A Framework for Supervisory Control of Probabilistic Discrete Event Systems

Vera Pantelic, Mark Lawford and Steven Postma

McMaster Centre for Software Certification, Department of Computing and Software, Faculty of Engineering, McMaster University, 1280 Main Street West, Hamilton, ON, Canada L8S 4K1 (e-mail: pantelv at mcmaster dot ca, lawford at mcmaster dot ca, steven.m.postma at gmail dot com)

Abstract: This paper focuses on a framework for probabilistic supervisory control of probabilistic discrete event systems (PDES). PDES are modelled as generators of probabilistic languages, and the supervisors used are probabilistic. In our previous work, we presented and solved a number of supervisory control problems inside the framework. We also suggested a pseudometric to measure the behavioural similarity between PDES, and used the pseudometric in the solution of two optimal supervisory control problems defined in the framework. In this paper, we survey these results and introduce a real-world application of the framework. Further, we investigate a relationship between our framework and that of Markov Decision Processes, that could prove beneficial for both control synthesis and probabilistic model checking.

Keywords: Supervisory control, stochastic systems, discrete event systems, metrics, MDPs.

1. INTRODUCTION

A framework for supervisory control of *probabilistic discrete event systems* (PDES) was developed in our previous work (Lawford and Wonham, 1993; Postma and Lawford, 2004; Pantelic et al., 2009; Pantelic and Lawford, 2010, 2009; Pantelic, 2011; Pantelic and Lawford, 2012a,b). This paper integrates the precursory work with novel results pointing to important avenues for future research.

In the framework, PDES are modelled as *probabilistic generators*: an extension of regular generators used in supervisory control theory. More precisely, every transition in the regular generator is extended with its probability of occurrence: the probabilities of all the transitions from a state should not be greater than 1. The control used in our framework is probabilistic: instead of only being able to enable/disable a controllable event, the probabilistic supervisor enables a controllable event with a certain probability. First, Lawford and Wonham (1993), Postma and Lawford (2004) and Pantelic et al. (2009) focused on the solution of the Probabilistic Supervisory Control Problem (PSCP). The PSCP tries to find a probabilistic supervisor such that the plant's behaviour under probabilistic control satisfies a given probabilistic specification. The solution gives necessary and sufficient conditions for the existence of a probabilistic supervisor, and, if the conditions are satisfied, synthesizes the supervisor. Then, in Pantelic and Lawford (2009) and Pantelic and Lawford (2012a), an optimal supervisory control problem inside the probabilistic framework is posed and solved. As in classical supervisory theory, if there does not exist a (probabilistic) supervisor such that the controlled plant's behaviour can exactly match a prespecified probabilistic behaviour, a supervisor is synthesized such that the controlled plant's behaviour is "as close as possible" to the desired behaviour. The measure of similarity is a *pseudometric* on states of probabilistic generators. The concept of the pseudometric is useful outside the control framework: the pseudometric measures behavioural similarity between probabilistic generators, and can be used for e.g., model reduction as explored in Pantelic and Lawford (2012b).

The contributions of this paper are as following. First, we survey the previous results in the framework. Then, an example of an application of the framework is presented that has not been published anywhere except in the Ph.D. thesis of the first author (Pantelic, 2011). Then, we focus on *Markov Decision Processes* (MDPs), a widely used framework for control of probabilistic discrete event systems, with the goal of exploring the relationship between our framework and MDPs. Initial results on the relationship were presented in the Ph.D. thesis of the first author (Pantelic, 2011). We show that a probabilistic generator can be viewed as a (probabilistic) policy for MDPs (see Pantelic (2011)). On the other hand, a probabilistic supervisor as defined in our framework can be represented as an MDP. This duality between the plant to be controlled and the controller might provide interesting connections between probabilistic model checking and supervisory control theory. Therefore, the duality might pave the road for the exchange of results between the two frameworks.

The outline of the paper is as following. Section 2 presents the main results in the framework. More precisely, some previous results are summarized (solutions of probabilistic matching and optimal control problems), and a new, real-world application is presented. Then, Section 3 discusses the interplay between the framework and MDPs. Section 4 concludes the paper and sketches future work.

2. THE FRAMEWORK

This section introduces probabilistic generators as the model for PDES. Then, probabilistic control is defined, and some previous results are presented: control problems in the framework and their solutions, and a pseudometric that measures the behavioural similarity of probabilistic generators. Also, an application of the results is presented.

In the sequel, for given sets A and B, the power set of A will be denoted by $\mathcal{P}(A)$, and the set difference of A and B by $A\backslash B$. Further, B^A will be used to denote a set of functions from A to B.

2.1 Probabilistic Generators: The Model

In our framework, a PDES is modeled following Lawford and Wonham (1993) as a *probabilistic generator* $G = (Q, \Sigma, \delta, q_0, p)$, where Q is the nonempty finite set of states, Σ is a finite alphabet whose elements we will refer to as event labels, $\delta : Q \times \Sigma \to Q$ is the (partial) transition function, $q_0 \in Q$ is the initial state, and $p : Q \times \Sigma \to [0, 1]$ is the statewise event probability distribution, i.e. for any $q \in Q$, $\sum_{\sigma\in\Sigma} p(q, \sigma) \leq 1$. The probability that the event $\sigma \in \Sigma$ is going to occur at the state $q \in Q$ is $p(q, \sigma)$. For the generator G to be well-defined, $p(q, \sigma) = 0$ should hold if and only if $\delta(q, \sigma)$ is undefined. The probabilistic generator G is nonterminating if, for every reachable state $q \in Q$, $\sum_{\sigma\in\Sigma} p(q, \sigma) = 1$. Conversely, G is terminating if there is at least one reachable state $q \in Q$ such that $\sum_{\sigma\in\Sigma} p(q, \sigma) < 1$. The probability that the system terminates at state q is $1 - \sum_{\sigma\in\Sigma} p(q, \sigma)$. Throughout the sequel, unless stated otherwise, we assume nonterminating generators. If a PDES is terminating, it can easily be transformed into a nonterminating one using the technique described in Lawford and Wonham (1993).

