Optimal Supervisory Control of Probabilistic Discrete Event Systems

Vera Pantelic and Mark Lawford, *Senior Member, IEEE*

***Abstract*—Probabilistic discrete event systems (PDES) are modeled as generators of probabilistic languages and the supervisors employed are a probabilistic generalization of deterministic supervisors used in standard supervisory control theory. In the case when there exists no probabilistic supervisor such that the behavior of a plant under control exactly matches the probabilistic language given as the requirements specification, we want to find a probabilistic control such that the behavior of the plant under control is "as close as possible" to the desired behavior. First, as a measure of this proximity, a pseudometric on states of generators is defined. Two algorithms for the calculation of the distance between states in this pseudometric are described. Then, an algorithm to synthesize a probabilistic supervisor that minimizes the distance between generators representing the achievable and required behavior of the plant is presented.**

***Index Terms*—Discrete event systems, optimal control, stochastic systems, supervisory control.**

I. Introduction

THE supervisory control theory of discrete event systems (DES) was developed in the seminal work of Ramadge and Wonham [1]. A supervisor (controller) controls a plant by enabling/disabling controllable events based on the observation of the previous behavior of the plant. DES are most often modeled by finite automata whose transitions are labeled with events: therefore, the behavior of a DES can be represented as a regular language. The supervisory control problem considered is to supervise the plant so it generates a given specification language.

Recently probabilistic models have attracted considerable attention in modeling systems with uncertainty. They are of interest in many application areas, e.g., communication protocols, distributed computing, performance analysis, and fault tolerance. Many probabilistic logics are used in the specification and verification of probabilistic systems (excellent overviews are offered in [2], [3]). Probabilistic model checking provides for (limited) guarantees when conventional model checking is not possible (e.g., the state space is too large) [4]. Many of probabilistic models have been widely researched and applied. While reactive models have been predominantly used in probabilistic model checking tools as well as in the control of probabilistic systems, generative models have also found their applications, especially in the control and modeling of robot systems [5]–[7], prediction of human behavior [8], *etc*. The difference between reactive and generative systems is in the treatment of events [9], [10]. In a reactive system, an event is seen as an input from the environment: the system reacts to it by choosing a next state according to a probability distribution on the states of the system. On the other hand, in a generative system, an event is seen as an output: the system chooses a transition according to a probability distribution, and generates as an output the event the chosen transition is labeled with. In terms of expressiveness, in general, a generative model is more expressive than a reactive model.

The control of different models of stochastic discrete event systems has been investigated in [11], [12], etc.. Rabin's probabilistic automata are used in [11] as the underlying model. The optimal control theory of Markov chains is widely investigated in the framework of controlled Markov chains, also known as Markov decision processes (MDPs), e.g., see [12]. A deterministic supervisory control framework for stochastic discrete event systems was developed in [13] using the model of [14]–[16]. The control objective considered in [13] is to construct a supervisor such that the controlled plant does not execute specified illegal traces, and occurrences of the legal traces in the system are greater than or equal to specified values. Necessary and sufficient conditions for the existence of a supervisor are given. Further, in [17], a technique to compute a maximally permissive supervisor on-line is given. In [18], [19], the deterministic version of probabilistic automata used in [14]–[16] (our model) is used. The requirements specification is given by weights assigned to states of a plant and the control goal is, roughly speaking, to reach the states with more weight (more desirable states) more often. A deterministic control is synthesized for a given requirements specification so that a measure based on the specification and probabilities of the plant is optimized. Optimal supervisory theory of probabilistic systems was considered in [5], where the system is allowed to violate the specification, but with a probability lower than a specified value. Controller synthesis for probabilistic systems has also attracted attention in the formal methods community (e.g., [20]).

We seek for a comprehensive theoretical framework for probabilistic supervisory control theory of *probabilistic discrete event systems* (PDES). We build upon the framework introduced in [21]–[23]. Finite state machines with transitions labeled with events, *generators*, commonly used in classical supervisory theory to represent discrete event systems, are generalized to a generative probabilistic model called *probabilistic generators* (a version of automata from [14]–[16]), which are generative models. Roughly speaking, probabilistic generators are generators extended with probabilities attached

Manuscript received July 27, 2010; revised April 20, 2011; revised July 27, 2011; revised September 07, 2011; revised September 09, 2011; accepted September 30, 2011. Date of publication October 25, 2011; date of current version April 19, 2012. Recommended by Associate Editor S. A. Reveliotis.

The authors are with the Department of Computing and Software, Faculty of Engineering, McMaster University, Hamilton ON Canada L8S 4K1 (e-mails: pantelv@mcmasterdotca; lawford@mcmasterdotca).

Digital Object Identifier 10.1109/TAC.2011.2173420

to each transition. A probability attached to an event that can occur from a state represents the probability of occurrence of that event from the state. A probabilistic generator generates a *probabilistic language*: each string generated by the underlying (nonprobabilistic) generator has a probability attached to it that represents the probability of occurrence of that particular string in the language. The control used in the framework is probabilistic. Although probabilistic control of PDES is harder to deal with than deterministic control, both from the viewpoint of analysis, and practice, it is much more powerful: it has been shown in [21], [23] that probabilistic supervisory control of PDES can generate a much larger class of probabilistic languages than deterministic control. In the sense of a supervisory control problem that would require that the controlled plant generates a specified probabilistic language, the use of deterministic control might be too restrictive for a designer. In [21], the standard supervisory control problem is accordingly generalized to the *probabilistic supervisory control problem (PSCP)*: find, if possible, a probabilistic supervisor for a plant so that the plant under control generates a given probabilistic language. This problem has been solved in [21]–[23].

This paper focuses on optimal supervisory control theory inside this framework. Analogous to a problem in classical supervisory control theory, it can happen that, given a plant to be controlled and a probabilistic specification language, no supervisor exists such that the plant under control generates the specified language. In this case, when the exact solution is not achievable, a designer tries to find a supervisor such that the plant under control generates a *closest approximation* of the desired behavior. The problem is called the *Optimal Probabilistic Supervisory Control Problem (OPSCP)*. The nonprobabilistic behavior of the requirements specification is considered to be a safety constraint in the standard supervisory control sense similar to [13].

The closest approximation is then generated such that its nonprobabilistic behavior is the maximally permissive behavior of the controlled plant. Then, using an iterative algorithm, the probabilities of the closest approximation are determined such that the approximation is the most similar to the (appropriately modified) requirements specification. As a measure of similarity, a *pseudometric* on states of probabilistic generators is used.

The pseudometric is adopted from the theoretical computer science community. It is a measure of behavioral similarity between states: the smaller the distance in the pseudometric, the more similar the behavior of the states is. A distance of zero corresponds to probabilistic bisimilarity. The pseudometric was defined in [24] as a greatest fixed point of a monotone function. It is based on Kantorovich metric [25] (also known as Hutchinson metric [26], and Wasserstein metric [27]) that has been widely applied in many domains [28]. The work of [24] is closely related to the well-established work on pseudometrics presented in [29]–[35]. In [36], [37], we further characterize our pseudometric by giving it logical and trace characterizations. The logical characterization measures the distance between two systems by a real-valued formula that distinguishes between the systems the most. The trace characterization shows that two systems similar in the pseudometric have similar (appropriately discounted) probabilities of traces, certain sets of traces and certain properties of traces.

The work of [24] did not offer any algorithm to calculate distances in the pseudometric. In this paper, we suggest two algorithms to calculate/approximate the distance between two generators in the pseudometric. The first algorithm calculates distances by solving a system of linear equations, and is, to the best of our knowledge, a novel result in the field. The second algorithm approximates distances with a specified accuracy. It is iterative and bears some similarities to the algorithms of [32], [35], [38]–[40]. However, our algorithm is for a different type of probabilistic transition systems and is derived using a different mathematical machinery, resulting in a much simpler algorithm due to the nature of the underlying models. While interesting in its own right, the main significance of this iterative algorithm is that its proof of correctness can be reused in the solution of the OPSCP.

Our research is likely to find an application in the field of robotics as probabilistic generators have been used to model systems in the control of robot systems [5]–[7] (see [41] for a simple, illustrative application). Also, the Kantorovich metric has been used in a number of applications [28]. The most promising area in regard to our research is the field of bionformatics, where the metric has been used to measure similarities between DNA sequences. Another route to explore is the use of our research in the generation of test cases (adversaries) for MDPs. More precisely, a probabilistic generator can be viewed as a supervisor for MDPs. On the other hand, the probabilistic supervisors defined in our framework can be represented as an MDP (see [41]). This duality between the plant to be controlled and the probabilistic supervisor performing the control might provide interesting connections between probabilistic model checking and supervisory control theory.

In Section II, PDES are presented as generators of probabilistic languages and the probabilistic control of PDES is introduced before stating the PSCP and reviewing its solution. The OPSCP is formulated in Section III. The pseudometric to be used in the solution of the OPSCP is presented in Section IV. Section V derives two algorithms for the calculation of the pseudometric. Section VI solves the OPSCP: it presents the algorithm for finding the closest approximation to within a specified accuracy. Section VII concludes with avenues for future work.

This paper represents the journal version of results of [42]. The paper offers a thorough literature review and additional informal reasoning and formal proofs that are lacking from [42].

II. Preliminaries

In this section, PDES are modeled as generators of probabilistic languages. Then, the probabilistic control of PDES, the PSCP, and the solution of the PSCP are introduced.

A. Modeling PDES

Following [21], [23], a probabilistic DES is modeled as a probabilistic generator.

Definition 1: A probabilistic generator G is a tuple $G = (Q, \Sigma, \delta, q_0, p)$, 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 $q \in Q$, $\sum_{\sigma\in\Sigma} p(q,\sigma) = 1$. The probability that the event $\sigma \in \Sigma$ is going to occur at the state $q \in Q$ is $p(q,\sigma)$. For generator G to be well-defined, $p(q,\sigma) = 0$ should hold if and only if $\delta(q,\sigma)$ is undefined.

Remark 1: Relaxing the condition $\sum_{\sigma\in\Sigma} p(q,\sigma) = 1$ into $\sum_{\sigma\in\Sigma} p(q,\sigma) \leq 1$ would allow for modeling termination. The probability that the system terminates at state q would then be $1 - \sum_{\sigma\in\Sigma} p(q,\sigma)$. However, since a terminating PDES can easily be transformed into a probabilistic generator of Definition 1 using the technique described in [21], we find the model of Definition 1 general enough.

The state 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 definition of probabilistic generators does not contain marking states since the probabilistic specification languages considered in this paper are prefix closed.

The language $L(G)$ generated by G is $L(G) = \{s \in \Sigma^* \mid \delta(q_0,s)!\}$. 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 $Pos(q) := \{\sigma \in \Sigma | p(q,\sigma) > 0\}$, or, equivalently, $Pos(q) := \{\sigma \in \Sigma | \delta(q,\sigma)!\}$.

Next, the synchronous product of (nonprobabilistic) 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 that underlies G will be denoted G^{np} (i.e., $G^{np} = (Q,\Sigma,\delta,q_0)$) throughout the sequel. 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., $G_1^{np} = (Q_1,\Sigma_1,\delta_1,q_{0_1})$ and $G_2^{np} = (Q_2,\Sigma,\delta_2,q_{0_2})$.

Definition 2: 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 $\delta_2(q_2,\sigma)!$.

B. Probabilistic Supervisors: Existence and Synthesis

As in classical supervisory control theory, the set Σ is partitioned into Σ_c and Σ_u, the sets of controllable and uncontrollable events, respectively. Deterministic supervisors for DES are generalized to *probabilistic supervisors*. The control technique used is called *random disablement*. 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.

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

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

Therefore, after observing a string s, 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 a set of controllable events to be enabled has been decided upon (uncontrollable events are always enabled), the system acts as if supervised by a deterministic supervisor. Given sets A, B, we will denote the power set of A by $\mathcal{P}(A)$, and the set difference of A and B by $A \backslash B$. 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:

$$P(\alpha \text{ in } V_p/G|s) = \sum_{\Theta\in\mathcal{P}(Pos(q)\cap\Sigma_c)} P(\alpha|V_p \text{ enables } \Theta \text{ after } s)\cdot P(V_p \text{ enables } \Theta|s) \quad (1)$$

where

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

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

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

Given a plant PDES G_p and a 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 a requirements specification is shown in Fig. 1. Controllable events are marked with a bar on their edges.

Next, we present the conditions for the existence of a probabilistic supervisor for the PSCP from [21], [23].

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 PDES with disjoint state sets Q_p and Q_r. Then, let G_p^{np} and G_r^{np} be the nonprobabilistic generators underlying G_p and G_r, respectively, i.e., $G_p^{np} = (Q_p,\Sigma,\delta_p,q_{p_0})$ and $G_r^{np} = (Q_r,\Sigma,\delta_r,q_{r_0})$. Also, let

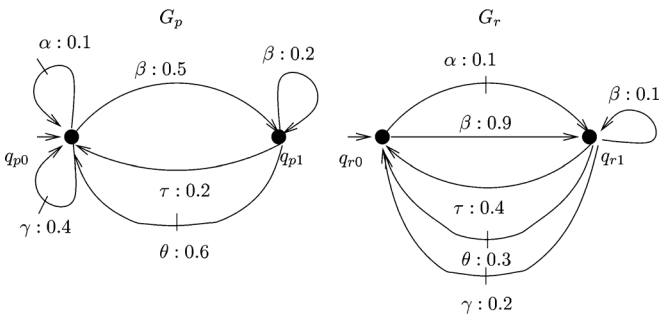


Fig. 1. Plant G_p, and requirements specification G_r.

