CONTROL OF STOCHASTIC DISCRETE EVENT SYSTEMS: EXISTENCE

Ratnesh Kumar, Vijay K. Garg [1]

Dept. of Elec. Eng., Univ. of Kentucky, and Applied Research Lab., Penn. State Univ.
[1] Dept. of ECE, Univ. of Texas, Austin, TX

Keywords
Stochastic Discrete Event Systems, Probabilistic Language, Supervisory Control [1]

Abstract

In earlier papers [3, 2, 1] we introduced the formalism of *probabilistic languages* for modeling the stochastic qualitative behavior of discrete event systems (DESs). In this paper we present a framework for their supervisory control, where control is exercised by dynamically disabling certain controllable events. The control objective is to design a supervisor such that the controlled system never executes any illegal traces (their occurrence probability is zero), and legal traces occur with minimum prespecified occurrence probabilities. We provide a condition for the existence of a supervisor. We also present an algorithm to test this existence condition when the probabilistic languages are regular (so that they admit probabilistic automata representation with finitely many states).

1 Introduction

The *non-stochastic* behavior of a discrete event system can be viewed as a binary valued map over the set of all possible sequences of events, called traces. (A trace is mapped to one if and only if it belongs to the system behavior.) We call such a map to be *non-probabilistic language*. In [3] we introduced a more general map over the set of all traces that takes values in the closed unit interval (instead of just in the binary set) to describe the stochastic qualitative behavior of a DES. The interpretation being that value associated with a certain trace under such a map is its occurrence probability. In order for such a probabilistic map to describe the stochastic behavior of a DES it must satisfy certain consistency properties obtained in [3]: (i) the probability of the zero length trace is one, and (ii) the probability of any trace is at least as much as the cumulative probability of all its extensions. We call such maps to be *probabilistic languages* and use them for modeling the stochastic qualitative behavior of DESs.

Sengupta [8, Chapter 5] studies the problem of optimal control of stochastic behaviors of DESs. The controller changes the occurrence probabilities of events. A cost is assigned with each control action, and the control objective is to minimize the cumulative cost over an infinite horizon. The optimal control problem is the classical infinite horizon optimal control of Markov processes.

In contrast, in this paper we study the problem of supervisory control of stochastic behavior of DESs modeled as probabilistic languages. A supervisor restricts the behavior of the plant by dynamically disabling certain set of controllable events based on its observations of the executed traces. Thus the occurrence probability of disabled events becomes zero, whereas the occurrence probability of the enabled ones increases proportionately. Hence supervision restricts the *support* of the probabilistic language of the plant, but increases the occurrence probabilities of the surviving traces within this restricted support.

The control objective is specified as a lower and an upper bound constraint. The upper bound constraint imposes a legality constraint specifying that a trace be enabled if and only if it is legal. Thus the upper bound constraint can be represented as a non-probabilistic language that maps a trace to the value one if and only if it is legal. The second constraint imposes a level of desirability on the legal traces by specifying a lower bound on their occurrence probabilities. This constraint is given as a probabilistic map over the set of traces, and the control objective is to ensure that each legal trace occurs with probability at least as much as specified by this lower bound. Intuitively, we are interested in designing a supervisor so that "bad" traces never occur, whereas the "good" traces occur with certain minimum probabilities. This generalizes the supervisory control problem studied in the non-stochastic setting where both the upper and lower bounds are non-probabilistic languages.

Proc. of the Fourth Workshop on Discrete Event Systems
[1] This research is supported in part by the National Science Foundation under Grants NSF-ECS-9409712, NSF-ECS-9709796, ECS-9414780, and CCR-9520540, in part by the Office of Naval Research under the Grant ONR-N00014-96-1-5026, a General Motors Fellowship, a Texas Higher Education Coordinating Board Grant ARP-320, and an IBM grant.

We obtain a necessary and sufficient condition for the existence of the supervisor for the above control problem. The complexity of verifying the condition is the same as that of shortest path computation, i.e., $O(n^3)$, where n is the product of the number of states in the automata representations of the two probabilistic languages.

2 Notation and Preliminaries

We use Σ to denote the universe of events over which a given DES evolves. The set Σ^* is the set of all finite length event sequences, called traces, including the zero length trace, denoted ϵ. A subset of Σ^* is called a language. Given traces s and t, we use $s \leq t$ to denote that s is a *prefix* of t, in which case the notation $s^{-1}t$ is used to denote the suffix of t obtained by removing the prefix s, i.e., $t = ss^{-1}t$. Given a language K, we use $pr(K)$, prefix closure of K, to denote the set of all prefixes of K; K is said to be prefix closed if $K = pr(K)$.

