi \to Q$ and accepting states $F \subseteq Q$, whose corresponding language equals $\overline{\phi}$.*

[4] We use the term ω-regular property as a synonym for ω-regular language [27].

Definition 11 (Probabilistic safety property). *A* probabilistic safety property $[\phi]_{\geq p}$ *is a probabilistic predicate comprising a regular safety property* ϕ *and a rational (lower) probability bound* p*. The probability of* ϕ *and satisfaction of* $[\phi]_{\geq p}$ *by* $\mathcal{M}$*, under adversary* σ*, are defined as for probabilistic predicates above (see Definition 8).*

Regular safety properties, by their definition, are a strict subclass of ω-regular properties and thus probabilistic safety properties are a special case of the probabilistic predicates introduced in Definition 8. Probabilistic 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 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 an important class of properties for *timed* probabilistic systems, perhaps not typically considered as safety properties. Using the *digital clocks* approach of [29], 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 probabilistic safety properties.

Example 2. Consider the two PAs $\mathcal{M}_s$ and $\mathcal{M}_d$ from Example 1 (see Figure 1) and their parallel composition $\mathcal{M}_s \parallel \mathcal{M}_d$, shown in Figure 2(a). We define a simple regular safety property ϕ_G for $\mathcal{M}_s \parallel \mathcal{M}_d$ with the meaning "action *fail* never occurs". This is represented by the DFA $\mathcal{A}^{err}_{\phi_G}$, over alphabet $\{fail\}$, shown in Figure 2(b). The accepting state of the DFA is shaded grey. In this simple example, $\mathcal{M}_s \parallel \mathcal{M}_d$ has just a single adversary, under which the probability of satisfying ϕ_G is $1 - 0.2 \cdot 0.1 = 0.98$. Thus, we have that $\mathcal{M}_s \parallel \mathcal{M}_d \models [\phi_G]_{\geq 0.98}$.

3.2. Model Checking for PAs

As illustrated in the previous section, we typically verify properties of PA $\mathcal{M}$ by quantifying over all possible adversaries of $\mathcal{M}$. For example, for a quantitative predicate $[\psi]_{\sim x}$, we have:

$$\mathcal{M} \models [\psi]_{\sim x} \;\Leftrightarrow\; \forall \sigma \in Adv_{\mathcal{M}} \,.\, \mathcal{M}, \sigma \models [\psi]_{\sim x} \,.$$

A model checking procedure for verifying whether $\mathcal{M}$ satisfies a probabilistic predicate $[\phi]_{\sim p}$ reduces to the computation of the minimum or maximum probability of satisfying ϕ:

$$Pr^{\min}_{\mathcal{M}}(\phi) \stackrel{\text{def}}{=} \inf_{\sigma \in Adv_{\mathcal{M}}} Pr^{\sigma}_{\mathcal{M}}(\phi) \quad \text{and} \quad Pr^{\max}_{\mathcal{M}}(\phi) \stackrel{\text{def}}{=} \sup_{\sigma \in Adv_{\mathcal{M}}} Pr^{\sigma}_{\mathcal{M}}(\phi) \,.$$

In particular, if $\sim \in \{\geq, >\}$, then $\mathcal{M} \models [\phi]_{\sim p}$ if and only if $Pr^{\min}_{\mathcal{M}}(\phi) \sim p$, while if $\sim \in \{<, \leq\}$, then $\mathcal{M} \models [\phi]_{\sim p}$ if and only if $Pr^{\max}_{\mathcal{M}}(\phi) \sim p$. Similarly, for a reward predicate $[\rho]_{\sim r}$, minimum or maximum expected total rewards are required:

$$ExpTot^{\min}_{\mathcal{M}}(\rho) \stackrel{\text{def}}{=} \inf_{\sigma \in Adv_{\mathcal{M}}} ExpTot^{\sigma}_{\mathcal{M}}(\rho) \quad \text{and} \quad ExpTot^{\max}_{\mathcal{M}}(\rho) \stackrel{\text{def}}{=} \sup_{\sigma \in Adv_{\mathcal{M}}} ExpTot^{\sigma}_{\mathcal{M}}(\rho) \,.$$

In the remainder of this section, we discuss how these values can be computed. For the probabilistic case, the problem reduces to the calculation of *reachability probabilities*. For an atomic

proposition q from the set AP used to label states of PA $\mathcal{M}$, we define the probability of reaching a q-labelled state under adversary σ, denoted $Pr^{\sigma}_{\mathcal{M}}(\Diamond\, q)$, as:

$$Pr^{\sigma}_{\mathcal{M}}(\Diamond\, q) \stackrel{\text{def}}{=} Pr^{\sigma}_{\mathcal{M}}(\{\, \pi \in IPaths_{\mathcal{M}} \mid q \in L(\pi(i)) \text{ for some } i \geq 0 \,\})$$

and, as for other probabilistic properties, we define:

$$Pr^{\min}_{\mathcal{M}}(\Diamond\, q) \stackrel{\text{def}}{=} \inf_{\sigma \in Adv_{\mathcal{M}}} Pr^{\sigma}_{\mathcal{M}}(\Diamond\, q) \quad \text{and} \quad Pr^{\max}_{\mathcal{M}}(\Diamond\, q) \stackrel{\text{def}}{=} \sup_{\sigma \in Adv_{\mathcal{M}}} Pr^{\sigma}_{\mathcal{M}}(\Diamond\, q)\,.$$

Computation of the values $Pr^{\min}_{\mathcal{M}}(\Diamond\, q)$ or $Pr^{\max}_{\mathcal{M}}(\Diamond\, q)$ for $\mathcal{M}$ can be achieved by solving a linear programming problem of size $|\mathcal{M}|$ [30, 1], and thus performed in time polynomial in $|\mathcal{M}|$. In practice, other techniques, such as value iteration or policy iteration, are often used [31].

The probabilities $Pr^{\min}_{\mathcal{M}}(\phi)$ or $Pr^{\max}_{\mathcal{M}}(\phi)$ for an ω-regular property ϕ can be computed by reducing the problem to calculating reachability probabilities in a product PA composed from the PA $\mathcal{M}$ and an ω-automaton for the property ϕ. In the implementation developed for this work, we follow the approach of [3], which is based on the use of deterministic Rabin automata and then determining the maximum probability of reaching a set of accepting end components. The presentation in [3] uses ω-regular properties over atomic propositions (on states) rather than the action labels (on transitions) used here, but the procedure is essentially the same. Full details for the action-based case can be found in [28]. The total time complexity for computing probabilities is polynomial in the sizes of both the PA $\mathcal{M}$ and the deterministic ω-automaton for ϕ.

We now describe in more detail the computation of probabilities for regular safety properties, since these are used in various places throughout the paper. Although a regular safety property ϕ is an instance of an ω-regular property, it is more efficient to bypass the use of ω-automata and end component identification, instead computing reachability probabilities directly from the product of the PA $\mathcal{M}$ and the error automaton for ϕ.

Definition 12 (PA-DFA product). *Let $\mathcal{M}$=$(S, \overline{s}, \alpha_{\mathcal{M}}, \delta_{\mathcal{M}}, L)$ be a PA and $\mathcal{A}^{err}_{\phi_A}$=$(Q, \overline{q}, \alpha_A, \delta_A, F)$ be the DFA for a regular safety property ϕ_A with $\alpha_A \subseteq \alpha_{\mathcal{M}}$. The* product *of $\mathcal{M}$ and $\mathcal{A}^{err}_{\phi_A}$ is given by the PA $\mathcal{M}\otimes\mathcal{A}^{err}_{\phi_A} = (S\times Q, (\overline{s}, \overline{q}), \alpha_{\mathcal{M}}, \delta', L')$, where δ' is defined such that we have $(s, q) \xrightarrow{a} \mu\times\eta_{q'}$ if and only if one of the following holds:*

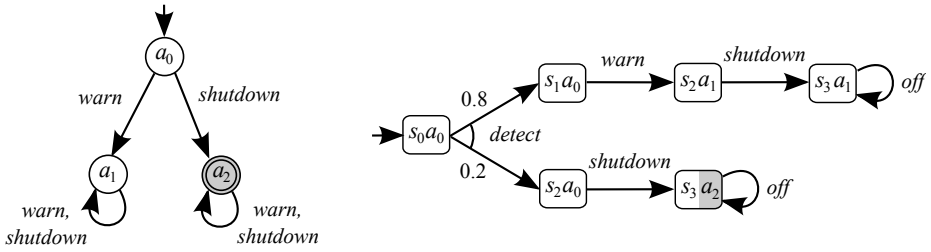
- *$s \xrightarrow{a} \mu$, $q' = \delta_A(q, a)$ and $a \in \alpha_A$;*
- *$s \xrightarrow{a} \mu$, $q' = q$ and $a \notin \alpha_A$;*

and $L'((s, q)) = L(s) \cup \{err_A\}$ if $q \in F$ and $L'((s, q)) = L(s)$ otherwise.

Intuitively, the product $\mathcal{M}\otimes\mathcal{A}^{err}_{\phi_A}$ records both the state of the PA $\mathcal{M}$ and the state of the DFA $\mathcal{A}^{err}_{\phi_A}$, based on the actions seen so far in the history of $\mathcal{M}$. States of the product that correspond to accepting states of the DFA are labelled with atomic proposition err_A. The key property of a PA-DFA product is the following.

Lemma 1. *For a PA $\mathcal{M}$ and DFA $\mathcal{A}^{err}_{\phi_A}$, there is a bijection $f_{\mathcal{M},A}$ between the adversaries of $\mathcal{M}$ and the adversaries of the product $\mathcal{M}\otimes\mathcal{A}^{err}_{\phi_A}$. Moreover, for any adversary σ of $\mathcal{M}$, we have the following correspondence between the probability of satisfying the regular safety property ϕ_A and the probability of reaching an err_A-labelled state in the product $\mathcal{M}\otimes\mathcal{A}^{err}_{\phi_A}$:*

$$Pr^{\sigma}_{\mathcal{M}}(\phi_A) \;=\; 1 - Pr^{f_{\mathcal{M},A}(\sigma)}_{\mathcal{M}\otimes\mathcal{A}^{err}_{\phi_A}}(\Diamond\, err_A)\,.$$



(a) DFA $\mathcal{A}^{err}_{\phi_A}$ (b) PA $\mathcal{M}_s \otimes \mathcal{A}^{err}_{\phi_A}$

Figure 3: A DFA $\mathcal{A}^{err}_{\phi_A}$ for safety property ϕ_A and the product $\mathcal{M}_s \otimes \mathcal{A}^{err}_{\phi_A}$ (see Example 3)

Proof. The existence of bijection $f_{\mathcal{M},A}$ follows directly from the definition of the PA-DFA product $\mathcal{M}\otimes\mathcal{A}^{err}_{\phi_A}$ (which relies on the fact that the automaton $\mathcal{A}^{err}_{\phi_A}$ is both deterministic and complete). The remainder of the lemma then follows from Theorem 10.51 of [32]. □

Using Lemma 1, the (minimum) probability of satisfying regular safety property ϕ_A relates to (maximum) reachability probabilities in the product as follows:

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

and therefore:

$$\mathcal{M} \models [\phi_A]_{\geq p_A} \;\Leftrightarrow\; Pr^{\max}_{\mathcal{M}\otimes\mathcal{A}^{err}_{\phi_A}}(\Diamond\, err_A) \leq 1 - p_A$$

which means that satisfaction of a probabilistic safety property $[\phi_A]_{\geq p_A}$ can be checked in time polynomial in the size of $\mathcal{M}\otimes\mathcal{A}^{err}_{\phi_A}$.

Example 3. We return to the PA $\mathcal{M}_s$ from Example 1 (see Figure 1(a)) and consider a probabilistic safety property $[\phi_A]_{\geq 0.8}$ where ϕ_A is a regular safety property over alphabet $\alpha_A = \{warn, shutdown\}$ with the meaning "*warn* occurs before *shutdown*". The error automaton $\mathcal{A}^{err}_{\phi_A}$ for ϕ_A is shown in Figure 3(a). To verify $[\phi_A]_{\geq 0.8}$ against $\mathcal{M}_s$, we construct the product PA $\mathcal{M}_s \otimes \mathcal{A}^{err}_{\phi_A}$, shown in Figure 3(b). States labelled with atomic proposition err_A, i.e. those corresponding to the accepting state a_2 of the DFA are marked with grey shading. We have $Pr^{\min}_{\mathcal{M}_s}(\phi_A) = 1 - Pr^{\max}_{\mathcal{M}_s\otimes\mathcal{A}^{err}_{\phi_A}}(\Diamond\, err_A) = 1 - 0.2 = 0.8$, and thus $\mathcal{M}_s \models [\phi_A]_{\geq 0.8}$.

Finally, we mention the computation required for expected total reward properties, i.e. to calculate $ExpTot^{\min}_{\mathcal{M}}(\rho)$ or $ExpTot^{\max}_{\mathcal{M}}(\rho)$ for a reward structure ρ. In fact, for this, we can use very similar techniques to those for reachability probabilities, such as value iteration or linear programming. We refer the reader to, for example, [3] for details.

3.3. Multi-objective Model Checking for PAs

In addition to conventional quantitative verification techniques for probabilistic automata, the approach presented in this paper requires the use of *multi-objective* model checking [14, 15]. The conventional approach, described in the previous section, allows us to check whether a single quantitative (probabilistic or reward) predicate holds for all adversaries of a PA (or, dually, for at least one adversary). Multi-objective queries allow us to reason about whether adversaries satisfy *multiple* properties of this form.

Definition 13 (qmo-property). *A* quantitative multi-objective property (qmo-property) *for PA $\mathcal{M}$ is a finite conjunction*[5] *of quantitative predicates $\Psi_P = [\psi_1]_{\sim_1 x_1} \wedge \ldots \wedge [\psi_k]_{\sim_k x_k}$, where each $[\psi_i]_{\sim_i x_i}$ is a quantitative predicate for $\mathcal{M}$, i.e. a probabilistic predicate $[\phi_i]_{\sim_i p_i}$ or a reward predicate $[\rho_i]_{\sim_i r_i}$. We use α_P to denote the set of all actions in Ψ_P, i.e., $\alpha_{\psi_1} \cup \ldots \cup \alpha_{\psi_k}$. We say that $\mathcal{M}$ satisfies Ψ_P under adversary σ, denoted $\mathcal{M}, \sigma \models \Psi_P$, if $\mathcal{M}, \sigma \models [\psi_i]_{\sim_i x_i}$ for all $1 \leq i \leq k$.*

We first observe that verifying whether a qmo-property $\Psi_P = [\psi_1]_{\sim_1 x_1} \wedge \ldots \wedge [\psi_k]_{\sim_k x_k}$ is satisfied *for all* adversaries of a PA, denoted $\mathcal{M} \models \Psi_P$, reduces simply to k separate checks:

$$\begin{array}{rcl} \mathcal{M} \models \Psi_P & \Leftrightarrow & \forall \sigma \in Adv_{\mathcal{M}} \,.\, \left(\wedge_{i=1}^{k} \mathcal{M}, \sigma \models [\psi_i]_{\sim_i x_i} \right) \\ & \Leftrightarrow & \wedge_{i=1}^{k} \mathcal{M} \models [\psi_i]_{\sim_i x_i} \,. \end{array}$$

However, a typical multi-objective query (often called an *achievability query*) asks if *there exists* an adversary of $\mathcal{M}$ satisfying all k predicates, i.e. whether:

$$\exists \sigma \in Adv_{\mathcal{M}} \,.\, \left(\wedge_{i=1}^{k} \mathcal{M}, \sigma \models [\psi_i]_{\sim_i x_i} \right) .$$

For the case where a qmo-property comprises only probabilistic ω-regular properties, an algorithm for checking the existence of such an adversary is given in [14]. This is based on a reduction to a linear programming (LP) problem, yielding a time complexity polynomial in the sizes of the PA $\mathcal{M}$ and the ω-automata representing the k ω-regular properties ψ_i. Whenever such an adversary exists, the solution of the LP problem yields a randomised, finite-memory adversary σ of $\mathcal{M}$ satisfying all k predicates. In [15], this LP-based approach to multi-objective model checking is extended to include expected total reward properties, i.e. for the class of qmo-properties described in Definition 13. In [33], an alternative approach to model checking qmo-properties is presented, based on value iteration. This has higher time complexity (exponential in the size of the model, in the worst case) but tends to perform better in practice.

Another useful class of multi-objective properties, also treated in [15, 33], is that of *numerical* queries. These yield the minimum or maximum achievable value for the probability of an ω-regular property ϕ or the expected total value of a reward structure ρ, whilst still satisfying some qmo-property Ψ_P. For example, in the maximum case:

$$\begin{array}{rcl} Pr_{\mathcal{M}}^{\max}(\phi \mid \Psi_P) & \stackrel{\text{def}}{=} & \sup \{ Pr_{\mathcal{M}}^{\sigma}(\phi) \mid \sigma \in Adv_{\mathcal{M}} \wedge \mathcal{M}, \sigma \models \Psi_P \}, \\ ExpTot_{\mathcal{M}}^{\max}(\rho \mid \Psi_P) & \stackrel{\text{def}}{=} & \sup \{ ExpTot_{\mathcal{M}}^{\sigma}(\rho) \mid \sigma \in Adv_{\mathcal{M}} \wedge \mathcal{M}, \sigma \models \Psi_P \}. \end{array}$$

The techniques for numerical queries in [33] are further extended to the class of *Pareto* queries, which can be used for a more detailed analysis of the trade-off between two or more properties. For example, given two ω-regular properties ϕ_1 and ϕ_2, we can consider the *Pareto curve* of points (p_1, p_2) such that $[\phi_1]_{\geq p_1} \wedge [\phi_2]_{\geq p_2}$ is achievable but any increase in either p_1 or p_2 would necessitate a decrease in the other. We discuss the use of numerical and Pareto queries for compositional verification in Section 6.

