Probabilistic Supervisory Control of Probabilistic Discrete Event Systems

Vera Pantelic, Steven M. Postma, and Mark Lawford, *Senior Member, IEEE*

***Abstract*—This paper considers supervisory control of probabilistic discrete event systems (PDESs). PDESs are modeled as generators of probabilistic languages. The supervisory control problem considered is to find, if possible, a supervisor under whose control the behavior of a plant is identical to a given probabilistic specification. The probabilistic supervisors we employ are a generalization of the deterministic ones previously employed in the literature. At any state, the supervisor enables/disables events with certain probabilities. Necessary and sufficient conditions for the existence of such a supervisor, and an algorithm for its computation are presented.**

***Index Terms*—Probabilistic discrete event systems, random disablement, supervisory control.**

I. Introduction

The supervisory control theory of discrete event systems (DESs) was developed in the seminal work of Ramadge and Wonham [1]. A supervisor (controller) controls a plant by enabling/disabling controllable events based on observation of the previous behavior of the plant. The supervisory control problem considered is to supervise the plant so that it generates a given specification language. In order to model stochastic behavior of the plant, many models of stochastic behavior of discrete event systems have been proposed (e.g., Markov chains [2], Rabin's probabilistic automata [3], stochastic Petri nets [4]). We follow the theory of stochastic discrete event systems that was developed in [5], [6] using an algebraic approach. A stochastic discrete event system is represented as an automaton with transitions labeled with probabilities. The probabilities of all the events in a certain state add up to at most one. This differs from Rabin's probabilistic automata [3], where the sum of the probabilities of all instances of an event at a state is one. Also, unlike the Markov chains [2], the emphasis of the approach of [5], [6] is on event traces rather than state traces.

The control of different models of stochastic discrete event systems has been investigated in [7], [8], etc. Rabin's probabilistic automata are used in [7] as the underlying model, while [8] uses Markov chains. In [9], the model of [5], [6] is adopted and deterministic supervisors for DESs are generalized to *probabilistic supervisors*. The control method of *random disablement* is used: after observing a string s, the probabilistic supervisor enables an event σ with a certain probability. When controllable events are disabled, the probabilities of their execution become zero, and the probabilities of occurrence of other events proportionally increase. Necessary and sufficient conditions for the existence of a supervisor for a probabilistic discrete event system (PDES) to meet a specification are given. These conditions reduce to checking whether certain linear equalities and inequalities hold.

The work of [10] builds upon [9] by giving a formal proof of the necessity and sufficiency of the conditions presented in [9]. It also gives an algorithm for the calculation of the supervisor, if it exists. A supervisory control framework for stochastic discrete event systems that was developed in [11] represents a special case of the control introduced in [9]. The control is deterministic. The control objective considered in [11] is to construct a supervisor such that the controlled plant does not execute specified illegal traces, and the probabilities of occurrences of the events in the system are greater than or equal to specified values. The paper gives necessary and sufficient condition for the existence of a supervisor. Further, in [12], a technique to compute a maximally permissive supervisor on-line is given. In [13], [14], the same model of [5], [6] is used. The requirements specification is given by weights assigned to the states of a plant and the control goal is, roughly speaking, to reach the states with more weight (more desired 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.

Controller synthesis for probabilistic systems has also attracted attention in the formal methods community. E.g., [15], [16] consider different control policies: deterministic or randomized (probabilistic) on one hand; memoryless (Markovian) or history-independent on the other. The systems considered are finite Markov decision processes, where the state space is divided into two disjoint sets: controllable states and uncontrollable states. In [15], the controller synthesis problem for a requirements specification given as a probabilistic computation tree logic (PCTL) formula is shown to be NP-hard, and a synthesis algorithm for automata specification is presented. Controller synthesis was considered in [16] for requirements specification given as a formula of PCTL extended with long-run average propositions. It is shown that the existence of such a controller is decidable, and an algorithm for the synthesis of a controller, when it exists, is presented. Further, controller robustness with respect to slight changes in the probabilities of the plant is discussed. The paper shows that the existence of robust controllers is decidable and the controller, if it exists, is effectively computable.

Deterministic control is easier to deal with than probabilistic control, both from the viewpoint of analysis, and practice. However, probabilistic control is much more powerful. It has been shown in [9] that probabilistic supervisory control can generate a much larger class of probabilistic languages than deterministic control. In the sense of the supervisory control problem discussed in this paper, the use of deterministic control might be too restrictive for a designer. Hence, [9] and [10] investigate probabilistic supervisory control: conditions under which a probabilistic control can generate a prespecified probabilistic language, and, if the supervisor exists, an algorithm for its synthesis.

This paper merges the works of [9] and [10]. The notation used in [10] differs from the classical notation introduced in the seminal work of [1] and used in [9]. We will use this classical notation. Further, a more detailed literature review is presented. Some of the proofs from [10] have been reworked, and a new, more general example for the computation of a supervisor is used. We modify the main results of [9] and [10] to include a special case when only controllable events can occur in a plant. This case has not been solved in any of the previous work. We also offer time complexity analysis of both the controller synthesis problem and the proposed synthesis algorithm.

Section II introduces PDESs modeled as generators of probabilistic languages. The technique of random disablement is presented in Section III. The probabilistic supervisory control problem is stated in Section IV. This section also introduces the main results of [9] and [10]. Section V gives the formal proof of the main result. The complexity analysis of the synthesis problem and algorithm is presented in Section VI. Finally, Section VII concludes with avenues for future work.