Qualitative behavior of DESs is described by languages. A language $L \subseteq \Sigma^*$ can be viewed as a unit interval valued map—a *probabilistic map*—over Σ^*, $L : \Sigma^* \to [0,1]$. For a probabilistic map L, its *support*, denoted $supp(L) \subseteq \Sigma^*$, is the set of traces such that $L(s) > 0$. L is said to be a *non-probabilistic map* if $L(s) \in \{0,1\}$ for each trace s. Clearly, languages can also be represented by non-probabilistic maps. A non-probabilistic map L models the non-stochastic qualitative behavior of a DES if

$$L(\epsilon) = 1; \forall s \in \Sigma^*, \sigma \in \Sigma : L(s\sigma) = 1 \Rightarrow L(s) = 1.$$

This is because a system can always execute the epsilon trace, and if it can execute a trace, then it can also execute all its prefixes. We call such maps to be *non-probabilistic languages* or np-languages.

The notion of *probabilistic languages* or p-languages was introduced in [3] to model the stochastic qualitative behavior of DESs. A definition of p-languages based on their underlying probability measure space was presented in [3]. A p-language L can alternatively be viewed as a probabilistic map satisfying the following constraints:

P1: $L(\epsilon) = 1$
P2: $\forall s \in \Sigma^* : \sum_{\sigma \in \Sigma} L(s\sigma) \leq L(s)$

Here for each trace s, $L(s)$ gives its probability of occurrence. Condition P1 follows from the fact that a system can always execute the epsilon trace, whereas the condition P2 follows from the fact that for any extension of a trace s to be executable, s must itself be executable.

It follows from the definition of a p-language L that $\Delta(L) : \Sigma^* \to [0,1]$ defined as:

$$\forall s \in \Sigma^* : \Delta(L)(s) := L(s) - \sum_{\sigma \in \Sigma} L(s\sigma)$$

satisfies $\Delta(L) \geq 0$. $\Delta(L)(s)$ gives the probability that the system modeled as p-language L terminates following the execution of s.

Qualitative behavior of DESs can alternatively be represented by automata. An automaton G over the event set Σ is a quadruple, $G := (X, \Sigma, x_{init}, P)$, where X is the set of states of G, $x_{init} \in X$ is the initial state of G, and $P : X \times \Sigma \times X \to [0,1]$ is the state transition function of G. A triple $(x, \sigma, x') \in X \times \Sigma \times X$ is called a *transition*. G is called a *non-probabilistic automaton* or np-automaton if $P(x, \sigma, x') \in \{0,1\}$ for each transition (x, σ', x); it is said to be a *probabilistic automaton* or p-automaton [3] if

$$\forall x \in X : \sum_{x' \in X} \sum_{\sigma \in \Sigma} P(x, \sigma, x') \leq 1.$$

For a p-automaton G, we define

$$\forall x \in X : \Delta(G)(x) := 1 - \sum_{x' \in X} \sum_{\sigma \in \Sigma} P(x, \sigma, x')$$

to be the *probability of termination at state* x. An np-automaton (resp., p-automaton) is said to be *deterministic np-automaton* or dnp-automaton (resp., *deterministic p-automaton* or dp-automaton) if

$$\forall x \in X, \sigma \in \Sigma : |\{x' \in X \mid P(x, \sigma, x') > 0\}| \leq 1.$$

The state transition function of G can be extended to the set of *paths* $X(\Sigma X)^*$, where a path is obtained by concatenating transitions such that the end and start states of consecutive transitions are the same. Given a path $\pi = x_0\sigma_1 x_1 \ldots \sigma_n x_n \in X(\Sigma X)^*$, we use $|\pi| = n$ to denote its length; for each $k \leq |\pi|$, $\pi^k := x_0\sigma_1 x_1 \ldots \sigma_k x_k$ to denote its initial sub-path of length k; and $tr(\pi) := \sigma_1 \ldots \sigma_n$ to denote its trace. The state transition function is extended inductively to the set of paths as follows:

$$\forall x \in X : P(x) = 1$$
$$\forall \pi \in X(\Sigma X)^*, \sigma \in \Sigma, x' \in X : P(\pi, \sigma, x') = P(\pi)P(x_{|\pi|}, \sigma, x').$$