3.4. Model Checking PAs under Partial Adversaries

In this section, we explain how to perform model checking of PAs over the class of partial (rather than complete) adversaries, which will be needed later in the paper. We cover two cases.

[5] Quantitative multi-objective properties are introduced in [15], where a more general form is proposed: arbitrary Boolean combinations of predicates, rather than just conjunctions.

Firstly, we show that, for model checking of probabilistic safety properties, these two classes of adversaries are equivalent. Secondly, we show any other kind of property can be handled using a simple transformation of the PA. In both cases, we then have a reduction from model checking of PAs over partial adversaries to the problem of model checking over complete adversaries, as described in the preceding sections.

The following proposition states that satisfaction of probabilistic safety properties is equivalent whether quantifying over complete adversaries or over the larger class of partial adversaries. This is because checking probabilistic safety properties reduces to the computation of maximum reachability probabilities.

Proposition 1. *For any PA $\mathcal{M}$ and probabilistic safety property $[\phi_A]_{\geq p}$, we have:*

$$\mathcal{M} \models [\phi_A]_{\geq p} \quad \Leftrightarrow \quad \forall \sigma \in Adv_{\mathcal{M}}^{\text{part}} \,.\, \mathcal{M}, \sigma \models [\phi_A]_{\geq p} \,.$$

Proof. Consider any PA $\mathcal{M}$ and probabilistic safety property $[\phi_A]_{\geq p_A}$. By Definition 8, we have:

$$\mathcal{M} \models [\phi_A]_{\geq p} \quad \Leftrightarrow \quad \forall \sigma \in Adv_{\mathcal{M}} \,.\, \mathcal{M}, \sigma \models [\phi_A]_{\geq p}$$

and from Lemma 1 there is a bijection $f_{\mathcal{M},A}$ between the adversaries of PA $\mathcal{M}$ and the PA-DFA product $\mathcal{M} \otimes \mathcal{A}_{\phi_A}^{err}$.

For the $\Rightarrow$ direction, we assume $\mathcal{M} \models [\phi_A]_{\geq p}$ and consider any partial adversary σ of $\mathcal{M}$. Now, letting σ' be a completion of σ, it follows that $f_{\mathcal{M},A}(\sigma')$ is a completion of $f_{\mathcal{M},A}(\sigma)$ and, by the definition of reachability probabilities, we have:

$$Pr_{\mathcal{M} \otimes \mathcal{A}_{\phi_A}^{err}}^{f_{\mathcal{M},A}(\sigma')}(\Diamond\, err_A) \geq Pr_{\mathcal{M} \otimes \mathcal{A}_{\phi_A}^{err}}^{f_{\mathcal{M},A}(\sigma)}(\Diamond\, err_A) \,. \tag{1}$$

Next, using Lemma 1, we have:

$$\begin{aligned}
Pr_{\mathcal{M}}^{\sigma}(\phi_A) = \;& 1 - Pr_{\mathcal{M} \otimes \mathcal{A}_{\phi_A}^{err}}^{f_{\mathcal{M},A}(\sigma)}(\Diamond\, err_A) \\
\geq \;& 1 - Pr_{\mathcal{M} \otimes \mathcal{A}_{\phi_A}^{err}}^{f_{\mathcal{M},A}(\sigma')}(\Diamond\, err_A) && \text{by (1)} \\
= \;& Pr_{\mathcal{M}}^{\sigma'}(\phi_A) && \text{by Lemma 1} \\
\geq \;& p && \text{since } M \models [\phi_A]_{\geq p}.
\end{aligned}$$

Therefore, since σ was an arbitrary partial adversary of $\mathcal{M}$, this half of the proof is complete.

For the $\Leftarrow$ direction, the result follows directly from the fact that $Adv_{\mathcal{M}} \subset Adv_{\mathcal{M}}^{\text{part}}$. □

The next proposition shows that satisfaction of qmo-properties in a PA $\mathcal{M}$ using partial adversaries is equivalent to satisfaction under complete adversaries after performing a simple transformation of $\mathcal{M}$ which adds a probability 1 self-loop to every state. We formalise this transformation as a special case of alphabet extension (see Definition 7) using a fresh action b. The transformed PA is thus denoted $\mathcal{M}[\{b\}]$.

Proposition 2. *For any PA $\mathcal{M}$, action $b \notin \alpha_{\mathcal{M}}$ and qmo-property Ψ_P, we have:*

$$\exists \sigma \in Adv_{\mathcal{M}}^{\text{part}} \,.\, \mathcal{M}, \sigma \models \Psi_P \quad \Leftrightarrow \quad \exists \sigma' \in Adv_{\mathcal{M}[\{b\}]} \,.\, \mathcal{M}[\{b\}], \sigma' \models \Psi_P \,.$$

Proof. Consider any PA $\mathcal{M} = (S, \overline{s}, \alpha_{\mathcal{M}}, \delta_{\mathcal{M}}, L)$ and action $b \notin \alpha_{\mathcal{M}}$. By Definition 7 the only difference between the PAs $\mathcal{M}$ and $\mathcal{M}[\{b\}]$ is that the probabilistic transition relation of $\mathcal{M}[\{b\}]$ extends the one of $\mathcal{M}$ with a 'self-loop' transition for each state, labelled with the action b.

For the direction $\Rightarrow$, the proof follows by showing that, for any partial adversary σ of $\mathcal{M}$, we can construct a complete adversary σ' of $\mathcal{M}[\{b\}]$ which satisfies precisely the same qmo-properties that do not include b in their alphabet. The construction proceeds as follows. For any finite path $\pi' \in \mathit{FPaths}_{\mathcal{M}[\{b\}]}$ we have the following two cases to consider.

- If $\pi' \in \mathit{FPaths}_{\mathcal{M}}$, then for any action distribution pair $(a', \mu') \in \mathit{Dist}(\alpha_{\mathcal{M}[\{b\}]} \times \mathit{Dist}(S))$:

$$\sigma'(\pi')(a', \mu') = \begin{cases} \sigma(\pi')(a', \mu') & \text{if } a \in \alpha_{\mathcal{M}} \\ 1 - \sum_{(a,\mu) \in \mathit{Dist}(\alpha_{\mathcal{M}} \times \mathit{Dist}(S))} \sigma(\pi')(a, \mu) & \text{if } (a', \mu') = (b, \eta_{\mathit{last}(\pi)}) \\ 0 & \text{otherwise.} \end{cases}$$

- If $\pi' \notin \mathit{FPaths}_{\mathcal{M}}$, then for any action distribution pair $(a', \mu') \in \mathit{Dist}(\alpha_{\mathcal{M}[\{b\}]} \times \mathit{Dist}(S))$:

$$\sigma'(\pi')(a', \mu') = \begin{cases} 1 & \text{if } (a', \mu') = (b, \eta_{\mathit{last}(\sigma')}) \\ 0 & \text{otherwise.} \end{cases}$$

The fact that these adversaries satisfy the same qmo-properties (not including b in their alphabet) follows by construction of the probability measure for σ and σ'.

For the direction $\Leftarrow$, the proof follows by showing that, for any complete adversary σ' of $\mathcal{M}[\{b\}]$, we can construct a partial adversary σ of $\mathcal{M}[\{b\}]$ which satisfies precisely the same qmo-properties that do not include b in their alphabet. Before giving the construction, we need to define a mapping from paths of $\mathcal{M}[\{b\}]$ to paths of $\mathcal{M}$. This mapping essentially removes the transitions of the path labelled with the action b. This always yields feasible paths of $\mathcal{M}$ since all transitions of $\mathcal{M}[\{b\}]$ labelled with the action b correspond to self-loops. Formally, for any finite path $\pi' \in \mathit{FPaths}_{\mathcal{M}[\{b\}]}$, let $\pi' \restriction_{\alpha_{\mathcal{M}}}$ be the finite path of $\mathit{FPaths}_{\mathcal{M}}$ where:

$$\pi' \restriction_{\alpha_{\mathcal{M}}} = \begin{cases} \pi'' \restriction_{\alpha_{\mathcal{M}}} \xrightarrow{(a,\mu)} s & \text{if } \pi' \text{ is of the form } \pi'' \xrightarrow{(a,\mu)} s \text{ and } a \neq b \\ \pi'' \restriction_{\alpha_{\mathcal{M}}} & \text{if } \pi' \text{ is of the form } \pi'' \xrightarrow{(b,\eta_s)} s \\ \pi' & \text{otherwise.} \end{cases}$$

From Definition 7, in the 'otherwise' case above, it follows that π' is a path of length 0, that is, a state of $\mathcal{M}$.

Now, considering any complete adversary σ' of $\mathcal{M}[\{b\}]$, we construct the following partial adversary σ of $\mathcal{M}$. For any finite path $\pi \in \mathit{FPaths}_{\mathcal{M}}$ and action distribution pair $(a, \mu) \in \mathit{Dist}(\alpha_{\mathcal{M}} \times \mathit{Dist}(S))$:

$$\sigma(\pi)(a, \mu) = \frac{\sum \left\{ \mathit{Pr}^{\sigma'}_{\mathcal{M}[\{b\}]}(\pi') \cdot \sigma'(\pi')(a, \mu) \;\middle|\; \pi' \in \mathit{FPaths}_{\mathcal{M}[\{b\}]} \wedge \pi' \restriction_{\alpha_{\mathcal{M}}} = \pi \right\}}{\mathit{Pr}^{\sigma}_{\mathcal{M}}(\pi)}$$

The remainder of the proof follows from the construction of the probability measures for the adversaries σ' and σ. □

3.5. Model Checking PAs under Fairness

We conclude our discussion of model checking techniques for PAs with the topic of *fairness*. When verifying systems where multiple PAs are composed in parallel and can execute asynchronously, it is often necessary to impose conditions that ensure the PAs are scheduled in a fair manner. These fairness conditions, which typically correspond to reasonable assumptions about the system being modelled, may be essential in order to verify even trivial properties.

In this paper, we use a simple but effective notion of fairness called *unconditional fairness*, in which it is required that each component makes a transition infinitely often. For probabilistic automata, a natural approach to incorporating fairness (as taken in, e.g., [34, 35]) is to restrict analysis of the system to a class of adversaries in which fair behaviour occurs with probability 1.

If $\mathcal{M} = \mathcal{M}_1 \| \ldots \| \mathcal{M}_n$ is a PA comprising n components, then an (unconditionally) *fair path* of $\mathcal{M}$ is an infinite path $\pi \in IPaths_{\mathcal{M}}$ in which, for each component $\mathcal{M}_i$, there exists an action $a \in \alpha_{\mathcal{M}_i}$ that appears infinitely often. A *fair adversary* σ of $\mathcal{M}$ is an adversary for which $Pr^{\sigma}_{\mathcal{M}}\{\, \pi \in IPaths_{\mathcal{M}} \mid \pi \text{ is fair} \,\} = 1$. We let $Adv^{\text{fair}}_{\mathcal{M}}$ denote the set of fair adversaries of $\mathcal{M}$.

An important consequence of our definition of fairness is that the projection of a fair adversary onto its component PAs results in a complete, rather than partial, adversaries.

Lemma 2. *If $\mathcal{M}_1, \mathcal{M}_2$ are PAs and $\sigma \in Adv^{\text{fair}}_{\mathcal{M}_1 \| \mathcal{M}_2}$, then $\sigma{\restriction}_{\mathcal{M}_i}$ is a complete adversary for i=1, 2.*

Proof. Consider any PAs $\mathcal{M}_1$ and $\mathcal{M}_2$ and adversary $\sigma \in Adv^{\text{fair}}_{\mathcal{M}_1 \| \mathcal{M}_2}$. Suppose for a contradiction that $\sigma{\restriction}_{\mathcal{M}_i}$ is a partial adversary for some i=1, 2. Since $\sigma{\restriction}_{\mathcal{M}_i}$ is partial, by definition there exists some finite path π_i^{fin} of $\mathcal{M}_i$ such that:

$$\textstyle\sum_{(last(\pi_i^{fin}),a,\mu_i) \in \delta_{\mathcal{M}_i}} \sigma{\restriction}_{\mathcal{M}_i}(\pi_i^{fin})(a,\mu_i) < 1 \,.$$

Now, by Definition 6, it follows that:

$$1 - \textstyle\sum_{(last(\pi_i^{fin}),a,\mu_i) \in \delta_{\mathcal{M}_i}} \sigma{\restriction}_{\mathcal{M}_i}(\pi_i^{fin})(a,\mu_i) = Pr^{\sigma}_{\mathcal{M}_1 \| \mathcal{M}_2}(\{\, \pi \in IPaths^{\sigma}_{\mathcal{M}_1 \| \mathcal{M}_2} \mid \pi{\restriction}_{\mathcal{M}_i} = \pi_i^{fin} \,\}) \,.$$

By the definition of a fair path, we have that the projection of a fair path is always infinite and, combining this fact with the above, yields:

$$Pr^{\sigma}_{\mathcal{M}_1 \| \mathcal{M}_2}(\{\, \pi \in IPaths^{\sigma}_{\mathcal{M}} \mid \pi \text{ is not fair } \}) \geq Pr^{\sigma}_{\mathcal{M}_1 \| \mathcal{M}_2}(\{\, \pi \in IPaths^{\sigma}_{\mathcal{M}_1 \| \mathcal{M}_2} \mid \pi{\restriction}_{\mathcal{M}_i} = \pi_i^{fin} \,\}) > 0$$

which contradicts the fact that σ is a fair adversary as required. □

Verification of a quantitative predicate $[\psi]_{\sim x}$ against a multi-component PA $\mathcal{M} = \mathcal{M}_1 \| \cdots \| \mathcal{M}_n$ *under fairness* is defined by quantifying only over fair adversaries:

$$\mathcal{M} \models^{\text{fair}} [\psi]_{\sim x} \;\Leftrightarrow\; \forall \sigma \in Adv^{\text{fair}}_{\mathcal{M}} \,.\, \mathcal{M}, \sigma \models [\psi]_{\sim x} \,.$$

Model checking algorithms for verifying PAs under fairness are presented in [35]. These techniques apply to more general notions of strong and weak fairness, of which unconditional fairness is a special case. The algorithms work by restricting the set of end components of the PA under consideration. A related problem is *realisability*: determining whether a PA has any (unconditionally) fair adversaries. Efficient techniques for this problem, based on an analysis of the underlying graph of a PA, can be found in [36].

We can also pose multi-objective queries, as discussed in the previous section, in the context of fairness, i.e., ask whether there exists a fair adversary satisfying all conjuncts of a qmo-property $\Psi_P = [\psi_1]_{\sim_1 x_1} \wedge \ldots \wedge [\psi_k]_{\sim_k x_k}$:

$$\exists \sigma \in Adv^{\text{fair}}_{\mathcal{M}} \,.\, \left(\wedge_{i=1}^{k} \mathcal{M}, \sigma \models [\psi_i]_{\sim_i x_i} \right) .$$

This can be reduced to a normal multi-objective query, i.e. over *all* adversaries. Our definition of a fair adversary can be captured by a single ω-regular probabilistic predicate $[\phi_{\text{fair}}]_{\geq 1}$, where:

$$\phi_{\text{fair}} = \cap_{i=1}^{n} \{ w \in (\alpha_{\mathcal{M}})^{\omega} \mid \text{some action } a \in \alpha_{\mathcal{M}_i} \text{ appears in } w \text{ infinitely often} \} .$$

Thus, we can check for a fair adversary satisfying Ψ_P by checking for an arbitrary adversary that satisfies $\Psi_P \wedge [\phi_{\text{fair}}]_{\geq 1}$.

4. Assume-Guarantee Verification for Probabilistic Automata

We now present our compositional verification framework for probabilistic automata. The approach is based on the well-known *assume-guarantee* paradigm and builds on the *multi-objective* model checking techniques summarised in Section 3.3. In this section, we define the basic underlying ideas and then introduce our two main classes of assume-guarantee proof rules, for safety properties and general quantitative properties. In Section 5, we will consider a number of further assume-guarantee proof rules.