Manuscript received July 25, 2008; revised March 16, 2009. Current version published August 05, 2009. Recommended by Associate Editor S. Haar.

The authors are with the SQRL, Department of Computing and Sofware, Faculty of Engineering, McMaster University, Hamilton, ON, Canada L8S 4K1 (e-mail: pantelv@ mcmaster.ca; steven.m.postma@gmail.com; lawford@mcmaster.ca).

Digital Object Identifier 10.1109/TAC.2009.2024376

II. Preliminaries

A *probabilistic DES (PDES)* is modeled as a probabilistic generator $G = (Q, \Sigma, \delta, q_0, Q_m, p)$, where Q is a nonempty set of states (at most countable), Σ is a finite alphabet whose elements we will refer to as event labels, $\delta : Q \times \Sigma \to Q$ is a (partial) transition function, $q_0 \in Q$ is the initial state, $Q_m \subseteq Q$ is the set of marking states, which represent the completed tasks, and $p : Q \times \Sigma \to [0, 1]$ is the statewise event probability distribution [9]. 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. In this paper, we consider only finite state PDESs (Q is a nonempty finite set).

The probability that the event $\sigma \in \Sigma$ is going to occur at the state $q \in Q$ is $p(q, \sigma)$. For the generator G to be well-defined, (*i*) $p(q, \sigma) = 0$ should hold if and only if $\delta(q, \sigma)$ is undefined, and (*ii*) $\forall q \ \sum_{\sigma \in \Sigma} p(q, \sigma) \leq 1$. The probabilistic generator G is *nonterminating* if, for every reachable state $q \in Q$, $\sum_{\sigma \in \Sigma} p(q, \sigma) = 1$. The probabilistic generator G is *terminating* if there is at least one reachable state $q \in Q$ such that $\sum_{\sigma \in \Sigma} p(q, \sigma) < 1$. Upon entering state q, the probability that the system terminates at that state is $1 - \sum_{\sigma \in \Sigma} p(q, \sigma)$. Throughout the sequel, unless stated otherwise, we will be considering nonterminating generators. Terminating generators can be transformed into nonterminating ones in a straightforward manner as presented in [9], [17].

The language $L(G)$ generated by a PDES $G = (Q, \Sigma, \delta, q_0, Q_m, p)$ is given by $L(G) = \{s \in \Sigma^* \mid \delta(q_0, s)!\}$. The marked language of G is given by $L_m(G) = \{s \in \Sigma^* \mid \delta(q_0, s) \in Q_m\}$, whereas 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 $s \in L(G)$.

III. Control of PDES

As in classical supervisory theory, the set Σ is partitioned into Σ_c and Σ_u, the sets of controllable and uncontrollable events, respectively. Given probabilistic generator G_2 of a probabilistic specification language E (i.e., $L_p(G_2) = E$) and probabilistic generator G of probabilistic language $L_p(G)$ representing a plant, the goal is to find a supervisor V such that the language generated by the plant under supervision, $L_p(V/G)$, is equal to E. A classical, deterministic supervisor can only disable controllable events $\sigma \in \Sigma_c$. It can be defined using a function $V : L(G) \to \{0, 1\}^{\Sigma}$:

$$(\forall s \in L(G))(\forall \sigma \in \Sigma) V(s)(\sigma) = \begin{cases} 1, & \text{if } \sigma \in \Sigma_u \text{ or } s\sigma \in E \\ 0, & \text{otherwise.} \end{cases}$$

We now explore the limited effect a classical supervisor can have on a PDES. Fig. 1 shows two PDESs: the first one, G, represents a plant, and the second one, G_2, is a requirements specification. Controllable events are marked with a bar on their edges. A number next to an event represents the probability distribution of that event. G has alphabet $\Sigma = \{\alpha, \beta, \gamma\}$ and is nonterminating. The event γ is uncontrollable, and, therefore, always enabled. We also make an important assumption about the behavior of a supervisor: *After an event is disabled, the probabilities of the remaining enabled events proportionally increase.* The question we want to answer is: Does there exist a deterministic supervisor V such that $L_p(V/G) = L_p(G_2)$?

We first consider the case when the PDES G is in the state q_0 and the PDES G_2 is in the state q_{20}. The required probabilities of all the events in the state q_{20} are nonzero. Therefore, the deterministic supervisor V should enable all (controllable) events (state q_{V0} of DES V in

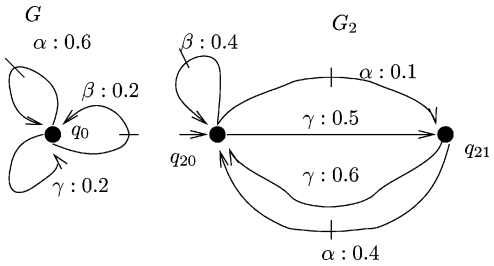

Fig. 1. Plant G and requirements specification G_2.

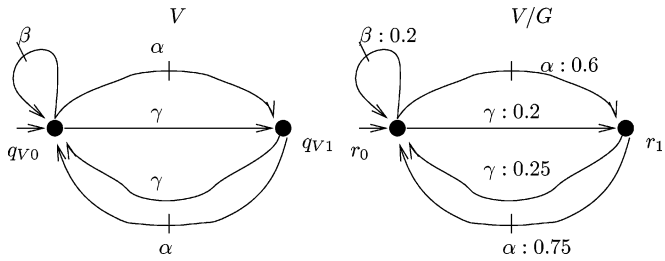

Fig. 2. Deterministic supervisor V and controlled plant V/G.