The transition function is traditionally extended by induction on the length of strings to $\delta : Q \times \Sigma^* \to Q$ in a natural way. For a state q, and a string s, the expression $\delta(q, s)!$ will denote that δ is defined for the string s in the state q. The language $L(G)$ generated by a probabilistic DES generator $G = (Q, \Sigma, \delta, q_0, p)$ is $L(G) = \{s \in \Sigma^* \,|\, \delta(q_0, s)!\}$. The probabilistic language generated by G is defined as:

$$L_p(G)(\epsilon) = 1,$$

$$L_p(G)(s\sigma) = \begin{cases} L_p(G)(s) \cdot p(\delta(q_0, s), \sigma), & \text{if } \delta(q_0, s)! \\ 0, & \text{otherwise.} \end{cases}$$

Informally, $L_p(G)(s)$ is the probability that the string s is executed in G. Also, $L_p(G)(s) > 0$ iff $s \in L(G)$.

For each state $q \in Q$, we define the function $\rho_q : \Sigma \times Q \to [0, 1]$ such that for any $q' \in Q$, $\sigma \in \Sigma$, we have $\rho_q(\sigma, q') = p(q, \sigma)$ if $q' = \delta(q, \sigma)$, and 0 otherwise. The function ρ_q is a probability distribution on the set $\Sigma \times Q$ induced by q. Also, for a state q, we define *the set of possible events* to be $Pos(q) := \{\sigma \in \Sigma | p(q, \sigma) > 0\}$.

Next, the synchronous product of (nonprobabilistic) discrete event systems (DES) that underlie PDES is defined in a standard manner. For a probabilistic generator $G = (Q, \Sigma, \delta, q_0, p)$, the (nonprobabilistic) discrete event system (DES) that underlies G will be denoted G^{np}, i.e., $G^{np} = (Q, \Sigma, \delta, q_0)$ throughout this paper. Let G_1^{np} and G_2^{np} be the nonprobabilistic generators (DES) underlying $G_1 = (Q_1, \Sigma, \delta_1, q_{0_1}, p_1)$ and $G_2 = (Q_2, \Sigma, \delta_2, q_{0_2}, p_2)$, respectively, i.e., $G_1^{np} = (Q_1, \Sigma, \delta_1, q_{0_1})$ and $G_2^{np} = (Q_2, \Sigma, \delta_2, q_{0_2})$.

Definition 1. The synchronous product of $G_1^{np} = (Q_1, \Sigma, \delta_1, q_{0_1})$ and $G_2^{np} = (Q_2, \Sigma, \delta_2, q_{0_2})$, denoted $G_1^{np} \parallel G_2^{np}$, is the reachable sub-DES of DES $G_a = (Q_a, \Sigma, \delta, q_0)$, where $Q_a = Q_1 \times Q_2$, $q_0 = (q_{0_1}, q_{0_2})$, and, for any $\sigma \in \Sigma$, $q_i \in Q_i$, $i = 1, 2$, it holds that $\delta((q_1, q_2), \sigma) = (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma))$ whenever $\delta_1(q_1, \sigma)!$ and $\delta_2(q_2, \sigma)!$.

2.2 Probabilistic Supervisory Control of PDES

The set Σ is partitioned into the uncontrollable event set Σ_u and the controllable event set Σ_c. Deterministic supervisors for DES are generalized to probabilistic supervisors. Instead of deterministically enabling or disabling controllable events, probabilistic supervisors enable them with certain probabilities. This means that, upon reaching a certain state q, the control pattern is chosen according to supervisor's probability distributions of controllable events. Consequently, the controller does not always enable the same events when in the state q.

Definition 2. Let $x : L(G) \to [0, 1]^{\Sigma_c}$. For a PDES $G = (Q, \Sigma, \delta, q_0, p)$, a *probabilistic supervisor* is a function $V_p : L(G) \to [0, 1]^{\Sigma}$ such that

$$(\forall s \in L(G))(\forall \sigma \in \Sigma)V_p(s)(\sigma) = \begin{cases} 1, & \text{if } \sigma \in \Sigma_u \\ x(s)(\sigma), & \text{otherwise.} \end{cases}$$

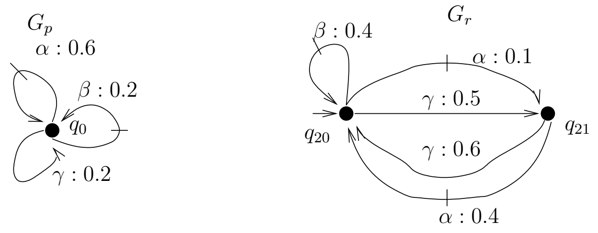

Fig. 1. Plant G_p and requirements specification G_r

Therefore, after observing a string $s \in L(G)$ (all the events are assumed to be observable), the supervisor enables event σ with probability $V_p(s)(\sigma)$. More precisely, for event σ, the supervisor performs a Bernoulli trial with possible outcomes *enable* (that has the probability $V_p(s)(\sigma)$), and *disable* (with probability $1 - V_p(s)(\sigma)$), and, depending on the outcome of the trial, decides whether to enable or disable the event. After (independent) Bernoulli trials have been performed for all controllable events, control pattern Θ is determined as a set of controllable events such that a controllable event belongs to Θ if and only if its corresponding Bernoulli trial resulted in outcome *enable.* Thus the *controllable event set probability* of Θ, i.e., the probability that V_p enables the controllable event in Θ after observing string s is given by:

$$P(V_p \text{ enables } \Theta|s) = \prod_{\sigma\in\Theta} V_p(s)(\sigma) \quad \cdot \prod_{\sigma\in(Pos(q)\cap\Sigma_c)\backslash\Theta} (1 - V_p(s)(\sigma)) \qquad (1)$$

After Θ has been decided upon, the system acts as if supervised by a deterministic supervisor. Let $q \in Q$ be the state of the plant after $s \in L(G)$ has been observed. The plant G under the control of the supervisor V_p will be denoted V_p/G. The probability that the event $\alpha \in \Sigma$ will occur in the controlled plant V_p/G after string s has been observed is equal to:

$$P(\alpha \text{ in } V_p/G|s) \tag{2}$$
$$= \sum_{\Theta\in\mathcal{P}(Pos(q)\cap\Sigma_c)} P(\alpha|V_p \text{ enables } \Theta \text{ after } s)\cdot P(V_p \text{ enables } \Theta|s)$$

where

$$P(\alpha|V_p \text{ enables } \Theta \text{ after } s)$$
$$= \begin{cases} \dfrac{p(q,\alpha)}{\sum\limits_{\sigma\in\Theta\cup\Sigma_u} p(q,\sigma)} & \text{if } \alpha\in\Theta\cup\Sigma_u \\ 0 & \text{otherwise.}\end{cases}$$

2.3 Probabilistic Supervisory Control Problem (PSCP)

The goal is to match the behaviour of the controlled plant with a given probabilistic specification language. We call this problem the *Probabilistic Supervisory Control Problem (PSCP)*, or, alternatively, *Probabilistic Model Matching Problem.* More formally:

Given a plant PDES G_p and a requirements specification PDES G_r, find, if possible, a probabilistic supervisor V_p such that $L_p(V_p/G_p) = L_p(G_r)$.