4.1. Assume-Guarantee Triples

The key ingredient of classical assume-guarantee reasoning is the *assume-guarantee triple*, comprising a component, an assumption and a guarantee. To enable compositional verification of probabilistic systems, we introduce *probabilistic assume-guarantee triples*. These triples take the form $\langle \Psi_A \rangle \, \mathcal{M}^{\star} \, \langle \Psi_G \rangle$, where $\mathcal{M}$ is a PA, representing a component of a probabilistic system, and Ψ_A and Ψ_G are qmo-properties, representing an assumption and a guarantee, respectively. Triples are also parameterised by a class of adversaries for $\mathcal{M}$, denoted by the symbol $\star \in \{\text{part}, \text{comp}, \text{fair}\}$, indicating the set of partial, complete or fair adversaries, respectively. The triple $\langle \Psi_A \rangle \, \mathcal{M}^{\star} \, \langle \Psi_G \rangle$ asserts that any adversary for $\mathcal{M}$ (of type $\star$) which satisfies the assumption Ψ_A also satisfies the guarantee Ψ_G. Formally, we have the following definition.[6]

Definition 14 (Probabilistic assume-guarantee triple). *Let $\mathcal{M}$ be a PA, Ψ_A and Ψ_G be qmo-properties such that $\alpha_G \subseteq \alpha_A \cup \alpha_{\mathcal{M}}$ and $\star \in \{\text{part}, \text{comp}, \text{fair}\}$ be a class of adversaries. Then $\langle \Psi_A \rangle \, \mathcal{M}^{\star} \, \langle \Psi_G \rangle$ is a* probabilistic assume-guarantee triple *with the following semantics:*

$$\langle \Psi_A \rangle \, \mathcal{M}^{\star} \, \langle \Psi_G \rangle \;\Leftrightarrow\; \forall \sigma \in Adv^{\star}_{\mathcal{M}[\alpha_A]} \,.\, \left(\mathcal{M}[\alpha_A], \sigma \models \Psi_A \rightarrow \mathcal{M}[\alpha_A], \sigma \models \Psi_G \right) .$$

Notice that Definition 14 quantifies over adversaries of the alphabet extension $\mathcal{M}[\alpha_A]$ of $\mathcal{M}$ (see Definition 7), rather than $\mathcal{M}$ itself. This is because, when we come to use these triples to formulate assume-guarantee proof rules, Ψ_G will be a property of the overall system being verified, of which $\mathcal{M}$ is just one component. Property Ψ_G may therefore be defined over a

[6] Our definition of assume-guarantee triple generalises the one we originally presented in [17], where the parameter $\star$ is omitted. In that work, the set of partial adversaries is always used, i.e. $\star$ = part.

superset of the actions in the alphabet $\alpha_{\mathcal{M}}$ of $\mathcal{M}$ and, as a result, the assumption Ψ_A may also include additional actions not in $\alpha_{\mathcal{M}}$.

Because probabilistic assume-guarantee triples are stated in terms of quantification over adversaries, checking whether or not a triple is true can be reduced to a multi-objective model checking problem of the form described in the previous section. More precisely, we ascertain whether $\langle\Psi_A\rangle\, \mathcal{M}^{\star}\, \langle\Psi_G\rangle$ is *not* true by checking for the existence of an adversary that satisfies the assumption Ψ_A but does not satisfy the property Ψ_G.

Proposition 3. *Let PA $\mathcal{M}$, qmo-properties Ψ_A, Ψ_G and $\star \in \{\text{part}, \text{comp}, \text{fair}\}$ be as given in Definition 14. Furthermore, let Ψ_G be of the form $[\psi_G]_{\sim x_G}$, i.e., a single quantitative predicate.*[7] *Then, checking whether the assume-guarantee triple $\langle\Psi_A\rangle\, \mathcal{M}^{\star}\, \langle\Psi_G\rangle$ holds reduces to multi-objective model checking as follows:*

$$\langle\Psi_A\rangle\, \mathcal{M}^{\star}\, \langle\Psi_G\rangle \;\Leftrightarrow\; \neg\exists\, \sigma \in Adv^{\star}_{\mathcal{M}[\alpha_A]} \,.\, \left(\, \mathcal{M}[\alpha_A], \sigma \models \Psi_A \wedge \mathcal{M}[\alpha_A], \sigma \not\models [\psi_G]_{\sim x_G}\right) .$$

This can be done using the techniques from [15], described in Section 3.3, with time complexity polynomial in the sizes of the PA $\mathcal{M}$ and the (deterministic Rabin) automata representing the probabilistic predicates that occur in Ψ_A and Ψ_G.

Proof. The result follows directly from Definition 14. □

Moreover, for the special case of assume guarantee triples restricted to probabilistic safety properties and partial adversaries, checking whether the triple holds reduces to the simpler multi-objective model checking algorithm of [14].

Proposition 4. *Let $\mathcal{M}$ be a PA, $\Psi_A^{safe} = [\phi_1]_{\geq p_1} \wedge \ldots \wedge [\phi_n]_{\geq p_n}$ be a conjunction of probabilistic safety properties and $\Psi_G^{safe} = [\phi_G]_{\geq p_G}$ be a single probabilistic safety property. If $\mathcal{M}'$ is the product PA $\mathcal{M}[\alpha_A] \otimes \mathcal{A}^{err}_{\phi_1} \otimes \cdots \otimes \mathcal{A}^{err}_{\phi_n} \otimes \mathcal{A}^{err}_{\phi_G}$, then:*

$$\langle\Psi_A^{safe}\rangle\, \mathcal{M}^{\text{part}}\, \langle\Psi_G^{safe}\rangle \;\Leftrightarrow\; \neg\exists\sigma' \in Adv^{\text{part}}_{\mathcal{M}'} \,.\, \left(\left(\wedge_{i=1}^{n} Pr^{\sigma'}_{\mathcal{M}'}(\Diamond err_i) \leq 1-p_i\right) \wedge Pr^{\sigma'}_{\mathcal{M}'}(\Diamond err_G) > 1-p_G\right).$$

This can be checked in time polynomial in the sizes of $\mathcal{M}$, $\mathcal{A}^{err}_{\phi_1}, \ldots, \mathcal{A}^{err}_{\phi_n}$ and $\mathcal{A}^{err}_{\phi_G}$.

Proof. Consider any PA $\mathcal{M}$, conjunction of probabilistic safety properties $\Psi_A^{safe} = [\phi_1]_{\geq p_1} \wedge \ldots \wedge [\phi_n]_{\geq p_n}$ and single probabilistic safety property $\Psi_G^{safe} = [\phi_G]_{\geq p_G}$. Letting $\mathcal{M}'$ be the product $\mathcal{M}[\alpha_A] \otimes \mathcal{A}^{err}_{\phi_1} \otimes \cdots \otimes \mathcal{A}^{err}_{\phi_n} \otimes \mathcal{A}^{err}_{\phi_G}$, through repeated application of Lemma 1, we can construct a bijective function $f : Adv^{\text{part}}_{\mathcal{M}[\alpha_A]} \to Adv^{\text{part}}_{\mathcal{M}'}$ such that, for any (partial) adversary σ of $\mathcal{M}[\alpha_A]$ and $\phi_i \in \{\phi_1, \ldots, \phi_n, \phi_G\}$:

$$Pr^{\sigma}_{\mathcal{M}[\alpha_A]}(\phi_i) = 1 - Pr^{f(\sigma)}_{\mathcal{M}'}(\Diamond err_i)\,. \tag{2}$$

Now, using Definition 14, we have:

$$\begin{aligned}
&\langle\Psi_A^{safe}\rangle\, \mathcal{M}^{\text{part}}\, \langle\Psi_G^{safe}\rangle \Leftrightarrow\; \forall\sigma \in Adv^{\text{part}}_{\mathcal{M}[\alpha_A]} . \left(\left(\wedge_{i=1}^{n} Pr^{\sigma}_{\mathcal{M}[\alpha_A]}(\phi_i) \geq p_i\right) \rightarrow Pr^{\sigma}_{\mathcal{M}[\alpha_A]}(\phi_G) \geq p_G\right) \\
&\Leftrightarrow\; \forall\sigma \in Adv^{\text{part}}_{\mathcal{M}[\alpha_A]} . \left(\left(\wedge_{i=1}^{n} Pr^{f(\sigma)}_{\mathcal{M}'}(\Diamond err_i) \leq 1-p_i\right) \rightarrow Pr^{f(\sigma)}_{\mathcal{M}'}(\Diamond err_G) \leq 1-p_G\right) && \text{using (2)} \\
&\Leftrightarrow\; \forall\sigma' \in Adv^{\text{part}}_{\mathcal{M}'} . \left(\left(\wedge_{i=1}^{n} Pr^{\sigma'}_{\mathcal{M}'}(\Diamond err_i) \leq 1-p_i\right) \rightarrow Pr^{\sigma'}_{\mathcal{M}'}(\Diamond err_G) \leq 1-p_G\right) && \text{since } f \text{ is a bijection} \\
&\Leftrightarrow\; \neg\exists\sigma' \in Adv^{\text{part}}_{\mathcal{M}'} . \left(\left(\wedge_{i=1}^{n} Pr^{\sigma'}_{\mathcal{M}'}(\Diamond err_i) \leq 1-p_i\right) \wedge Pr^{\sigma'}_{\mathcal{M}'}(\Diamond err_G) > 1-p_G\right) && \text{rearranging}
\end{aligned}$$

which completes the proof. □

[7] For clarity of presentation, we restrict our attention in Proposition 3 to the case where Ψ_G is a single predicate and then later generalise to arbitrary triples in Theorem 3.

4.2. Assume-Guarantee Verification for Safety Properties

We now present, using the definitions above, *asymmetric* assume-guarantee proof rules for compositional verification of probabilistic automata, that is, those which use only a single assumption about one system component. Experience in the non-probabilistic setting [7] indicates that, despite their simplicity, rules of this form are widely applicable. First, in this section, we discuss an approach in which both assumptions and guarantees are probabilistic safety properties. Then, in Section 4.3, we describe how to handle a wider class of quantitative properties, including ω-regular and reward properties, by incorporating fairness. To simplify the presentation, here and in Section 4.3 we restrict our attention to guarantees consisting of a single predicate, however in Section 4.4 we extend the approach to allow general guarantees.

Our first proof rule, (Asym-Safety), can be stated as follows.[8]

Theorem 1. *If $\mathcal{M}_1, \mathcal{M}_2$ are PAs, Ψ_A^{safe} is a conjunction of probabilistic safety properties and Ψ_G^{safe} is a single probabilistic safety property such that $\alpha_A \subseteq \alpha_{\mathcal{M}_1}$ and $\alpha_G \subseteq \alpha_{\mathcal{M}_2} \cup \alpha_A$, then the following proof rule holds:*

$$\frac{\mathcal{M}_1 \models \Psi_A^{safe} \qquad \langle \Psi_A^{safe} \rangle\, \mathcal{M}_2 \,^{\text{part}}\, \langle \Psi_G^{safe} \rangle}{\mathcal{M}_1 \,\|\, \mathcal{M}_2 \models \Psi_G^{safe}} \qquad \text{(Asym-Safety)}$$

Theorem 1 means that, given an appropriate assumption $\Psi_A^{safe} = [\phi_1]_{\geq p_1} \wedge \ldots \wedge [\phi_n]_{\geq p_n}$, we can check the correctness of a probabilistic safety property $[\phi_G]_{\geq p_G}$ on a two-component system, $\mathcal{M}_1 \| \mathcal{M}_2$, without constructing and model checking the full system. Instead, we perform one instance of (standard) model checking on $\mathcal{M}_1$, to check the first premise of rule (Asym-Safety), and one instance of multi-objective model checking on $\mathcal{M}_2[\alpha_A]$, to check the assume-guarantee triple in the second premise (recall, from Definition 14, that a triple for $\mathcal{M}_2$ is defined in terms of $\mathcal{M}_2[\alpha_A]$). If the error automata $\mathcal{A}_{\phi_i}^{err}$ for the regular safety properties ϕ_i in the assumption are much smaller than the PA for component $\mathcal{M}_1$, we can expect significant gains in terms of the overall performance for verification. Our experimental results, described later in Section 7, show that this can indeed be the case in practice.

Before proving Theorem 1, we give an example of its usage.

Example 4. We illustrate the application of rule (Asym-Safety) using the PAs $\mathcal{M}_s$ and $\mathcal{M}_d$ from Example 1 (see Figure 1) and property $\Psi_G^{safe} = [\phi_G]_{\geq 0.98}$ from Example 2, letting $\mathcal{M}_1 = \mathcal{M}_s$ and $\mathcal{M}_2 = \mathcal{M}_d$. As an assumption, we use the probabilistic safety property $\Psi_A^{safe} = [\phi_A]_{\geq 0.8}$ from Example 3, where ϕ_A means "*warn* occurs before *shutdown*".

Checking the first premise of (Asym-Safety) amounts to verifying that $\mathcal{M}_s \models [\phi_A]_{\geq 0.8}$, which was illustrated previously in Example 3. To complete the verification, we need to check the second premise, i.e. $\langle \Psi_A^{safe} \rangle\, \mathcal{M}_d \,^{\text{part}}\, \langle \Psi_G^{safe} \rangle$, which reduces to a multi-objective model checking problem on $\mathcal{M}_d$ (notice that $\alpha_A = \{warn, shutdown\} \subseteq \alpha_{\mathcal{M}_d}$ so $\mathcal{M}_d[\alpha_A] = \mathcal{M}_d$). More precisely, from Proposition 4, we need to check that, for the product PA $\mathcal{M}_d \otimes \mathcal{A}_{\phi_A}^{err} \otimes \mathcal{A}_{\phi_G}^{err}$, there is no adversary under which the probability of reaching err_A states is at most $1 - 0.8 = 0.2$ *and* the probability of reaching an err_G state is strictly above $1-0.98 = 0.02$.

[8] This rule is equivalent to the rule (Asym) we originally presented in [17]. In that work, the first premise and conclusion quantify over partial adversaries, rather than all adversaries. However, as Proposition 1 shows, these are equivalent.

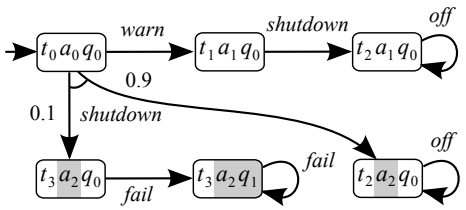


Figure 4: The product PA $\mathcal{M}_d \otimes \mathcal{A}^{err}_{\phi_A} \otimes \mathcal{A}^{err}_{\phi_G}$ (see Example 4)

The product is shown in Figure 4, where we indicate states satisfying err_A and err_G by shading in grey the accepting states a_2 and q_1 of the corresponding DFAs $\mathcal{A}^{err}_{\phi_A}$ and $\mathcal{A}^{err}_{\phi_G}$. By inspection, we see that no such adversary exists, so we can deduce that $\mathcal{M}_s \| \mathcal{M}_d \models [\phi_G]_{\geq 0.98}$. Consider, however, the adversary σ which, in the initial state, chooses *warn* with probability 0.8 and *shutdown* with probability 0.2. This reaches err_A with probability 0.2 and err_G with probability 0.02. So, $\langle \Psi_A^{safe} \rangle \, \mathcal{M}_d \, ^{\text{part}} \langle \Psi_G^{safe} \rangle$ does *not* hold for any value of $p_G > 1-0.02 = 0.98$.

We now prove Theorem 1. For this, and for the other results in this section, we require the following lemma, adapted from [8].

Lemma 3. *Let $\mathcal{M}_1, \mathcal{M}_2$ be PAs, σ an adversary (complete or partial) of $\mathcal{M}_1 \| \mathcal{M}_2$, $\alpha \subseteq \alpha_{\mathcal{M}_1 \| \mathcal{M}_2}$ and i=1, 2. If ϕ_i and ϕ_i' are ω-regular properties over α_i and α_i' such that $\alpha_i \subseteq \alpha_{\mathcal{M}_i}$ and $\alpha_i' \subseteq \alpha_{\mathcal{M}_i[\alpha]}$, then:*

$$\text{(a)} \quad Pr^{\sigma}_{\mathcal{M}_1 \| \mathcal{M}_2}(\phi_i) = Pr^{\sigma \restriction_{\mathcal{M}_i}}_{\mathcal{M}_i}(\phi_i)$$
$$\text{(b)} \quad Pr^{\sigma}_{\mathcal{M}_1 \| \mathcal{M}_2}(\phi_i') = Pr^{\sigma \restriction_{\mathcal{M}_i[\alpha]}}_{\mathcal{M}_i[\alpha]}(\phi_i') \, .$$

If ρ and ρ_i' are reward structures over α_i and α_i', then:

$$\text{(c)} \quad ExpTot^{\sigma}_{\mathcal{M}_1 \| \mathcal{M}_2}(\rho_i) = ExpTot^{\sigma \restriction_{\mathcal{M}_i}}_{\mathcal{M}_i}(\rho_i)$$
$$\text{(d)} \quad ExpTot^{\sigma}_{\mathcal{M}_1 \| \mathcal{M}_2}(\rho_i') = ExpTot^{\sigma \restriction_{\mathcal{M}_i[\alpha]}}_{\mathcal{M}_i[\alpha]}(\rho_i').$$

If Ψ_i and Ψ_i' are qmo-properties over α_i and α_i', then:

$$\text{(e)} \quad \mathcal{M}_1 \| \mathcal{M}_2, \sigma \models \Psi_i \quad \Leftrightarrow \quad \mathcal{M}_i, \sigma \restriction_{\mathcal{M}_i} \models \Psi_i$$
$$\text{(f)} \quad \mathcal{M}_1 \| \mathcal{M}_2, \sigma \models \Psi_i' \quad \Leftrightarrow \quad \mathcal{M}_i[\alpha], \sigma \restriction_{\mathcal{M}_i[\alpha]} \models \Psi_i'.$$

Note that the projections onto $\mathcal{M}_i[\alpha]$ in the above are well defined since the condition $\alpha \subseteq \alpha_{\mathcal{M}_1 \| \mathcal{M}_2}$ implies that $\mathcal{M}_1 \| \mathcal{M}_2 = \mathcal{M}_1[\alpha] \| \mathcal{M}_2 = \mathcal{M}_1 \| \mathcal{M}_2[\alpha]$.