If G is a np-automaton, then *np-language generated by* G is given by

$$[L_G(s) := 1] \Leftrightarrow [\exists \pi \in X(\Sigma X)^* : tr(\pi) = s, P(\pi) = 1].$$

If G is a p-automaton, then the *p-language generated by* G is given by

$$L_G(s) := \sum_{\pi : tr(\pi) = s, \pi^0 = x_{init}} P(\pi).$$

It is easy to see that L_G is a np-language when G is a np-automaton, and it was shown in [3] that L_G is a p-language when G is a p-automaton. Conversely,

given a np-language (resp., p-language) there exists a deterministic np-automaton (resp., deterministic p-automaton) that generates it [3].

A np-language (resp., p-language) L is said to be *regular* if there exists a np-automaton (resp., p-automaton) G with finitely many states such that $L_G = L$. A regular np-language (resp., regular p-language) L is called *deterministic regular* if there exists a dnp-automaton (resp., dp-automaton) with finite states such that $L_G = L$.

Given a pair of automata $G^i := (X^i, \Sigma, x^i_{init}, P^i)$, $(i = 1, 2)$, their synchronous composition is another automaton $G := (X, \Sigma, x_{init}, P)$, where $X := X^1 \times X^2$, $x_{init} := (x^1_{init}, x^2_{init})$, and

$$P((x^1, x^2), \sigma, (\overline{x}^1, \overline{x}^2)) := P^1(x^1, \sigma, \overline{x}^1) P^2(x^2, \sigma, \overline{x}^2),$$

where $x^1, \overline{x}^1 \in X^1, x^2, \overline{x}^2 \in X^2, \sigma \in \Sigma$. It is easy to see that if G^i's are deterministic, regular, p-automata, np-automata, respectively, then so is G. Furthermore, $supp(L_G) = supp(L_{G^1}) \cap supp(L_{G^2})$.

Given a set X, a partial order on X, denoted $\preceq$, is a binary relation that is reflexive, antisymmetric, and transitive. The pair $(X, \preceq)$ is called a partially order set or a *poset*. For a pair of elements $x, y \in X$, their infimum and supremum whenever defined are unique. A poset $(x, \preceq)$ is said to be a upper (resp., lower) semi-lattice if supremum (resp., infimum) for any pair of elements in X exists; it is said to be a complete upper (resp., lower) semi-lattice if supremum (resp., infimum) of any subset of X exists; it is said to be a (complete) lattice if it is both (complete) upper and lower semi-lattice. A set $Y \subseteq X$ is called a chain if it is totally ordered, in which case Y can be written as a monotonically increasing sequence of poset elements $Y = \{x_i\}_{i \geq 0}$ with $x_i \preceq x_j$ whenever $i \leq j$. A poset $(X, \preceq)$ is called a complete partial order or a cpo if it has the bottom element and every chain has the supremum element. A function $f : X \to X$ is called monotone if it preserves ordering under its transformation; it is said to be continuous if it distributes with supremum taken over a chain.

The set of unit interval valued probabilistic maps over Σ^* forms a poset under the following natural ordering relation introduced in [3]:

$$\forall K, L : \Sigma^* \to [0,1] : [K \preceq L] \Leftrightarrow [\forall s \in \Sigma^* : K(s) \leq L(s)].$$

It is easy to see that the set of all non-probabilistic maps is a complete lattice under this ordering. Also, it was shown in [3] that the set of all p-languages forms a cpo under this ordering. It was also shown that the set of p-languages forms a complete lower semi-lattice under this ordering.

For supervisory control of the qualitative behavior of a discrete event plant the set of events is partitioned into $\Sigma_u \cup (\Sigma - \Sigma_u)$, the sets of uncontrollable and controllable events. For a discrete event plant with behavior modeled by a np-language or a p-language L, a supervisor S with complete observation of traces is a map $S : supp(L) \to 2^{\Sigma - \Sigma_u}$ that, following the occurrence of a trace $s \in supp(L)$, disables the controllable events in the set $S(s) \subseteq \Sigma - \Sigma_u$ from occurring next. The behavior of the controlled plant is denoted by L^S. For a np-language L, the controlled behavior L^S is also a np-language defined inductively as:

$$L^S(\epsilon) := 1; L^S(s) = 1, L(s\sigma) = 1, \sigma \notin S(s) \Rightarrow L^S(s\sigma) := 1.$$