An example of probabilistic generators representing a plant and its desired behaviour (a requirements specification) is shown in Fig. 1. Controllable events are marked with a bar on their edges.

The following theorem presents the conditions for the existence of a probabilistic supervisor for the PSCP (Lawford and Wonham, 1993; Pantelic et al., 2009).

Theorem 1. Let $G_p = (Q_p, \Sigma, \delta_p, q_{p_0}, p_p)$ and $G_r = (Q_r, \Sigma, \delta_r, q_{r_0}, p_r)$ be two nonterminating PDES with disjoint state sets Q_p and Q_r. There exists a probabilistic supervisor V_p such that $L_p(V_p/G_p) = L_p(G_r)$ iff for all $s\in L(G_r)$ there exists $q\in Q_p$ such that $\delta_p(q_{p_0}, s) = q$ and, letting $r = \delta_r(q_{r_0}, s)$, the following two conditions hold:

(i) $Pos(q)\cap\Sigma_u = Pos(r)\cap\Sigma_u$, and for all $\sigma\in Pos(q)\cap\Sigma_u$,
$$\frac{p_p(q,\sigma)}{\sum\limits_{\alpha\in\Sigma_u} p_p(q,\alpha)} = \frac{p_r(r,\sigma)}{\sum\limits_{\alpha\in\Sigma_u} p_r(r,\alpha)}$$

(ii) $Pos(r)\cap\Sigma_c \subseteq Pos(q)\cap\Sigma_c$, and, if $Pos(q)\cap\Sigma_u\neq\emptyset$, then for all $\sigma\in Pos(q)\cap\Sigma_c$,
$$\frac{p_r(r,\sigma)}{p_p(q,\sigma)}\sum_{\alpha\in\Sigma_u} p_p(q,\alpha) + \sum_{\alpha\in Pos(q)\cap\Sigma_c} p_r(r,\alpha) \le 1.$$

The first part of both conditions of Theorem 1 corresponds to controllability from classical supervisory control theory (namely, the condition $Pos(q)\cap\Sigma_u = Pos(r)\cap\Sigma_u$ of (i), and $Pos(r)\cap\Sigma_c\subseteq Pos(q)\cap\Sigma_c$ of (ii)). The remaining equations and inequalities correspond to the conditions for probability matching.

For each uncontrollable event possible from a state in a plant, the equation to be checked reflects the fact that the ratio of probabilities of uncontrollable events remains the same under supervision. This comes from the fact that after a control pattern has been chosen, the probabilities of disabled events in the plant are redistributed over enabled events in proportion to their probabilities. All possible uncontrollable events are always enabled, hence the ratios of their probabilities remain unchanged. Also, an inequality for each possible controllable event σ is derived from the upper bound on the probability of the occurrence of σ in the supervised plant, that is reached when the controllable event is always enabled.

When the conditions are satisfied, a solution to the PSCP exists. After a string has been observed, the control input is given as a solution to the system of nonlinear equations given by (2). This solution can be approximated by the fixpoint iteration algorithm as presented in the following theorem (Postma and Lawford, 2004; Pantelic et al., 2009).

Theorem 2. Assume that conditions (i) and (ii) of Theorem 1 are satisfied. Let $\Gamma(s) = Pos(q)\cap\Sigma_c$ if $Pos(q)\cap\Sigma_u\neq\emptyset$, and $\Gamma(s) = (Pos(q)\cap\Sigma_c)\backslash\{\gamma\}$ otherwise, where $\gamma\in Pos(q)$ is chosen such that for every $\sigma\in Pos(q)$, $\frac{p_r(r,\gamma)}{p_p(q,\gamma)} \ge \frac{p_r(r,\sigma)}{p_p(q,\sigma)}$ is satisfied. Let $x^0(s)\in[0,1]^{\Gamma(s)}$ and $f(s):\mathbb{R}^{\Gamma(s)}\to\mathbb{R}^{\Gamma(s)}$. For $x^0(s) = 0$, the sequence

$$x^{k+1}(s) = f(s)(x^k(s)), \qquad k = 0,1,\ldots, \text{ where} \tag{3}$$

$$f(s)(x)(\sigma) = \frac{p_r(r,\sigma)}{p_p(q,\sigma)h(s)(x)(\sigma)}, \sigma\in\Gamma(s), x\in\mathbb{R}^{\Gamma(s)} \text{ and}$$
$$h(s)(x)(\sigma)$$
$$= \sum_{\Theta\in\mathcal{P}(\Gamma(s)\backslash\{\sigma\})} \frac{1}{1-\sum\limits_{\alpha\in\Theta} p_p(q,\alpha)} \prod_{\alpha\in\Theta}(1-x(s)(\alpha)) \prod_{\alpha\in\Gamma(s)\backslash\{\sigma\}\backslash\Theta} x(s)(\alpha)$$

converges to the control input $x^*(s)$ (i.e., $V_p(s) = x^*(s)$).

For the example from Fig. 1, the probabilistic supervisor for the PSCP is given in Fig. 2.

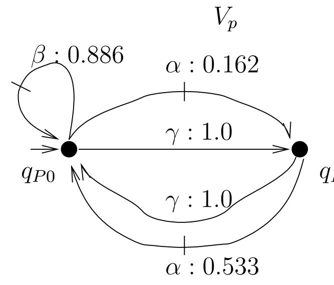

Fig. 2. Probabilistic supervisor V_p such that $L(V_p/G_p) = L(G_r)$ for G_p and G_r from Fig. 1

We note the special case from Theorem 2: in a certain state, only controllable events can happen in the plant ($Pos(q)\cap\Sigma_u = \emptyset$). Then, a probabilistic supervisor can disable them all which would cause termination. However, as we consider nonterminating generators, this is not allowed. An elegant solution turns out to be to always enable an appropriately chosen event (an event γ with the maximal ratio $p_{r,\gamma}/p_{p,\gamma}$): this event effectively becomes uncontrollable.