Fig. 2). Hence, the probabilities of events in the controlled plant remain unchanged (see state r_0 of the controlled plant V/G in Fig. 2).

Next, after an odd number of α or γ events (PDES G is in state q_0 and PDES G_2 is in state q_{21}), the supervisor should disable β. When V disables only β, the plant can choose between α and γ. The probabilities of these events occuring in the resulting system are increased proportional to their original probabilities. Therefore, the probability of α occuring in state r_1 of the controlled plant is equal to:

$$P(\alpha | \sigma \in \{\gamma, \alpha\}) = \frac{p(q_0, \alpha)}{p(q_0, \alpha) + p(q_0, \gamma)} = \frac{0.6}{0.6 + 0.2} = 0.75.$$

Similarly, the probability of γ ocurring is 0.25.

Therefore, although the requirement was met nonprobabilistically (meaning $L(V/G) = L(G_2)$), it is obvious that there is no deterministic control such that $L_p(V/G) = L_p(G_2)$. This example illustrates that application of deterministic supervisors to PDES results in a rather limited class of probabilistic languages. Hence, applying a deterministic supervisor to a PDES might be unacceptable for a designer.

We now generalize deterministic supervisors for DES to *probabilistic supervisors*. The control technique used is called *random disablement*. Instead of deterministically enabling or disabling controllable events, probabilistic supervisors enable events with certain probabilities. This means that, upon reaching a certain state q, the control pattern (a set of events to be enabled) 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.

For a PDES $G = (Q, \Sigma, \delta, q_0, Q_m, p)$, a *probabilistic supervisor* is a function $V_p : L(G) \to [0, 1]^{\Sigma}$ such that for $s \in L(G), \sigma \in \Sigma$:

$$V_p(s)(\sigma) = \begin{cases} 1, & \text{if } \sigma \in \Sigma_u \\ x(s)(\sigma), & \text{otherwise, where } x(s)(\sigma) \in [0, 1]. \end{cases}$$

Therefore, after observing a string s, the supervisor enables the event σ with probability $V_p(s)(\sigma)$. After a set of controllable events to be enabled, Θ, has been decided upon (uncontrollable events are always enabled), the system acts as if supervised by a deterministic supervisor. An example of a probabilistic supervisor is given in Fig. 3. Note that the probabilities of all the events that can execute in a state of this

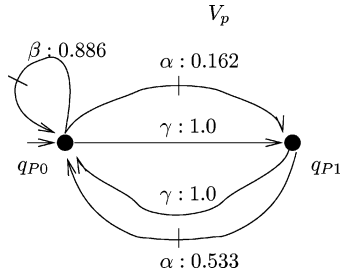

Fig. 3. Probabilistic supervisor V_p.

generator do not, in general, add up to 1. This is because those are not the probabilities of events occurring, but rather being enabled.

What is the probability that an event α will occur in a plant G under the control of probabilistic supervisor V_p when the string $s \in L(G)$ has been observed? First, the control pattern is chosen according to controllable event probabilities of the supervisor, and then, under that pattern, the plant makes a choice according to its events probabilities. Let $q \in Q$ be the state of the plant after s. We define *the set of possible events* at q to be $Pos(q) := \{\sigma \in \Sigma | p(q,\sigma) > 0\}$. Given sets A, B, the power set of A will be denoted by $\mathcal{P}(A)$, and the set difference of A and B by $A \backslash B$.

The probability that the event $\alpha \in \Sigma$ will occur after string s has been observed is equal to

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

where

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

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

It is now easy to show that the plant in Fig. 1 under the control of the probabilistic supervisor depicted in Fig. 3 succeeds in generating the probabilistic language $L_p(G_2)$, whereas a deterministic controller failed (see [9] and [17]). However, in the general case, for a given plant, there might not exist a probabilistic supervisor for a given probabilistic specification language. In the next section, we will explore the conditions under which a probabilistic supervisor exists.

IV. Probabilistic Supervisory Control Problem

A. Problem Statement

Our 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_1 and a specification PDES G_2, find, if possible, a probabilistic supervisor V_p such that $L_p(V_p/G_1) = L_p(G_2)$.

B. Main Result

We now present slightly modified main results of [9] and [10]. The results are modified to account for the special case when $Pos(q) \cap \Sigma_u = \emptyset$ discussed in detail in Section V-B.

First, we develop necessary and sufficient conditions for the existence of a solution to the PSCP problem for nonterminating PDESs.

Theorem 1: Let $G_1 = (Q, \Sigma, \delta_1, q_0, Q_m, p_1)$ and $G_2 = (R, \Sigma, \delta_2, r_0, R_m, p_2)$ be two nonterminating PDESs with disjoint state sets Q and R. There exists a probabilistic supervisor V_p such that $L_p(V_p/G_1) = L_p(G_2)$ iff for all $s \in L(G_2)$ there exists $q \in Q$ such that $\delta_1(q_0, s) = q$ and, letting $r = \delta_2(r_0, s)$, the following two conditions hold:

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

$$\frac{p_1(q,\sigma)}{\sum_{\alpha \in \Sigma_u} p_1(q,\alpha)} = \frac{p_2(r,\sigma)}{\sum_{\alpha \in \Sigma_u} p_2(r,\alpha)}$$

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

$$\frac{p_2(r,\sigma)}{p_1(q,\sigma)} \sum_{\alpha \in \Sigma_u} p_1(q,\alpha) + \sum_{\alpha \in Pos(q)\cap\Sigma_c} p_2(r,\alpha) \leq 1.$$

The first parts of conditions (i) and (ii) of Theorem 1 correspond 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. All 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 is always enabled (for details, see [9], [17]).