Given a language $K \subseteq \Sigma^*$, and a plant with np-language or p-language L, K is said to be *controllable* with respect to L if $pr(K)\Sigma_u \cap supp(L) \subseteq pr(K)$. It is known that there exists a supervisor S for a plant with np-language L such that $supp(L^S) = K$ if and only if K is nonempty, prefix closed, and controllable [6]. The set of prefix closed and controllable languages forms a complete lattice. So the *infimal prefix closed and controllable superlanguage* of a language K, denoted $inf\overline{PC}(K)$ [4], and its *supremal prefix closed and controllable sublanguage*, denoted $sup\underline{PC}(K)$ [6], exist and are effectively computable.

3 Existence of Supervisor

In this section we extend the supervisory control framework to the stochastic setting. Given a DES with stochastic behavior modeled as p-language L, we use $(s^{-1}L)(t)$ to denote the probability of trace t given that trace s has already occurred:

$$\forall s, t \in \Sigma^* : (s^{-1}L)(t) := L(st|s) = \frac{L(st \wedge s)}{L(s)} = \frac{L(st)}{L(s)},$$

where the last equality follows from the fact that the outcome that trace st and trace s have occurred is equivalent to the outcome that the trace st has occurred, since occurrence of st implies occurrence of the prefix s also. We thus have

$$L(st) = L(st \wedge s) = L(st|s)L(s) = (s^{-1}L)(t)L(s).$$

We have the following simple lemma about $s^{-1}L$.

Lemma 1 Let L be a p-language. Then for each $s \in \Sigma^*$ we have:

1. $\forall t \in \Sigma^* : \Delta(s^{-1}L)(t) = \frac{\Delta(L)(st)}{L(s)}$.
2. $s^{-1}L$ is a p-language.

Remark 1 If L is a deterministic p-language, so that there exists a dp-automaton $G := (X, \Sigma, x_{init}, P)$ such that $L_G = L$, then for each trace s there exists a unique path $\pi_s \in X(\Sigma X)^*$ such that $tr(\pi) = s$, and $L(s) = P(\pi_s)$. This implies

$$P(x_{|\pi_s|}, \sigma, x_{|\pi_{s\sigma}|}) = (s^{-1}L)(\sigma); \Delta(G)(x_{|\pi_s|}) = \Delta(s^{-1}L)(\epsilon).$$

As described above, a supervisor $S : supp(L) \rightarrow 2^{\Sigma - \Sigma_u}$ determines the set of controllable events $S(s) \subseteq \Sigma - \Sigma_u$ to be disabled following the execution of trace s. In the next lemma we obtain the value of $(s^{-1}L^S)(\sigma)$, the occurrence probability of event σ in the controlled plant given that trace s has already occurred. It states that this probability is zero when $\sigma \in S(s)$, and otherwise it equals the corresponding occurrence probability in the uncontrolled plant scaled by the appropriate normalization factor. For the notational convenience, given a plant with p-language L and a supervisor $S : supp(L) \rightarrow 2^{\Sigma - \Sigma_u}$, we define a probabilistic map $\Phi^S(L) : \Sigma^* \rightarrow [0, 1]$:

$$\forall s \in \Sigma^* : \Phi^S(L)(s) := \Delta(s^{-1}L)(\epsilon) + \sum_{\hat{\sigma} \in \Sigma - S(s)} (s^{-1}L)(\hat{\sigma}).$$

$\Phi^S(L)(s)$ computes *for the controlled plant the probability of either termination or execution of an enabled event given that the trace s has already occurred.*

Lemma 2 Let L be the p-language of a DES, and $S : supp(L) \rightarrow 2^{\Sigma - \Sigma_u}$ be a supervisor. Then

$$(s^{-1}L^S)(\sigma) = \begin{cases} 0 & \text{if } \sigma \in S(s) \\ \frac{(s^{-1}L)(\sigma)}{\Phi^S(L)(s)} & \text{otherwise} \end{cases}$$

Using the result of Lemma 2 we define the p-language of the controlled plant next (that it is indeed a p-language is established below in Theorem 1):

Definition 1 Let L be a p-language of a DES, and $S : supp(L) \rightarrow 2^{\Sigma - \Sigma_u}$ be a supervisor. The p-language of the controlled plant, denoted L^S, is defined inductively as:

$$L^S(\epsilon) := 1; L^S(s\sigma) := L^S(s)(s^{-1}L^S)(\sigma).$$

The following corollary states that the effect of the control is to restrict the support, but to increase the occurrence probabilities of the surviving traces within this restricted support.