2.4 Application: An Example

We now present an application for the framework. We adapt the simplified model of a distributed robot control system presented in Li et al. (1998) to our setting. The example is simple, but illustrative of one class of application. A processor processes the readings from two sensors. The sensors models are shown at the top of Figure 3. For $i = 1, 2$, event α_i stands for "sensor i requests the processor", event β_i for "sensor i uses the processor", and γ_i is "sensor i releases the processor". The resulting plant is given in the same figure. Its nonprobabilistic behaviour is given as the shuffle of two DES that represent the sensors. Sensor 1 uses the processor more frequently. For example, sensor 1 reads the speed of the robot at a fixed rate, while sensor 2 sends warning signals when the robot approaches an obstacle. Probabilities attached to transitions are such that the probabilities of $\alpha_1, \beta_1, \gamma_1$ are 0.95, while the probabilities of $\alpha_2, \beta_2, \gamma_2$ are 0.05.

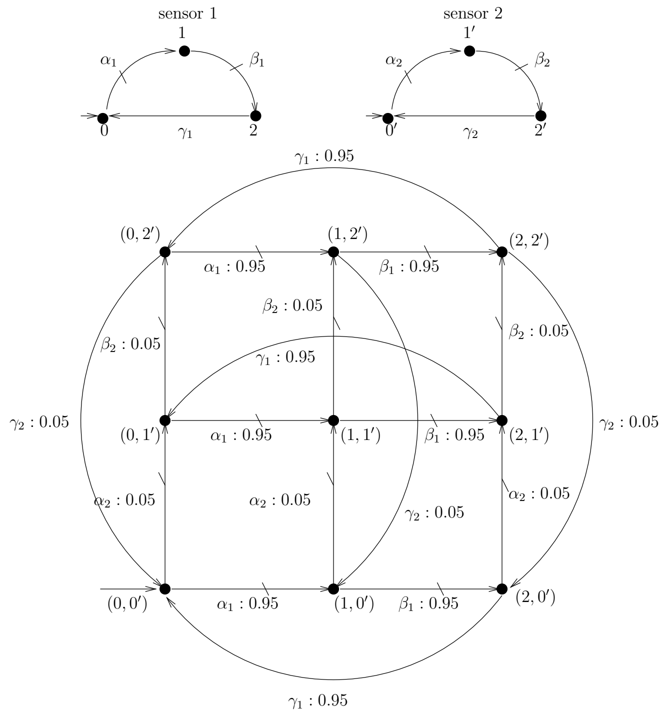

Fig. 3. Sensors and resulting plant

The requirements specification G_{r_1} is given in Figure 4. The nonprobabilistic part of the specification expresses the mutual exclusion requirement: two sensors cannot use the processor at the same time. Therefore, state $(2, 2')$ in Figure 3 is the forbidden state of the plant. The probabilistic part of the requirements specification reflects a need for a prioritization of sensor 2 when both sensors have requested the processor. More precisely, at state r_4, after both sensors have requested the processor, sensor 2 should be four times more likely to use the processor than sensor 1.

In order to solve the PSCP for the given plant and requirements specification, the results of Section 2.3 are used. The probabilistic supervisor for the PSCP exists, and it disables event β_1 at state $(1, 2')$ and event β_2 at state $(2, 1')$, while event β_1 is enabled with probability 0.2105 at state $(1, 1')$ (as calculated by Theorem 2).

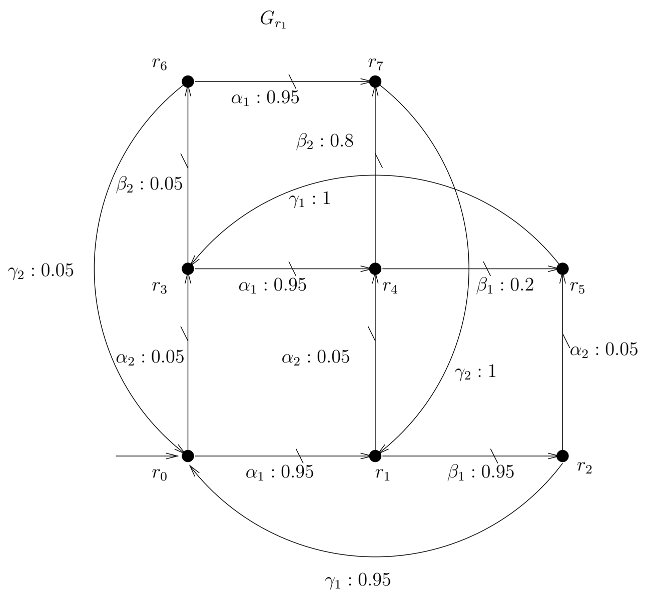

Fig. 4. Requirements specification G_{r_1}

2.5 *Measuring Similarity Between PDES*

Behavioural similarity between the systems in our framework is measured using a *pseudometric* [1]. The pseudometric is based on the pseudometric of Deng et al. (2006) suggested for a large class of automata which includes our generator. The pseudometric is defined on the states of automata as a fixed point of a function that is given as a linear programming problem.

Let $G = (Q, \Sigma, \delta, q_0, p)$ be a PDES, where $Q = \{q_0, q_1, \ldots q_{N-1}\}$. First, Deng et al. (2006) introduce the class $\mathcal{M}$ of 1-bounded pseudometrics on states with the (partial) ordering

$$d_1 \preceq d_2 \text{ if } \forall q_q, q_r \in Q \quad d_1(q_q, q_r) \geq d_2(q_q, q_r) \qquad (4)$$

as was initially suggested in Desharnais et al. (2002). The ordering in (4) is reverse so that bisimilarity can be characterized as the greatest fixed point of a function. It can be proved that $(\mathcal{M}, \preceq)$ is a complete lattice. Then, let $d \in \mathcal{M}$, and let the constant $e \in (0, 1]$ be a *discount factor* that determines the degree to which the difference in the probabilities of transitions further in the future is discounted: the smaller the value of e, the greater the discount on future transitions. Next, we introduce some useful notation. Let $q_q, q_r \in Q$ and let ρ_{q_q} and ρ_{q_r} be the distributions on $\Sigma \times Q$ induced by the states q_q and q_r, respectively. Assume $0 \leq i, j \leq N - 1$. For notational convenience, we will write $\rho_{\sigma,i}$ instead of $\rho_{q_q}(\sigma, q_i)$, and, similarly, $\rho'_{\sigma,j}$ instead of $\rho_{q_r}(\sigma, q_j)$. As stated previously, the pseudometric of Deng et al. (2006) is defined on the states of automata as a fixed point of a function that is given as a linear programming problem. For the case of probabilistic generators, the function is simplified by solving the linear programming problem (Pantelic and Lawford, 2009, 2012a). Consequently, the pseudometric