Proof. Parts (a)-(d) are a simple extension of [8, Lemma 7.2.6, page 141]. Parts (e) and (f) follow directly from parts (a)-(d). □

Lemma 3 is fundamental to the proof rules that we present in this paper. It relates an adversary σ of a composed PA $\mathcal{M}_1 \| \mathcal{M}_2$ and the adversary $\sigma \restriction_{\mathcal{M}_i}$ that results when projecting σ onto one of the component PAs $\mathcal{M}_i$. In essence, it says that, if a property (e.g. an ω-regular property ϕ_i) refers only to actions that appear in $\mathcal{M}_i$ (i.e., $\alpha_i \subseteq \alpha_{\mathcal{M}_i}$), then the probability of that property is the same in $\mathcal{M}_1 \| \mathcal{M}_2$ under σ, and in $\mathcal{M}_i$ under $\sigma \restriction_{\mathcal{M}_i}$.

Proof (of Theorem 1). Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be PAs, Ψ_A^{safe} a conjunction of probabilistic safety properties and Ψ_G^{safe} is a single probabilistic safety property such that $\alpha_A \subseteq \alpha_{\mathcal{M}_1}$, $\alpha_G \subseteq \alpha_{\mathcal{M}_2} \cup \alpha_A$, $\mathcal{M}_1 \models \Psi_A^{safe}$ and $\langle \Psi_A^{safe} \rangle \, \mathcal{M}_2 \, ^{\text{part}} \langle \Psi_G^{safe} \rangle$. For any adversary σ of $\mathcal{M}_1 \| \mathcal{M}_2$ by definition we have $\mathcal{M}_1, \sigma \models \Psi_A^{safe}$, and hence since $\sigma \restriction_{\mathcal{M}_1}$ is a partial adversary of $\mathcal{M}_1$, using Proposition 1 we have:

$$
\begin{aligned}
\mathcal{M}_1 \models \Psi_A^{safe} &\Rightarrow \mathcal{M}_1, \sigma \restriction_{\mathcal{M}_1} \models \Psi_A^{safe} \\
&\Rightarrow \mathcal{M}_1 \| \mathcal{M}_2, \sigma \models \Psi_A^{safe} && \text{by Lemma 3(e) since } \alpha_A \subseteq \alpha_{\mathcal{M}_1} \\
&\Rightarrow \mathcal{M}_2[\alpha_A], \sigma \restriction_{\mathcal{M}_2[\alpha_A]} \models \Psi_A^{safe} && \text{by Lemma 3(f) since } \alpha_A \subseteq \alpha_{\mathcal{M}_2[\alpha_A]} \\
&\Rightarrow \mathcal{M}_2[\alpha_A], \sigma \restriction_{\mathcal{M}_2[\alpha_A]} \models \Psi_G^{safe} && \text{since } \langle \Psi_A^{safe} \rangle \, \mathcal{M}_2 \, ^{\text{part}} \langle \Psi_G^{safe} \rangle \\
&\Rightarrow \mathcal{M}_1 \| \mathcal{M}_2, \sigma \models \Psi_G^{safe} && \text{by Lemma 3(f) since } \alpha_G \subseteq \alpha_{\mathcal{M}_2[\alpha_A]}
\end{aligned}
$$

Since σ was an arbitrary adversary of $\mathcal{M}_1 \| \mathcal{M}_2$, it follows that $\mathcal{M}_1 \,\|\, \mathcal{M}_2 \models \Psi_G^{safe}$, as required. □

4.3. Assume-Guarantee Verification for Quantitative Properties

Next, we extend our assume-guarantee framework with a proof rule in which the properties to be proved and the assumptions used to do so can comprise arbitrary quantitative predicates, rather than just probabilistic safety properties. More precisely, we use a qmo-property Ψ_A for an assumption and a single quantitative predicate $\Psi_G = [\psi_G]_{\sim_G x_G}$ for the property to be proved. Recall that quantitative predicates include probabilistic ω-regular properties (which subsume both probabilistic safety properties and probabilistic LTL) and expected total reward properties.

To make this possible, we need an additional ingredient: fairness. Our proof rule provides compositional verification of a multi-component system $\mathcal{M}_1 \| \mathcal{M}_2$ under (unconditional) fairness, i.e. it guarantees that Ψ_G holds under any fair adversary of $\mathcal{M}_1 \| \mathcal{M}_2$. The reason that fairness enables this rule to work stems from the fact that the projection of a fair adversary of $\mathcal{M}_1 \| \mathcal{M}_2$ onto a component $\mathcal{M}_i$ is a complete adversary, as will become clear when we prove the rule subsequently.

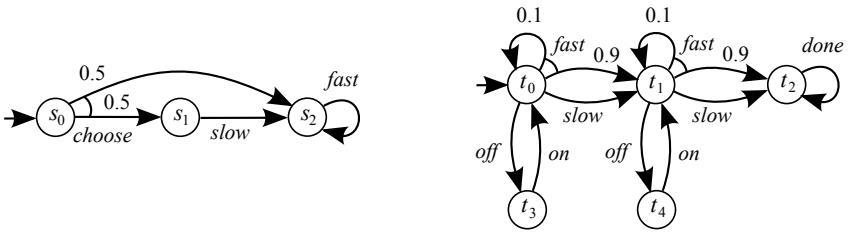
As in the previous section, we initially restrict our attention to an asymmetric assume-guarantee proof rule. Formally, we state this as follows.

Theorem 2. *Let $\mathcal{M}_1, \mathcal{M}_2$ be PAs, Ψ_A a qmo-property and $\Psi_G = [\psi_G]_{\sim_G x_G}$ a single quantitative predicate, such that $\alpha_A \subseteq \alpha_{\mathcal{M}_1}$ and $\alpha_G \subseteq \alpha_A \cup \alpha_{\mathcal{M}_2}$. Then the following proof rule holds:*

$$
\frac{\mathcal{M}_1 \models^{\text{fair}} \Psi_A \qquad \langle \Psi_A \rangle \, \mathcal{M}_2 \, ^{\text{fair}} \langle \Psi_G \rangle}{\mathcal{M}_1 \,\|\, \mathcal{M}_2 \models^{\text{fair}} \Psi_G} \qquad \text{(Asym-Quant)}
$$

Theorem 2 provides the means to verify an ω-regular or reward property Ψ_G on a composed system $\mathcal{M}_1 \| \mathcal{M}_2$. Like (Asym-Safety) in Theorem 1, we can do this without constructing $\mathcal{M}_1 \| \mathcal{M}_2$ by decomposing into two-sub problems: one instance of normal verification (premise 1) and one instance of multi-objective model checking (premise 2). Again, these two steps have the potential to be significantly cheaper than verifying the combined system $\mathcal{M}_1 \| \mathcal{M}_2$.

The inclusion of fairness in the two premises of (Asym-Quant) is to permit recursive applications of the rule, in order to compositionally verify systems comprising more than two components. If $\mathcal{M}_1$ is just a single PA, the stronger (but easier) check $\mathcal{M}_1 \models \Psi_A$ suffices for premise 1. Similarly, if $\mathcal{M}_2$ is a single PA, we can check $\langle \Psi_A \rangle \, \mathcal{M}_2 \, ^{\text{comp}} \langle \Psi_G \rangle$ for premise 2.



(a) PA $\mathcal{M}_c$ (b) PA $\mathcal{M}_m$

Figure 5: Two probabilistic automata $\mathcal{M}_c$ and $\mathcal{M}_m$ (see Example 5)

Example 5. We illustrate proof rule (Asym-Quant) from Theorem 2 using another pair of PAs, $\mathcal{M}_m$ and $\mathcal{M}_c$, shown in Figure 5, representing a machine and its controller, respectively. The machine $\mathcal{M}_m$ executes 2 consecutive jobs, each in 1 of 2 ways: *fast*, which requires 1 second, but fails with probability 0.1; or *slow*, which requires 3 seconds, and never fails. PA $\mathcal{M}_c$ models a specific controller for the machine, which instructs it which way to execute jobs, when composed in parallel with $\mathcal{M}_m$. With probability 0.5, the controller sends an initial instruction *slow*. Subsequently, it only sends the instruction *fast*.

Our aim is to verify that the expected total execution time for the controlled machine is at most $\frac{19}{6}$. Since PAs do not explicitly model time, we achieve this by using a reward structure $\rho_{time}=\{fast\mapsto 1, slow\mapsto 3\}$ to capture the passage of time and reward predicate $\Psi_G = [\rho_{time}]_{\leq \frac{19}{6}}$. We apply rule (Asym-Quant), with $\mathcal{M}_1=\mathcal{M}_c$ and $\mathcal{M}_2=\mathcal{M}_m$. Let ϕ_{on} be the ω-regular property stating that action *off* never occurs (i.e. $\Box\neg$*off* in LTL notation) and $\rho_{slow}=\{slow\mapsto 1\}$ be a reward structure that counts the number of occurrences of action *slow*. We use the assumption $\Psi_A=[\phi_{on}]_{\geq 1} \wedge [\rho_{slow}]_{\leq \frac{1}{2}}$, i.e. we assume the controller never issues an *off* instruction and that the expected number of *slow* jobs requested is at most 0.5.

We first verify (separately) that $\mathcal{M}_c \models [\phi_{on}]_{\geq 1}$ and $\mathcal{M}_c \models [\rho_{slow}]_{\leq \frac{1}{2}}$, from which we conclude that $\mathcal{M}_c \models \Psi_A$. Next, we check the assume-guarantee triple $\langle \Psi_A \rangle\, \mathcal{M}_m\, {}^{\text{comp}}\, \langle \Psi_G \rangle$. The triple is checked by verifying that no (complete) adversary of $\mathcal{M}_m$ satisfies $[\phi_{on}]_{\geq 1} \wedge [\rho_{slow}]_{\leq \frac{1}{2}} \wedge [\rho_{time}]_{> \frac{19}{6}}$, which we see to be true from inspection. Thus, we can conclude that $\mathcal{M}_c \| \mathcal{M}_m \models^{\text{fair}} [\rho_{time}]_{\leq \frac{19}{6}}$.

Proof (of Theorem 2). Suppose $\mathcal{M}_1$ and $\mathcal{M}_2$ are PAs, Ψ_A is a qmo-property and $\Psi_G = [\psi_G]_{\sim_G x_G}$ is a single quantitative predicate, such that both $\mathcal{M}_1 \models^{\text{fair}} \Psi_A$ and $\langle \Psi_A \rangle\, \mathcal{M}_2\, {}^{\text{fair}}\, \langle \Psi_G \rangle$ hold. Consider an arbitrary fair adversary σ of $\mathcal{M}_1 \| \mathcal{M}_2$. Now, from Lemma 2, we know that $\sigma{\restriction}_{\mathcal{M}_1}$ is a complete adversary of $\mathcal{M}_1$. Furthermore, we have that $\mathcal{M}_1, \sigma{\restriction}_{\mathcal{M}_1}$ is also fair, and hence by definition of $\models^{\text{fair}}$:

$$
\begin{aligned}
\mathcal{M}_1 \models^{\text{fair}} \Psi_A \;&\Rightarrow\; \mathcal{M}_1, \sigma{\restriction}_{\mathcal{M}_1} \models \Psi_A \\
&\Rightarrow\; \mathcal{M}_1 \| \mathcal{M}_2, \sigma \models \Psi_A && \text{by Lemma 3(e) since } \alpha_A \subseteq \alpha_{\mathcal{M}_1} \\
&\Rightarrow\; \mathcal{M}_2[\alpha_A], \sigma{\restriction}_{\mathcal{M}_2[\alpha_A]} \models \Psi_A && \text{by Lemma 3(f) since } \alpha_A \subseteq \alpha_{\mathcal{M}_2[\alpha_A]} \\
&\Rightarrow\; \mathcal{M}_2[\alpha_A], \sigma{\restriction}_{\mathcal{M}_2[\alpha_A]} \models \Psi_G && \text{since } \langle \Psi_A \rangle\, \mathcal{M}_2\, {}^{\text{fair}}\, \langle \Psi_G \rangle \\
&\Rightarrow\; \mathcal{M}_1 \| \mathcal{M}_2, \sigma \models \Psi_G && \text{by Lemma 3(f) since } \alpha_G \subseteq \alpha_{\mathcal{M}_2[\alpha_A]}.
\end{aligned}
$$

Since σ was an arbitrary fair adversary of $\mathcal{M}_1 \| \mathcal{M}_2$, we have $\mathcal{M}_1 \| \mathcal{M}_2 \models^{\text{fair}} \Psi_G$, as required. □

4.4. Extensions

Next, we discuss two ways in which the proof rules presented so far can be extended. Subsequently, in Section 5, we will present several additional rules.

Conjunctions of predicates. Firstly, we remark that it is straightforward to extend our approach to verify properties that are conjunctions of predicates, rather than single predicates. For presentational simplicity, we so far assumed (in Propositions 3 and 4, and Theorems 1 and 2) that the qmo-property Ψ_G used on the right hand side of a triple $\langle \Psi_A \rangle \, \mathcal{M}^\star \, \langle \Psi_G \rangle$ comprised a single predicate. However, checking a triple in which Ψ_G is a conjunction of k predicates can be decomposed into k separate checks, i.e, k applications of Proposition 3 or Proposition 4.

Theorem 3. *Let $\mathcal{M}$ be a PA and, for i=1, 2, let $\Psi^{safe}_{A_i}$, $\Psi^{safe}_{G_i}$, Ψ_{A_i} and Ψ_{G_i} be qmo-properties, where those marked with the superscript "safe" comprise only probabilistic safety properties. Then the following proof rules hold:*

$$\frac{\langle \Psi^{safe}_{A_1} \rangle \, \mathcal{M}^{\text{part}} \, \langle \Psi^{safe}_{G_1} \rangle \quad \langle \Psi^{safe}_{A_2} \rangle \, \mathcal{M}^{\text{part}} \, \langle \Psi^{safe}_{G_2} \rangle}{\langle \Psi^{safe}_{A_\wedge} \rangle \, \mathcal{M}^{\text{part}} \, \langle \Psi^{safe}_{G_\wedge} \rangle} \text{ (Conj-Safety)} \qquad \frac{\langle \Psi_{A_1} \rangle \, \mathcal{M}^{\text{fair}} \, \langle \Psi_{G_1} \rangle \quad \langle \Psi_{A_2} \rangle \, \mathcal{M}^{\text{fair}} \, \langle \Psi_{G_2} \rangle}{\langle \Psi_{A_\wedge} \rangle \, \mathcal{M}^{\text{fair}} \, \langle \Psi_{G_\wedge} \rangle} \text{ (Conj-Quant)}$$

where $\Psi^{safe}_{A_\wedge} = \Psi^{safe}_{A_1} \wedge \Psi^{safe}_{A_2}$, $\Psi^{safe}_{G_\wedge} = \Psi^{safe}_{G_1} \wedge \Psi^{safe}_{G_2}$, $\Psi_{A_\wedge} = \Psi_{A_1} \wedge \Psi_{A_2}$ and $\Psi_{G_\wedge} = \Psi_{G_1} \wedge \Psi_{G_2}$.

Proof. The result follows directly from the definition of assume-guarantee triples (see Definition 14). □

Theorem 3 provides a way to check assume-guarantee triples containing arbitrary qmo-properties. Furthermore, observe that Theorems 1 and 2 can easily be generalised to the case where the property Ψ^{safe}_G or Ψ_G being proved on the system $\mathcal{M}_1 \| \mathcal{M}_2$ comprises more than one predicate, thus yielding proof rules for arbitrary qmo-properties.

Notice that, in Theorem 3, unlike the previous theorems, we omitted explicit statements about the restrictions imposed on the alphabets of the PAs and properties. There, and from this point on, we will implicitly assume that, if a rule contains an occurrence of the triple $\langle \Psi_A \rangle \, \mathcal{M}^\star \, \langle \Psi_G \rangle$, then $\alpha_G \subseteq \alpha_A \cup \alpha_{\mathcal{M}}$; similarly, for a premise that checks Ψ_A against $\mathcal{M}$, we assume that $\alpha_A \subseteq \alpha_{\mathcal{M}}$.

Multiple components. Secondly, we observe that, simply through repeated application of either (Asym-Safety) or (Asym-Quant), we obtain proof rules for systems consisting of n components.