$G_s = (Q_s, \Sigma, \delta_s, q_{s_0})$ be the synchronous product of generators G_p^{np} and G_r^{np}, $G_s = G_p^{np} \parallel G_r^{np}$. There exists a probabilistic supervisor V_p such that $L_p(V_p/G_p) = L_p(G_r)$ iff for all $(q, r) \in Q_s$, the following two conditions hold:

i) $Pos(q) \cap \Sigma_u = Pos(r) \cap \Sigma_u$, and for all $\sigma \in Pos(q) \cap \Sigma_u$,

$$\frac{p_p(q,\sigma)}{\sum_{\alpha \in \Sigma_u} p_p(q,\alpha)} = \frac{p_r(r,\sigma)}{\sum_{\alpha \in \Sigma_u} p_r(r,\alpha)}$$

ii) $Pos(r) \cap \Sigma_c \subseteq Pos(q) \cap \Sigma_c$, and, if $Pos(q) \cap \Sigma_u \neq \emptyset$, then for all $\sigma \in Pos(q) \cap \Sigma_c$,

$$\frac{p_r(r,\sigma)}{p_p(q,\sigma)} \sum_{\alpha \in \Sigma_u} p_p(q,\alpha) + \sum_{\alpha \in Pos(q) \cap \Sigma_c} p_r(r,\alpha) \leq 1.$$

Conditions (i) and (ii) together are necessary and sufficient for the existence of a probabilistic supervisor. The first part of both conditions corresponds to controllability as used in classical supervisory 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. Any possible uncontrollable events are always enabled, hence the ratios of their probabilities remain unchanged. 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 $\sigma \in Pos(q)$ is always enabled. In this case we could add σ to Σ_u and use (i) to derive that $p_p(q,\sigma)/p_p(q,\alpha) = p_r(r,\sigma)/p_r(r,\alpha)$ for an $\alpha \in Pos(q) \cap \Sigma_u$. Reducing probability of the controller enabling σ would have the effect of reducing this ratio on the right side, so we must have $p_p(q,\sigma)/p_p(q,\alpha) \geq p_r(r,\sigma)/p_r(r,\alpha)$ or equivalently $p_r(r,\alpha) \geq (p_r(r,\sigma)/p_p(q,\sigma)) \cdot p_p(q,\alpha)$. Since we assume the systems are nonterminating, we know that:

$$\sum_{\alpha \in \Sigma} p_r(r,\alpha) = \sum_{\alpha \in \Sigma_u} p_r(r,\alpha) + \sum_{\alpha \in \Sigma_c} p_r(r,\alpha) = 1, \text{ so,}$$

$$\frac{p_r(r,\sigma)}{p_p(q,\sigma)} \sum_{\alpha \in \Sigma_u} p_p(r,\alpha) + \sum_{\alpha \in \Sigma_c} p_r(r,\alpha) \leq 1.$$

This last equation simplifies to (ii) with equality holding when event σ is always enabled.

Also, it should be stressed that a special case was implicitly considered in the previous theorem. This special case arises when all the possible events in a state of the plant are controllable. Then, disabling all the events can cause termination. In order to avoid this (as the generators considered do not terminate), there is always at least one event enabled (for details, an interested reader is referred to [23]).

When the conditions are satisfied, a solution to the PSCP exists. The probabilistic supervisor can then be computed by the fixed point iteration algorithm as presented in [22], [23].

III. Problem Formulation

In the case when the conditions for the existence of a solution of the probabilistic supervisory control problem are not satisfied, we search for a suitable approximation. We define the problem as follows.

1) 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 in the nonprobabilistic sense (i.e., $L(V_p/G_p)$ is the supremal controllable sublanguage of $L(G_r)$ with the respect to G_p).
2) The probabilistic behavior of the controlled plant is "as close as possible" to the probabilistic behavior of the requirements specification restricted to the supremal controllable sublanguage of $L(G_r)$ with the respect to G_p.

Let $G = V_p/G_p = (S, \Sigma, \delta, s_0, p)$ be the closest approximation.

The first criterion is straightforward. The requirement G_r represents a safety constraint: the controlled plant is not allowed to generate strings not in $L(G_r)$ even with the smallest of probabilities. Further, the criterion of maximal permissiveness is a standard one for optimality of supervisory control. The second criterion, on the other hand, is probabilistic: a distance in a *pseudometric* between the initial states of the probabilistic generators G and an appropriately modified G_r is chosen as a measure of probabilistic similarity. The requirements specification G_r is modified such that its nonprobabilistic behavior is reduced to the maximal permissible legal nonprobabilistic behavior of the plant under control. In other words, the (nonprobabilistic) language of the modified specification is the supremal controllable sublanguage of $L(G_r)$ with respect to G_p. Consequently, the probabilities of the specification are revised so that the probabilities of the events inadmissible for not satisfying the first criterion are redistributed over the admissible ones. The rationale behind the modified specification is as follows:

- It makes sense for a designer to modify the (probabilistic) requirements specification as she cannot do better in a nonprobabilistic sense. So, after realizing that only a subset of the desired nonprobabilistic behavior is achievable, the designer sees no reason to insist on the probabilities suggested for a behavior that cannot be achieved. We assume that the designer wants to, for each state, distribute the

probabilities of the events not possible anymore over the remaining ones so that the new probabilities are proportional to the old ones. However, the designer might want to rebalance the probabilities any way it suites her. The algorithm to be proposed should be able to handle any such rebalancing.

- Obviously, it might be the case that the designer prefers to leave the specification intact. Then, the problem to solve becomes the OPSCP with criterion (2) modified so that the difference between the controlled plant and the original specification is minimized. As it turns out, the solution of the original OPSCP is an important part en route to the solution of this modified OPSCP. More precisely, with some preprocessing, the algorithm that we propose in Section VI can be reused for the modified OPSCP (for details, see [37]).

IV. Pseudometric for the Closest Approximation

In this section, we first motivate our choice of the pseudometric, especially in the context of related research done on pseudometrics on states of probabilistic systems. The chosen pseudometric is then presented.

A. Probabilistic Pseudometrics

Probabilistic bisimulation, introduced in [43], is commonly used to define an equivalence relation between probabilistic systems. However, probabilistic bisimulation is not a robust relation: the probabilities of corresponding transitions must match exactly.

The formal definition follows and represents a modified version of the definition of probabilistic bisimulation given in [44].

Definition 3: Let $G = (Q, \Sigma, \delta, q_0, p)$ be a PDES. A *probabilistic bisimulation* on Q is the binary relation $\equiv$ such that for any $\sigma \in \Sigma$, and any $q_1, q_2 \in Q$ such that $q_1 \equiv q_2$, the following holds:

1) For every $q_1' \in Q$ such that $\delta(q_1, \sigma) = q_1'$, there is $q_2' \in Q$ such that $\delta(q_2, \sigma) = q_2'$, $p(q_1, \sigma) = p(q_2, \sigma)$, and $q_1' \equiv q_2'$.
2) For every $q_2' \in Q$ such that $\delta(q_2, \sigma) = q_2'$, there is $q_1' \in Q$ such that $\delta(q_1, \sigma) = q_1'$, $p(q_1, \sigma) = p(q_2, \sigma)$, and $q_1' \equiv q_2'$.

States q_1 and q_2 are probabilistic bisimilar if there exists a probabilistic bisimulation $\equiv$ such that $q_1 \equiv q_2$.

As a more robust way to compare probabilistic systems, a notion of *pseudometric* is introduced. 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 $x, y, z \in Q$. A pseudometric generalizes a metric in that two distinct points are allowed to be at the distance 0. If all distances are in $[0, 1]$, the pseudometric is *1-bounded*.

Little work on distance between probabilistic systems has focused on generative models. The first paper that discussed the use of a pseudometric as a way to measure the distance between two probabilistic processes is [45]. This early work considers deterministic generative probabilistic systems. The distance between processes is a number between 0 and 1, and represents a measure of a behavioral proximity between the processes: the smaller the number, the smaller the distance. The work of [14] suggests a pseudometric based on probabilities of occurrence of strings in languages generated by two automata. More precisely, the distance between two automata in the pseudometric is defined as a maximal difference in occurrence probabilities of strings in the corresponding languages. Probabilistic generators are used to model probabilistic systems in [46]. In a symbolic pattern recognition application, a pseudometric is introduced to measure the distance between the original model and the transformed one, where the transformed model has the same long term distribution over the states as the original one.

The work of [24] introduces a pseudometric on states for a large class of probabilistic automata, including reactive and generative probabilistic automata. This metric is inspired by the Kantorovich metric [25] which is used in transport problems, and more recently has been used by Hutchinson in his theory of fractals [26]. The metric is also known as Wasserstein metric [27]. The pseudometric is characterized as the greatest fixed point of a function. Two states are at distance 0 in this pseudometric if and only if they are probabilistic bisimilar. This is the pseudometric we will use in the solution of our problem. The pseudometric has a discount factor $e \in (0, 1]$: the smaller the factor, the greater the discount of future. The work of [24] is closely related to [29]–[35], [38]–[40], [47], which consider reactive systems. In [32], [35], [38]–[40], polynomial algorithms to approximate (with a specified accuracy) the distances in respective pseudometrics (also based on the Kantorovich metric with a discount factor) are presented. The algorithms are applicable for $e \in (0, 1)$. In general, for this type of pseudometric, the value of e determines the topology of the pseudometric: a pseudometric with factor $e \in (0, 1)$ has a different topology than the one with $e = 1$ (see [31]). Further, efficient algorithms typically exist that approximate the distances for $e \in (0, 1)$, while no such algorithms are known to exist for $e = 1$. It is to expect, then, that for $e \in (0, 1)$, there is a simple algorithm to compute distances in our pseudometric for our generative, deterministic model (see Section V). Conveniently enough, the concept of discount has been widely applied in game theory, economics and optimal control [48]. From an engineering point of view, one cares more about an error in the near future than one in the distant future [49]. Also, the pseudometric intuitively matches our notion of the distance between PDES and accounts for all differences between corresponding transition probabilities, as opposed to, for example, that of [45] that, roughly speaking, considers only the maximum of the differences between the corresponding probabilities. Furthermore, as the pseudometric is suggested for a large class of systems, it allows for an extension of our work to, for example, nondeterministic systems. Further, as presented in [36], [37], the pseudometric has both logical and trace characterizations. The logical characterization measures the distance between two systems by a $[0, 1]$-valued formula that distinguishes between the systems the most, while the trace characterization describes the similarity between the probabilistic traces of similar systems. More precisely, the trace characterization shows that the pseudometric measures not only the difference in (appropriately discounted) occurrence probabilities of strings in two systems, but also differences in (appropriately discounted) occurrence probabilities of certain sets of strings as well as some properties of strings.

Our generative model can be transformed to the models of [2], [3], [29], [31], but with a state space expansion by a factor of $O(|\Sigma|)$. However, as the current mathematical apparatus allows for direct reasoning about distance between our generators, no benefits in regards to the optimal supervisory control of PDES would have been gained by a transformation to one of the aforementioned models.

For a more detailed discussion on probabilistic pseudometrics, see [41].

B. Pseudometric

Let $G = (Q, \Sigma, \delta, q_0, p)$ be a PDES, where $Q = \{q_0, q_1, \ldots q_{N-1}\}$. This is the system that will be used throughout the sequel. Our pseudometric is based on the pseudometric suggested in [24] for a large class of automata which includes our generator: the two pseudometrics are the same up to a constant.