[1] A pseudometric on a set of states Q is a function $d : Q \times Q \to \mathbb{R}$ that defines a distance between two elements of Q, and satisfies the following conditions: $d(x, y) \geq 0$, $d(x, x) = 0$, $d(x, y) = d(y, x)$, and $d(x, z) \leq d(x, y) + d(y, z)$, for any $x, y, z \in Q$. If all distances are in $[0, 1]$, the pseudometric is *1-bounded*.

on states of probabilistic generators, d_{fp}, is given as the greatest fixed-point of the function $\mathcal{D}$ on $\mathcal{M}$:

$$\mathcal{D}(d)(q_q, q_r) = \sum_{\sigma \in \Sigma} max(\rho_{\sigma,i} - \rho'_{\sigma,j} + c_{ij}\rho'_{\sigma,j}, c_{ij}\rho_{\sigma,i}) \quad (5)$$

where $c_{ij} = e \cdot d(q_i, q_j)$, and i and j denote $i(q_q, \sigma)$ and $j(q_r, \sigma)$, respectively, where $i(q_q, \sigma) = i$ such that $q_i = \delta(q_q, \sigma)$ if $\delta(q_q, \sigma)!$, and $i(q_q, \sigma) = 0$, otherwise. We arbitrarily choose $i(q_q, \sigma)$ to be 0 when $\delta(q_q, \sigma)$ is not defined although we could have chosen any other $i \in \{1, \ldots, N-1\}$. This is because when $\delta(q_q, \sigma)!$ does not hold, then $\rho_{\sigma, i(q_q,\sigma)} = 0$ for any $i(q_q, \sigma) \in \{1, \ldots, N-1\}$. Similarly, $j(q_r, \sigma) = j$ such that $q_j = \delta(q_r, \sigma)$ if $\delta(q_r, \sigma)!$, and $j(q_r, \sigma) = 0$, otherwise. Roughly speaking, the distance sums up the differences between probabilities

Two efficient algorithms for calculation of the distances in the pseudometric have been developed in Pantelic and Lawford (2009, 2012a). The first algorithm reduces to finding the (unique) solution of a system of linear equations. The second algorithm approximates the distances with a prespecified accuracy and is iterative.

The pseudometric is sensitive to all differences between corresponding transition probabilities, as opposed to e.g., the pseudometric of Giacalone et al. (1990) that, roughly speaking, considers only the maximum of the differences between the corresponding probabilities. The logical and trace characterization of the pseudometric were given in Pantelic and Lawford (2010), Pantelic and Lawford (2012b). In logical characterization, the pseudometric is characterized via a real-valued logic: the distance in the pseudometric between two systems is measured by a formula that distinguishes between the systems the most. Then, from this logical characterizations follows the trace characterization: systems are similar if the probabilities of their (appropriately discounted) traces, certain sets of traces, and certain properties of traces are similar. The goal of alternative characterizations is to deepen the understanding of what similarity between systems as measured by the pseudometric means in terms of similarities of their logical properties, and similarities of their probabilistic traces. Also, the pseudometric discounts the future. The concept of discount has been widely applied in game theory, economics and optimal control. From an engineering point of view, one cares more about an error in the near future than the one in the distant future (de Alfaro et al. (2003)).

2.6 Optimal Probabilistic Supervisory Control

In the case when the conditions for the existence of a solution of the PSCP are not satisfied, we search for a suitable approximation. In Pantelic and Lawford (2012a,b), two similar optimal probabilistic supervisory control problems were defined and solved. Given the space limitations, we define one problem and only sketch its solution as a complete presentation of the algorithm is quite lengthy. We define the problem first.

The Optimal Probabilistic Supervisory Control Problem (OPSCP): Let $G_p = (Q_p, \Sigma, \delta_p, q_{p_0}, p_p)$ be a plant PDES, and let $G_r = (Q_r, \Sigma, \delta_r, q_{r_0}, p_r)$ be a requirements specification represented as a PDES. If there is no probabilistic supervisor V_p such that $L_p(V_p/G_p) = L_p(G_r)$ (i.e., the conditions of Theorem 1 fail), find, if it exists, V_p such that

(1) $L(V_p/G_p) \subseteq L(G_r)$ and supervisor V_p is maximally permissive and deadlock-free in the nonprobabilistic sense (i.e., $L(V_p/G_p)$ is the supremal deadlock-free and controllable sublanguage of $L(G_r)$ with respect to G_p).

(2) The probabilistic behaviour of the controlled plant is "as close as possible" to the probabilistic behaviour of the requirements specification restricted to the supremal deadlock-free and controllable sublanguage of $L(G_r)$ with the respect to G_p.

The requirement is considered a hard safety requirement. Therefore, the supremal deadlock-free and controllable sublanguage of the requirement with respect to the plant is generated as the maximal deadlock-free behaviour of the controlled plant. The requirements specification is constrained to the same sublanguage, with appropriately normalized probabilities. The generators representing achievable behaviour of the controlled plant and the modified requirements specification are isomorphic. The distance in the pseudometric from Section 2.5 between the two generators is then minimized. The algorithm is iterative and works for $e \in (0, 1)$. In each iteration, for every two isomorphic states, a linear programming problem is solved: the distance between two states in the pseudometric is minimized under the controllability conditions of Theorem 1. The algorithm iterates until a prespecified accuracy of the distance between the systems is reached.

The second optimal supervisory problem is very similar to the OPSCP, without a restriction of the probabilistic behaviour of the requirements specification to the supremal deadlock-free and controllable sublanguage of $L(G_r)$ with the respect to G_p. Its solution is presented in Pantelic and Lawford (2012b) and is similar to the solution of the OPSCP.