Theorem 4. *Let $\mathcal{M}_1, \ldots, \mathcal{M}_n$ be PAs and $\Psi^{safe}_{A_1}, \ldots, \Psi^{safe}_{A_{n-1}}$, Ψ^{safe}_G, $\Psi_{A_1}, \ldots, \Psi_{A_{n-1}}$ and Ψ_G be qmo-properties, where those marked with the superscript "safe" comprise only probabilistic safety properties. Then the following proof rules hold:*

$$\frac{\begin{array}{c} \mathcal{M}_1 \models \Psi^{safe}_{A_1} \\ \langle \Psi^{safe}_{A_1} \rangle \, \mathcal{M}_2 \, ^{\text{part}} \, \langle \Psi^{safe}_{A_2} \rangle \\ \cdots \\ \langle \Psi^{safe}_{A_{n-1}} \rangle \, \mathcal{M}_n \, ^{\text{part}} \, \langle \Psi^{safe}_{G} \rangle \end{array}}{\mathcal{M}_1 \| \cdots \| \mathcal{M}_n \models \Psi^{safe}_G} \textit{(Asym-Safety-N)} \qquad \frac{\begin{array}{c} \mathcal{M}_1 \models^{\text{fair}} \Psi_{A_1} \\ \langle \Psi_{A_1} \rangle \, \mathcal{M}_2 \, ^{\text{fair}} \, \langle \Psi_{A_2} \rangle \\ \cdots \\ \langle \Psi_{A_{n-1}} \rangle \, \mathcal{M}_n \, ^{\text{fair}} \, \langle \Psi_{G} \rangle \end{array}}{\mathcal{M}_1 \| \cdots \| \mathcal{M}_n \models^{\text{fair}} \Psi_G} \textit{(Asym-Quant-N)}$$

PROOF. The result follows by iteratively applying rules (ASYM-SAFETY) and (CONJ-SAFETY) or (ASYM-QUANT) and (CONJ-QUANT), respectively, to the parallel composition of $\mathcal{M}_1 \| \cdots \| \mathcal{M}_{i-1}$ and $\mathcal{M}_i$, for $i = 2, \ldots, n$. □

Elsewhere [20], we have already successfully applied rule (ASYM-SAFETY-N), showing that it can be significantly more efficient to use an N-way, rather than 2-way, decomposition for assume-guarantee verification. It also worth noting at this point that the parallel composition operator $\|$ for PAs is both associative and commutative, which allows for flexibility in the way that multi-component rules are applied, and the way that several separate proof rules are combined.

5. Further Proof Rules

In this section, we consider three additional classes of assume-guarantee proof rules for compositional verification of probabilistic automata.

5.1. Circular Proof Rules

One potential limitation of the various proof rules considered so far is that they are all asymmetric, i.e., they may analyse, for example, a component $\mathcal{M}_2$ using an assumption Ψ_A about another component $\mathcal{M}_1$, but checking whether $\mathcal{M}_1$ satisfies Ψ_A cannot make any assumptions about $\mathcal{M}_2$. Below, we give proof rules that we call *circular*, which do allow the use of additional assumptions in this way.

Theorem 5. *If $\mathcal{M}_1$ and $\mathcal{M}_2$ are PAs, $\Psi^{safe}_{A_1}$, $\Psi^{safe}_{A_2}$, Ψ^{safe}_{G} are qmo-properties comprising only safety properties and Ψ_{A_1}, Ψ_{A_2} and Ψ_G are qmo-properties, then the following circular assume-guarantee proof rules hold:*

$$\frac{\begin{array}{c}\mathcal{M}_2 \models \Psi^{safe}_{A_2} \\ \langle \Psi^{safe}_{A_2} \rangle \, \mathcal{M}_1 \, {}^{\text{part}} \langle \Psi^{safe}_{A_1} \rangle \\ \langle \Psi^{safe}_{A_1} \rangle \, \mathcal{M}_2 \, {}^{\text{part}} \langle \Psi^{safe}_{G} \rangle\end{array}}{\mathcal{M}_1 \| \mathcal{M}_2 \models \Psi^{safe}_{G}} \quad \text{(CIRC-SAFETY)} \qquad \frac{\begin{array}{c}\mathcal{M}_2 \models^{\text{fair}} \Psi_{A_2} \\ \langle \Psi_{A_2} \rangle \, \mathcal{M}_1 \, {}^{\text{fair}} \langle \Psi_{A_1} \rangle \\ \langle \Psi_{A_1} \rangle \, \mathcal{M}_2 \, {}^{\text{fair}} \langle \Psi_{G} \rangle\end{array}}{\mathcal{M}_1 \| \mathcal{M}_2 \models^{\text{fair}} \Psi_{G}} \quad \text{(CIRC-QUANT)}$$

PROOF. We give the proof of (CIRC-QUANT); the proof for (CIRC-SAFETY) follows similarly using Proposition 1 as opposed to Lemma 2 (see the proof of Theorem 1).

Suppose $\mathcal{M}_1$ and $\mathcal{M}_2$ are PAs, Ψ_{A_1}, Ψ_{A_2} and Ψ_G are qmo-properties, such that $\mathcal{M}_2 \models^{\text{fair}} \Psi_{A_2}$, $\langle \Psi_{A_2} \rangle \, \mathcal{M}_1 \, {}^{\text{fair}} \langle \Psi_{A_1} \rangle$ and $\langle \Psi_{A_1} \rangle \, \mathcal{M}_2 \, {}^{\text{fair}} \langle \Psi_{G} \rangle$ hold. Now, consider an arbitrary fair adversary σ of $\mathcal{M}_1 \| \mathcal{M}_2$. Since $\mathcal{M}_2 \models^{\text{fair}} \Psi_{A_2}$, using Lemma 2, we have $\mathcal{M}_2, \sigma{\restriction_{\mathcal{M}_2}} \models \Psi_{A_2}$ and therefore, since $\alpha_{A_2} \subseteq \alpha_{\mathcal{M}_2}$ and applying Lemma 3(e), we have:

$$\begin{array}{rll}
\mathcal{M}_2, \sigma{\restriction_{\mathcal{M}_2}} \Psi_{A_2} \Rightarrow & \mathcal{M}_1 \| \mathcal{M}_2, \sigma \models \Psi_{A_2} & \\
\Rightarrow & \mathcal{M}_1[\alpha_{A_2}], \sigma{\restriction_{\mathcal{M}_1[\alpha_{A_2}]}} \models \Psi_{A_2} & \text{by Lemma 3(f) since } \alpha_{A_2} \subseteq \alpha_{\mathcal{M}_1[\alpha_{A_2}]} \\
\Rightarrow & \mathcal{M}_1[\alpha_{A_2}], \sigma{\restriction_{\mathcal{M}_1[\alpha_{A_2}]}} \models \Psi_{A_1} & \text{since } \langle \Psi_{A_2} \rangle \, \mathcal{M}_1 \, {}^{\text{fair}} \langle \Psi_{A_1} \rangle \\
\Rightarrow & \mathcal{M}_1 \| \mathcal{M}_2, \sigma \models \Psi_{A_1} & \text{by Lemma 3(f) since } \alpha_{A_2} \subseteq \alpha_{\mathcal{M}_2} \\
\Rightarrow & \mathcal{M}_2[\alpha_{A_1}], \sigma{\restriction_{\mathcal{M}_2[\alpha_{A_1}]}} \models \Psi_{A_1} & \text{by Lemma 3(f) since } \alpha_{A_1} \subseteq \alpha_{\mathcal{M}_2[\alpha_{A_1}]} \\
\Rightarrow & \mathcal{M}_2[\alpha_{A_1}], \sigma{\restriction_{\mathcal{M}_2[\alpha_{A_1}]}} \models \Psi_{G} & \text{since } \langle \Psi_{A_1} \rangle \, \mathcal{M}_2 \, {}^{\text{fair}} \langle \Psi_{G} \rangle \\
\Rightarrow & \mathcal{M}_1 \| \mathcal{M}_2, \sigma \models \Psi_{G} & \text{by Lemma 3(f) since } \alpha_{G} \subseteq \alpha_{\mathcal{M}_2[\alpha_{A_1}]}.
\end{array}$$

Therefore, since σ was an arbitrary fair adversary of $\mathcal{M}_1||\mathcal{M}_2$, we have $\mathcal{M}_1||\mathcal{M}_2 \models^{\text{fair}} \Psi_G$ as required. □

5.2. Asynchronous Proof Rules

Our next class of rules 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 each coin *not* returning a tail is $1/2$. 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 $1-1/2^n$. In the context of our assume-guarantee proof rules, we can reason about systems with asynchronous components as follows.

Theorem 6. *For i=1, 2, let $\mathcal{M}_i$ be a PA, $\Psi^{safe}_{A_i}$ a qmo-property comprising only safety properties and $\Psi^{safe}_{G_i} = [\phi^{safe}_{G_i}]_{\geq p_{G_i}}$ be a single probabilistic safety property. If $(\alpha_{\mathcal{M}_1} \cup \alpha_{A_1}) \cap (\alpha_{\mathcal{M}_2} \cup \alpha_{A_2}) = \emptyset$ (which implies that $\alpha_{\mathcal{M}_1} \cap \alpha_{\mathcal{M}_2} = \emptyset$, i.e., the components are asynchronous), then the following proof rule holds:*

$$\frac{\langle \Psi^{safe}_{A_1} \rangle\ \mathcal{M}_1\ {}^{\text{part}}\ \langle \Psi^{safe}_{G_1} \rangle \qquad \langle \Psi^{safe}_{A_2} \rangle\ \mathcal{M}_2\ {}^{\text{part}}\ \langle \Psi^{safe}_{G_2} \rangle}{\langle \Psi^{safe}_{A} \rangle\ \mathcal{M}_1||\mathcal{M}_2\ {}^{\text{part}}\ \langle \Psi^{safe}_{G} \rangle} \quad \text{(Async-Safety)}$$

where $\Psi^{safe}_A = \Psi^{safe}_{A_1} \wedge \Psi^{safe}_{A_2}$ and $\Psi^{safe}_G = [\phi^{safe}_{G_1} \cup \phi^{safe}_{G_2}]_{\geq p_{G_1} + p_{G_2} - p_{G_1} \cdot p_{G_2}}$. Note that the union of two regular safety properties is also a regular safety property.

In addition, if $\sim \in \{<, \leq, \geq, >\}$ and, for i=1, 2, we have that Ψ_{A_i} is a qmo-property and $\Psi_{G_i} = [\phi_{G_i}]_{\sim p_{G_i}}$ is a single probabilistic predicate such that $(\alpha_{\mathcal{M}_1} \cup \alpha_{A_1}) \cap (\alpha_{\mathcal{M}_2} \cup \alpha_{A_2}) = \emptyset$, then the following proof rule holds:

$$\frac{\langle \Psi_{A_1} \rangle\ \mathcal{M}_1\ {}^{\text{fair}}\ \langle \Psi_{G_1} \rangle \qquad \langle \Psi_{A_2} \rangle\ \mathcal{M}_2\ {}^{\text{fair}}\ \langle \Psi_{G_2} \rangle}{\langle \Psi_{A} \rangle\ \mathcal{M}_1||\mathcal{M}_2\ {}^{\text{fair}}\ \langle \Psi_{G} \rangle} \quad \text{(Async-Quant)}$$

where $\Psi_A = \Psi_{A_1} \wedge \Psi_{A_2}$ and $\Psi_G = [\phi_{G_1} \cup \phi_{G_2}]_{\sim p_{G_1} + p_{G_2} - p_{G_1} \cdot p_{G_2}}$. Note that ω-regular properties are closed under union (and also intersection and complementation).

Proof. We consider the case of general quantitative predicates, i.e. rule (Async-Quant). The case for safety properties follows similarly. Therefore, suppose for i=1, 2 that $\mathcal{M}_i$ is a PA, Ψ_{A_i} is a qmo-property and $\Psi_{G_i} = [\phi_{G_i}]_{\sim p_{G_i}}$ is a single probabilistic predicate such that $(\alpha_{\mathcal{M}_1} \cup \alpha_{A_1}) \cap (\alpha_{\mathcal{M}_2} \cup \alpha_{A_2}) = \emptyset$ and $\langle \Psi_{A_i} \rangle\ \mathcal{M}_i\ {}^{\text{fair}}\ \langle \Psi_{G_i} \rangle$ for i=1, 2. In addition, let $\Psi_A = \Psi_{A_1} \wedge \Psi_{A_2}$ and $\Psi_G = [\phi_{G_1} \cup \phi_{G_2}]_{\sim p_{G_1} + p_{G_2} - p_{G_1} \cdot p_{G_2}}$.

Now, to show $\langle \Psi_A \rangle\ \mathcal{M}_1||\mathcal{M}_2\ {}^{\text{fair}}\ \langle \Psi_G \rangle$ holds, we consider any fair adversary σ of $(\mathcal{M}_1||\mathcal{M}_2)[\alpha_A]$ such that $(\mathcal{M}_1||\mathcal{M}_2)[\alpha_A], \sigma \models \Psi_A$. Letting $\overline{\phi}$ be the complement of an ω-regular property, it follows that:

$$Pr^{\sigma}_{(\mathcal{M}_1||\mathcal{M}_2)[\alpha_A]}(\phi_{G_1} \cup \phi_{G_2}) = 1 - Pr^{\sigma}_{(\mathcal{M}_1||\mathcal{M}_2)[\alpha_A]}(\overline{\phi}_{G_1} \cap \overline{\phi}_{G_2})\,. \tag{3}$$

Next, since $\alpha_A = \alpha_{A_1} \cup \alpha_{A_2}$ and the parallel composition operator is commutative and associative, using Definition 7 we have:

$$(\mathcal{M}_1 \| \mathcal{M}_2)[\alpha_A] \;=\; (\mathcal{M}_1[\alpha_{A_1}]) \,\|\, (\mathcal{M}_2[\alpha_{A_2}])\,.$$

Using this result and the fact that $(\alpha_{\mathcal{M}_1} \cup \alpha_{A_1}) \cap (\alpha_{\mathcal{M}_2} \cup \alpha_{A_2}) = \emptyset$ and $\alpha_{G_i} \subseteq \alpha_{\mathcal{M}_i} \cup \alpha_{A_i}$ for $i{=}1,2$, we can derive the equality:

$$\begin{aligned} Pr^{\sigma}_{(\mathcal{M}_1\|\mathcal{M}_2)[\alpha_A]}(\overline{\phi}_{G_1} \cap \overline{\phi}_{G_2}) &= Pr^{\sigma\restriction_{\mathcal{M}_1[\alpha_{A_1}]}}_{\mathcal{M}_1[\alpha_{A_1}]}(\overline{\phi}_{G_1}) \cdot Pr^{\sigma\restriction_{\mathcal{M}_2[\alpha_{A_2}]}}_{\mathcal{M}_2[\alpha_{A_2}]}(\overline{\phi}_{G_2}) \\ &= \left(1 - Pr^{\sigma\restriction_{\mathcal{M}_1[\alpha_{A_1}]}}_{\mathcal{M}_1[\alpha_{A_1}]}(\phi_{G_1})\right) \cdot \left(1 - Pr^{\sigma\restriction_{\mathcal{M}_2[\alpha_{A_2}]}}_{\mathcal{M}_2[\alpha_{A_2}]}(\phi_{G_2})\right). \end{aligned} \tag{4}$$

Since $(\mathcal{M}_1\|\mathcal{M}_2)[\alpha_A], \sigma \models \Psi_A$, by construction of Ψ_A, it follows that $(\mathcal{M}_1\|\mathcal{M}_2)[\alpha_A], \sigma \models \Psi_{A_i}$ for $i{=}1,2$, and hence using Lemma 3(f) we have $\mathcal{M}_i[\alpha_{A_i}], \sigma\restriction_{\mathcal{M}_1[\alpha_{A_1}]} \models \Psi_{A_i}$ for $i{=}1,2$. Combining this with the hypothesis $\langle \Psi_{A_i} \rangle\, \mathcal{M}_i \,^{\text{fair}}\, \langle \Psi_{G_i} \rangle$ for $i{=}1,2$ and Lemma 2, it follows that:

$$Pr^{\sigma\restriction_{\mathcal{M}_1[\alpha_{A_1}]}}_{\mathcal{M}_1[\alpha_{A_1}]}(\phi_{G_1}) \sim p_{G_1} \;\text{ and }\; Pr^{\sigma\restriction_{\mathcal{M}_2[\alpha_{A_2}]}}_{\mathcal{M}_2[\alpha_{A_2}]}(\phi_{G_2}) \sim p_{G_2}$$

which, together with (4) and (3), yields the inequality:

$$Pr^{\sigma}_{(\mathcal{M}_1\|\mathcal{M}_2)[\alpha_A]}(\phi_{G_1} \cup \phi_{G_2}) \sim 1 - (1{-}p_{G_1}) \cdot (1{-}p_{G_2}) = p_{G_1} + p_{G_2} - p_{G_1} \cdot p_{G_2}$$

demonstrating that $(\mathcal{M}_1\|\mathcal{M}_2)[\alpha_A], \sigma \models \Psi_G$. Therefore, since σ was an arbitrary fair adversary of $(\mathcal{M}_1\|\mathcal{M}_2)[\alpha_A]$ for which $(\mathcal{M}_1\|\mathcal{M}_2)[\alpha_A], \sigma \models \Psi_A$, we have $\langle \Psi_A \rangle\, \mathcal{M}_1\|\mathcal{M}_2 \,^{\text{fair}}\, \langle \Psi_G \rangle$ as required. □

We remark that the rules in Theorem 6 can be adapted to prove properties with multiple, not single, predicates. This can be achieved as discussed in Section 4.4 for the earlier rules.

5.3. Decomposition of Reward-based Properties

The third class of proof rules that we present in this section concerns the verification of reward properties. We show that, to prove properties of reward structures over a complete system, we can divide the reward structure into sub-structures over the system's components, prove properties for each sub-structure and then combine the results to prove properties of the original reward structure over the complete system.