First, [24] introduces the class $\mathcal{M}$ of 1-bounded pseudometrics on states with the (partial) ordering

$$d_1 \preceq d_2 \text{ if } \forall q_q, q_r \in Q,\ d_1(q_q, q_r) \geq d_2(q_q, q_r) \qquad (2)$$

as was initially suggested in [30]. It is 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 $\rho_{q_r}(\sigma, q_j)$. Then, the distance between the distributions ρ_{q_q} and ρ_{q_r}, $d(\rho_{q_q}, \rho_{q_r})$ (note the slight abuse of notation) for our generators is given as:

$$\begin{aligned}
&\text{Maximize } \left(\sum_{\substack{\sigma \in \Sigma \\ 0 \leq i \leq N-1}} a_{\sigma,i} \rho_{\sigma,i} \right) - \left(\sum_{\substack{\sigma \in \Sigma \\ 0 \leq i \leq N-1}} a_{\sigma,i} \rho'_{\sigma,i} \right) \\
&\text{subject to} \\
&0 \leq a_{\sigma,i} \leq 1, \quad \sigma \in \Sigma, 0 \leq i \leq N-1 \\
&a_{\sigma,i} - a_{\alpha,j} \leq c_{ij}^{\sigma\alpha}, \quad \sigma, \alpha \in \Sigma, 0 \leq i, j \leq N-1 \\
&\text{where} \\
&c_{ij}^{\sigma\alpha} = \begin{cases} e \cdot d(q_i, q_j), & \text{if } \sigma = \alpha \\ 1, & \text{otherwise.} \end{cases}
\end{aligned} \qquad (3)$$

The dual of this problem is:

$$\begin{aligned}
&\text{Minimize } \left(\sum_{\substack{\sigma, \alpha \in \Sigma \\ 0 \leq i, j \leq N-1}} \lambda_{\sigma,\alpha,i,j} \cdot c_{ij}^{\sigma\alpha} \right) \\
&\text{subject to} \\
&\sum_{\substack{\alpha \in \Sigma \\ 0 \leq j \leq N-1}} \lambda_{\sigma,\alpha,i,j} = \rho_{\sigma,i}, \quad \sigma \in \Sigma, 0 \leq i \leq N-1, \\
&\sum_{\substack{\sigma \in \Sigma \\ 0 \leq i \leq N-1}} \lambda_{\sigma,\alpha,i,j} = \rho'_{\alpha,j}, \quad \alpha \in \Sigma, 0 \leq j \leq N-1, \\
&\lambda_{\sigma,\alpha,i,j} \geq 0, \quad \sigma, \alpha \in \Sigma, 0 \leq i, j \leq N-1
\end{aligned} \qquad (4)$$

where $c_{ij}^{\sigma\alpha}$ is given as in (3). The problem (4) is a minimum cost flow problem in the network made of two copies of state space Q, where one state space represents the set of source nodes, and the other is the set of sink nodes. Let $c_{ij}^{\sigma\alpha}$ be the cost of moving one unit of mass from source state i to sink state j through σ and α. Then, the problem becomes finding the most economical way to transport mass from source nodes to sink nodes such that the mass to be transported from source node i through σ is $\rho_{\sigma,i}$, and the mass to be transported to sink node j through α is $\rho'_{\alpha,j}$.

The pseudometric on states, d_{fp}, is now given as the greatest fixed-point of the following function $\mathcal{D}$ on $\mathcal{M}$ (here, for probabilistic generators, we give a simplified version of that in [24]):

$$\mathcal{D}(d)(q_q, q_r) = d(\rho_{q_q}, \rho_{q_r}), d \in \mathcal{M}, q_q, q_r \in Q. \qquad (5)$$

The definition of the pseudometric on distributions is a modified version of that of [24]: the pseudometric is changed such that the distances between the states in d_{fp} are larger by a factor of $1/e$ than the distances in the pseudometric defined in [24]. This is done so that the distances in our pseudometric are in $[0, 1]$ interval instead of $[0, e]$. A number of existing results can be reused in reasoning about our pseudometric. The distance between distributions (3) is a 1-bounded pseudometric, and is consistent with the ordering (2) (see [30], [32]). The proofs that the function defined by (5) is monotone on $\mathcal{M}$, and that it does have a greatest fixed point originate from [30]. Also, according to Tarski's fixed point theorem, the greatest fixed point of function $\mathcal{D}$ can be reached through an iterative process that starts from the greatest element. As the number of transitions from a state of a probabilistic generator is finite, the greatest fixed point of the function $\mathcal{D}$ is reached after at most ω iterations ([24], [30]), where ω is the first infinite ordinal.

The definition of the pseudometric as a greatest fixed point of function (5) is a quantitative analogue of the definition of (non-probabilistic) bisimilarity as a greatest fixed point of a monotone function. Consequently, the pseudometric is an analogue of probabilistic bisimulation: two states are at distance 0 if and only if they are probabilistic bisimilar (for proof, see [24]). Furthermore, the pseudometric measures behavioral similarity between states: greater distances correspond to greater differences in behavior.

The work of [24] does not offer any algorithms for calculation of their pseudometric. In Section V, two algorithms for calculating our pseudometric are proposed.

V. Calculating the Pseudometric

As a prelude to the solution of the optimal approximation problem, two algorithms that calculate/approximate distances in the pseudometric d_{fp} are suggested. First, the function $\mathcal{D}$ is simplified, and then the algorithms are described, and their correctness proven.

A. Simplifying Function $\mathcal{D}$ for Deterministic Generators

The function that represents the pseudometric on distributions is defined as the linear programming problem (3). We now show that, for deterministic generators, this function, and consequently, function $\mathcal{D}$ as defined by (5), can be simplified by explicitly solving the linear programming problem (3).

First, recall that our generators are deterministic: for an event σ and a state q, there is at most one state q' such that $q' =$

$\delta(q,\sigma)$. For the purposes of the following analysis of our deterministic generators, we rewrite the objective function of the optimization problem of (3) as:

$$\sum_{\sigma\in\Sigma}(a_{\sigma,i(q_q,\sigma)}\rho_{\sigma,i(q_q,\sigma)} - a_{\sigma,j(q_r,\sigma)}\rho'_{\sigma,j(q_r,\sigma)}) \tag{6}$$

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 $i \in \{1,\ldots,N-1\}$. 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. For readability purposes, we will write i instead of $i(q_q,\sigma)$, and j instead of $j(q_r,\sigma)$.

We are now ready to state our first result.

Lemma 1: Let $G = (Q,\Sigma,\delta,q_0,p)$ be a PDES. Then, the function $\mathcal{D}$ simplifies to:

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

where, again, $c_{ij} = e\cdot d(q_i,q_j)$ as before, and i and j denote $i(q_q,\sigma)$ and $j(q_r,\sigma)$, respectively, as defined in (6).

Proof: The objective function (6) can be maximized by maximizing each of its summands separately. In order to explain this observation, we consider a summand $a_{\sigma,i}\rho_{\sigma,i} - a_{\sigma,j}\rho'_{\sigma,j}$. Due to the generator's determinism, there is no other nonzero summand containing $a_{\sigma,k}$, $0 \le k \le N-1$, $k \ne i$, $k \ne j$. Therefore, the last constraint of (3) for any two coefficients $a_{\sigma,i}$ and $a_{\alpha,j}$ $(0 \le i,j \le N-1)$ from different summands becomes $a_{\sigma,i} - a_{\alpha,j} \le 1$. This constraint is already implied by the first constraint, so we can independently pick the coefficients a in different summands, and, consequently, independently maximize the summands in order to maximize the sum.

In order to maximize a summand of the objective function (6), we solve the following linear programming problem for $\sigma \in \Sigma$:

$$\begin{aligned}&\text{Maximize } (a_{\sigma,i}\rho_{\sigma,i} - a_{\sigma,j}\rho'_{\sigma,j})\\ &\text{subject to } 0 \le a_{\sigma,i}, a_{\sigma,j} \le 1, \quad a_{\sigma,i} - a_{\sigma,j} \le c_{ij}\end{aligned}$$

where i and j are defined as in (6), and $c_{ij} = ed(q_i,q_j)$ as before. Also, note that the set of constraints does not contain the inequality $a_{\sigma,j} - a_{\sigma,i} \le c_{ji}$. In order to maximize the given function, the coefficient $a_{\sigma,i}$ is to be chosen to be greater than $a_{\sigma,j}$ since the given constraints allow it. In that case, since $c_{ij} = c_{ji}$, if $a_{\sigma,i} - a_{\sigma,j} \le c_{ij}$, then $a_{\sigma,j} - a_{\sigma,i} \le c_{ji}$ follows, so the latter constraint is redundant. Further, it is not hard to see that the solution of the given linear programming problem for $\rho_{\sigma,i} \ge \rho'_{\sigma,j}$ is equal to $\rho_{\sigma,i} - \rho'_{\sigma,j} + c_{ij}\rho'_{\sigma,j}$. We can solve this problem using graphical method, simplex method or using the following line of reasoning. In order to maximize the given function, we can either choose $a_{\sigma,i}$ to be 1 and then pick $a_{\sigma,j}$ so that it has the minimal value for the given constraints, or we choose $a_{\sigma,j}$ to be 0, and then pick $a_{\sigma,i}$ so that it has the maximal value under the given constraints. In the first case, we pick $a_{\sigma,i}$ to be 1, $a_{\sigma,j}$ to be $1 - c_{ij}$, and value of the objective function is $\rho_{\sigma,i} - \rho'_{\sigma,j} + c_{ij}\rho'_{\sigma,j}$. In the second case, since $a_{\sigma,j}$ is 0, then $a_{\sigma,i}$ is equal to c_{ij}, and the objective function becomes $c_{ij}\rho_{\sigma,i}$. The latter is our solution, since $c_{ij}\rho_{\sigma,i} = c_{ij}\left(\rho_{\sigma,i} - \rho'_{\sigma,j} + \rho'_{\sigma,j}\right) \le \rho_{\sigma,i} - \rho'_{\sigma,j} + c_{ij}\rho'_{\sigma,j}$ (for $\rho_{\sigma,i} \ge \rho'_{\sigma,j}$ and $c_{ij} \in [0,1]$). Using the same reasoning, for $\rho_{\sigma,i} < \rho'_{\sigma,j}$, the maximum is reached at $(a_i,a_j) = (c_{ij},0)$ and its value is $c_{ij}\rho_{\sigma,i}$.

Now, we put together the presented solution of the linear programming problem (3). The distance between the distributions ρ_{q_q} and ρ_{q_r} is then:

$$d(\rho_{q_q},\rho_{q_r}) = \sum_{\sigma\in\Sigma} f(d,q_q,q_r,\sigma), \text{ where}$$

$$f(d,q_q,q_r,\sigma) = \begin{cases}\rho_{\sigma,i} - \rho'_{\sigma,j} + c_{ij}\rho'_{\sigma,j}, & \text{if } \rho_{\sigma,i} \ge \rho'_{\sigma,j}\\ c_{ij}\rho_{\sigma,i}, & \text{otherwise}\end{cases}$$

or, equivalently,

$$f(d,q_q,q_r,\sigma) = \max(\rho_{\sigma,i} - \rho'_{\sigma,j} + c_{ij}\rho'_{\sigma,j}, c_{ij}\rho_{\sigma,i})$$

where $c_{ij} = e\cdot d(q_i,q_j)$ as before, and i and j denote $i(q_q,\sigma)$ and $j(q_r,\sigma)$, respectively, as defined as in (6).

To summarize, the function $\mathcal{D}(d)$ for our model is given as:

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

where, again, $c_{ij} = e\cdot d(q_i,q_j)$ as before, and i and j denote $i(q_q,\sigma)$ and $j(q_r,\sigma)$, respectively, as defined in (6). ■

As stated in Section IV-B, the pseudometric d_{fp} is now characterized as the greatest fixed point of function $\mathcal{D}$.

B. Calculating the Pseudometric

For $e \in (0,1)$, we will prove that the function $\mathcal{D}$ has only one fixed point, d^*, and, consequently, $d_{fp} = d^*$. Then, two algorithms for calculating the distances in pseudometric d_{fp} are suggested.

First, some useful definitions and results from linear algebra are introduced.

A real $n \times n$ matrix $A = (a_{ij})$ defines a linear mapping from $\mathbb{R}^n$ to $\mathbb{R}^n$, and we will write $A \in L(\mathbb{R}^n)$ to denote either the matrix or linear function, as no distinction between the two is necessary. Also, the absolute value of column vector $x = (x_1,\ldots,x_n)^T \in \mathbb{R}^n$ will be denoted by $|x|$, and defined as $|x| = (|x_1|,\ldots,|x_n|)^T$. A partial ordering on $\mathbb{R}^n$ is defined as the product order:

$$\forall x,y \in \mathbb{R}^n \quad x \le y \Leftrightarrow (\forall i = 1,\ldots,n \quad x_i \le y_i)$$

Definition 4: For any complex $n \times n$ matrix A, the spectral radius of A is defined as the maximum of $|\lambda_1|,\ldots,|\lambda_n|$, where $\lambda_1,\ldots,\lambda_n$ are the eigenvalues of A.

The spectral radius of A, denoted $\varphi(A)$, satisfies $\varphi(A) \le \|A\|$, where $\|A\|$ is an arbitrary norm on $\mathbb{R}^n$. During the course of the following proof we will make use of the infinity norm $\|A\|_\infty = \max_{1\le i\le n}\sum_{j=1}^n |a_{ij}|$. Also, the proof will use functions $div : N\times N \to N$ and $mod : N\times N \to N$ defined in the standard manner to be the quotient and remainder, respectively, of the division of the first argument by the second.