When the conditions are satisfied, a solution to PSCP exists. After a string has been observed, the control input is given as a solution to the system of nonlinear equations given by (1). This solution is computed by the fixpoint iteration algorithm as presented in the following theorem.

Theorem 2: Assume that conditions (i) and (ii) of Theorem 1 are satisfied. Let

$$\Gamma(q) = \begin{cases} Pos(q) \cap \Sigma_c, & \text{if } Pos(q) \cap \Sigma_u \neq \emptyset \\ (Pos(q) \cap \Sigma_c) \backslash \{\gamma\}, & \text{otherwise} \end{cases}$$

where $\gamma \in Pos(q)$ is chosen such that for every $\sigma \in Pos(q)$, $p_2(r,\gamma)/p_1(q,\gamma) \geq p_2(r,\sigma)/p_1(q,\sigma)$ is satisfied. Let $x^0(s) \in [0,1]^{\Gamma(q)}$ and $f(s) : \mathbb{R}^{\Gamma(q)} \to \mathbb{R}^{\Gamma(q)}$. For $x^0(s) = 0$, the sequence

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

$$f(s)(x)(\sigma) = \frac{p_2(r,\sigma)}{p_1(q,\sigma) h(s)(x)(\sigma)}, \sigma \in \Gamma(q), x \in \mathbb{R}^{\Gamma(q)} \text{ and}$$

$$h(s)(x)(\sigma) = \sum_{\Theta \in \mathcal{P}(\Gamma(q)\backslash\{\sigma\})} \frac{1}{1 - \sum_{\alpha \in \Theta} p_1(q,\alpha)} \cdot \prod_{\alpha \in \Theta} (1 - x(s)(\alpha)) \prod_{\alpha \in \Gamma(q)\backslash\{\sigma\}\backslash\Theta} x(s)(\alpha) \tag{2}$$

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

V. Formal Proof

If there exists a probabilistic supervisor V_p such that $L_p(V_p/G_1) = L_p(G_2)$, then $L(G_2) \subseteq L(G_1)$. Therefore, let $s \in L(G_2)$ and assume there exists $q \in Q$ such that $q = \delta_1(q_0, s)$, and $r = \delta_2(r_0, s)$. For notational convenience, whenever obvious from the context, we will omit the symbols for strings and states so that, e.g., instead of $p_1(q, \sigma)$, we shall write $p_{1,\sigma}$, and instead of $p_2(r, \sigma)$, we shall write $p_{2,\sigma}$. We will use small Greek letters to denote events and capital Greek letters for sets of events. To denote sets whose elements are not necessarily events, capital Roman letters will be used, and small Roman letters will denote functions. Also, we assume the set difference operation to be left-associative.

Further, without loss of generality, we assume that in state q, not all the possible events are controllable, that is $\sum_{\sigma \in \Sigma_c} p_{1,\sigma} < 1$, or, equivalently, $Pos(q) \cap \Sigma_u \neq \emptyset$. This assumption is safe since if $p_{1,\sigma} = 0$ for all $\sigma \in \Sigma_u$, then the PSCP reduces to the PSCP with only controllable events which can be transformed into a problem with exactly one uncontrollable event (we will discuss this further in Section V-B). Note that in the case of at least one possible uncontrollable event, we have $\Gamma(q) = Pos(q) \cap \Sigma_c$. We will write Γ instead of $\Gamma(q)$.

After a string $s \in L(G_2)$ has been observed, the supervisory problem is effectively the problem of finding the control input vector $x(s) \in [0,1]^\Gamma$ such that $P(\sigma \text{ in } V_p/G|s) = p_{2,\sigma}$ for all $\sigma \in \Sigma$, where $P(\sigma \text{ in } V_p/G|s)$ is given by (1).

Proof

The detailed proofs are omitted due to space restrictions, and can be found in [17]. For the purposes of the proof, we will write x instead of $x(s)$, x_σ instead of $x(s)(\sigma)$, $f_\sigma(x)$ instead of $f(s)(x)(\sigma)$, $h_\sigma(x)$ instead of $h(s)(x)(\sigma)$, for $\sigma \in \Sigma$. Also, we will denote $P(\sigma \text{ in } V_p/G|s)$ by $P_\sigma(x)$.

Lemma 1: Let $\Psi \subseteq \Gamma$ and $x \in \mathbb{R}^\Psi$. Then:

$$\sum_{\Phi \subseteq \Psi} \prod_{\sigma \in \Phi} x_\sigma \prod_{\sigma \in \Psi \setminus \Phi} (1 - x_\sigma) = 1.$$

Lemma 2: Let $\Delta \subseteq \Gamma$ and $l : \mathcal{P}(\Gamma) \to \mathbb{R}$ be positive and monotone. The function $f_\Delta : \mathbb{R}^\Gamma \to \mathbb{R}$ given by

$$f_\Delta(x) = \sum_{\Phi \in \mathcal{P}(\Delta)} l(\Phi) \prod_{\sigma \in \Phi} (1 - x_\sigma) \prod_{\sigma \in \Delta \setminus \Phi} x_\sigma$$

is positive and antitone on $[0,1]^\Gamma$.