First, for reward structures $\rho_1 : \alpha_{\rho_1} \to \mathbb{R}_{>0}$ and $\rho_2 : \alpha_{\rho_2} \to \mathbb{R}_{>0}$, we define the composition $\rho_1{+}\rho_2 : \alpha_{\rho_1} \cup \alpha_{\rho_2} \to \mathbb{R}_{>0}$ such that, for any $a \in \alpha_{\rho_1} \cup \alpha_{\rho_2}$:

$$(\rho_1{+}\rho_2)(a) = \begin{cases} \rho_1(a){+}\rho_2(a) & \text{if } a \in \alpha_{\rho_1} \cap \alpha_{\rho_2} \\ \rho_1(a) & \text{if } a \in \alpha_{\rho_1} \setminus \alpha_{\rho_2} \\ \rho_2(a) & \text{if } a \in \alpha_{\rho_2} \setminus \alpha_{\rho_1}. \end{cases}$$

Using this definition we have the following result.

Theorem 7. *If $\mathcal{M}_1, \mathcal{M}_2$ are PAs, $\sim \in \{<, \leq, \geq, >\}$ and, for $i{=}1,2$, we have that Ψ_{A_i} is a qmo-property and $\Psi_{G_i} = [\rho_{G_i}]_{\sim r_{G_i}}$ is a single reward predicate, then the following proof rule holds:*

$$\frac{\begin{array}{c}\langle \Psi_{A_1} \rangle\, \mathcal{M}_1 \,^{\text{fair}}\, \langle \Psi_{G_1} \rangle \\ \langle \Psi_{A_2} \rangle\, \mathcal{M}_2 \,^{\text{fair}}\, \langle \Psi_{G_2} \rangle\end{array}}{\langle \Psi_A \rangle\, \mathcal{M}_1\|\mathcal{M}_2 \,^{\text{fair}}\, \langle \Psi_G \rangle} \quad \text{(Sum-Reward)}$$

where $\Psi_A = \Psi_{A_1} \wedge \Psi_{A_2}$ and $\Psi_G = [\rho_{G_1} + \rho_{G_2}]_{\sim r_{G_1} + r_{G_2}}$.

Proof. Suppose $\mathcal{M}_1$ and $\mathcal{M}_2$ are PAs, $\sim \in \{<, \leq, \geq, >\}$ and, for i=1, 2, we have that Ψ_{A_i} is a qmo-property and $\Psi_{G_i} = [\phi_{G_i}]_{\sim r_{G_i}}$ is a single reward predicate such that $\langle \Psi_{A_i} \rangle \, \mathcal{M}_i \, ^{\text{fair}} \langle \Psi_{G_i} \rangle$. In addition, let $\Psi_A = \Psi_{A_1} \wedge \Psi_{A_2}$ and $\Psi_G = [\rho_{G_1} + \rho_{G_2}]_{\sim r_{G_1} + r_{G_2}}$. Since $\alpha_A = \alpha_{A_1} \cup \alpha_{A_2}$ and the parallel composition operator is commutative and associative, using Definition 7 we have:

$$(\mathcal{M}_1 \| \mathcal{M}_2)[\alpha_A] \;=\; (\mathcal{M}_1[\alpha_{A_1}]) \,\|\, (\mathcal{M}_2[\alpha_{A_2}])\,. \tag{5}$$

Now, to show that $\langle \Psi_A \rangle \, \mathcal{M}_1 \| \mathcal{M}_2 \, ^{\text{fair}} \langle \Psi_G \rangle$, we consider any fair adversary σ of $(\mathcal{M}_1 \| \mathcal{M}_2)[\alpha_A]$ such that $(\mathcal{M}_1 \| \mathcal{M}_2)[\alpha_A], \sigma \models \Psi_A$. By Definition 9, we have:

$$\begin{aligned}
&ExpTot^{\sigma}_{(\mathcal{M}_1 \| \mathcal{M}_2)[\alpha_A]}(\rho_1 + \rho_2) = \int_{\pi} (\rho_1 + \rho_2)(\pi)\, dPr^{\sigma}_{(\mathcal{M}_1 \| \mathcal{M}_2)[\alpha_A]} \\
&= \int_{\pi} \rho_1(\pi)\, dPr^{\sigma}_{(\mathcal{M}_1 \| \mathcal{M}_2)[\alpha_A]} + \int_{\pi} \rho_2(\pi)\, dPr^{\sigma}_{(\mathcal{M}_1 \| \mathcal{M}_2)[\alpha_A]} && \text{rearranging} \\
&= ExpTot^{\sigma}_{(\mathcal{M}_1 \| \mathcal{M}_2)[\alpha_A]}(\rho_1) + ExpTot^{\sigma}_{(\mathcal{M}_1 \| \mathcal{M}_2)[\alpha_A]}(\rho_2) && \text{by Definition 9} \\
&= ExpTot^{\sigma}_{\mathcal{M}_1[\alpha_{A_1}] \| \mathcal{M}_2[\alpha_{A_2}]}(\rho_1) + ExpTot^{\sigma}_{\mathcal{M}_1[\alpha_{A_1}] \| \mathcal{M}_2[\alpha_{A_2}]}(\rho_2) && \text{by (5)} \\
&= ExpTot^{\sigma \restriction_{\mathcal{M}_1[\alpha_{A_1}]}}_{\mathcal{M}_1[\alpha_{A_1}]}(\rho_1) + ExpTot^{\sigma \restriction_{\mathcal{M}_2[\alpha_{A_2}]}}_{\mathcal{M}_2[\alpha_{A_2}]}(\rho_2) && \text{by Lemma 3(d).}
\end{aligned} \tag{6}$$

Since $(\mathcal{M}_1 \| \mathcal{M}_2)[\alpha_A], \sigma \models \Psi_A$, from construction of Ψ_A we have $(\mathcal{M}_1 \| \mathcal{M}_2)[\alpha_A], \sigma \models \Psi_{A_i}$ for i=1, 2. Hence, using Lemma 3(d) and (5), it follows that $\mathcal{M}_1[\alpha_{A_i}], \sigma \restriction_{\mathcal{M}_i[\alpha_{A_i}]} \models \Psi_{A_i}$ for i=1, 2, which, together with the fact that $\langle \Psi_{A_i} \rangle \, \mathcal{M}_i \, ^{\text{fair}} \langle \Psi_{G_i} \rangle$ for i=1, 2, implies:

$$ExpTot^{\sigma \restriction_{\mathcal{M}_1[\alpha_{A_1}]}}_{\mathcal{M}_1[\alpha_{A_1}]}(\rho_1) \sim r_{G_1} \quad \text{and} \quad ExpTot^{\sigma \restriction_{\mathcal{M}_2[\alpha_{A_2}]}}_{\mathcal{M}_2[\alpha_{A_2}]}(\rho_2) \sim r_{G_2} \tag{7}$$

Combining (6) and (7) we have:

$$ExpTot^{\sigma}_{(\mathcal{M}_1 \| \mathcal{M}_2)[\alpha_A]}(\rho) \sim r_{G_1} + r_{G_2},$$

and hence $(\mathcal{M}_1 \| \mathcal{M}_2)[\alpha_A], \sigma \models \Psi_G$. Since σ was an arbitrary fair adversary of $(\mathcal{M}_1 \| \mathcal{M}_2)[\alpha_A]$ for which $(\mathcal{M}_1 \| \mathcal{M}_2)[\alpha_A], \sigma \models \Psi_A$, it follows that $\langle \Psi_A \rangle \, \mathcal{M}_1 \| \mathcal{M}_2 \, ^{\text{fair}} \langle \Psi_G \rangle$ as required. □

Although the rule (Sum-Reward) is defined in terms of assume-guarantee triples, it also allows us to prove the satisfaction of reward properties. In particular, assuming $\alpha_{\rho_i} \subseteq \alpha_{\mathcal{M}_i}$ for i=1, 2, then as a special case of the rule we have:

$$\frac{\begin{array}{c} \mathcal{M}_1 \models^{\text{fair}} [\rho_{G_1}]_{\sim r_{G_1}} \\ \mathcal{M}_2 \models^{\text{fair}} [\rho_{G_2}]_{\sim r_{G_2}} \end{array}}{\mathcal{M}_1 \| \mathcal{M}_2 \models^{\text{fair}} [\rho_{G_1} + \rho_{G_2}]_{\sim r_{G_1} + r_{G_2}}}$$

6. Numerical Assume-Guarantee Queries

Practical experience with probabilistic verification suggests that it is often more instructive to analyse models using *numerical*, rather than Boolean, queries. For example, instead of checking the correctness of a (lower-bounded) probabilistic predicate $[\phi]_{\geq p}$ against a PA $\mathcal{M}$ for some bound p, it may be preferable to just directly compute the *minimum* probability $Pr^{\min}_{\mathcal{M}}(\phi)$ that ϕ is satisfied. This is because $Pr^{\min}_{\mathcal{M}}(\phi)$ gives the *maximum* value of p for which $[\phi]_{\geq p}$ is true. In similar fashion, we can compute the maximum probability $Pr^{\max}_{\mathcal{M}}(\phi)$ instead of checking an

upper-bounded predicate $[\phi]_{\leq p}$ and the minimum or maximum expected reward, $ExpTot^{\min}_{\mathcal{M}}(\rho)$ or $ExpTot^{\max}_{\mathcal{M}}(\rho)$, instead of a predicate $[\rho]_{\sim r}$ over reward structure ρ.

In this section, we discuss how to formulate such numerical queries in the context of assume-guarantee reasoning. Consider, first, an assume-guarantee triple $\langle \Psi_A \rangle \, \mathcal{M}^{\star} \, \langle \Psi_G \rangle$ where $\Psi_A = [\phi_A]_{\geq p_A}$ and $\Psi_G = [\phi_G]_{\geq p_G}$ are both lower-bounded probabilistic predicates. Recall, from Definition 14, that this triple is said to be satisfied if, for any adversary (of class $\star$) of PA $\mathcal{M}[\alpha_A]$ under which assumption Ψ_A is true, guarantee Ψ_G is also true. We can instead ask "what is the *minimum* probability that ϕ_G holds, assuming that Ψ_A is true?". This can be determined with a numerical multi-objective query of the kind mentioned in Section 3.3:

$$Pr^{\min}_{\mathcal{M}[\alpha_A]}(\phi_G \downharpoonleft \Psi_A) \;=\; \inf \{ \, Pr^{\sigma}_{\mathcal{M}[\alpha_A]}(\phi_G) \mid \sigma \in Adv^{\star}_{\mathcal{M}[\alpha_A]} \wedge \mathcal{M}[\alpha_A], \sigma \models \Psi_A \, \}$$

which can be determined with essentially the same procedure that checks whether assume-guarantee triple $\langle \Psi_A \rangle \, \mathcal{M}^{\star} \, \langle \Psi_G \rangle$ is true for a fixed p_G (i.e., as described in Proposition 3).

We call expressions such as the one above *numerical assume-guarantee queries*. The example above is equivalent to the *maximum* value of p_G for which $\langle \Psi_A \rangle \, \mathcal{M}^{\star} \, \langle \Psi_G \rangle$ holds:

$$\begin{aligned} Pr^{\min}_{\mathcal{M}[\alpha_A]}(\phi_G \downharpoonleft \Psi_A) &= \inf \{ \, Pr^{\sigma}_{\mathcal{M}[\alpha_A]}(\phi_G) \mid \sigma \in Adv^{\star}_{\mathcal{M}[\alpha_A]} \wedge \mathcal{M}[\alpha_A], \sigma \models \Psi_A \, \} \\ &= \sup \{ \, p_G \in [0,1] \mid \forall \sigma \in Adv^{\star}_{\mathcal{M}[\alpha_A]} \, . \, \mathcal{M}[\alpha_A], \sigma \models \Psi_A \rightarrow \; Pr^{\sigma}_{\mathcal{M}[\alpha_A]}(\phi_G) \geq p_G \, \} \\ &= \sup \{ \, p_G \in [0,1] \mid \langle \Psi_A \rangle \, \mathcal{M}^{\star} \, \langle \Psi_G \rangle \, \} \, . \end{aligned}$$

Optimal bounds. In the rest of this section, we will illustrate how to use numerical assume-guarantee queries, such as the one above, in the context of our compositional proof rules. For simplicity, we will focus on proof rule (Asym-Safety) (see Theorem 1) in which $\Psi^{safe}_A = [\phi_A]_{\geq p_A}$ and $\Psi^{safe}_G = [\phi_G]_{\geq p_G}$ are single, lower-bounded probabilistic predicates. We can apply similar ideas for our other proof rules. For clarity, in the remainder of this section, we will abbreviate Ψ^{safe}_A and Ψ^{safe}_G to Ψ_A and Ψ_G, respectively.

Rule (Asym-Safety) allows us to verify that $[\phi_G]_{\geq p_G}$ holds in $\mathcal{M}_1 \| \mathcal{M}_2$ for some p_G, i.e., it can be used to establish *lower bounds* p_G on the probability $Pr^{\min}_{\mathcal{M}_1 \| \mathcal{M}_2}(\phi_G)$. An appropriate *numerical* interpretation of this rule would therefore be to instead determine the *maximal lower bound* p_G, say $p^{\sharp}_G$, for which $[\phi_G]_{\geq p_G}$ can be shown to hold using the (Asym-Safety) rule (assuming the use of property ϕ_A in the assumption). This can be achieved as follows. First, we note that, from Definition 14, it is clear that the highest value of p_G for which $\langle \Psi_A \rangle \, \mathcal{M}_2 \, ^{\star} \langle \Psi_G \rangle$ holds will be obtained by using the maximum possible value of p_A, which we will call $p^{\sharp}_A$. For rule (Asym-Safety) to be applicable, $p^{\sharp}_A$ is equal to $Pr^{\min}_{\mathcal{M}_1}(\phi_A)$, since any higher value of p_A will result in the first premise failing to hold.

Now, to find $p^{\sharp}_G$, we use a numerical assume-guarantee query that computes the maximum value of p_G for which $\langle \Psi_A \rangle \, \mathcal{M}_2 \, ^{\star} \langle \Psi_G \rangle$ holds, assuming that the bound p_A in Ψ_A is taken to be $p^{\sharp}_A$. Reasoning as above, this can be obtained through the multi-objective numerical query $Pr^{\min}_{\mathcal{M}_2[\alpha_A]}(\phi_G \downharpoonleft [\phi_A]_{\geq p^{\sharp}_A})$. In summary, we have a numerical interpretation of the rule (Asym-Safety) which reduces to a two-step procedure, with one instance of (standard) model checking and one of multi-objective model checking:

(i) $p^{\sharp}_A := Pr^{\min}_{\mathcal{M}_1}(\phi_A)$

(ii) $p^{\sharp}_G := Pr^{\min}_{\mathcal{M}_2[\alpha_A]}(\phi_G \downharpoonleft [\phi_A]_{\geq p^{\sharp}_A})$

Through similar reasoning, we can determine either *maximal lower bounds*, or *minimal upper bounds*, for many of the other rules presented in this paper, where the property being proved is a single quantitative predicate.

Example 6. We give an illustration of the above process by adapting the earlier Example 4, which shows how to apply rule (Asym-Safety) to verify the probabilistic safety property $[\phi_G]_{\geq 0.98}$ on the parallel composition of the PAs $\mathcal{M}_s$ and $\mathcal{M}_d$ from Example 1 (see Figure 1). Again, we let $\mathcal{M}_1 = \mathcal{M}_s$ and $\mathcal{M}_2 = \mathcal{M}_d$, and use an assumption based on regular safety property ϕ_A.

Firstly, we compute $p_A^\sharp := Pr^{\min}_{\mathcal{M}_s}(\phi_A)$, which, as shown earlier in Example 3, gives $p_A^\sharp = 0.8$. Secondly, we compute $p_G^\sharp := Pr^{\min}_{\mathcal{M}_d[\alpha_A]}(\phi_G \downharpoonright [\phi_A]_{\geq 0.8})$, which, using Lemma 1, equals $1 - Pr^{\max}_{\mathcal{M}'}(\Diamond err_G \downharpoonright [\Diamond err_A]_{\leq 0.2})$ on the product $\mathcal{M}' = \mathcal{M}_d \otimes \mathcal{A}^{err}_{\phi_A} \otimes \mathcal{A}^{err}_{\phi_G}$ shown in Figure 4. This gives $p_G^\sharp = 0.98$, which shows that the lower bound of 0.98 on the probability $Pr^{\min}_{\mathcal{M}_s \| \mathcal{M}_d}(\phi_G)$ shown in Example 4 is indeed the highest lower bound that we can obtain with assumption ϕ_A using the (Asym-Safety) rule.

Parameterised queries. Above, we found the maximum lower bound on $Pr^{\min}_{\mathcal{M}_1 \| \mathcal{M}_2}(\phi_G)$ that can be shown using an assumption of the form $[\phi_A]_{\geq p_A}$ and rule (Asym-Safety). Let us now consider the reverse problem, i.e., finding which values of p_A suffice to guarantee that $\mathcal{M}_1 \| \mathcal{M}_2 \models [\phi_G]_{\geq p_G}$ for some fixed threshold p_G. Let us further assume that component $\mathcal{M}_1$ is parameterised by a variable x in such a way that varying x changes the probability of $\mathcal{M}_1$ satisfying the property ϕ_A used in the assumption. This may in turn affect the probability of ϕ_G holding on $\mathcal{M}_1 \| \mathcal{M}_2$.