3. THE FRAMEWORK AND MDPS

Markov Decision Processes (MDPs, also known as Markov Controlled Processes) are commonly used to model systems that exhibit both probabilistic and nondeterministic behaviour. Most state-of-the-art probabilistic model checkers support this model directly. MDPs also represent the most widely used framework for control of PDES. The generative model used in our framework can be straightforwardly transformed into an MDP (with the introduction of a special event) so that the probabilistic model checking tools can be used for the analysis of these systems. The details of the transformation will be discussed later in this section. However, simply transforming our model to an MDP and applying the control as defined in the MDP framework on the resulting MDP is pointless: as will be shown in this section, the control applied to the transformed model in the MDP framework cannot change the dynamics of the system. Also, as will be shown, any probabilistic supervisor in our framework can be represented as an MDP, and any control policy or "adversary" as defined in the MDP framework can be represented using a (possibly nondeterministic) probabilistic generator. This duality of the frameworks has the potential to connect these two distinct research areas, allowing the transfer of results between the communities, possibly leading to novel results in both control theory and probabilistic verification.

3.1 MDPs

An MDP is a tuple (S, s_s, A, P), where S is a finite set of states, s_s is the initial state, A is a finite set of actions, and $P : S \times A \times S \to [0, 1]$ is a transition function such that for every $s \in S$, $a \in A$, either $\sum_{s' \in S} P(s, a, s') = 1$ or $\sum_{s' \in S} P(s, a, s') = 0$ holds. $P(s, a, s')$ is interpreted as the probability of the action a taking the system from the state s to state s'. At least one action should be enabled at each state.

A *finite path* through the MDP is the sequence $h = (s_0, a_0, \ldots, s_{t-1}, a_{t-1}, s_t)$, $t \in \mathbb{N}$, with $P(s_i, a_i, s_{i+1}) > 0$ for $i = 0, \ldots, t-1$, where $s_i \in S$ and $a_i \in A$. Let H be the set of all finite paths in the MDP. A (randomized) *policy* (*adversary* or *scheduler*) is function $\pi : H \times A \to [0, 1]$, such that $\sum_{a \in A} \pi(h, a) = 1$, for each finite path $h = (s_0, a_0, \ldots, s_{t-1}, a_{t-1}, s_t)$ through the MDP, and $\pi(h, a) > 0$ only if a is enabled in s_t. A *memoryless policy* is a policy that does not depend on the finite path, but only its last state s_t. Note that the defined policy is probabilistic: the deterministic one is a special case (the choice of action to be taken is deterministic).

An MDP often has an associated cost function, giving each transition or state a cost. Alternatively, a reward function can be defined. The optimal control problem is to find the admissible control policy such that a certain criterion is optimized (possibly with some additional constraints). Different cost evaluation criteria have been used. Some of the most commonly used are total cost, discounted cost, average cost, and sample path average cost. For more details, the interested reader is referred to Borkar (1991); Arapostathis et al. (1993).

In the computer science community, there has been a lot of effort in extending classical model checking techniques to probabilistic transition systems. Quantitative properties of MDPs can be expressed in probabilistic extensions of temporal logic and checked on the model (Rutten et al., 2004; Kwiatkowska et al., 2007).

3.2 Probabilistic Generators and MDPs

Compared to classical reactive models like MDP, generative models are more general: for every state, it contains not only the information about relative probabilities of transitions on a particular event, but also information on the relative probabilities of transitions on different events (Schröder and Mateus, 2002; Glabbeek et al., 1995). Therefore, as a generative model contains more information than a reactive one, a direct transformation of a generative model to a reactive model results in an abstraction that loses this additional information. For example, a naive transformation of the generator G on the left of Figure 5 to an MDP would have all the probabilities of events' occurrences become 1, effectively leaving out all the information about the probabilistic language of the original generator. However, a probabilistic generator can be transformed into an MDP so that it is possible to reconstruct the generator's probabilistic language from the MDP (see Figure 5). The transformation requires the introduction of a special event τ, with a state expansion factor of $O(|\Sigma|)$. The new model is an MDP, with a special event τ. Let G be a probabilistic generator representing a plant, and G_{MDP} its equivalent MDP, as in Figure 5. For every state $s \in S$ of G_{MDP}, only one action is admissible from a state. That means that a controller in the MDP framework would always pick that one action available, and, consequently, would not be able to affect the probabilistic behaviour of the plant. Therefore, the control in the MDP framework would not make sense for an MDP equivalent to a probabilistic generator.

3.3 Duality

In general, a probabilistic supervisor is not a probabilistic generator (an example is the probabilistic supervisor in Figure 2). This comes from the fact that, with a probabilistic supervisor, the probabilities attached to events are the probabilities of the events being enabled, not the probabilities of events occurring (see Section 2.2). Therefore, the probabilities of enabling events from a state of the supervisor in Figure 2 are not distributed according to a probability distribution.

A probabilistic supervisor, however, can be represented using a reactive model. In Figure 6, probabilistic supervisor V_p from Figure 2 is represented using an MDP. The probabilities in Figure 6 are denoted by symbols, instead of their numerical values, to facilitate our exposition. The transformation illustrated in Figure 6 will now be briefly explained.

Each state $q \in Q$ of probabilistic supervisor V_p maps to a set of states $\{(q, \Phi_1) | \Phi_1 \subseteq Pos(q) \backslash \Sigma_u\}$. Events admissible from state (q, Φ_1) are exactly those from Φ_1 together with uncontrollable events possible from state q. The probability attached to event σ going from state (q, Φ_1) to (q', Θ) (where $q' \in Q$, $\Theta \subseteq Pos(q') \backslash \Sigma_u$) is the probability of Θ being enabled at state q' as given in Equation 1 (not the probability of Φ_1 being enabled at state q). Event $\eta \notin \Sigma$ and state $r_0 \notin Q$ are used to initialize the supervisor properly.

On the other hand, a probabilistic control policy in the MDP framework cannot be represented with our probabilistic generator in a straightforward manner. This comes from the fact that, even when a memoryless policy is being implemented (the control decision only depends only on the current system state), there might be two different paths corresponding to the same sequence of events leading to two different states of the system. However, in that case, a nondeterministic probabilistic generator can be used, given that the states of the system are observable. Alternatively, we would be able to encode two different decisions at the states reached by the same string using our probabilistic generator with event augmentation. More precisely, it would be possible to encode information about the two states through event labelling: transitions would be labeled not only with events, but also the states of the system that these events lead to.