Corollary 1 Let L be the p-language of a DES, and $S : supp(L) \rightarrow 2^{\Sigma - \Sigma_u}$ be a supervisor. Then

1. $supp(L^S) \subseteq supp(L)$.
2. $supp(L^S)$ is nonempty, prefix closed, and controllable.
3. $\forall s \in supp(L^S) : L(s) \leq L^S(s)$.

The following theorem shows that L^S is indeed a p-language.

Theorem 1 Let L be the p-language of a DES, and $S : supp(L) \rightarrow 2^{\Sigma - \Sigma_u}$ be a supervisor. Then

1. $\forall s \in \Sigma^* : \Delta(L^S)(s) = L^S(s) \left[\frac{\Delta(s^{-1}L)(\epsilon)}{\Phi^S(L)(s)} \right]$
2. L^S in Definition 1 is a p-language.

For a plant with p-language L we have defined a supervisor S and the resulting controlled plant p-language L^S. The control objective is to ensure that the controlled plant p-language lies within a certain prespecified range, i.e., given a pair of probabilistic maps $K \preceq D$, the task is to design a supervisor such that $K \preceq L^S \preceq D$. Here D is actually a non-probabilistic map, i.e., $D : \Sigma^* \rightarrow \{0, 1\}$, and specifies a legality constraint. D maps the legal traces to one, and the illegal traces to zero. The control objective is to ensure $L^S \preceq D$, i.e., illegal traces never occur in the controlled plant implying their occurrence probabilities is each zero, whereas no constraint is imposed on the occurrence probabilities of legal traces in the upper bound specification D. The lower bound K on the other hand specifies the level of desirability of legal traces by specifying their minimum acceptable occurrence probabilities. The control objective is to ensure $K \preceq L^S$, i.e., legal traces in the controlled plant occur with at least as much probability as specified by the lower bound constraint K. The existence of S such that $K \preceq L^S \preceq D$ also implies $supp(K) \subseteq supp(L^S) \subseteq supp(D)$, which is the supervisory control problem studied in the non-stochastic setting [7]. Thus the supervisory control problem formulated here generalizes the one studied in the non-stochastic setting.

In the following theorem we give a necessary and sufficient condition for the existence of a supervisor for the supervisory control problem described above.

Theorem 2 Let L be the p-language of a DES, K be a probabilistic map representing the lower bound constraint, and D be a non-probabilistic map representing the upper bound constraint. Then there exists a supervisor $S : supp(L) \rightarrow 2^{\Sigma - \Sigma_u}$ such that $K \preceq L^S \preceq D$ if and only if

$$K \preceq L^{S^{\downarrow}} \preceq D,$$

where $S^{\downarrow}$ is the supervisor that restricts the support of the plant to $inf\overline{PC}(supp(K))$. In this case $S^{\downarrow}$ can be used as a supervisor.

In the following remark we compare the existence condition of Theorem 2 with the corresponding condition in the non-stochastic setting.

Remark 2 Given a plant with p-language L, a probabilistic map lower bound specification K, and

a non-probabilistic map upper bound specification D such that $K \preceq D$, in the non-stochastic setting we are interested in designing a supervisor S such that $supp(K) \subseteq supp(L^S) \subseteq supp(D)$. It can be shown using known results that such a supervisor exists if and only if

$$supp(K) \subseteq supp(L^{S^\downarrow}) = inf\overline{PC}(supp(K)) \subseteq supp(D).$$

Clearly this condition is implied by the condition of Theorem 2. Thus the condition for the existence of a supervisor in the non-stochastic setting is weaker than that in the stochastic setting, as expected.

To see that the condition in the non-stochastic setting is strictly weaker consider the following example with $\Sigma = \{a, b, c\}$ and $\Sigma_u = \emptyset$:
$L(\epsilon) = 1, L(a) = L(b) = L(c) = \frac{1}{3}, L(s) = 0,$ otherwise
$K(a) = \frac{3}{4}, K(b) = \frac{1}{4}, K(s) = 0,$ otherwise
$D(c) = 0, D(s) = 1,$ otherwise

Then $supp(K) = \{a, b\} \subset supp(D) = \{\epsilon, a, b\} \subset supp(L) = \{\epsilon, a, b, c\}$. This implies that c must be disabled initially, and a, b must be enabled initially. So there is only one choice for the supervisor S with $S(\epsilon) = \{c\}$. This gives:

$$L^S(\epsilon) = 1, L^S(a) = L^S(b) = \frac{1}{2}, \& L^S(s) = 0, \text{ otherwise.}$$

So clearly, $supp(K) \subset supp(L^S) = \{\epsilon, a, b\} = supp(D)$. However, since $K(a) = \frac{3}{4} > \frac{1}{2} = L^S(a)$, $K \not\preceq L^S \preceq D$.