Lemma 3: Let $x \in [0,1]^\Gamma$. Then, $P_\sigma(x) = p_{1,\sigma} x_\sigma h_\sigma(x)$ for every $\sigma \in \Sigma$, where $h_\sigma : \mathbb{R}^\Gamma \to \mathbb{R}$ is given by

$$h_\sigma(x) = \sum_{\Theta \in \mathcal{P}(\Gamma \setminus \{\sigma\})} \frac{1}{1 - \sum_{\alpha \in \Theta} p_{1,\alpha}} \prod_{\alpha \in \Theta} (1 - x_\alpha) \prod_{\alpha \in \Gamma \setminus \{\sigma\} \setminus \Theta} x_\alpha. \quad (3)$$

Next, we introduce a partial order on $\mathbb{R}^\Gamma$. For $x, y \in \mathbb{R}^\Gamma$, $x \leq y$ iff $\forall \sigma \in \Gamma \; x_\sigma \leq y_\sigma$. Let $f : (X, \leq) \to (Y, \leq)$ be a mapping between posets. This mapping is monotone if whenever $x \leq y$, then $f(x) \leq f(y)$; it is antitone if whenever $x \leq y$, then $f(x) \geq f(y)$. Also, for $a \in \mathbb{R}$, let $\bar{a}$ denote $x \in \mathbb{R}^\Gamma$ such that $x_\sigma = a$ for all $\sigma \in \Gamma$.

Lemma 4: The functions $h_\sigma(x)$, $\sigma \in \Sigma$, as defined in Lemma 3 are positive antitone, and such that $x_\sigma h_\sigma(x) \leq h_\gamma(x)$ on $[0,1]^\Gamma$ for all $\sigma \in \Sigma_c, \gamma \in \Sigma_u$.

Let $\{x^k\}$ be a sequence of real numbers, and $x \in \mathbb{R}$. We will write $x^k \uparrow x$ iff $x^k \leq x^{k+1}$ for all $k \in \mathbb{N}$, and $x^k \to x$ as $k \to \infty$. The following lemma gives sufficient conditions for the existence of a fixpoint of the function f.

Lemma 5: Let $f : D \to \mathbb{R}^\Gamma$ be a monotone function on $D \subseteq \mathbb{R}^\Gamma$, $a_0, b_0 \in \mathbb{R}$, such that $a_0 \leq b_0$, $[a_0, b_0]^\Gamma \subseteq D$ and $\bar{a}_0 \leq f(\bar{a}_0)$. Assume that for every $x \in [a_0, b_0]^\Gamma$ such that $x \leq f(x)$, we have $f(x) \leq \bar{b}_0$. Then, the sequence $\{x^k\}$ given by

$$x^0 = \bar{a}_0, \quad x^{k+1} = f(x^k), \quad k = 0, 1, \ldots$$

exists and is such that $x^k \uparrow x^*$ for some $x^* \in [a_0, b_0]^\Gamma$. If, furthermore, f is lower continuous ($\lim_{x^k \uparrow x} f(x^k) = f(x)$), then $x^* = f(x^*)$.

The following theorem presents necessary and sufficient conditions for controllable events' probabilities to be assignable to given probabilities. If the conditions hold, the fixpoint algorithm to calculate the control input is given.

Theorem 3: Assume that $Pos(r) \cap \Sigma_c \subseteq \Gamma$, and, for every $\sigma \in \Gamma$:

$$\frac{p_{2,\sigma}}{p_{1,\sigma}} \sum_{\alpha \in \Sigma_u} p_{1,\alpha} + \sum_{\alpha \in \Gamma} p_{2,\alpha} \leq 1. \quad (4)$$

Then, the sequence $\{x^k\}$, $k = 0, 1, \ldots$, given by

$$x^0 = 0, \quad x^{k+1} = f(x^k), \quad \text{where } f_\sigma(x) = \frac{p_{2,\sigma}}{p_{1,\sigma} h_\sigma(x)}, \sigma \in \Gamma$$

exists and is such that $x^k \uparrow x^*$ for some $x^* \in [0,1]^\Gamma$. Furthermore, $P_\sigma(x^*) = p_{2,\sigma}$ for all $\sigma \in \Sigma_c$. Conversely, for any $x \in [0,1]^\Gamma$, if $p_{2,\sigma} = P_\sigma(x)$ for all $\sigma \in \Sigma_c$, then $Pos(r) \cap \Sigma_c \subseteq \Gamma$ and (4) holds.

So far we have considered the conditions under which the probabilities of controllable events can be assigned to specified probabilities. Next, we consider uncontrollable events as well.

Lemma 6: There exists $x \in [0,1]^\Gamma$ such that $P_\sigma(x) = p_{2,\sigma}$ for every $\sigma \in \Sigma$ iff $Pos(r) \cap \Sigma_c \subseteq \Gamma$, $Pos(q) \cap \Sigma_u = Pos(r) \cap \Sigma_u$, (4) holds for every $\sigma \in \Gamma$, and for every $\sigma \in Pos(q) \cap \Sigma_u$

$$\frac{p_{1,\sigma}}{\sum_{\alpha \in \Sigma_u} p_{1,\alpha}} = \frac{p_{2,\sigma}}{\sum_{\alpha \in \Sigma_u} p_{2,\alpha}}. \quad (5)$$

A. Special Case: $Pos(q) \cap \Sigma_u = \emptyset$

We now address the issue previously mentioned: in a certain state, only controllable events can happen in the plant. Then, a probabilistic supervisor can disable them all which would cause termination. However, as we consider nonterminating generators, this is not allowed. An elegant solution is to always enable one event: this event effectively becomes uncontrollable and the problem reduces to the one already proved. We now show that, if an event γ with the maximal ratio $p_{2,\gamma}/p_{1,\gamma}$ is chosen, then the condition (4) is satisfied.