Definition 5 ([50]): An operator $G : D \subseteq \mathbb{R}^n \to \mathbb{R}^n$ is called a *P-contraction* on a set $D_0 \subseteq D$ if there exists a linear operator $P \in L(\mathbb{R}^n)$ such that $P \ge 0$, $\varphi(P) < 1$ and

$$|G(x) - G(y)| \le P|x-y| \text{ for all } x,y \in D_0.$$

Now, let $d \in \mathcal{M}$. Next, we define the function $\mathcal{V} : \mathcal{M} \to [0,1]^{N^2}$:

$$\mathcal{V}(d) = (d(q_0,q_0), d(q_0,q_1), \ldots, d(q_{N-1},q_{N-2}), d(q_{N-1},q_{N-1}))^T.$$

Note that the vector $\mathcal{V}(d)$ could be further cut down, as $d(s,s) = 0$ and $d(s,t) = d(t,s)$ for any $s, t \in Q$. However, for ease of presentation, we will not decrease the size of the vector. Therefore, $\mathcal{V}(d) = (\mathcal{V}_1(d), \mathcal{V}_2(d), \ldots, \mathcal{V}_{N^2}(d))^T$, where $\mathcal{V}_k(d)$ for $k \in \{1, \ldots, N^2\}$ is given as:

$$\mathcal{V}_k(d) = d(q_i, q_j),\ i = k\ div\ (N+1),\ j = (k-1)\ mod\ N.$$

Now, the function $\mathcal{D}$ is redefined in a natural way as $\mathcal{D}(\mathcal{V}(d)) = (\mathcal{D}_1(\mathcal{V}(d)), \ldots, \mathcal{D}_{N^2}(\mathcal{V}(d)))^T$, where for any $k \in \{1, \ldots, N^2\}$:

$$\begin{aligned} \mathcal{D}_k(\mathcal{V}(d)) &= d(\rho_{q_i}, \rho_{q_j}), \\ i &= k\ div\ (N+1), \\ j &= (k-1)\ mod\ N. \end{aligned} \tag{7}$$

Further, let $D_0 = \{\mathcal{V}(d) | d \in \mathcal{M}\}$.

Lemma 2: The function $\mathcal{D}$ is a P-contraction on D_0.

Proof: Let $d', d'' \in \mathcal{M}$, and $\mathfrak{d}' = \mathcal{V}(d')$, and $\mathfrak{d}'' = \mathcal{V}(d'')$. Let $k \in \{1, \ldots, N^2\}$, and let i and $j (0 \le i, j \le N-1)$ be given as in (7). Also, let $t(i,\sigma) = t$ such that $\delta(q_i, \sigma) = q_t$ if $\delta(q_i,\sigma)!$, and $t(i,\sigma) = 0$, otherwise. Similarly, $l(j,\sigma) = l$ such that $\delta(q_j,\sigma) = q_l$ if $\delta(q_j,\sigma)!$, and $l(j,\sigma) = 0$, otherwise. Again, for notational convenience, we will write t instead of $t(i,\sigma)$, and l instead of $l(j,\sigma)$. Also, we will write $\rho_{\sigma,t}$ instead of $\rho_{q_i}(\sigma, q_t)$, and, similarly, $\rho'_{\sigma,l}$ instead of $\rho_{q_j}(\sigma, q_l)$ for $q_t, q_l \in Q$. Then:

$$\begin{aligned} &|\mathcal{D}_k(\mathfrak{d}') - \mathcal{D}_k(\mathfrak{d}')| \\ &= \left|d'(\rho_{q_i}, \rho_{q_j}) - d''(\rho_{q_i}, \rho_{q_j})\right| \\ &= \Bigg|\sum_{\sigma\in\Sigma} \max(\rho_{\sigma,t} - \rho'_{\sigma,l} + ed'(q_t,q_l)\rho'_{\sigma,l}, ed'(q_t,q_l)\rho_{\sigma,t}) \\ &\qquad - \sum_{\sigma\in\Sigma} \max(\rho_{\sigma,t} - \rho'_{\sigma,l} + ed''(q_t,q_l)\rho'_{\sigma,l}, ed''(q_t,q_l)\rho_{\sigma,t})\Bigg| \\ &= \Bigg|\sum_{\sigma\in\Sigma} (\max(\rho_{\sigma,t} - \rho'_{\sigma,l} + ed'(q_t,q_l)\rho'_{\sigma,l}, ed'(q_t,q_l)\rho_{\sigma,t}) \\ &\qquad - \max(\rho_{\sigma,t} - \rho'_{\sigma,l} + ed''(q_t,q_l)\rho'_{\sigma,l}, ed''(q_t,q_l)\rho_{\sigma,t}))\Bigg| \\ &= \left|\sum_{\sigma\in\Sigma} e(d'(q_t,q_l) - d''(q_t,q_l)) \min(\rho_{\sigma,t}, \rho'_{\sigma,l})\right| \\ &\left(\begin{array}{l} \text{since, for any } d \in \mathcal{M}, \text{it holds that:} \\ \max(\rho_{\sigma,t} - \rho'_{\sigma,l} + ed(q_t,q_l)\rho'_{\sigma,l}, ed(q_t,q_l)\rho_{\sigma,t}) \\ = \begin{cases} \rho_{\sigma,t} - \rho'_{\sigma,l} + ed(q_t,q_l)\rho'_{\sigma,l}, & \text{if } \rho_{\sigma,t} \ge \rho'_{\sigma,l} \\ ed(q_t,q_l)\rho_{\sigma,t}, & \text{otherwise} \end{cases} \end{array}\right) \\ &\le e \sum_{\sigma\in\Sigma} \min(\rho_{\sigma,t}, \rho'_{\sigma,l}) |d'(q_t,q_l) - d''(q_t,q_l)| \qquad (8) \\ &\le e \sum_{\substack{\sigma\in\Sigma \\ m = tN+l+1}} \min(\rho_{\sigma,t}, \rho'_{\sigma,l}) |\mathfrak{d}'_m - \mathfrak{d}''_m|. \qquad (9) \end{aligned}$$

Note that $t = t(i,\sigma)$ and $l = l(j,\sigma)$ are also functions of k (since i and j are functions of k). Now, without the explicit construction of matrix P, we can see from (9) that there exists P such that $|\mathcal{D}(\mathfrak{d}') - \mathcal{D}(\mathfrak{d}'')| \le P|\mathfrak{d}' - \mathfrak{d}''|$ where

$$\begin{aligned} \|P\|_\infty &= \max_k \{e \sum_{\sigma\in\Sigma} \min(\rho_{\sigma,t}, \rho'_{\sigma,l})\} \\ &\le e\ (\text{since} \sum_{\substack{\sigma\in\Sigma \\ t\in\{0,\ldots,N-1\}}} \rho_{\sigma,t} = 1, \sum_{\substack{\sigma\in\Sigma \\ l\in\{0,\ldots,N-1\}}} \rho'_{\sigma,l} = 1) \\ &< 1\ (\text{since } e \in (0,1)). \end{aligned}$$

Therefore, $\varphi(P) < 1$ and, since, obviously, $P \ge 0$, then $\mathcal{D}$ is P-contraction. ∎

Lemma 3: Let $d', d'' \in \mathcal{M}$, and $\mathfrak{d}' = \mathcal{V}(d')$, and $\mathfrak{d}' = \mathcal{V}(d'')$. For any $k \in \{1, \ldots, N^2\}$, there exists $m \in \{1, \ldots, N^2\}$ such that:

$$|\mathcal{D}_k(\mathfrak{d}') - \mathcal{D}_k(\mathfrak{d}'')| \le e\,|\mathfrak{d}'_m - \mathfrak{d}''_m|$$

Proof: We use the notation from Lemma 2.

$$\begin{aligned} &|\mathcal{D}_k(\mathfrak{d}') - \mathcal{D}_k(\mathfrak{d}'')| \\ &\le e \sum_{\sigma\in\Sigma} \min(\rho_{\sigma,t}, \rho'_{\sigma,l}) |d'(q_t,q_l) - d''(q_t,q_l)| \ \text{(from (8))} \\ &\le e \sum_{\sigma\in\Sigma} \min(\rho_{\sigma,t}, \rho'_{\sigma,l}) \max_{\substack{(t,l)\in \\ \{(t(\sigma,i),l(\sigma,j))|\sigma\in\Sigma\}}} \{|d'(q_t,q_l) - d''(q_t,q_l)|\} \\ &\le e \sum_{\sigma\in\Sigma} \min(\rho_{\sigma,t}, \rho'_{\sigma,l}) |d'(q_r,q_s) - d''(q_r,q_s)| \\ &\text{for some } r, s \in \{0, \ldots, N-1\} \\ &\le e|d'(q_r,q_s) - d''(q_r,q_s)| \\ &\left(\text{since} \sum_{\substack{\sigma\in\Sigma \\ t\in\{0,\ldots,N-1\}}} \rho_{\sigma,t} = \sum_{\substack{\sigma\in\Sigma \\ l\in\{0,\ldots,N-1\}}} \rho'_{\sigma,l} = 1\right) \\ &\le e\,|\mathfrak{d}'_m - \mathfrak{d}''_m| \ \text{for some } m \in \{1, \ldots, N^2\}. \end{aligned}$$

∎

Theorem 2: For any $\mathfrak{d}^0 \in D_0$, the sequence

$$\mathfrak{d}^{n+1} = \mathcal{D}(\mathfrak{d}^n), \quad n = 0, 1, \ldots$$

converges to the unique fixed point of $\mathcal{D}$ in D_0, $\mathfrak{d}^*$, and the error estimate is given componentwise ($k \in \{1, \ldots, N^2\}$) as:

$$|\mathfrak{d}^n_k - \mathfrak{d}^*_k| \le e^n,\ n = 0, 1, \ldots \tag{10}$$

Proof: Note that this is a variant of the contraction-mapping theorem extended to P-contractions [50, Theorem 13.1.2]. A similar proof technique is employed. For details, see the Appendix. ∎

Now, using the presented analysis, the following two algorithms for the calculation of the distances between the states of PDES in the chosen pseudometric are suggested.

Algorithm 1: Theorem 2 proves that the system of equations

$$\mathfrak{d} = \mathcal{D}(\mathfrak{d}) \tag{11}$$

has a unique solution. The system (11) is a system of linear equations. Therefore, the system (11) can be rewritten into the standard form $A\mathfrak{d} = b$, where A is a $N^2 \times N^2$ matrix and b is a column vector of dimension N^2. Therefore, the distances in the pseudometric d_{fp} can be calculated by solving this system of linear equations.

Algorithm 2: Theorem 2 also suggests an iterative algorithm to approximate distances in the pseudometric d_{fp} between the states of a probabilistic generator. Let $d^0(q_q, q_r) = 0$ for any two states $q_q, q_r \in Q$. As before, let ρ_{q_q} and ρ_{q_r} be the distributions induced by the states q_q and q_r, respectively. The n-th iteration of the algorithm calculates the distance d^n between each two states $q_q, q_r \in Q$:

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

where $c_{ij} = e \cdot d^{n-1}(q_i, q_j)$, and $i = i(q_q, \sigma)$ and $j = j(q_r, \sigma)$ are defined as in (6). The accuracy of the solution found at the n-th iteration is e^n.

The pseudometric of [24] is closely related to the ones suggested in [32], [35], [38]–[40]. It is not a surprise then that our iterative algorithm turns out to be similar to those of [32], [35], [38]–[40] that calculate distances in similar pseudometrics suggested for different kinds of probabilistic system. Those algorithms are similar to ours in that they all approximate the distances by fixed point iterations. In each iteration, however, the algorithms of [32], [35], [38]–[40] solve a special case of the linear programming problem—the transshipment problem—for each pair of states. The transshipment problem can be solved in polynomial time. On the other hand, in each iteration, for each pair of states, our algorithm simply evaluates an expression. The evaluation is done in linear time. The simplification is possible due to the deterministic nature of our generators. Also, while our algorithm is derived from the characterization of the pseudometric as the greatest fixed point of a monotone function, the pseudometric of [32], [35] is defined as the metric kernel induced by the unique map from the probabilistic transition system, viewed as a coalgebra, to the terminal coalgebra. In that regard, the derivation of [39], [40] is more similar to ours since they start from the fixed point characterization, and then use the Banach fixed point theorem, whereas we use its generalization to P-contractions.

The iterative method can be useful for systems with large N^2, where the direct method can be rather expensive. More importantly, the mathematical apparatus used to reach the iterative method will be reused in the solution of the OPSCP in Section VI.

An important feature of d_{fp} is to be noted: metric d_{fp} is defined on any two states of a single PDES, not on two states that belong to different PDES. In order to define the distance between two PDES (with disjoint sets of states) as the distance between their initial states, a new PDES is created that represents the union of the two PDES (the union is defined in a natural way as will be presented formally in Section VI-B).

Also, it should be stressed that the presented algorithm works for $e \in (0, 1)$. However, [51] presented an algorithm for calculating distances in the pseudometric of [31] for a variant of Markov chains for the case when $e = 1$. The key element in the algorithm is Tarski's decision procedure for the first order theory of real closed fields. We believe that this algorithm can be modified to calculate d_{fp} between the states of probabilistic generators. However, as this algorithm is impractical, an efficient calculation of the pseudometric for $e = 1$ is still an open problem.

VI. Closest Approximation: Algorithm

In this section the algorithm that solves the closest approximation problem is presented. All the results in the sequel are applicable for $e \in (0, 1)$.