To give a simple illustration, consider the PA $\mathcal{M}_s$ from the running example (see Figure 1(a)), representing a sensor that may fail to send a *warn* message upon detection of a system failure. Let parameter x be the probability that *warn* is not sent (x=0.2 in the running example). For the assumption ϕ_A ("*warn* occurs before *shutdown*"), we then have $Pr^{\min}_{\mathcal{M}_s}(\phi_A) = 1 - x$. However, varying x may simultaneously worsen some other performance measure or cost. For example, it may only be possible to reduce the probability of a failure in the sensor by replacing it with a more reliable, but more expensive, component.

In this case, it is natural to ask for the minimum value of p_A, say $p_A^\flat$, such that we can guarantee $\mathcal{M}_1 \| \mathcal{M}_2 \models [\phi_G]_{\geq p_G}$ is true. This means determining the minimum value of p_A such that $\langle [\phi_A]_{\geq p_A} \rangle \, \mathcal{M}_2 \, ^\star \langle [\phi_G]_{\geq p_G} \rangle$ holds. This can again be computed using a numerical multi-objective query as $p_A^\flat := Pr^{\max}_{\mathcal{M}_2[\alpha_A]}(\phi_A \downharpoonright [\phi_G]_{<p_G})$ since, by definition of numerical queries:

$$
\begin{aligned}
& Pr^{\max}_{\mathcal{M}_2[\alpha_A]}(\phi_A \downharpoonright [\phi_G]_{<p_G}) \;=\; \sup \{\, Pr^{\sigma}_{\mathcal{M}_2[\alpha_A]}(\phi_A) \mid \sigma \in Adv^{\star}_{\mathcal{M}_2[\alpha_A]} \wedge Pr^{\sigma}_{\mathcal{M}_2[\alpha_A]}(\phi_G) < p_G \,\} \\
&= \inf \{\, p_A \in [0,1] \mid \forall \sigma \in Adv^{\star}_{\mathcal{M}_2[\alpha_A]} \,.\, Pr^{\sigma}_{\mathcal{M}_2[\alpha_A]}(\phi_G) < p_G \rightarrow\; Pr^{\sigma}_{\mathcal{M}_2[\alpha_A]}(\phi_A) < p_A \,\} \qquad \text{rearranging} \\
&= \inf \{\, p_A \in [0,1] \mid \forall \sigma \in Adv^{\star}_{\mathcal{M}_2[\alpha_A]} \,.\, Pr^{\sigma}_{\mathcal{M}_2[\alpha_A]}(\phi_A) \geq p_A \rightarrow\; Pr^{\sigma}_{\mathcal{M}_2[\alpha_A]}(\phi_G) \geq p_G \,\} \qquad \text{rearranging} \\
&= \inf \{\, p_A \in [0,1] \mid \langle [\phi_A]_{\geq p_A} \rangle \, \mathcal{M}_2 \, ^\star \langle [\phi_G]_{\geq p_G} \rangle \,\} \qquad \text{by Definition 14.}
\end{aligned}
$$

Recall that, to check a standard (non-numerical) assume-guarantee triple (see Proposition 3), we in fact determine whether it is *false*, by checking for the existence of an adversary σ of $\mathcal{M}_2[\alpha_A]$ that satisfies the assumption $[\phi_A]_{\geq p_A}$ but does *not* satisfy the guarantee $[\phi_G]_{\geq p_G}$. Analogously, the numerical query given above computes $p_A^\flat$ as the maximum probability of satisfying ϕ_A over all adversaries σ of $\mathcal{M}_2[\alpha_A]$ for which $[\phi_G]_{\geq p_G}$ is *not* true. We then know that, for any value of p_A that is *strictly* less than $p_A^\flat$, there *is* such an adversary σ. Hence, the assume-guarantee triple is true for any $p_A \geq p_A^\flat$ and $p_A^\flat$ is the minimum value that we can use for p_A.

Finally, having found $p_A^\flat$, we can choose a suitable value for the parameter x of PA $\mathcal{M}_1$ which ensures that the assumption $[\phi_A]_{\geq p_A^\flat}$ holds. It some cases, determining the possible values of x might be straightforward (such as in the example below); in others, we might use, for example, parametric probabilistic model checking techniques for PAs [37, 38].

Example 7. Let us return to the PAs from Example 6 but, as suggested in the text above, assume that the probability of PA $\mathcal{M}_s$ (see Figure 1(a)) moving from state s_0 to s_1 upon taking action *detect* is x, not 0.2. Thus, $Pr^{\min}_{\mathcal{M}_s}(\phi_A) = 1-x$. Let us also assume that we wish to guarantee that the full system $\mathcal{M}_s \| \mathcal{M}_d$ satisfies $[\phi_G]_{\geq 0.97}$ (note that this bound of 0.97 is lower than the bound of 0.98 checked in Example 6). Proceeding as described above, we compute $p_A^\flat := Pr^{\max}_{\mathcal{M}_d[\alpha_A]}(\phi_A \downharpoonright [\phi_G]_{<0.97})$ which yields a value of 0.7. This means that, to verify $[\phi_G]_{\geq 0.97}$ compositionally, it suffices to use the assumption $[\phi_A]_{\geq 0.7}$ and, from above, this requires that $x \geq 0.3$.

Pareto queries. We can take the idea of numerical assume-guarantee queries one step further by using the class of *Pareto* queries [33], mentioned previously in Section 3.3. Earlier in this section, we looked at two ways of investigating the values of p_A and p_G for which the assume-guarantee triple $\langle \Psi_A \rangle \, \mathcal{M}^\star \, \langle \Psi_G \rangle$ holds: we considered the largest value of p_G that we can guarantee for a given p_A and the smallest value of p_A that is required to guarantee a fixed p_G. More generally, it is useful to examine the trade-off between these two values, by determining the set of all possible values of p_A, p_G for which $\langle \Psi_A \rangle \, \mathcal{M}^\star \, \langle \Psi_G \rangle$ holds.

In fact, we are interested in maximising the value of p_G (to give us as strong a guarantee as possible) and minimising the value of p_A (to permit the use of an assumption that is as weak as possible). So, as explained below, we can study *Pareto curves* for these values. Intuitively, these are sets of pairs (p_A, p_G) such that $\langle \Psi_A \rangle \, \mathcal{M}^\star \, \langle \Psi_G \rangle$ holds, but increasing p_G any further would necessitate also increasing p_A to ensure that $\langle \Psi_A \rangle \, \mathcal{M}^\star \, \langle \Psi_G \rangle$ is still true and, conversely, decreasing p_A would require a lower value of p_G.

In a similar fashion to the numerical query used to compute $p_A^\flat$ in the section above, we actually use a query that searches for adversaries that make the assume-guarantee triple false, namely those for which $[\phi_A]_{\geq p_A}$ is true but $[\phi_G]_{\geq p_G}$ is *not* true. More precisely, we use a Pareto query which maximises the probability $Pr^\sigma_{\mathcal{M}_2[\alpha_A]}(\phi_A)$ and minimises the probability $Pr^\sigma_{\mathcal{M}_2[\alpha_A]}(\phi_G)$. The set of *achievable points*, which we denote $X_a \subseteq [0,1]^2$, contains all pairs (p_A, p_G) such that:

- there exists an adversary σ of $\mathcal{M}_2[\alpha_A]$ for which $Pr^\sigma_{\mathcal{M}_2[\alpha_A]}(\phi_A) \geq p_A$ and $Pr^\sigma_{\mathcal{M}_2[\alpha_A]}(\phi_G) \leq p_G$

and the *Pareto curve* is the biggest set $X_p \subseteq X_a$ such that, for each pair of values $(p_A, p_G) \in X_p$:

- there is no other $(p'_A, p'_G) \in X_p$ such that either $p'_A \geq p_A$ and $p'_G < p_G$, or $p'_A > p_A$ and $p'_G \leq p_G$.

We can then determine from the Pareto curve X_p the values p_A and p_G for which the assume-guarantee triple holds. More precisely, $\langle [\phi_A]_{\geq p_A} \rangle \, \mathcal{M}_2 \, ^\star \langle [\phi_G]_{\geq p_G} \rangle$ holds for (p_A, p_G) if and only if there is some pair $(p_A^-, p_G^+) \in X_p$ such that $p_A \geq p_A^-$ or $p_G \leq p_G^+$.

Example 8. We return to the compositional verification problem studied in Examples 6 and 7, and study the values of p_A and p_G for which $\langle [\phi_A]_{\geq p_A} \rangle \, \mathcal{M}_d \, ^\star \langle [\phi_G]_{\geq p_G} \rangle$ holds. Following the procedure outlined above, we obtain the Pareto curve which maximises $Pr^\sigma_{\mathcal{M}_d[\alpha_A]}(\phi_A)$ and minimises $Pr^\sigma_{\mathcal{M}_d[\alpha_A]}(\phi_G)$. This is shown in Figure 6. The set X_a of achievable points is shown shaded grey and the black dashed line represents the Pareto curve X_p.

Thus, for each point $(p_A, p_G) \in X_p$, there is an adversary of $\mathcal{M}_d[\alpha_A]$ for which $p_A = Pr^\sigma_{\mathcal{M}_d[\alpha_A]}(\phi_A)$ and $p_G = Pr^\sigma_{\mathcal{M}_d[\alpha_A]}(\phi_G)$. For this simple example, we can find such adversaries

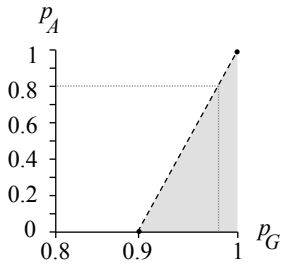


Figure 6: Pareto curve to show the trade-off between satisfying an assumption and a guarantee (see Example 8).

by inspection of the PA $\mathcal{M}_d[\alpha_A]$. Recall that, in this example, $\alpha_A = \{warn, shutdown\} \subseteq \alpha_{\mathcal{M}_d}$, so $\mathcal{M}_d[\alpha_A]$ is identical to $\mathcal{M}_d$, which can be seen in Figure 1(b). Consider the set of (randomised) adversaries σ_y (for $y \in [0, 1]$) which, in state s_0, choose action *warn* with probability y and action *shutdown* with probability $1 - y$. For adversary σ_y, this results in a probability of y for ϕ_A and $1 - 0.1{\cdot}(1{-}y) = 0.9 + 0.1{\cdot}y$ for ϕ_G. Varying $y \in [0, 1]$ gives the Pareto curve.

From Figure 6, we can conclude that $\langle [\phi_A]_{\geq p_A} \rangle \, \mathcal{M}_d \, {}^{\star} \langle [\phi_G]_{\geq p_G} \rangle$ holds for any values of p_A, p_G such that $p_G \leq 0.9 + 0.1{\cdot}p_A$. Notice that the Pareto curve allows us to determine values for the numerical queries used in Examples 6 and 7 (without any further computation). For example, if we know that $p_A \leq 0.8$ (as in Example 6), then the maximum value of p_G for which we can show $[\phi_G]_{\geq p_G}$ to hold is $0.9 + 0.1{\cdot}0.8 = 0.98$ (see also the grey dotted line in Figure 6). Alternatively, if we want $p_G = 0.97$ (as in Example 7), then we must have $p_A \geq (0.97 - 0.9)/0.1 = 0.7$.

7. Implementation and Case Studies

We have implemented our compositional verification approach by extending the probabilistic model checker PRISM [4]. Recall that, using the rules presented in Sections 4 and 5, verification requires both (standard) model checking of ω-regular or reward-based properties and multi-objective model checking. PRISM already provided support for verifying probabilistic automata against the logic LTL and expected total reward properties. Furthermore, it can easily verify the action-based safety or ω-regular properties used in this paper, by encoding the required deterministic finite or Rabin automata directly in PRISM's modelling language. For multi-objective model checking, we extended PRISM with an implementation of the techniques in [14, 15]. 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.

This section evaluates the performance and scalability of our techniques on two large case studies: the randomised consensus algorithm of Aspnes & Herlihy [39]; and a model of the Zeroconf protocol [29]. All models and properties are available online [40]. This includes the automata that are needed to form the assumptions used in our compositional verification techniques. Currently, all assumptions are constructed manually, based on user knowledge of the models, their structure and the properties that are being checked. An important direction of future work in this area is the automatic generation of such counterexamples. Some progress has already been made on this topic, using algorithmic learning to generate probabilistic safety properties to be used as assumptions [19, 20].

The next two sections describe in more detail how we verified the two case studies compositionally; in the following section, we summarise the experimental results.

7.1. Aspnes & Herlihy's Randomised Consensus Algorithm

The first case study we consider is the randomised consensus algorithm of Aspnes & Herlihy [39]. The algorithm allows N processes in a distributed network to reach a consensus and employs, in each round, a shared coin protocol parameterised by a constant K and the number of processes N. Each shared coin protocol is based on a random walk where the boundaries of the walk are derived from the values of K and N.

The model of the algorithm used here is based on [41] and comprises one PA for each process and one for the shared coin protocol used in each round. Although the number of rounds of the protocol can be unbounded, the properties that we check here relate only to the behaviour within a finite number of rounds R. The combined PA is:

$$(P_1 \,||\, P_2 \,||\, \cdots \,||\, P_N) \,||\, (SCP_1 \,|||\, SCP_2 \,||| \cdots |||\, SCP_R)$$

where P_i is the PA corresponding to process i, SCP_j is the PA corresponding to the shared coin protocol used in round j and we write $|||$ to denote parallel composition between PAs with disjoint alphabets. Although each shared chain protocol SCP_j does interact with the N processes, there is no communication between the protocols themselves, i.e., the shared coin protocols for each round are independent. This lets us use the asynchronous proof rules introduced in Section 5.2.

We analyse the following two properties:

(i) the minimum probability that the processes decide by round R;

(ii) the maximum expected number of steps required in the first R rounds.

We adopt the numerical approach described in Section 6, computing lower/upper bounds on these values, rather than proving that they are above or below a given threshold.

Property (i) can be represented by a probabilistic safety property. We use the rule (Asym-Safety), in conjunction with the rule (Async-Safety), to establish the required assumptions. More precisely, compositional verification is achieved by:

- calculating the minimum probability that the coin protocols in earlier rounds return the same coin value for all processes, and then combining these results using rule (Async-Safety) to prove a probabilistic safety property satisfied by the (asynchronous) composition of the coin protocols;
- using this probabilistic safety property as the assumption for an application of the (Asym-Safety) rule, yielding the final property of interest on the combined system, namely, the minimum probability that agreement is reached by round R.

Compositional verification of property (ii) is performed by:

- splitting the reward structure for the property into several parts (one for the N processes and one for each coin protocol) and then using rule (Async-Quant) to determine an upper bound on the maximum total expected value for each one;
- combining these results, using the (Sum-Reward) rule, to compute an upper bound on the maximum total reward.

7.2. The Zeroconf Protocol

The second case study is the Zeroconf protocol [42], for configuring IP addresses in a local network. We use the model from [29], composed of two PAs: one representing a new host joining the network and the second representing the environment, i.e. the existing network. The model is parameterised by K, the number of messages ("probes") that the new host sends before using its chosen IP address. We consider the following four properties:

(i) the minimum probability that the new host employs a fresh IP address;

(ii) the minimum probability that the new host is configured by time T;

(iii) the minimum probability that the protocol terminates;

(iv) the minimum and maximum expected time for the protocol to terminate.

Properties (i) and (ii) are probabilistic safety properties, which are verified compositionally by:

- applying the rule (Circ-Safety), where the first and second assumptions concern the new host and environment, respectively, and the first assumption is a qualitative (probability 1) safety property.

For properties (iii) and (iv), verification is performed by:

- applying the rule (Circ-Quant), where the first assumption is a either a qualitative safety or qualitative liveness property, about the new host, and the second assumption is a probabilistic ω-regular property, concerning the environment.

7.3. Experimental Results

In Tables 1 and 2, we summarise the experimental results for the randomised consensus and Zeroconf case studies, respectively. Experiments were run on a 2.8GHz PC with 8GB RAM. Any run exceeding a time-limit of 6 hours was disregarded. The tables show the time taken for verification, performed both compositionally and non-compositionally. For the former, we proceed as described as above, using PRISM to check each individual model checking problem. For the latter, we also use PRISM, selecting its fastest available model checking engine. To give an indication of the scale of the verification tasks performed, the tables also show, for the non-compositional case, the size (number of states) of the PA for the composed system and, for the compositional case, the size (number of variables) of the largest LP problem solved for multi-objective model checking. Finally, the table includes the (numerical) results obtained from verification. For probabilistic safety properties, we show the maximum probabilities of violating the property, so the actual values are these subtracted from 1.