Formally, let $Pos(q) \cap \Sigma_u = \emptyset$. Then, only for this local problem, we declare event $\gamma \in Pos(q) \cap \Sigma_c$ to be uncontrollable. Then, $\Gamma(q) = (Pos(q) \cap \Sigma_c) \setminus \{\gamma\}$, denoted Γ for simplicity. The left hand side of the condition in (4), for $\sigma \in \Gamma$, becomes

$$\begin{aligned} \frac{p_{2,\sigma}}{p_{1,\sigma}} p_{1,\gamma} + \sum_{\alpha \in \Gamma} p_{2,\alpha} &= \frac{p_{2,\sigma}}{p_{1,\sigma}} p_{1,\gamma} - p_{2,\gamma} + \sum_{\alpha \in \Gamma} p_{2,\alpha} + p_{2,\gamma} \\ &= \frac{p_{2,\sigma}}{p_{1,\sigma}} p_{1,\gamma} - p_{2,\gamma} + \sum_{\alpha \in Pos(q) \cap \Sigma_c} p_{2,\alpha} \\ &= \frac{p_{2,\sigma}}{p_{1,\sigma}} p_{1,\gamma} - p_{2,\gamma} + 1. \end{aligned}$$

Then, the condition (4) becomes

$$\frac{p_{2,\sigma}}{p_{1,\sigma}} p_{1,\gamma} - p_{2,\gamma} \leq 0$$

which, since $p_{1,\gamma} > 0$, is equivalent to

$$\frac{p_{2,\sigma}}{p_{1,\sigma}} - \frac{p_{2,\gamma}}{p_{1,\gamma}} \leq 0.$$

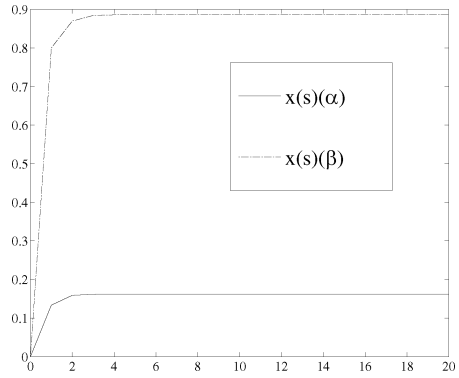

Fig. 4. Fixpoint iteration.

If the event γ is one with the maximal ratio $p_{2,\gamma}/p_{1,\gamma}$ (meaning, for every $\sigma \in Pos(q)$, $p_2(r,\gamma)/p_1(q,\gamma) \geq p_2(r,\sigma)/p_1(q,\sigma)$), it is obvious that the condition is satisfied for any $\sigma \in \Gamma$. Now, it is easy to show that this case and Lemma 6 result in Theorem 1. Further, if the conditions of Theorem 1 are satisfied, the algorithm for computation of control input from Theorem 3 can be applied to this special case, with γ considered an uncontrollable event.

B. Example

We now present the calculation of a probabilistic supervisor for the example from Fig. 1, where $\Sigma_c = \{\alpha, \beta\}$, and $\Sigma_u = \{\gamma\}$. The case when the string $s \in L(G_2)$ has been observed such that G is at the state q_0, and G_2 is at q_{20} will be presented in detail. Again, for notational convenience, we shall write $p_{1,\sigma}$ instead of $p_1(q_0,\sigma)$, and $p_{2,\sigma}$ instead of $p_2(q_{20},\sigma)$, where $\sigma \in \Sigma$. Let $p_1 = (p_{1,\alpha}, p_{1,\beta}, p_{1,\gamma})$, $p_2 = (p_{2,\alpha}, p_{2,\beta}, p_{2,\gamma})$, and $x(s) = (x(s)(\alpha), x(s)(\beta))$. From Fig. 1, it follows $p_1 = (0.6, 0.2, 0.2), p_2 = (0.1, 0.4, 0.5)$. First, we check the conditions of Theorem 1. The equality of Theorem 1 is trivially satisfied. We then check if the inequalities of Theorem 1 are satisfied: $1/6 \cdot p_{1,\gamma} + p_{2,\alpha} + p_{2,\beta} = 0.53 \leq 1, 2p_{1,\gamma} + p_{2,\alpha} + p_{2,\beta} = 0.9 \leq 1$. Then, the control input $x^*(s)$ can be calculated by the fixpoint iteration where $x^0(s) = (0,0)$, $x^k(s) = f(s)(x^{k-1}(s))$, and, for $x \in \mathbb{R}^{\{\alpha,\beta\}}$

$$f(s)(x)(\alpha) = \frac{p_{2,\alpha}}{p_{1,\alpha}\left(x(\beta) + \frac{1}{1-p_{1,\beta}}(1 - x(\beta))\right)}$$
$$f(s)(x)(\beta) = \frac{p_{2,\beta}}{p_{1,2}\left(x(\alpha) + \frac{1}{1-p_{1,\alpha}}(1 - x(\alpha))\right)}.$$

After just a few iterations, the sequence $\{x^k(s)\}$ converges to $x^*(s) = (0.162, 0.886)$ (see Fig. 4).