First, the formulation of the closest approximation problem is repeated. Assume that the plant is given as PDES $G_p = (Q_p, \Sigma, \delta_p, q_{p_0}, p_p)$, and the requirements specification is given as $G_r = (Q_r, \Sigma, \delta_r, q_{r_0}, p_r)$. If there is no probabilistic supervisor V_p such that $L_p(V_p/G_p) = L_p(G_r)$, the optimal solution is sought. The solution is optimal in the following sense. It is assumed that the nonprobabilistic language of the requirement is a safety requirement: no other strings are allowed in the plant. Then, it is required that maximal permissible behavior (in the nonprobabilistic sense) is achieved. In this case, the probabilistic behavior of the controlled plant should be as close as possible to the requirements specification that is now normalized so that it is constrained to the supremal controllable nonprobabilistic language. The proposed algorithm uses this separation of probabilistic and nonprobabilistic aspects of optimality so that it deals with each aspect separately: the first part handles the "nonprobabilistic optimality," and the second part handles the "probabilistic optimality."

A. Part I: Nonprobabilistic Optimality

Before we start looking for the closest approximation in the sense of probability matching, we resort to the classical supervisory theory of supremal controllable languages. First, the classical controllability condition that corresponds to the first parts of conditions (i) and (ii) of Theorem 1 is checked while constructing $L(G_p) \cap L(G_r)$. Then, if the condition is not satisfied, the goal is to find K, the deadlock-free supremal controllable sublanguage of $L(G_p) \cap L(G_r)$ (with respect to G_p). The language K is required to be deadlock-free as termination is not allowed. Then, the DES that represents this language K, further equipped with distribution p_p (appropriately normalized) becomes the modified plant PDES G_1. Also, a DES corresponding to language K equipped with the distribution p_r appropriately normalized, becomes the desired behavior PDES G_2. Formally, let the reachable and deadlock-free DES $G_{1a} = (T, \Sigma, \zeta, t_0)$ represent language K. We define a PDES $G_1 = (T, \Sigma, \zeta, t_0, p_1)$, where the distribution $p_1 : T \times \Sigma \to [0, 1]$, for any $q \in T$, $\sigma \in \Sigma$, is defined as:

$$p_1(q, \sigma) = \frac{p_p(q_p, \sigma)}{\sum\limits_{\sigma \in \{\sigma \in \Sigma | \zeta(q,\sigma)!\}} p_p(q_p, \sigma)}$$

where $q_p = \delta_p(q_{p_0}, s)$ for any $s \in K$ such that $q = \zeta(t_0, s)$.

Similarly, let $G_{2a} = (Q, \Sigma, \delta, q_0)$ be a DES isomorphic to G_{1a} up to renaming of states, and, without loss of generalization, assume $T \cap Q = \emptyset$. Obviously, the nonprobabilistic language generated by G_{2a} is K, too. Similarly, we define a PDES $G_2 = (Q, \Sigma, \delta, q_0, p)$ where the distribution $p : Q \times \Sigma \to [0, 1]$, for any $q \in Q$, $\sigma \in \Sigma$, is defined as:

$$p(q, \sigma) = \frac{p_r(q_r, \sigma)}{\sum_{\sigma \in \{\sigma \in \Sigma | \delta(q,\sigma)!\}} p_r(q_r, \sigma)}$$

where $q_r = \delta_r(q_{r_0}, s)$ for any $s \in K$ such that $q = \delta(q_0, s)$. Note that p_1 and p are well-defined as no state minimization on automaton representing language K is performed.

B. Part II: Probabilistic Optimality

Now, the probability matching equations and inequalities from Theorem 1 are checked. If they are not satisfied (i.e., there is no probabilistic supervisor V_p such that $L_p(V_p/G_1) = L_p(G_2)$), the goal is to find $G_2' = (Q', \Sigma, \delta', q_0', p')$ such that there exists a probabilistic supervisor V_p so that $L_p(V_p/G_1) = L_p(G_2')$ holds, and G_2' is closest to G_2 in our chosen pseudometric. Without loss of generality, it is assumed that $Q \cap Q' = \emptyset$. Also, without loss of generality, it is assumed that the nonprobabilistic automata underlying G_2 and G_2' are isomorphic (with labeling of events being preserved). Therefore, the nonprobabilistic automata underlying G_2 and G_2' are identical up to renaming of states. This assumption is not restrictive as there cannot be any string in the desired system that does not belong to K, and, therefore, since $K = L(G_2)$, there cannot be any string in the desired system that does not belong to $L(G_2)$. This comes from the fact that $L(G_2)$ is the reachable and deadlock-free supremal controllable sublanguage: if any string not in $L(G_2)$ would be allowed in the controlled plant, either the safety or nontermination requirement might not be met. As our pseudometric is defined on the states of a single system, in order to define distances between the states of different systems, namely G_2 and G_2', the union PDES $G_u = (Q \cup Q', \Sigma, \delta_u, q_0, p_u)$ is considered, where for $\sigma \in \Sigma$ and $q \in Q \cup Q'$:

$$\delta_u(q, \sigma) = \begin{cases} \delta(q, \sigma), & \text{if } q \in Q \\ \delta'(q, \sigma), & \text{otherwise} \end{cases}$$

$$p_u(q, \sigma) = \begin{cases} p(q, \sigma), & \text{if } q \in Q \\ p'(q, \sigma), & \text{otherwise.} \end{cases}$$

First, note that, considering the isomorphism between G_2^{np} and $G_2'^{np}$, only the distances between (probability measures on) states $q \in Q$ of G_2 and $q' = f(q) \in Q'$ of G_2' are of interest, where f is the isomorphism between G_2^{np} and $G_2'^{np}$. Also, let h be an isomorphism between G_2^{np} and G_1^{np}. It is assumed that PDES G_2 is in state q after the occurrence of string $s \in L(G_2)(\delta(q_0, s) = q)$. Then, the closest approximation G_2' is in state q', respectively, where $q' = f(q)$. Let ρ_q be the probability distribution induced by the state $q \in Q$ of PDES G_2 and let $\rho'_{q'}$ be the probability distribution induced by the state $q' \in Q'$ of PDES G_2'.

Next, a class $\mathcal{A}$ of partial functions $a : Q \times Q' \to [0, 1]$ is defined, such that $\forall q \in Q$, $q' = f(q) \in Q'$, $a(q, q') = d(q, q')$, where $d \in \mathcal{M}$. Therefore, the class $\mathcal{A}$ is the class of all 1-bounded pseudometrics with domain reduced to $Q \times Q'$, and only distances between $q \in Q$ and $q' = f(q) \in Q'$ defined since the algorithm is independent of the distance between the other states. Next, we define a family Δ as a set of probability distributions on $\Sigma \times Q'$. Now, for each $\rho' \in Q' \to \Delta$, we define function $\mathcal{D}^{\rho'} : \mathcal{A} \to \mathcal{A}$ as ($q \in Q$, $q' = f(q) \in Q'$, $d \in \mathcal{A}$):

$$\mathcal{D}^{\rho'}(d)(q, q') = d(\rho_q, \rho'_{q'}) \text{ and } \rho'(q') = \rho'_{q'},$$

where, as before, d is lifted to the pseudometric on distributions, and $d(\rho_q, \rho'_{q'})$ is defined as in (5). Also, the reversed ordering on $\mathcal{A}$ is introduced to match the one in (2):

$$d_1 \preceq' d_2 \text{ if } \forall q \in Q,\ q' = f(q),\ d_1(q, q') \geq d_2(q, q').$$

The fact that $(\mathcal{A}, \preceq')$ is a complete lattice follows from the fact that $(\mathcal{M}, \preceq)$ is a complete lattice. Further, for each $\rho' \in Q' \to \Delta$, we define function $d_{fp}^{\rho'}$ as the greatest fixed point of function $\mathcal{D}^{\rho'}$. The problem of finding the optimal approximation reduces now to finding $\rho'_m \in Q' \to \Delta$ such that $d_{fp}^{\rho'_m}(q_0, q_0') = \min_{\rho'} \in \{d_{fp}^{\rho'}(q_0, q_0') | \rho' \in Q' \to \Delta\}$ and the conditions for the existence of a probabilistic supervisor of Theorem 1 are satisfied. It follows straight from the definitions of $\mathcal{D}^{\rho'}$ and $d_{fp}^{\rho'}$ that, for any $\rho' \in Q' \to \Delta$, $q \in Q$, $q' = f(q) \in Q'$, the distances $d_{fp}^{\rho'}(q, q')$ are distances in our pseudometric.

We assume that $T = \{t_0, t_1, \ldots t_{N-1}\}$, $Q = \{q_0, q_1, \ldots q_{N-1}\}$, and $Q' = \{q_0', q_1', \ldots q_{N-1}'\}$, where $q_i' = f(q_i)$, $t_i = h(q_i)$, $i = 0, \ldots, N-1$. Note that, for probability distributions, a different notation will be used than the one used in the previous section. Let $d \in \mathcal{A}$, $0 \leq i \leq N-1$, $\Psi(q_i) = Pos(q_i)$, $\Psi_u(q_i) = Pos(q_i) \cap \Sigma_u$, and $\Psi_c(q_i) = Pos(q_i) \cap \Sigma_c$. Also, we will write j for $j(i, \sigma)$, then $\rho_{q_i,\sigma}$ instead of $\rho_{q_i}(\sigma, q_k)$, and $\rho'_{q_i',\sigma}$ instead of $\rho'_{q_i'}(\sigma, q_k')$, $k = 0, 1, \ldots N-1$. Now, the function $\mathcal{P} : \mathcal{A} \to \mathcal{A}$ is defined as:

$$\mathcal{P}(d)(q_i, q_i') = \underset{\rho'_{q_i,\sigma}}{\text{Minimize}} \sum_{\sigma \in \Psi(q_i)} \max(\rho_{q_i,\sigma} - \rho'_{q_i',\sigma} + c_j \rho'_{q_i',\sigma}, c_j \rho_{q_i,\sigma}) \tag{12}$$

where $c_j = e \cdot d(q_j, q_j')$ s.t. $q_j = \delta(q_i, \sigma)$

subject to

$$\frac{p_1(t_i, \sigma)}{\sum_{\alpha \in \Psi_u(q_i)} p_1(t_i, \alpha)} = \frac{\rho'_{q_{i'},\sigma}}{\sum_{\alpha \in \Psi_u(q_i)} \rho'_{q_i',\alpha}}, \quad \sigma \in \Psi_u(q_i) \tag{13}$$

$$\frac{\sum_{\alpha \in \Psi_u(q_i)} p_1(t_i, \alpha)}{p_1(t_i, \sigma)} \rho'_{q_{i'},\sigma} + \sum_{\alpha \in \Psi_c(q_i)} \rho'_{q_i,\alpha} \leq 1, \quad \sigma \in \Psi_c(q_i) \tag{14}$$

$$\sum_{\alpha \in \Psi(q_i)} \rho'_{q_i',\alpha} = 1 \tag{15}$$

$$\rho'_{q_i',\sigma} \geq 0, \quad \sigma \in \Psi(q_i). \tag{16}$$

Constraints (13) and (14) represent the conditions for the existence of probabilistic supervisor given by Theorem 1. The function $\mathcal{P}$ is well-defined since, if $\rho'_{q_i',\sigma} = p_1(t_i, \sigma)$ for all

$\sigma \in \Sigma$, the constraints (13)–(16) are satisfied. Therefore, the optimization problem has a feasible origin. Since $\mathcal{A}$ is a complete lattice, and the function $\mathcal{P}$ can be easily shown to be monotone, it has a greatest fixed point. Next, a useful lemma is stated.

Lemma 4: Let $(\mathcal{L}, \preceq)$ be a complete lattice, and let $f, g : \mathcal{L} \to \mathcal{L}$ be two monotone functions such that $\forall x \in \mathcal{L} : g(x) \preceq f(x)$. Let $gfp(f)$ and $gfp(g)$ denote the greatest fixed point of functions f and g, respectively. Then, $gfp(g) \preceq gfp(f)$.

Proof: See the Appendix. ∎

Obviously, because of the definition of function $\mathcal{P}$, for any function $\mathcal{D}^{\rho'}$, where $\rho' \in Q' \to \Delta$, it holds that $\forall d \in \mathcal{A}$, $\mathcal{D}^{\rho'}(d) \preceq' \mathcal{P}(d)$. Using Lemma 4, we conclude that the greatest fixed point of $\mathcal{P}$ is greater than or equal to any $d_{fp}^{\rho'}, \rho' \in Q \to \Delta$. This greatest fixed point corresponds to the minimal distance between q_0 and q_0' because of the reversed ordering on $\mathcal{A}$. Therefore, the greatest fixed point of function $\mathcal{P}$ corresponds to the distances in our pseudometric where the distance between q_0 and q_0' is minimized under the conditions of Theorem 1 for the existence of a probabilistic supervisor. Consequently, the values of decision variables $\rho'_{q'}$ for $q' \in Q'$ when the greatest fixed point of $\mathcal{P}$ is reached correspond to the statewise probability distributions of the optimal approximation.