Therefore, a probabilistic supervisor in our framework can be represented as an MDP. Similarly, a policy as defined

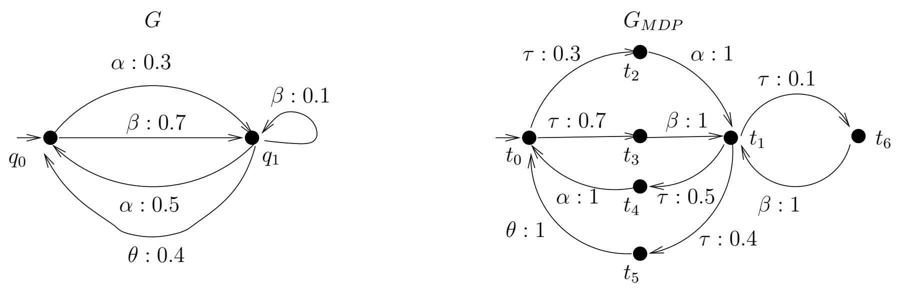

Fig. 5. Transformation from a probabilistic generator to an MDP

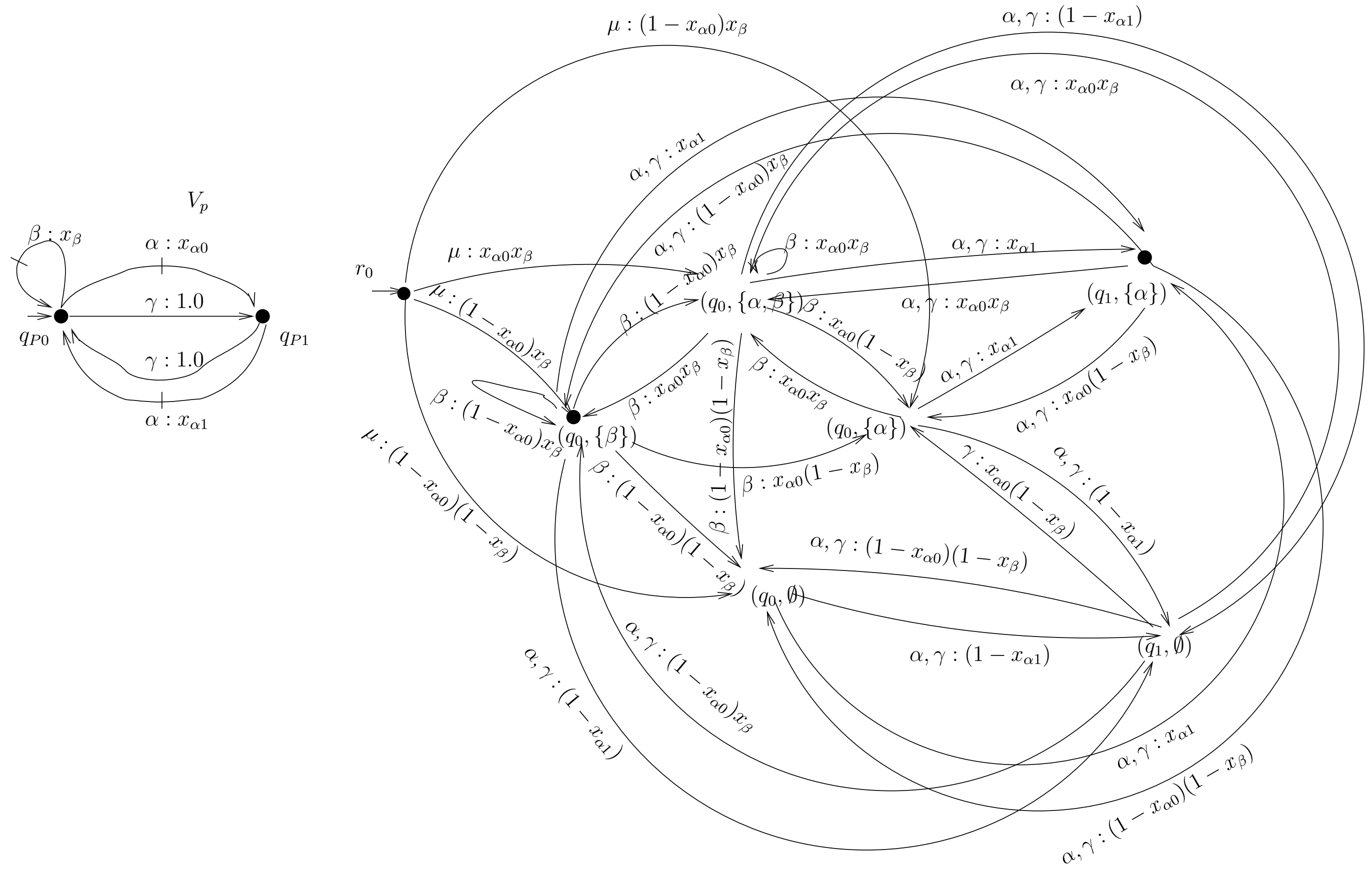
V_p
$\beta : x_\beta$
$\alpha : x_{\alpha 0}$
$\gamma : 1.0$
q_{P0}
q_{P1}
$\gamma : 1.0$
$\alpha : x_{\alpha 1}$
$\mu : (1 - x_{\alpha 0})x_\beta$
$\alpha, \gamma : (1 - x_{\alpha 1})$
$\alpha, \gamma : x_{\alpha 0}x_\beta$
$\alpha, \gamma : x_{\alpha 1}$
$\alpha, \gamma : (1 - x_{\alpha 0})x_\beta$
$\beta : x_{\alpha 0}x_\beta$
$\alpha, \gamma : x_{\alpha 1}$
r_0
$\mu : x_{\alpha 0}x_\beta$
$\beta : (1 - x_{\alpha 0})x_\beta$
$(q_0, \{\alpha, \beta\})$
$\alpha, \gamma : x_{\alpha 0}x_\beta$
$(q_1, \{\alpha\})$
$\mu : (1 - x_{\alpha 0})x_\beta$
$\beta : x_{\alpha 0}(1 - x_\beta)$
$\alpha, \gamma : x_{\alpha 1}$
$\beta : (1 - x_{\alpha 0})x_\beta$
$\beta : x_{\alpha 0}x_\beta$
$\beta : x_{\alpha 0}x_\beta$
$\alpha, \gamma : x_{\alpha 0}(1 - x_\beta)$
$(q_0, \{\beta\})$
$(q_0, \{\alpha\})$
$\mu : (1 - x_{\alpha 0})(1 - x_\beta)$
$\beta : x_{\alpha 0}(1 - x_\beta)$
$\beta : (1 - x_{\alpha 0})(1 - x_\beta)$
$\gamma : x_{\alpha 0}(1 - x_\beta)$
$\alpha, \gamma : (1 - x_{\alpha 1})$
$\beta : (1 - x_{\alpha 0})(1 - x_\beta)$
$\alpha, \gamma : (1 - x_{\alpha 0})(1 - x_\beta)$
$(q_0, \emptyset)$
$(q_1, \emptyset)$
$\alpha, \gamma : (1 - x_{\alpha 1})$
$\alpha, \gamma : (1 - x_{\alpha 1})$
$\alpha, \gamma : (1 - x_{\alpha 0})x_\beta$
$\alpha, \gamma : x_{\alpha 1}$
$\alpha, \gamma : (1 - x_{\alpha 0})(1 - x_\beta)$