4 Verification of Existence Condition

It follows from Theorem 2 that the existence of a supervisor S for a plant with p-language L and specifications $K \preceq D$ satisfying $K \preceq L^S \preceq D$ can be checked by testing whether $K \preceq L^{S^\downarrow} \preceq D$, where $S^\downarrow$ is the supervisor that restricts the support of the plant to $inf\overline{PC}(supp(K))$. We next provide an algorithm for testing this condition. For this we assume that all maps L, K, and D are deterministic regular. We first construct a dnp-automaton G such that $L_G = L^{S^\downarrow}$.

Algorithm 1 Given deterministic regular p-languages L, K, let finite state dp-automata $G^L := (X^L, \Sigma, x^L_{init}, P^L)$, $G^K := (X^K, \Sigma, x^K_{init}, P^K)$ be such that $L_{G^L} = L, L_{G^K} = K$. We construct a dnp-automaton $G := (X, \Sigma, x_{init}, P)$ with finite state such that $L_G = L^{S^\downarrow}$ as follows:

1. Obtain dnp-automata $\overline{G^L}, \overline{G^K}$ from G^L, G^K respectively by replacing each transition probability with non-zero value by that of value one, and deleting transitions with zero probability value.
2. Obtain dnp-automaton G^1 such that the support of its generated language equals $inf\overline{PC}(supp(K))$. This is done in two steps: first by adding a dump state in $\overline{G^K}$ with self-loops on uncontrollable events and also adding a transition from each state of $\overline{G^K}$ to the dump state on each uncontrollable event that is undefined in that state; and next taking the synchronous composition of the resulting dnp-automaton with $\overline{G^L}$.
3. Next obtain the dp-automaton G such that $L_G = L^{S^\downarrow}$. This is done by attaching appropriate probability values to each transition of G^1 as follows. Let x^L, x^K, respectively, denote typical states of $\overline{G^L}$, and $\overline{G^K}$ augmented with the dump state. Then (x^L, x^K) denotes a typical state of G^1. For each state (x^L, x^K) of G^1, let $\Sigma(x^L, x^K) \subseteq \Sigma$ denote the set of events that are defined at state (x^L, x^K) in G^1. Finally, since all automata are deterministic we suppress the destination state in any transition by representing it by a "$*$". For each transition $((x^L, x^K), \sigma, *)$ of G^1 define its probability $P((x^L, x^K), \sigma, *)$ to be

$$\frac{P^L(x^L, \sigma, *)}{\Delta(G^L)(x^L) + \sum_{\hat{\sigma} \in \Sigma(x^L, x^K)} P^L(x^L, \hat{\sigma}, *)},$$

where $P^L(\cdot, \cdot, \cdot)$ gives the transition probability of G^L.

The following theorem proves the correctness of Algorithm 1.

Theorem 3 Let L, K, G^L, G^K, G^1, G be as in Algorithm 1. Then $L_G = L^{S^\downarrow}$, where $S^\downarrow$ is a supervisor that restricts the support of the plant with p-language L to $inf\overline{PC}(supp(K))$.

Next we present an algorithm for testing whether a p-language K is upper bounded by another p-language L, assuming that both K, L are deterministic regular. This algorithm combined with Algorithm 1 can be used to test the condition of Theorem 2. We first present an algorithm to construct a deterministic automaton G such that for each trace $s \in supp(K)$, $L_G(s) = \frac{L(s)}{K(s)}$.