The calculation of the supervisor after the string $t \in L(G_2)$ has been observed such that G_2 is in the state q_{21} gives the result $x^*(t) = (0.533, 0)$. The supervisor is shown in Fig. 3.

VI. Complexity Analysis of the Synthesis Problem and Algorithm

For the purposes of complexity analysis, let us restate the synthesis problem and the algorithm. Let G_1' and G_2' be the nonprobabilistic automata underlying generators G_1 and G_2, i.e., $G_1' = (Q, \Sigma, \delta_1, q_0, Q_m)$ and $G_2' = (R, \Sigma, \delta_2, r_0, R_m)$. Also, let the nonprobabilistic automaton $G_s = (Q_s, \Sigma, \delta_s, q_{s0}, Q_{ms})$ be the synchronous product of G_1' and G_2'. The construction of G_s and the check of conditions of Theorem 1 can be performed in time $O(|Q| \cdot |R| \cdot |\Sigma|)$. Assume that the conditions of Theorem 1 are satisfied. Then, for each state $(q,r) \in Q_s$ of the automaton G_s (there are at most $|Q| \cdot |R|$ states), control input $x(s) \in [0,1]^{\Gamma(q)}$, where $s \in L(G_s)$ such that $q = \delta_1(q_0, s)$, and $r = \delta_2(r_0, s)$, is the solution of the system of nonlinear polynomial equations

$$x = f(s)(x) \tag{6}$$

on the interval $[0,1]^{\Gamma(q)}$, where $f(s)$ is defined as in (2). This control input can be calculated using the algorithm from Theorem 2.

We assume that the probabilities of both the plant and specification are rational. Even in this case, control inputs are, in general, irrational. E.g., let us consider the plant and specification in Fig. 1, after string s has been observed such that PDES G is in state q_0 and PDES G_2 is in state q_{20}. Then, solving for $x = x(s)(\alpha)$, the system of (6) reduces to the equation $45x^2 - 69x + 10 = 0$, whose roots are irrational numbers, and, therefore, cannot be computed exactly. Hence, the best we could do is approximate the supervisor's probabilities to a certain accuracy. The theoretical complexity of this problem is equal to the theoretical complexity of approximating the solution of the system of nonlinear polynomial (6). It is known that even for systems of quadratic equations, the problem is at least exponentially hard [18].

For deriving the upper bounds on complexity of the problem, we use reasoning similar to the one presented in [19]. We resort to the the results on complexity of the decision procedures of the Existential Theory of Reals, $ExTh(\mathbb{R})$. Results of [20], [21] give the upper time complexity bounds for deciding sentences in $ExTh(\mathbb{R})$. A sentence in $ExTh(\mathbb{R})$ is of the form: $\Phi \equiv \exists x_1, \ldots, x_n P(x_1, \ldots, x_n)$, where P is a quantifier free boolean formula with "atomic predicates" of the form $g_i(x_1, \ldots, x_n) \triangle 0$, where g_i is a (multivariate) polynomial with rational coefficients, and $\triangle \in \{>, \geq, =, \neq, \leq, <\}$. Let m be the number of atomic predicates g_i, and d the maximal degree of polynomials g_i. Then, there is an algorithm that decides if the sentence Φ is true over real numbers, that runs in PSPACE, and in time $O((md)^{O(n)})$. This complexity result contains an implicit assumption that the validity of P can be decided in constant time (given the truth values of its atomic predicates); this assumption serves to simplify the result and does not have a significant impact on the following complexity results.

It is easy to construct a sentence in $ExTh(\mathbb{R})$ that compares $x(s)(\alpha)$ ($\alpha \in \Gamma(q)$) to a rational number. The sentence

$$\Phi(s)(\alpha) \equiv \exists x(s)\left(x(s) = f(s)(x(s)) \wedge \bigwedge_{\sigma \in \Gamma(q)} 0 \leq x(s)(\sigma) \leq 1 \wedge x(s)(\alpha) < u\right)$$

checks if there exists a solution $x(s) \in [0,1]^{\Gamma(q)}$ of (6) such that $x(s)(\alpha)$ is less than a rational number u. Since each $x(s)(\sigma)$ ($\sigma \in \Gamma(q)$) is in the interval $[0,1]$, we can use binary search and queries similar to $\Phi(s)(\alpha)$ to close in on the value of a control input up to an accuracy 10^{-i}, $i \in \mathbb{N}$. In order to reach this accuracy, we need to use $O(i \cdot |\Gamma(q)|)$ queries. Therefore, there is an algorithm that approximates the solution of (6) up to prespecified accuracy 10^{-i}, and it runs in time $O(i \cdot |\Gamma(q)|^{O(|\Gamma(q)|)})$.

On the other hand, a straightforward analysis of (6) suggests that the worst-case running time of one iteration of the fixpoint algorithm that approximates the solution of (6) is $O(|\Gamma(q)|^2 \cdot 2^{|\Gamma(q)|})$. Since the number $|\Gamma(q)|$ is typically small in practical applications, this complexity does not represent a practical limitation of the algorithm. The exact rate of convergence is not yet known. However, experimental results indicate that for $Pos(r) \cap \Sigma_c = \emptyset$, the convergence is superlinear, whereas for $Pos(r) \cap \Sigma_c \neq \emptyset$, the convergence is linear, with the convergence factor of at most $K(r) = \sum_{\sigma \in Pos(r) \cap \Sigma_c} p_2(r,\sigma) < 1$. Practically, this means that when the value $K(r) \leq 0.6$, the algorithm

converges fast (gaining one decimal of precision in at most five iterations). When $0.6 < K(r) < 0.9$, it takes not more than 25 iterations per decimal of precision. For $K(r)$ very close to 1, the number of iterations per decimal of precision $(1/\log(1/K(r)))$ can become quite large.