Fig. 6. Probabilistic supervisor V_p represented as an MDP V_p^R

in the MDP framework can be represented using a (potentially nondeterministic or augmented) probabilistic generator. This fact is potentially useful, as it might allow for the exchange of results between the models and the frameworks. For example, the following is an illustration of how the duality between the frameworks could be exploited. In our framework, a plant is represented as a probabilistic generator and we solved the problem of finding a probabilistic supervisor (equivalently, an MDP) that minimizes the distance of the controlled plant from a specification given as a probabilistic generator. In the MDP framework, a plant is represented as an MDP, and a safety property can be specified as a desired behaviour of the controlled plant (typically in temporal logic). Probabilistic model checking verifies whether the safety property is satisfied with probability greater than a prespecified probability for all schedulers (equivalently, probabilistic generators). The verification can be reduced to finding the most adversarial policy, *i.e.*, the policy under which the behaviour of the plant is probabilistically the farthest from the required property. The duality between the two problems might be used to apply the solution of one problem to the solution of the other problem.

4. CONCLUSIONS

This work surveys the existing framework for control of probabilistic discrete event systems and further equips it with a concrete control application. Then, we present an initial discussion on the relationship between our framework and MDPs. The discussion should prompt further research on the interplay of the two frameworks that we hope to bring useful results in both control and verification of PDES.

REFERENCES

Arapostathis, A., Borkar, V.S., Fernández-Gaucherand, E., Ghosh, M.K., and Marcus, S.I. (1993). Discrete-time controlled Markov processes with average cost criterion: a survey. *SIAM Journal on Control and Optimization*, 31(2), 282–344. doi:http://dx.doi.org/10.1137/0331018.

Borkar, V.S. (1991). *Topics in controlled Markov chains.* Wiley, New York.

de Alfaro, L., Henzinger, T.A., and Majumdar, R. (2003). Discounting the future in systems theory. In J.C.M. Baeten, J.K. Lenstra, J. Parrow, and G.J. Woeginger (eds.), *Proceedings of International Colloquium on Automata, Languages and Programming*, volume 2719 of *Lecture Notes in Computer Science*, 1022–1037. Springer.

Deng, Y., Chothia, T., Palamidessi, C., and Pang, J. (2006). Metrics for action-labelled quantitative transition systems. *Electronic Notes in Theoretical Computer Science*, 153(2), 79–96.

Desharnais, J., Jagadeesan, R., Gupta, V., and Panangaden, P. (2002). The metric analogue of weak bisimulation for probabilistic processes. In *Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science*, 413–422. IEEE Computer Society, Washington, DC, USA.

Giacalone, A., Jou, C., and Smolka, S. (1990). Algebraic reasoning for probabilistic concurrent systems. In *M. Broy and C.B. Jones, eds, Proceedings of the Working Conference on Programming Concepts and Methods*, 443–458. North-Holland, Sea of Gallilee, Israel.

Glabbeek, R.J.V., Smolka, S.A., and Steffen, B. (1995). Reactive, generative and stratified models of probabilistic processes. *Information and Computation*, 121(1), 59–80.

Kwiatkowska, M., Norman, G., and Parker, D. (2007). Stochastic model checking. In M. Bernardo and J. Hillston (eds.), *Proceedings of Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation*, volume 4486 of *Lecture Notes in Computer Science (Tutorial Volume)*, 220–270. Springer.

Lawford, M. and Wonham, W. (1993). Supervisory control of probabilistic discrete event systems. In *Proceedings of the 36th IEEE Midwest Symposium on Circuits and Systems*, volume 1, 327–331. IEEE.

Li, Y., Lin, F., and Lin, Z.H. (1998). Supervisory control of probabilistic discrete event systems with recovery. *IEEE Transactions on Automatic Control*, 44(10), 1971–1975.

Pantelic, V. (2011). *Probabilistic supervisory control of probabilistic discrete event systems.* Ph.D. thesis, McMaster University, Hamilton, ON, Canada.

Pantelic, V. and Lawford, M. (2009). Towards optimal supervisory control of probabilistic discrete event systems. In *Proceedings of 2nd IFAC Workshop on Dependable Control of Discrete Systems (DCDS 2009)*, 85–90. Bari, Italy.

Pantelic, V. and Lawford, M. (2010). Use of a metric in supervisory control of probabilistic discrete event systems. In *Proceedings of the 10th International Workshop on Discrete Event Systems (WODES 2010), August 30 - September 1, 2010*, 227–232.

Pantelic, V. and Lawford, M. (2012a). Optimal supervisory control of probabilistic discrete event systems. *IEEE Transactions on Automatic Control*, 56(5), 16pp.

Pantelic, V. and Lawford, M. (2012b). A pseudometric in supervisory control of probabilistic discrete event systems. *Discrete Event Dynamic Systems*, 22(4), 479–510.

Pantelic, V., Postma, S., and Lawford, M. (2009). Probabilistic supervisory control of probabilistic discrete event systems. *IEEE Transactions on Automatic Control*, 54(8), 2013–2018.

Postma, S. and Lawford, M. (2004). Computation of probabilistic supervisory controllers for model matching. In *Venu Veeravalli and Geir Dullerud, editors, Proceedings of Allerton Conference on Communications, Control, and Computing.* Monticello, Illinois.

Rutten, J., Kwiatkowska, M., Norman, G., and Parker, D. (2004). *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems,* P. Panangaden and F. van Breugel (eds.), volume 23 of *CRM Monograph Series.* American Mathematical Society.

Schröder, L. and Mateus, P. (2002). Universal aspects of probabilistic automata. *Mathematical Structures in Computer Science*, 12(4), 481–512. doi: http://dx.doi.org/10.1017/S0960129502003614.