Algorithm 2 Given deterministic regular p-languages L, K, let finite state dp-automata $G^L := (X^L, \Sigma, x^L_{init}, P^L)$, $G^K := (X^K, \Sigma, x^K_{init}, P^K)$ be such that $L_{G^L} = L, L_{G^K} = K$. We construct a deterministic automaton $G = (X, \Sigma, x_{init}, P)$ with finite state such that for each trace $s \in supp(K)$, $L_G(s) = \frac{L(s)}{K(s)}$ as follows:

1. Define $X := X^L \times X^K$

2. Define $x_{init} := (x^L_{init}, x^K_{init})$

3. For each $x^L, \overline{x}^L \in X^L, x^K, \overline{x}^K \in X^K, \sigma \in \Sigma$ define $P((x^L, x^K), \sigma, (\overline{x}^L, \overline{x}^K))$ to be

$$\begin{cases} \frac{P^L(x^L,\sigma,\overline{x}^L)}{P^K(x^K,\sigma,\overline{x}^K)} & \text{if } P^K(x^K,\sigma,\overline{x}^K) \neq 0 \\ \infty & \text{otherwise} \end{cases}$$

Then we have the following straightforward theorem. The first part states the correctness of Algorithm 2, whereas the second part reduces the problem of checking $K \preceq L$ to that of a shortest path computation.

Theorem 4 Let L, K, G^L, G^K, G be as in Algorithm 2. Then

1. For each trace $s \in supp(K)$, $L_G(s) = \frac{L(s)}{K(s)}$.
2. $K \preceq L$ if and only if $\min_{s \in supp(K)} L_G(s) \geq 1$.

The second part of Theorem 4 reduces the problem of verification of $K \preceq L$ to that of computation of the "least probable path" in G. An algorithm for the shortest path computation can be modified to compute this as follows.

Algorithm 3 Relabel the states of G by numbers $1, \ldots, n$, where n is the total number of states in G. The notation $d^k[i,j]$ is used to denote the "least probable path" from state i to state j, visiting intermediate states with label at most k.

1. $k := 0; \forall i, j \leq n : d^0[i,j] := \min_{\sigma \in \Sigma} P(i, \sigma, j)$.
2. $k := k+1; \forall i, j \leq n : d^k[i,j] :=$
$$\begin{cases} \min\{d^{k-1}[i,j], d^{k-1}[i,k] d^{k-1}[k,j]\} & \text{if } d^{k-1}[k,k] \geq 1 \\ 0 & \text{otherwise} \end{cases}$$
3. If $\forall i, j \leq n : d^{k-1}[i,j] = d^k[i,j]$, then stop; else go to step 2.

Remark 3 Using a proof similar to the proof of the correctness of shortest path computation [5], it is easily shown that the above iterative computation terminates in at most n iterations performing $O(n^2)$ computations in each iteration, and upon termination $d^n[i,j]$ equals the "least probable" path from the state i to state j. So if the initial state is relabeled as say i, then $K \preceq L$ if and only if $\min_{j \leq n} d^n[i,j] \geq 1$. The computational complexity of this test is thus $O(n^3)$, where n is the product of the number of states of G^L and G^K.

5 Conclusion

In this paper we have formalized the supervisory control of stochastic qualitative behavior of DESs. It generalizes the supervisory control formalism of the non-stochastic setting in a natural way. The control objective in the stochastic setting is to design a supervisor so that the controlled plant only generates legal traces (specified as a non-probabilistic map), and that the traces it generates occur with certain minimum probabilities (specified as a probabilistic map). We have shown that the computational complexity of the test for the existence of a supervisor (in the case when the languages involved are deterministic regular) is $O(n^3)$.

References

[1] V. K. Garg. An algebraic approach to modeling probabilistic discrete event systems. In *Proceedings of 1992 IEEE Conference on Decision and Control*, pages 2348–2353, Tucson, AZ, December 1992.

[2] V. K. Garg. Probabilistic languages for modeling of deds. In *Proceedings of Conference on Information Sciences and Systems*, pages 198–203, Princeton, NJ, March 1992.

[3] V. K. Garg, R. Kumar, and S. I. Marcus. Probabilistic language formalism for stochastic discrete event systems. *IEEE Transactions on Automatic Control*, 1997. Accepted.

[4] S. Lafortune and E. Chen. On the infimal closed and controllable superlanguage of a given language. *IEEE Transactions on Automatic Control*, 35(4):398–404, 1990.

[5] E. Lawler. *Combinatorial Optimization - Networks and Matroids*. Holt Rinehart and Winston, 1976.

[6] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, 1987.

[7] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of IEEE: Special Issue on Discrete Event Systems*, 77:81–98, 1989.

[8] R. Sengupta. *Optimal control of discrete event systems*. PhD thesis, Department of Electrical Engineering and Computer Science, University of Michigan at Ann Arbor, 1995.