We suggest an iterative algorithm to calculate the minimum achievable distance (i.e., the only fixed point of the function $\mathcal{P}$) up to a desired accuracy and provide the probability distribution of the system's achievable behavior when this distance is reached. The proof pattern used for the algorithm from Section V-B is followed. However, as mentioned before, the only relevant distances are the ones between $q \in Q$ and $q' = f(q) \in Q'$.

Let $d \in \mathcal{A}$. Again, we assume that $Q = \{q_0, q_1, \dots q_{N-1}\}$, and $Q' = \{q_0', q_1', \dots q_{N-1}'\}$, where $q_i' = f(q_i), i = 0, \dots, N-1$. Further, let us define function $\hat{\mathcal{V}} : \mathcal{M} \to [0,1]^N$ as:

$$\hat{\mathcal{V}}(d) = (d(q_0, q_0'), d(q_1, q_1'), \dots, d(q_{N-1}, q_{N-1}'))^T.$$

Therefore, $\hat{\mathcal{V}}(d) = (\hat{\mathcal{V}}_1(d), \dots, \hat{\mathcal{V}}_N(d))^T$, where, for $k = 1, \dots, N$:

$$\hat{\mathcal{V}}_k(d) = d(q_{k-1}, q_{k-1}').$$

The function $\mathcal{P}$ is redefined in a natural way as $\mathcal{P}(\hat{\mathcal{V}}(d)) = (\mathcal{P}_1(\hat{\mathcal{V}}(d)), \dots, \mathcal{P}_N(\hat{\mathcal{V}}(d)))^T$, where for any $k \in \{1, \dots, N\}$:

$$\mathcal{P}_k(\hat{\mathcal{V}}(d)) = \mathcal{P}(d)(q_{k-1}, q_{k-1}'),$$

where $q_{k-1}' = f(q_{k-1})$. Also, let $P_0 = \{\hat{\mathcal{V}}(d) | d \in \mathcal{A}\}$.

Theorem 3: Function $\mathcal{P}$ is P-contractive on P_0.

Proof: Let $d', d'' \in \mathcal{A}$, and $\hat{\mathfrak{d}}' = \hat{\mathcal{V}}(d')$, and $\hat{\mathfrak{d}}'' = \hat{\mathcal{V}}(d'')$. Next, for $q \in Q$, $q' = f(q) \in Q'$, we define set $\Phi(q)$ to be the set of all distributions $\rho'_{q'}$ that satisfy conditions given by (13)–(16). Let $k \in \{1, \dots, N\}$. Then, $\mathcal{P}_k(\hat{\mathfrak{d}}') = \mathcal{P}(d')(q_{k-1}, q_{k-1}')$, and $\mathcal{P}_k(\hat{\mathfrak{d}}'') = \mathcal{P}(d'')(q_{k-1}, q_{k-1}')$. Assume that the minumum of the objective function in (12) in function $\mathcal{P}(d')(q_{k-1}, q_{k-1}')$ is reached for $\rho'_{q'} = \mu$ for $\mu \in \Phi(q)$. Further, assume that the minumum of the objective function in (12) in function $\mathcal{P}(d'')(q_{k-1}, q_{k-1}')$ is reached for $\rho'_{q'} = \nu$ for $\nu \in \Phi(q)$. Let $\Psi = \Psi(q_{k-1})$. Also, let $j(k, \sigma) = j$ such that $q_j = \delta(q_{k-1}, \sigma)$, and $f(q_j) = q_j'$. Assume that $\mathcal{P}_k(\hat{\mathfrak{d}}') \geq \mathcal{P}_k(\hat{\mathfrak{d}}'')$. Then:

$$\begin{aligned}
&\left|\mathcal{P}_k(\hat{\mathfrak{d}}') - \mathcal{P}_k(\hat{\mathfrak{d}}'')\right| \\
&= \Bigg| \sum_{\sigma \in \Psi} \max(\rho_{q_{k-1},\sigma} - \mu_{q'_{k-1},\sigma} \\
&\quad + ed'(q_j, q_j')\mu_{q'_{k-1},\sigma}, ed'(q_j, q_j')\rho_{q_{k-1},\sigma}) \\
&\quad - \sum_{\sigma \in \Psi} \max(\rho_{q_{k-1},\sigma} - \nu_{q'_{k-1},\sigma} \\
&\quad + ed''(q_j, q_j')\nu_{q'_{k-1},\sigma}, ed''(q_j, q_j')\rho_{q_{k-1},\sigma}) \Bigg| \\
&\leq \Bigg| \sum_{\sigma \in \Psi} \max(\rho_{q_{k-1},\sigma} - \nu_{q'_{k-1},\sigma} \\
&\quad + ed'(q_j, q_j')\nu_{q'_{k-1},\sigma}, ed'(q_j, q_j')\rho_{q_{k-1},\sigma}) \\
&\quad - \sum_{\sigma \in \Psi} \max(\rho_{q_{k-1},\sigma} - \nu_{q'_{k-1},\sigma} \\
&\quad + ed''(q_j, q_j')\nu_{q'_{k-1},\sigma}, ed''(q_j, q_j')\rho_{q_{k-1},\sigma}) \Bigg| \qquad (17) \\
&\quad \begin{pmatrix} \text{for } \rho'_{q'_{K-1},\sigma} = \mu_{q'_{k-1},\sigma} \text{ the minimum in} \\ \mathcal{P}_k(\hat{\mathfrak{d}}') \text{ is reached} \end{pmatrix} \\
&\leq \Bigg| \sum_{\sigma \in \Psi} (\max(\rho_{q_{k-1},\sigma} - \nu_{q'_{k-1},\sigma} \\
&\quad + ed'(q_j, q_j')\nu_{q'_{k-1},\sigma}, ed'(q_j, q_j')\rho_{q_{k-1},\sigma}) \\
&\quad - \max(\rho_{q_{k-1},\sigma} - \nu_{q'_{k-1},\sigma} \\
&\quad + ed''(q_j, q_j')\nu_{q'_{k-1},\sigma}, ed''(q_j, q_j')\rho_{q_{k-1},\sigma})) \Bigg| \\
&\leq \sum_{\sigma \in \Psi} \Bigg| \max(\rho_{q_{k-1},\sigma} - \nu_{q'_{k-1},\sigma} \\
&\quad + ed'(q_j, q_j')\nu_{q'_{k-1},\sigma}, ed'(q_j, q_j')\rho_{q_{k-1},\sigma}) \\
&\quad - \max(\rho_{q_{k-1},\sigma} - \nu_{q'_{k-1},\sigma} \\
&\quad + ed''(q_j, q_j')\nu_{q'_{k-1},\sigma}, ed''(q_j, q_j')\rho_{q_{k-1},\sigma}) \Bigg|. \qquad (18)
\end{aligned}$$

(Similarly, when $\mathcal{P}_k(\hat{\mathfrak{d}}') \leq \mathcal{P}_k(\hat{\mathfrak{d}}'')$, we get (18), with $\mu_{q'_{k-1},\sigma}$ instead of $\nu_{q'_{k-1},\sigma}$.) Every summand in (18) has one of the following forms:

$$\begin{aligned}
&|\rho_{q_{k-1},\sigma} - \nu_{q'_{k-1},\sigma} + ed'(q_j, q_j')\nu_{q'_{k-1},\sigma} \\
&\quad - (\rho_{q_{k-1},\sigma} - \nu_{q'_{k-1},\sigma} + ed''(q_j, q_j')\nu_{q'_{k-1},\sigma})| \text{ or} \\
&\left|ed'(q_j, q_j')\rho_{q_{k-1},\sigma} - ed''(q_j, q_j')\rho_{q_{k-1},\sigma})\right|, \text{ where} \\
&|\rho_{q_{k-1},\sigma} - \nu_{q'_{k-1},\sigma} + ed'(q_j, q_j')\nu_{q'_{k-1},\sigma} \\
&\quad - (\rho_{q_{k-1},\sigma} - \nu_{q'_{k-1},\sigma} + ed''(q_j, q_j')\nu_{q'_{k-1},\sigma})| \\
&= e\nu_{q'_{k-1},\sigma} \left|d'(q_j, q_j') - d''(q_j, q_j')\right| \\
&\leq e\rho_{q_{k-1},\sigma} \left|d'(q_j, q_j') - d''(q_j, q_j')\right| \\
&\text{and} \\
&\left|ed'(q_j, q_j')\rho_{q_{k-1},\sigma} - ed''(q_j, q_j')\rho_{q_{k-1},\sigma})\right| \\
&= e\rho_{q_{k-1},\sigma} \left|d'(q_j, q_j') - d''(q_j, q_j')\right|.
\end{aligned}$$

Hence,

$$\left|\mathcal{P}_k(\hat{\mathfrak{d}}') - \mathcal{P}_k(\hat{\mathfrak{d}}'')\right| \leq \sum_{\sigma\in\Psi} e\rho_{q_{k-1},\sigma} \left|d'(q_j,q_j') - d''(q_j,q_j')\right|.$$

Further, using the same reasoning as in the proof of Lemma 2, it is straightforward to show that $\mathcal{P}$ is P-contractive. ∎

Lemma 5: Let d', $d'' \in \mathcal{A}$, and $\hat{\mathfrak{d}}' = \hat{\mathcal{V}}(d')$, and $\hat{\mathfrak{d}}'' = \hat{\mathcal{V}}(d'')$. For any $k \in \{1,\ldots,N\}$, there exists $m \in \{1,\ldots,N\}$ such that:

$$\left|\mathcal{P}_k(\hat{\mathfrak{d}}') - \mathcal{P}_k(\hat{\mathfrak{d}}'')\right| \leq e\left|\hat{\mathfrak{d}}'_m - \hat{\mathfrak{d}}''_m\right|.$$

Proof: See the Appendix. ∎

Theorem 4: For any $\hat{\mathfrak{d}}^0 \in P_0$, the sequence

$$\hat{\mathfrak{d}}^{n+1} = \mathcal{P}(\hat{\mathfrak{d}}^n), \quad n = 0,1,\ldots$$

converges to the only fixed point of $\mathcal{P}$ in P_0, $\hat{\mathfrak{d}}^*$, and the error estimate is given componentwise ($k \in \{1,\ldots,N\}$) as:

$$|\hat{\mathfrak{d}}^n_k - \hat{\mathfrak{d}}^*_k| \leq e^n,\ n = 0,1,\ldots$$

Proof: See the Appendix. ∎

The problem of (12)–(16) is not a linear programming problem, but it is transformable into one by using additional variables $y_{q_i,\sigma}$, and by transforming (13) into (19):

$$\begin{aligned}
&\text{Minimize} \sum_{\sigma\in\Psi(q_i)} y_{q_i,\sigma}\\
&\text{subject to}\\
&\rho_{q_i,\sigma} - \rho'_{q'_i,\sigma} + c_j\rho'_{q'_i,\sigma} \leq y_{q_i,\sigma}, \qquad \sigma\in\Psi(q_i),\\
&c_j\rho_{q_i,\sigma} \leq y_{q_i,\sigma}, \qquad \sigma\in\Psi(q_i),\\
&\text{where } c_j = e\cdot d(q_i,q_i') \text{ s.t. } q_j = \delta(q_i,\sigma),\\
&p_1(t_i,\sigma)\sum_{\alpha\in\Psi_u(q_i)}\rho'_{q_i,\alpha}, = \rho'_{q'_i,\sigma}\sum_{\alpha\in\Psi_u(q_i)} p_1(t_i,\alpha),\ \sigma\in\Psi_u(q_i),\\
&\frac{\sum_{\alpha\in\Psi_u} p_1(t_i,\alpha)}{p_1(t_i,\sigma)}\rho'_{q'_i,\sigma} + \sum_{\alpha\in\Psi_c(q_i)}\rho'_{q'_i,\alpha} \leq 1, \quad \sigma\in\Psi_c(q_i),\\
&\sum_{\alpha\in\Psi(q_i)}\rho'_{q'_i,\alpha} = 1,\\
&\rho'_{q'_i,\sigma} \geq 0, \quad \sigma\in\Psi(q_i).
\end{aligned} \tag{19}$$

It might look as if (19) is weaker than (13) as it allows the possibility of $\rho'_{q'_i,\sigma} = 0$ for all $\sigma \in \Psi_u(q_i)$, which (13) forbids. However, this is not the case. Let $\rho'_{q'_i,\sigma} = 0$ for every $\sigma \in \Psi_u(q_i)$. From (14) it follows that:

$$\frac{\sum_{\alpha\in\Psi_u} p_1(t_i,\alpha)}{p_1(t_i,\sigma)}\rho'_{q'_i,\sigma} \leq \sum_{\alpha\in\Psi_u(q_i)}\rho'_{q'_i,\alpha} = 0$$

which would mean that $\rho'_{q'_i,\sigma} = 0$ for every $\sigma \in \Psi_c(q_i)$ which contradicts the condition (15).

We now present the iterative algorithm for finding the fixed point of function $\mathcal{P}$.