VII. Conclusion

In this paper, we presented the solution of the probabilistic supervisory control problem as introduced in [9] and [10]. The control technique used is called random disablement: events are enabled with certain probabilities. Necessary and sufficient conditions for the existence of the solution are introduced and a fixpoint algorithm for the computation of supervisor is given. Detailed formal proofs are presented in [17].

We are currently working on the problem of finding a supervisor that provides the closest approximation to a probabilistic specification language when there does not exist an exact solution to the probabilistic supervisory control problem (the conditions of Theorem 1 fail) [22].

References

[1] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optimiz,*, vol. 25, no. 1, pp. 206–230, 1987.

[2] C. G. Cassandras, *Discrete Event Systems: Modeling and Performance Analysis*. Homewood, IL: Richard D. Irwin, Inc. and Aksen Associates, Inc., 1993.

[3] M. O. Rabin, "Probabilistic automata," *Inform. and Control*, vol. 6, no. 3, pp. 230–245, Sep. 1963.

[4] M. Molloy, "Performance analysis using stochastic Petri Nets," *IEEE Trans. Compute.*, vol. C-39, no. 9, pp. 913–917, Sep. 1982.

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

[6] V. Garg, "Probabilistic languages for modeling of DEDS," in *Proc. 26th Conf. Information Sciences and Systems*, Princeton, NJ, Mar. 1992, vol. 1, pp. 198–203.

[7] H. Mortazavian, "Controlled stochastic languages," in *Proc. 31st Annu. Allerton Conf. Communications, Control, and Computing*, Urbana, IL, 1993, pp. 938–947.

[8] V. S. Borkar, *Topics in Controlled Markov Chains*. New York: Wiley, 1991.

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

[10] S. Postma and M. Lawford, "Computation of probabilistic supervisory controllers for model matching," in *Proc. Allerton Conf. Communications, Control, and Computing*, Monticello, CA, 2004.

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

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

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

[14] 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.

[15] C. Baier, M. Größer, M. Leucker, B. Bollig, and F. Ciesinski, J.-J. Lévy, E. W. Mayr, and J. C. Mitchell, Eds., "Controller synthesis for probabilistic systems," in *Proc. IFIP Int. Conf. Theoretical Computer Science*, 2004, pp. 493–506.

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

[17] V. Pantelic, S. Postma, and M. Lawford, Supervisory Control of Probabilistic Discrete Event Systems Software Quality Research Lab, McMaster Univ., Hamilton, ON, Canada, 2009, Tech. Rep. 21.

[18] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational Complexity and Feasibility of Data Processing and Interval Computations*. Dordrecht, The Netherlands: Kluwer, 1998.

[19] K. Etessami and M. Yannakakis, "Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations," *J. ACM*, vol. 56, no. 1, 2009.

[20] J. Canny, "Some algebraic and geometric computations in PSPACE," in *STOC '88: Proc. 20th Annu. ACM Symp. Theory of Computing*, New York, 1988, pp. 460–469.

[21] J. Renegar, "On the computational complexity and geometry of the first-order theory of the reals, Part I-III," *J. Symb. Comput.*, vol. 13, no. 3, pp. 255–352, 1992.

[22] V. Pantelic and M. Lawford, "Towards optimal supervisory control of probabilistic discrete event systems," in *Proc. 2nd IFAC Workshop on Dependable Control of Discrete Systems*, Bari, Italy, Jun. 2009.

Session-Level Load Balancing for High-Dimensional Systems

Dennis Roubos and Sandjai Bhulai

***Abstract*—Load balancing is critical for the performance of big server clusters. Although many load balancers are available for improving performance in parallel applications, the load-balancing problem is not fully solved yet. Recent advances in security and architecture design advocate load balancing on a session level. However, due to the high dimensionality of session-level load balancing, little attention has been paid to this new problem. In this paper, we formulate the session-level load-balancing problem as a Markov decision problem. Then, we use approximate dynamic programming to obtain approximate load-balancing policies that are scalable with the problem instance. Extensive numerical experiments show that the policies have nearly optimal performance.**

***Index Terms*—Approximate dynamic programming, Markov decision processes, session-level load balancing.**

I. Introduction

Many content-intensive applications have scaled beyond the point where a single server can provide adequate processing power. This raises the need for flexibility to deploy additional servers quickly and transparently to end-users. The technique that addresses this need is load balancing, i.e., the process of transparently distributing service requests across a group of servers. It also addresses several requirements that are becoming increasingly important in computer networks, such as *increased scalability*, *high performance*, and *high availability and disaster recovery*.

While many effective load-balancing strategies have been developed that perform load balancing on the level of service requests, new applications, and architectures require load balancing on the level of user sessions. In these cases, the load-balancing algorithm is carried out only when a user requests a new session. This load-balancing problem on a session level is not yet solved completely and has received little attention due to its complexity.

Manuscript received July 18, 2008; revised July 21, 2008 and March 17, 2009. Current version published August 05, 2009. Recommended by Associate Editor E. Fabre.

The authors are with the VU University, Faculty of Sciences, De Boelelaan 1081a, 1081 HV Amsterdam. (e-mail: droubos@few.vu.nl; sbhulai@few.vu.nl).

Digital Object Identifier 10.1109/TAC.2009.2024378