One the whole, compositional verification performs very well. For the randomised consensus models (Table 1), on all but the smallest examples, it is faster than the non-compositional case, often significantly so, and is able to scale up to larger models. In particular, this allows the verification of several models for which it is infeasible with conventional techniques (those marked with "mem-out" in the table). For the Zeroconf example (Table 2), we again see improved times for the compositional approach on the first property. For the other properties, the times for the two approaches are closer, but non-compositional verification is slightly faster. Encouragingly,

Model (Property) [parameters]			Non-compositional			Compositional		
			States	Time (s)	Result	LP size	Time (s)	Result
2 processes (min. probability decide) [*R K*]	3	2	5,158	0.6	0.108333	1,427	0.9	0.108333†
	3	20	40,294	42.2	0.012500	1,427	3.9	0.012500†
	4	2	20,886	1.4	0.011736	3,217	2.9	0.011736†
	4	20	166,614	129.1	0.000156	3,217	4.4	0.000156†
	5	2	83,798	2.9	0.001271	6,797	1.4	0.001271†
	5	20	671,894	472.5	0.000002	6,797	5.9	0.000002†
3 processes (min. probability decide) [*R K*]	3	2	1,418,545	3,589.0	0.229092	49,113	23.5	0.229092†
	3	12	16,674,145*	time-out	-	49,113	34.1	0.041643†
	3	20	39,827,233*	time-out	-	49,113	66.9	0.024960†
	4	2	150,487,585	20,372	0.052483	174,193	200.1	0.052483†
	4	12	1,053,762,385*	mem-out	-	174,193	210.6	0.001734†
	4	20	2,028,200,209*	mem-out	-	174,193	243.4	0.000623†
2 processes (max. expected steps) [*R K*]	3	2	1,806	0.2	89.00	974	1.2	89.65
	3	20	11,598	16.6	5,057	5,775	3.3	5,057
	4	2	7,478	0.7	89.00	2,184	3.1	98.42
	4	20	51,830	84.0	5,057	5,775	11.6	5,120
	5	2	30,166	1.7	89.00	4,604	4.9	100.1
	5	20	212,758	348.0	5,057	5,775	15.6	5,121
3 processes (max. expected steps) [*R K*]	3	2	114,559	12.9	212.0	10,294	9.4	214.3
	3	12	507,919	865.9	4,352	59,254	210.1	4,352
	3	20	822,607*	3,396.0	11,552	98,422	650.2	11,552
	4	2	3,669,649	406.5	212.0	82.9	118.9	260.3
	4	12	29,797,249*	mem-out	-	116,658	475.1	4,533
	4	20	65,629,249*	mem-out	-	116,658	1,231.9	11,840

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

† Results for probabilistic safety properties are shown as maximum probabilities of error so actual values are these subtracted from 1.

Table 1: Experimental results for the randomised consensus case study

the times for compositional verification tend to grow more slowly with model size. Since the time required for this is usually dominated by the time to solve LP problems, we anticipate better performance through enhancements to the underlying LP solver and optimisations for our implementation that will reduce LP problem sizes. In fact, LP solution represents the limiting factor with respect to the sizes of models that our techniques can be applied to, so such improvements will also help improve the scalability of compositional verification.

Lastly we also comment on the results obtained from assume-guarantee verification. As mentioned earlier, we use the approach discussed in Section 6 to obtain numerical values for each property of interest, representing a lower or upper bound on the actual value of the property. We observe that the bounds obtained using our assume-guarantee approach are generally quite precise. In fact, for several properties, the bounds are tight, matching the actual values exactly.

8. Conclusions

We have presented new techniques for compositional verification of probabilistic automata, based on the assume-guarantee paradigm. Our techniques can be used to verify probabilistic ω-regular properties (including the special case of probabilistic safety properties) and expected total reward properties. The key novelty of our approach is the use of multi-objective model checking, for which efficient techniques exist. We have presented a variety of assume-guarantee proof rules and also discussed how to formulate and evaluate numerical verification queries in

Property [parameters]		Non-compositional			Compositional		
		States	Time (s)	Result	LP size	Time (s)	Result
min. probability fresh [K]	2	91,041	7.1	2.0e-5	4,216	3.1	3.1e-4[†]
	4	313,541	21.1	7.3e-7	13,103	6.4	3.1e-4[†]
	6	811,290	58.6	2.6e-8	24,928	16.0	2.5e-4[†]
	8	1,892,952	141.5	9.5e-10	41,224	31.8	9.0e-6[†]
min. probability configured by T [K T]	2 10	665,567	14.0	5.9e-5	38,759	20.0	2.1e-4[†]
	2 14	1,061,771	21.3	2.0e-8	63,188	32.4	8.1e-8[†]
	4 10	976,247	29.6	3.3e+0	47,577	34.3	3.3e-1[†]
	4 14	2,288,771	49.6	7.0e-5	105,685	74.3	3.1e-4[†]
min. probability terminate [K]	2	13,474	1.4	1.0	151,503	15.3	1.0
	4	57,960	5.3	1.0	151,503	15.6	1.0
	6	125,697	9.5	1.0	151,503	16.1	1.0
	8	163,229	10.7	1.0	151,503	16.6	1.0
min. expected time [K]	2	13,474	1.0	9.419	151,503	14.9	8.90
	4	57,960	3.5	13.49	151,503	15.1	16.90
	6	125,697	7.4	17.49	151,503	15.2	12.90
	8	163,229	10.6	21.49	151,503	15.5	20.90
max. expected time [K]	2	13,474	0.9	10.22	151,503	15.2	12.00
	4	57,960	3.0	14.28	151,503	15.5	17.33
	6	125,697	6.8	18.28	151,503	15.9	22.67
	8	163,229	9.9	22.28	151,503	16.3	28.00

[†] Results for probabilistic safety properties are shown as maximum probabilities of error so actual values are these subtracted from 1.

Table 2: Experimental results for Zeroconf case study

a compositional manner. 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. For the fragment of our framework that uses probabilistic safety properties, algorithmic learning techniques have already been developed to automatically produce the assumptions required for compositional reasoning [19, 20]. This work adapts the L* algorithm for learning regular languages to the problem of synthesising assumptions expressed as probabilistic safety properties. It would be interesting to extend these methods to the more general class of properties considered in this paper, perhaps using techniques for learning of ω-automata [43, 44].

Another topic to investigate is that of completeness. Some assume-guarantee frameworks are *complete* in the sense that, if a property of a composed system is true, then there must exist an assumption that can be used to verify it compositionally. For example, this is trivially true for frameworks in which assumptions are expressed in the same manner as the models themselves: then, completeness can be shown by using the component itself as the assumption that represents it. In our approach, the formalisms for components (probabilistic automata) and assumptions (qmo-properties) are distinct, so this argument is not applicable. We would like to investigate for which classes of models and properties our framework *can* be shown to be complete.

Finally, we would also like to consider assume-guarantee techniques for richer classes of models such as probabilistic timed automata and continuous-time variants of probabilistic automata, such as interactive Markov chains [45] or Markov automata [46]. In order to adapt the multi-objective model checking approach used in this paper, the first step will be to develop efficient multi-objective techniques for timed properties of such models.

Acknowledgements

This work was part funded by ERC Advanced Grant VERIWARE, EPSRC grant EP/F001096/1 and European Commission FP7 projects CONNECT (IST 231167) and HIERATIC (316705). The authors are also grateful to the anonymous referees for their helpful comments.

References

[1] A. Bianco, L. de Alfaro, Model checking of probabilistic and nondeterministic systems, in: P. Thiagarajan (Ed.), Proc. 15th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'95), volume 1026 of *LNCS*, Springer, 1995, pp. 499–513.

[2] C. Courcoubetis, M. Yannakakis, Markov decision processes and regular events, in: M. Paterson (Ed.), Proc. 17th International Colloquium on Automata, Languages and Programming (ICALP'90), volume 443 of *LNCS*, Springer, 1990, pp. 336–349.

[3] L. de Alfaro, Formal Verification of Probabilistic Systems, Ph.D. thesis, Stanford University, 1997.

[4] M. Kwiatkowska, G. Norman, D. Parker, PRISM 4.0: Verification of probabilistic real-time systems, in: G. Gopalakrishnan, S. Qadeer (Eds.), Proc. 23rd International Conference on Computer Aided Verification (CAV'11), volume 6806 of *LNCS*, Springer, 2011, pp. 585–591.

[5] F. Ciesinski, C. Baier, Liquor: A tool for qualitative and quantitative linear time analysis of reactive systems, in: Proc. 3rd International Conference on Quantitative Evaluation of Systems (QEST'06), IEEE CS Press, 2006, pp. 131–132.

[6] C. Jones, Tentative steps towards a development method for interfering programs, ACM Transactions on Programming Languages and Systems 5 (1983) 596–619.

[7] C. Pasareanu, D. Giannakopoulou, M. Bobaru, J. Cobleigh, H. Barringer, Learning to divide and conquer: Applying the L* algorithm to automate assume-guarantee reasoning, Formal Methods in System Design 32 (2008) 175–205.

[8] R. Segala, Modelling and Verification of Randomized Distributed Real Time Systems, Ph.D. thesis, Massachusetts Institute of Technology, 1995.

[9] R. Segala, N. Lynch, Probabilistic simulations for probabilistic processes, Nordic Journal of Computing 2 (1995) 250–273.

[10] A. Pogosyants, R. Segala, N. Lynch, Verification of the randomized consensus algorithm of Aspnes and Herlihy: A case study, Distributed Computing 13 (2000) 155–186.

[11] R. Segala, A compositional trace-based semantics for probabilistic automata, in: I. Lee, S. Smolka (Eds.), Proc. 6th International Conference on Concurrency Theory (CONCUR'95), volume 962 of *LNCS*, Springer, 1995, pp. 234–248.

[12] L. de Alfaro, T. Henzinger, R. Jhala, Compositional methods for probabilistic systems, in: K. Larsen, M. Nielsen (Eds.), Proc. 12th International Conference on Concurrency Theory (CONCUR'01), volume 2154 of *LNCS*, Springer, 2001, pp. 351–365.

[13] L. Cheung, N. Lynch, R. Segala, F. Vaandrager, Switched probabilistic I/O automata, in: Proc. 1st International Colloquium on Theoretical Aspects of Computing (ICTAC'04), volume 3407 of *LNCS*, Springer, 2004, pp. 494–510.

[14] K. Etessami, M. Kwiatkowska, M. Vardi, M. Yannakakis, Multi-objective model checking of Markov decision processes, Logical Methods in Computer Science 4 (2008) 1–21.

[15] V. Forejt, M. Kwiatkowska, G. Norman, D. Parker, H. Qu, Quantitative multi-objective verification for probabilistic systems, in: P. Abdulla, K. Leino (Eds.), Proc. 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'11), volume 6605 of *LNCS*, Springer, 2011, pp. 112–127.

[16] R. Alur, T. Henzinger, Reactive modules, in: Proc. 11th Annual IEEE Symposium on Logic in Computer Science (LICS'96), IEEE Computer Society Press, 1996, pp. 207–218.

[17] M. Kwiatkowska, G. Norman, D. Parker, H. Qu, Assume-guarantee verification for probabilistic systems, in: J. Esparza, R. Majumdar (Eds.), Proc. 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'10), volume 6105 of *LNCS*, Springer, 2010, pp. 23–37.

[18] K. Etessami, M. Kwiatkowska, M. Vardi, M. Yannakakis, Multi-objective model checking of Markov decision processes, in: O. Grumberg, M. Huth (Eds.), Proc. 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'07), volume 4424 of *LNCS*, Springer, 2007, pp. 50–65.

[19] L. Feng, M. Kwiatkowska, D. Parker, Compositional verification of probabilistic systems using learning, in: Proc. 7th International Conference on Quantitative Evaluation of SysTems (QEST'10), IEEE CS Press, 2010, pp. 133–142.

[20] L. Feng, M. Kwiatkowska, D. Parker, Automated learning of probabilistic assumptions for compositional reasoning, in: D. Giannakopoulou, F. Orejas (Eds.), Proc. 14th International Conference on Fundamental Approaches to Software Engineering (FASE'11), volume 6603 of *LNCS*, Springer, 2011, pp. 2–17.

[21] A. Komuravelli, C. Pasareanu, E. Clarke, Assume-guarantee abstraction refinement for probabilistic systems, in: P. Madhusudan, S. Seshia (Eds.), Proc. 24th International Conference on Computer Aided Verification (CAV'12), volume 7358 of *LNCS*, Springer, 2012, pp. 310–326.

[22] J. A. Kumar, S. Vasudevan, Automatic compositional reasoning for probabilistic model checking of hardware designs, in: Proc. 7th International Conference on Quantitative Evaluation of SysTems (QEST'10), IEEE CS Press, 2010, pp. 143–152.

[23] B. Delahaye, B. Caillaud, A. Legay, Probabilistic contracts: A compositional reasoning methodology for the design of stochastic systems, in: Proc. 10th International Conference on Application of Concurrency to System Design (ACSD'10), IEEE CS Press, 2010, pp. 223–232.

[24] K. Chatterjee, L. de Alfaro, M. Faella, T. Henzinger, R. Majumdar, M. Stoelinga, Compositional quantitative reasoning, in: Proc. 3rd International Conference on Quantitative Evaluation of Systems (QEST'06), IEEE CS Press, 2006, pp. 179–188.

[25] A. David, K. G. Larsen, A. Legay, U. Nyman, A. Wasowski, ECDAR: An environment for compositional design and analysis of real time systems, in: A. Bouajjani, W.-N. Chin (Eds.), Proc. 8th International Symposium on Automated Technology for Verification and Analysis (ATVA'10), volume 6252 of *LNCS*, Springer, 2010, pp. 365–370.

[26] J. Kemeny, J. Snell, A. Knapp, Denumerable Markov Chains, Springer-Verlag, 2nd edition, 1976.

[27] W. Thomas, Automata on infinite objects, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, volume B: Formal Models and Semantics, Elsevier, 1990, pp. 133–192.

[28] V. Forejt, M. Kwiatkowska, G. Norman, D. Parker, Automated verification techniques for probabilistic systems, in: M. Bernardo, V. Issarny (Eds.), Formal Methods for Eternal Networked Software Systems (SFM'11), volume 6659 of *LNCS*, Springer, 2011, pp. 53–113.

[29] M. Kwiatkowska, G. Norman, D. Parker, J. Sproston, Performance analysis of probabilistic timed automata using digital clocks, Formal Methods in System Design 29 (2006) 33–78.

[30] C. Courcoubetis, M. Yannakakis, Markov decision processes and regular events, IEEE Transactions on Automatic Control 43 (1998) 1399–1418.

[31] M. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, John Wiley and Sons, 1994.

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

[33] V. Forejt, M. Kwiatkowska, D. Parker, Pareto curves for probabilistic model checking, in: S. Chakraborty, M. Mukund (Eds.), Proc. 10th International Symposium on Automated Technology for Verification and Analysis (ATVA'12), volume 7561 of *LNCS*, Springer, 2012, pp. 317–332.

[34] C. Baier, M. Kwiatkowska, Model checking for a probabilistic branching time logic with fairness, Distributed Computing 11 (1998) 125–155.

[35] C. Baier, M. Größer, F. Ciesinski, Quantitative analysis under fairness constraints, in: Proc. 7th International Symposium on Automated Technology for Verification and Analysis (ATVA'09), volume 5799 of *LNCS*, Springer, 2009, pp. 135–150.

[36] H. Hansen, M. Kwiatkowska, H. Qu, Partial order reduction for model checking Markov decision processes under unconditional fairness, in: Proc. 8th International Conference on Quantitative Evaluation of SysTems (QEST'11), IEEE CS Press, 2011, pp. 203–212.

[37] E. M. Hahn, H. Hermanns, L. Zhang, Probabilistic reachability for parametric Markov models, International Journal on Software Tools for Technology Transfer (STTT) 13 (2011) 3–19.

[38] T. Chen, E. M. Hahn, T. Han, M. Kwiatkowska, H. Qu, L. Zhang, Model repair for Markov decision processes, in: Proc. 7th International Symposium on Theoretical Aspects of Software Engineering (TASE'13), IEEE, 2013, pp. 85–92.

[39] J. Aspnes, M. Herlihy, Fast randomized consensus using shared memory, Journal of Algorithms 15 (1990) 441–460.

[40] 2013. http://www.prismmodelchecker.org/files/ic-probag/.

[41] M. Kwiatkowska, G. Norman, R. Segala, Automated verification of a randomized distributed consensus protocol using Cadence SMV and PRISM, in: G. Berry, H. Comon, A. Finkel (Eds.), Proc. 13th International Conference on Computer Aided Verification (CAV'01), volume 2102 of *LNCS*, Springer, 2001, pp. 194–206.

[42] S. Cheshire, B. Adoba, E. Gutterman, Dynamic configuration of IPv4 link local addresses, 2005. Available from http://www.ietf.org/rfc/rfc3927.txt.

[43] A. Farzan, Y.-F. Chen, E. Clarke, Y.-K. Tsay, B.-Y. Wang, Extending automated compositional verification to the full class of omega-regular languages, in: Proc. 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'08), volume 4963 of *LNCS*, Springer, 2008, pp. 2–17.

[44] S. Chaki, A. Gurfinkel, Automated assume-guarantee reasoning for omega-regular systems and specifications, in: Proc. 2nd NASA Formal Methods Symposium (NFM'10), pp. 57–66.

[45] H. Hermanns, Interactive Markov Chains and the Quest for Quantified Quality, volume 2428 of *LNCS*, Springer Verlag, 2002.

[46] C. Eisentraut, H. Hermanns, L. Zhang, On probabilistic automata in continuous time, in: Proc. 25th Annual IEEE Symposium on Logic in Computer Science (LICS'10), IEEE Computer Society, 2010, pp. 342–351.