Let $d^0(q_i,q_i') = 0, i = 0,1,\ldots N-1$. The distance $d^n(q_i,q_i')$ in the n-th iteration ($n > 0$) is:

$$\begin{aligned}
&\text{Minimize} \sum_{\sigma\in\Psi(q_i)} y_{q_i,\sigma}\\
&\text{subject to}\\
&\rho_{q_i,\sigma} - \rho'_{q'_i,\sigma} + c_j\rho'_{q'_i,\sigma} \leq y_{q_i,\sigma}, \qquad \sigma\in\Psi(q_i)\\
&c_j\rho_{q_i,\sigma} \leq y_{q_i,\sigma}, \qquad \sigma\in\Psi(q_i)\\
&\text{where } c_j = e\cdot d^{n-1}(q_j,q_j') \text{ s.t. } q_j = \delta(q_i,\sigma),\\
&p_1(t_i,\sigma)\sum_{\alpha\in\Psi_u(q_i)}\rho'_{q_i,\alpha} = \rho'_{q'_i,\sigma}\sum_{\alpha\in\Psi_u(q_i)} p_1(t_i,\alpha), \quad \sigma\in\Psi_u(q_i),\\
&\frac{\sum_{\alpha\in\Psi_u} p_1(t_i,\alpha)}{p_1(t_i,\sigma)}\rho'_{q'_i,\sigma} + \sum_{\alpha\in\Psi_c(q_i)}\rho'_{q'_i,\alpha} \leq 1, \quad \sigma\in\Psi_c(q_i),\\
&\sum_{\alpha\in\Psi(q_i)}\rho'_{q'_i,\alpha} = 1,\\
&\rho'_{q'_i,\sigma} \geq 0, \quad \sigma\in\Psi(q_i).
\end{aligned} \tag{20}$$

After the n-th iteration, the value of decision variables $\rho'_{q'_i,\sigma}$ that represent the unknown transition probabilities are such that the distance between the (initial states of) systems G_2 and G_2' is within e^n of the minimal achievable distance between the two systems (in our pseudometric). Note that the aforementioned results hold for $e \in (0,1)$.

Also, as previously mentioned, a modification of the presented algorithm can be used to solve the control problem presented in Section III with requirement 2) changed so that the distance between the controlled plant and the unmodified requirement is minimized (see [36], [37]).

C. Summarizing the Algorithm

We now summarize the presented algorithm and give a brief complexity analysis.

1) First, the classical algorithm for finding the supremal controllable sublanguage is modified. The automaton G_s, the synchronous product of the nonprobabilistic automata underlying G_p and G_r is constructed. While constructing the product, the classical controllability conditions are checked for each state. If the conditions are satisfied for each state of the product, then $G = G_s$, and go to 2). If there is at least one state of the product for which the classical conditions do not hold, the rest of the algorithm for finding the automaton representing the supremal controllable sublanguage is then applied. The algorithm can easily be modified to exclude deadlock states: these states are considered uncontrollable.
   Let (reachable and deadlock-free) DES $G = (Q,\Sigma,\delta,q_0)$ represent this supremal controllable language.
2) Let G_1, G_2, and G_2' be defined as previously in this section. Check the equalities and inequalities of Theorem 1 for each state: if they are satisfied, a supervisor exists, and G_2 is the optimal approximation. If not, then let $d^0(q_i,q_i') = 0$ for all $0 \leq i \leq N-1$. The distance $d^n(q_i,q_i')$ in the n-th iteration ($n > 0$) is given by (20).

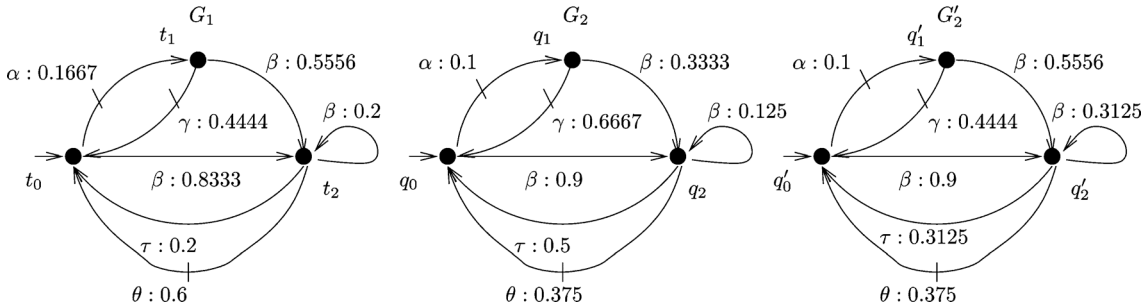


Fig. 2. Generators G_1, G_2, and optimal approximation G_2'.

TABLE I
EXAMPLE: FIXED POINT ITERATION

	$n=1$	n=2	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10
$d^n(q_0, q_0')$	-1.1e-12	0.0955	0.1023	0.1186	0.1209	0.1238	0.1244	0.1249	0.1251	0.1252
$d^n(q_1, q_1')$	0.2222	0.2535	0.2766	0.2838	0.2881	0.2896	0.2904	0.2908	0.2909	0.2910
$d^n(q_2, q_2')$	0.1875	0.1992	0.2328	0.2372	0.2431	0.2443	0.2453	0.2456	0.2458	0.2458
$\rho'_{q_0',\alpha}$	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000
$\rho'_{q_0',\beta}$	0.9000	0.9000	0.9000	0.9000	0.9000	0.9000	0.9000	0.9000	0.9000	0.9000
$\rho'_{q_1',\beta}$	0.5556	0.5556	0.5556	0.5556	0.5556	0.5556	0.5556	0.5556	0.5556	0.5556
$\rho'_{q_1',\gamma}$	0.4444	0.4444	0.4444	0.4444	0.4444	0.4444	0.4444	0.4444	0.4444	0.4444
$\rho'_{q_2',\beta}$	0.3125	0.3125	0.3125	0.3125	0.3125	0.3125	0.3125	0.3125	0.3125	0.3125
$\rho'_{q_2',\theta}$	0.3750	0.3750	0.3750	0.3750	0.3750	0.3750	0.3750	0.3750	0.3750	0.3750
$\rho'_{q_2',\tau}$	0.3125	0.3125	0.3125	0.3125	0.3125	0.3125	0.3125	0.3125	0.3125	0.3125

For each of the states of G_2 (typically, the number of states of G_2 is much smaller than $|Q_p| \cdot |Q_r|$), either the simplex method or an interior point method can be used to solve the linear programming problem (20). Depending on what method is used, the worst-case running time of the algorithm is either exponential (simplex method) or polynomial (interior point methods) in the maximal number of events possible from a state of the supremal controllable sublanguage of the specification (with respect to the plant). Even the worst-case exponential complexity of the simplex method is not problematic for two reasons: first, the method is very efficient in practice, and second, the number of possible events from a state is small in practical applications. Furthermore, the number of iterations sufficient to reach the accuracy of $\epsilon \in (0, 1)$ is $\lceil \log_e \epsilon \rceil$ (the smallest number for which $\epsilon \geq e^n$ is satisfied).

D. Example

For plant G_p depicted in Fig. 1, there does not exist a probabilistic supervisor V_p such that $G(V_p/G_p) = G_r$. Fig. 2 shows the modified plant G_1 and modified specification G_2, defined as suggested in Section VI-A. For PDES G_2, let ρ_q be the probability distribution induced by the state $q \in Q$ and, for PDES G_2', let $\rho'_{q'}$ be the probability distribution induced by the state $q' \in Q'$. As before, we will write $\rho_{q,\sigma}$ instead of $\rho_q(\sigma, q_i)$, and $\rho'_{q',\sigma}$ instead of $\rho'_{q'}(\sigma, q'_j)$, $i, j = 0, 1, 2$. For the accuracy $\epsilon = 0.001$, and $e = 0.5$, 10 iterations of the algorithm are suficient. The closest approximation is as given in Fig. 2. The computation is shown in detail in Table I. Note that the probabilities of the closest approximation do not change from the first iteration. Although this is not the case in general, for future work we would like to investigate how the unknown probabilities of the closest approximation change as the distance converges. The computation took 0.3 seconds on a 2.6 GHz dual core Opteron processor with 8 GB of RAM running Red Hat Enterprise Linux Server 5.5. In order to find the corresponding probabilistic supervisor, the algorithm of [22], [23] can be used. For the state q_0' and event α, the control input is 0.6, for q_1' and event γ, the input is 1, and for q_2' and event θ, the input is 0.625.

VII. CONCLUSIONS

This paper solves the classical problem of finding the closest approximation in the framework of probabilistic control of PDES. Two algorithms for the calculation of distances (in the chosen pseudometric) between the states of a probabilistic generator used to model a PDES are suggested. Then, a modification of the iterative algorithm is proposed to minimize the distance (in this pseudometric) between the desired behavior of the system and its achievable behavior.

Operators on probabilistic generators should be defined and their desired property of non-expansiveness with the respect to the pseudometric should be checked. This would allow for compositional reasoning. Further, the question of uniqueness of the closest approximation remains open as well as probabilistic control with marking.

APPENDIX

Proof of Theorem 2: Let $n, m \geq 1$. Then:

$$|\mathfrak{d}_k^{n+m} - \mathfrak{d}_k^n| \tag{21}$$

$$\leq \sum_{t=1}^{m} |\mathfrak{d}_k^{n+t} - \mathfrak{d}_k^{n+t-1}|$$
$$\leq \sum_{t=1}^{m} e^t |\mathfrak{d}_{i(t)}^{n} - \mathfrak{d}_{i(t)}^{n-1}|$$
$$\text{(applying Lemma 3 } t \text{ times, where } i(t) \in \{1, \ldots, N^2\})$$
$$\leq \sum_{t=1}^{m} e^t \max_{i(t)} \{|\mathfrak{d}_{i(t)}^{n} - \mathfrak{d}_{i(t)}^{n-1}|\}$$
$$\leq \left(\sum_{t=1}^{m} e^t \right) |\mathfrak{d}_j^{n} - \mathfrak{d}_j^{n-1}| \text{ (for some } j \in \{1, \ldots, N^2\})$$
$$\leq (1-e)^{-1} e |\mathfrak{d}_j^{n} - \mathfrak{d}_j^{n-1}|$$
$$\text{(since } \sum_{t=0}^{m} e^t \leq (1-e)^{-1} \text{ for } m \geq 0)$$
$$\leq (1-e)^{-1} e^n |\mathfrak{d}_l^{1} - \mathfrak{d}_l^{0}|$$
$$\text{(for some } l \in \{1, \ldots, N^2\},$$
$$\text{using Lemma 3 } (n-1) \text{ times}). \quad (22)$$

Therefore, the sequence $\{\mathfrak{d}_k^n\}_{n\geq 0}$ is a Cauchy sequence and hence converges to some $\mathfrak{d}_k^*$, and, consequently, the sequence $\{\mathfrak{d}^n\}_{n\geq 0}$ converges to some $\mathfrak{d}^* \in D_0$. Also, we have:

$$|\mathfrak{d}^* - \mathcal{D}(\mathfrak{d}^*)| \leq |\mathfrak{d}^* - \mathfrak{d}^{n+1}| + |\mathcal{D}(\mathfrak{d}^n) - \mathcal{D}(\mathfrak{d}^*)|$$
$$\leq |\mathfrak{d}^* - \mathfrak{d}^{n+1}| + P|\mathfrak{d}^n - \mathfrak{d}^*|$$

When we let $n \to \infty$, we see that $\mathfrak{d}^* = \mathcal{D}(\mathfrak{d}^*)$.

Remark 2: Note that the error in the n-th iteration can be estimated as

$$|\mathfrak{d}_k^n - \mathfrak{d}_k^*| \leq (1-e)^{-1} e^n \max_i \{|\mathfrak{d}_i^n - \mathfrak{d}_i^{n-1}|\}$$

that follows from (22) when $m \to \infty$ in (21).

Finally, it should be proven that $\mathfrak{d}^*$ is the only fixed point in D_0. Assume that there is another fixed point of $\mathcal{D}$ in the same set D_0, $\mathfrak{d}^+$. Then,

$$|\mathfrak{d}^* - \mathfrak{d}^+| = |\mathcal{D}(\mathfrak{d}^*) - \mathcal{D}(\mathfrak{d}^+)| \leq P|\mathfrak{d}^* - \mathfrak{d}^+|.$$

Hence, $(I-P)|\mathfrak{d}^* - \mathfrak{d}^+| \leq 0$. However, since $\varphi(P) < 1$, $(I-P)^{-1} = \sum_{i=0}^{\infty} P^i \geq 0$ (see [50], 2.4.5.), then $|\mathfrak{d}^* - \mathfrak{d}^+| \leq 0$. Therefore, $\mathfrak{d}^* = \mathfrak{d}^+$.

The error estimate of (10) follows by induction, using Lemma 3.

Proof of Lemma 4: According to Knaster-Tarski theorem, the functions f and g have the greatest fixed points $gfp(f)$ and $gfp(g)$, respectively, where $gfp(f) = sup(\{x | x \preceq f(x)\})$, and $gfp(g) = sup(\{x | x \preceq g(x)\})$. Since $\forall x : g(x) \preceq f(x)$, then $\{x | x \preceq g(x)\} \subseteq \{x | x \preceq f(x)\}$; hence $gfp(g) \preceq gfp(f)$.

Proof of Lemma 5: First, use the proof of Theorem 3 up to (17), and, then, analogous to the proof of Lemma 3.

Proof of Theorem 4: Analogous to the proof of Theorem 2 (with using Lemma 5 instead of Lemma 3).

REFERENCES

[1] P. Ramadge and W. Wonham, "On the supremal controllable sublanguage of a given language," *SIAM J. Control Opt.*, vol. 25, no. 3, pp. 637–659, 1987.

[2] J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker, *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems*, ser. CRM Monograph Series, P. Panangaden and F. van Breugel, Eds. Providence, RI: Amer. Math. Soc., 2004, vol. 23.

[3] M. Kwiatkowska, G. Norman, and D. Parker, "Stochastic model checking," in *Proc. Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation*, M. Bernardo and J. Hillston, Eds., (Tutorial Volume) ed. Berlin, Germany: Springer-Verlag, 2007, vol. 4486, Lecture Notes in Computer Science, pp. 220–270.

[4] M. Huth and M. Kwiatkowska, "Comparing CTL and PCTL on labeled Markov chains," in *Proc. PROCOMET*, Feb. 1998.

[5] Y. Li, F. Lin, and Z. H. Lin, "Supervisory control of probabilistic discrete event systems with recovery," *IEEE Trans. Autom. Control*, vol. 44, no. 10, pp. 1971–1975, Oct. 1998.

[6] G. Mallapragada, I. Chattopadhyay, and A. Ray, "Autonomous robot navigation using optimal control of probabilistic regular languages," *Int. J. Control*, vol. 82, no. 1, pp. 13–26, Jan. 2009.

[7] I. Chattopadhyay, G. Mallapragada, and A. Ray, "$\nu^\star$: A robot path planning algorithm based on renormalized measure of probabilistic regular languages," *Int. J. Control*, vol. 82, no. 5, pp. 849–867, May 2009.

[8] J. Feldman and J. F. Hanna, "The structure of responses to a sequence of binary events," *J. Math. Psych.*, vol. 3, no. 2, pp. 371–387, 1966.

[9] L. Schröder and P. Mateus, "Universal aspects of probabilistic automata," *Math. Struct. Comput. Sci.*, vol. 12, no. 4, pp. 481–512, 2002.

[10] R. J. V. Glabbeek, S. A. Smolka, and B. Steffen, "Reactive, generative and stratified models of probabilistic processes," *Inf. Comput.*, vol. 121, no. 1, pp. 59–80, 1995.

[11] H. Mortazavian, "Controlled stochastic languages," in *Proc. 31st Annu. Allerton Conf. Commun. , Control, Comput.*, Urbana, IL, 1993, pp. 938–947.

[12] V. S. Borkar, *Topics in Controlled Markov Chains*. Hoboken, NJ: Wiley, 1991.

[13] R. Kumar and V. Garg, "Control of stochastic discrete event systems: Existence," in *Proc. Int. Workshop Discrete Event Syst.*, Cagliari, Italy, Aug. 1998, pp. 24–29.

[14] V. Garg, R. Kumar, and S. Marcus, "Probabilistic language formalism for stochastic discrete event systems," *IEEE Trans. Autom. Control*, vol. 44, no. 2, pp. 280–293, 1999.

[15] V. Garg, "An algebraic approach to modeling probabilistic discrete event systems," in *Proc. 31st IEEE Conf. Decision Control*, Tucson, AZ, Dec. 1992, pp. 2348–2353.

[16] V. Garg, "Probabilistic languages for modeling of DEDS," in *Proc. 26th Conf. Inf. Sci. Syst.*, Princeton, NJ, Mar. 1992, vol. 1, pp. 198–203.

[17] R. Kumar and V. Garg, "Control of stochastic discrete event systems modeled by probabilistic languages," *IEEE Trans. Autom. Control*, vol. 46, no. 4, pp. 593–606, Apr. 2001.

[18] I. Chattopadhyay and A. Ray, "Language-measure-theoretic optimal control of probabilistic finite state systems," in *Proc. 46th IEEE Conf. Decision Control*, New Orleans, LA, Dec. 2007, pp. 5930–5935.

[19] I. Chattopadhyay and A. Ray, "Language-measure-theoretic optimal control of probabilistic finite-state systems," *Int. J. Control*, vol. 80, no. 8, pp. 1271–1290, 2007.

[20] A. Kučera and O. Stražovský, "On the controller synthesis for finite-state Markov decision processes," *Fundamenta Inf.*, vol. 82, no. 1–2, pp. 141–153, 2008.

[21] M. Lawford and W. Wonham, "Supervisory control of probabilistic discrete event systems," in *Proc. 36th IEEE Midwest Symp. Circuits Syst.*, Aug. 1993, vol. 1, pp. 327–331.

[22] S. Postma and M. Lawford, "Computation of probabilistic supervisory controllers for model matching," in *Proc. Allerton Conf. Commun. , Control, Comput.*, V. Veeravalli and G Dullerud, Eds., Monticello, IL, 2004.

[23] V. Pantelic, S. Postma, and M. Lawford, "Probabilistic supervisory control of probabilistic discrete event systems," *IEEE Trans. Autom. Control*, vol. 54, no. 8, pp. 2013–2018, Aug. 2009.

[24] Y. Deng, T. Chothia, C. Palamidessi, and J. Pang, "Metrics for action-labelled quantitative transition systems," *Electron. Notes Theoret. Comput. Sci.*, vol. 153, no. 2, pp. 79–96, 2006.

[25] L. Kantorovich, "On the transfer of masses (in Russian)," *Doklady Akademii Nauk*, vol. 37, no. 2, pp. 227–229, 1942.

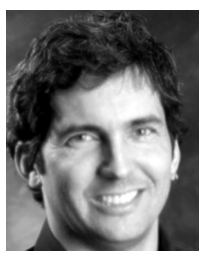
[26] J. E. Hutchinson, "Fractals and self-similarity," *Indiana Univ. Math. J.*, vol. 30, no. 5, pp. 713–747, 1981.

[27] L. Wasserstein, "Markov processes over denumerable products of spaces describing large systems of automata," *Prob. Inf. Trans.*, vol. 5, no. 3, pp. 47–52, 1969.

[28] Y. Deng and W. Du, "The Kantorovich metric in computer science: A brief survey," *Electron. Notes Theoret. Comput. Sci.*, vol. 253, pp. 73–82, Nov. 2009.
[29] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden, J. C. M. Baeten and S. Mauw, Eds., "Metrics for labeled Markov systems," in *Proc. 10th Int. Conf. Concurrency Theory*, 1999, vol. 1664, pp. 258–273.
[30] J. Desharnais, R. Jagadeesan, V. Gupta, and P. Panangaden, "The metric analogue of weak bisimulation for probabilistic processes," in *Proc. 17th Annu. IEEE Symp. Logic in Comput. Sci.*, Washington, DC, 2002, pp. 413–422.
[31] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden, "Metrics for labelled Markov processes," *Theoret. Comput. Sci.*, vol. 318, no. 3, pp. 323–354, 2004.
[32] F. van Breugel and J. Worrell, K. G. Larsen and M. Nielsen, Eds., "An algorithm for quantitative verification of probabilistic transition systems," in *Proc. Int. Conf. Concurrency Theory*, 2001, vol. 2154, pp. 336–350.
[33] F. van Breugel and J. Worrell, "A behavioural pseudometric for probabilistic transition systems," *Theoret. Comput. Sci.*, vol. 331, no. 1, pp. 115–142, 2005.
[34] F. van Breugel, C. Hermida, M. Makkai, and J. Worrell, "An accessible approach to behavioural pseudometrics," in *Automata, Languages and Programming*, L. Caires, G. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, Eds. Berlin, Germany: Springer-Verlag, 2005, vol. 3580, Lecture Notes in Computer Science, pp. 1018–1030.
[35] F. van Breugel and J. Worrell, "Approximating and computing behavioural distances in probabilistic transition systems," *Theoret. Comput. Sci.*, vol. 360, no. 1–3, pp. 373–385, 2006.
[36] V. Pantelic and M. Lawford, "Use of a metric in supervisory control of probabilistic discrete event systems," in *Pro. 10th Int. Workshop on Discrete Event Systems*, Berlin, Germany, Aug. 2010, pp. 227–232.
[37] V. Pantelic and M. Lawford, "A pseudometric in supervisory control of probabilistic discrete event systems," *Discr. Event Dynam. Syst.*, 2012, to be published.
[38] N. Ferns, P. Panangaden, and D. Precup, "Metrics for finite Markov decision processes," in *AAAI*, D. L. McGuinness and G. Ferguson, Eds. Cambridge, MA: AAAI Press/ MIT Press, 2004, pp. 950–951.
[39] N. Ferns, P. Panangaden, and D. Precup, "Metrics for Markov Decision Processes with infinite state spaces," in *UAI*. Arlington, VA: AUAI Press, 2005, pp. 201–208.
[40] N. Ferns, P. S. Castro, D. Precup, and P. Panangaden, "Methods for computing state similarity in Markov decision processes," in *UAI*. Arlington, VA: AUAI Press, 2006.
[41] V. Pantelic, "Probabilistic supervisory control of probabilistic discrete event systems," Ph.D. dissertation, McMaster Univ., Hamilton, ON, Canada, 2011.
[42] V. Pantelic and M. Lawford, "Towards optimal supervisory control of probabilistic discrete event systems," in *Proc. 2nd IFAC Workshop Dependable Control of Discrete Syst.*, Bari, Italy, Jun. 2009, pp. 85–90.
[43] K. G. Larsen and A. Skou, "Bisimulation through probabilistic testing," *Inf. Comput.*, vol. 94, no. 1, pp. 1–28, 1991.
[44] G. Barrett and S. Lafortune, "Using bisimulation to solve discrete event control problems," in *Proc. Amer. Control Conf.*, Albuquerque, NM, Jun. 1997, pp. 2337–2341.
[45] A. Giacalone, C. Jou, and S. Smolka, M. Broy and C. B. Jones, Eds., "Algebraic reasoning for probabilistic concurrent systems," in *Proc. Working Conf. Programming Concepts Methods*, Sea of Gallilee, Israel, 1990, pp. 443–458.
[46] I. Chattopadhyay and A. Ray, "Structural transformations of probabilistic finite state machines," *Int. J. Control*, vol. 81, no. 5, pp. 820–835, 2008.
[47] F. van Breugel and J. Worrell, "Towards quantitative verification of probabilistic transition systems," in *Int. Colloq. Automata, Lang. Programming*, F. Orejas, P. G. Spirakis, and J. van Leeuwen, Eds. Berlin, Germany: Springer, 2001, vol. 2076, Lecture Notes in Computer Science, pp. 421–432.
[48] D. Blackwell, "Discrete dynamic programming," *Annals Math. Stat.*, vol. 33, no. 2, pp. 719–726, 1962.
[49] L. de Alfaro, T. A. Henzinger, and R. Majumdar, J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger, Eds., "Discounting the future in systems theory," in *Proc. Int. Colloq. Automata, Languages and Programming*, 2003, vol. 2719, pp. 1022–1037.
[50] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*. New York: Academic Press, 1970.
[51] F. van Breugel, B. Sharma, and J. Worrell, "Approximating a behavioural pseudometric without discount for probabilistic systems," *Logical Methods Comput. Sci.*, vol. 4, no. 2:2, pp. 1–23, 2008.

Vera Pantelic received the B.Eng. in Electrical Engineering from University of Belgrade, Belgrade, Serbia, in 2001, and the M.A.Sc, and Ph.D. in Software Engineering from McMaster University, Hamilton, ON, Canada, in 2005, and 2011, respectively.

She is currently working as a Postdoctoral Fellow with the McMaster Centre for Software Certification. Her main research interests include supervisory control of discrete event systems, and verification and certification of safety-critical software systems.



Mark Lawford (SM'08) received the B.Sc. degree in Engineering Mathematics from Queen's University, Kingston, in 1989, where he also received the University Medal in Engineering Mathematics, and the M.A.Sc. and the Ph.D. degrees from the Systems Control Group, Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada in 1992 and 1997, respectively.

His research interests include software certification, application of formal methods to safety critical real-time systems, and supervisory control of discrete event systems. He worked at Ontario Hydro as a real-time software verification consultant on the Darlington Nuclear Generating Station Shutdown Systems Redesign project, receiving the Ontario Hydro New Technology Award for Automation of Systematic Design Verification of Safety Critical Software in 1999. He joined the Department of Computing and Software, McMaster University, Hamilton, ON, Canada, in 1998 where he helped to develop the Software Engineering and Mechatronics Engineering programs. In 2003 he was a Guest Co-Editor of joint special issues on software inspection for the IEEE SOFTWARE MAGAZINE and the IEEE TRANSACTIONS ON SOFTWARE ENGINEERING. He is a licensed Professional Engineer in the province of Ontario.