Control of Stochastic Discrete Event Systems: Synthesis [*]

Ratnesh Kumar
Dept. of Elec. Eng., Univ. of KY, &
Applied Research Lab., Penn. State

Vijay K. Garg
Dept. of Elec. & Comp. Eng.
Univ. of Texas at Austin

Abstract

In our earlier papers [7, 6, 5] we introduced the formalism of *probabilistic languages* for modeling the stochastic qualitative behavior of discrete event systems (DESs). We presented a framework for their supervisory control in [11], where control is exercised by dynamically disabling certain controllable events thereby nulling the occurrence probabilities of disabled events, and increasing the occurrence probabilities of enabled events proportionately. 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. In other words, the probabilistic language of the controlled system lies within a prespecified range, where the upper bound is a "non-probabilistic language" representing a legality constraint. In [11] we provided a condition for the existence of a supervisor, and also presented an algorithm to test this condition when the probabilistic languages are regular (so that they admit probabilistic automata representation with finitely many states). In this paper we give a technique to compute on-line a maximally permissive supervisor achieving the specified lower and upper bound constraints.

Keywords: Stochastic discrete event systems, supervisory control, probabilistic languages.

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 [7] 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 set of binary numbers) to describe the stochastic qualitative behavior of a DES. The interpretation being that the 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 [7]: (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.

A probabilistic language can be viewed as a *formal power series* [21], but satisfying the constraints mentioned above. As discussed in [7], this probabilistic language model differs in various ways from other existing models of stochastic behavior of DESs such as Markov chains [1], stochastic Petri nets [15], Rabin's probabilistic automata [18, 17, 4] and their supervisory control as studied by Mortzavian [16], fuzzy set theory [14], etc., and it is better suited for modeling stochastic qualitative behavior of DESs.

In [7] we defined the set of regular language operators for the probabilistic languages, and also introduced the notion of regularity, i.e., finiteness of automata representation, that is preserved under the operations of regular language operators. We also endowed the set of probabilistic languages with a partial order under which it forms a complete partial order [3]: A probabilistic language is bounded above by another one if the occurrence probability of each trace in the first is bounded above by that in the second.

Sengupta [22, 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 [10].

In [11] we introduced and studied the problem of *supervisory control* of stochastic behavior of DESs modeled as probabilistic languages. In this setting, 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 is obtained as conditionals given that certain controllable events are disabled, which increases the occurrence probabilities of the enabled events 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

[*]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.

by this lower bound. Summarizing, upper bound constraint is given as a non-probabilistic language, whereas the lower bound constraint given as a probabilistic map, and the control objective is to ensure that the probabilistic language of the controlled plant lies within the two bounds. 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.

In [11] we obtained a necessary and sufficient condition for the existence of the supervisor for the above control problem: A supervisor exists if and only if the probabilistic language of the system, controlled so that the set of surviving traces is the infimal controllable superlanguage of the support of the lower bound constraint, itself lies within the prescribed bound. Thus to test the existence of a supervisor one needs to (i) compute infimal controllable superlanguage of a non-probabilistic language, which is known [13, 12], and (ii) check whether a given probabilistic language is bounded above by another one, which we show can be effectively checked. The complexity of the later 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.

In this paper we study the problem of finding a *maximally permissive* supervisor for the above control problem, where a supervisor is said to be more permissive than another one if the first one does not disable a trace that is allowed by the second one. We show that unlike in the non-stochastic case no unique maximally permissive supervisor exists. This is because the set of probabilistic languages is not a complete upper semi-lattice [7]. However, since the set of controllable non-probabilistic languages is a complete upper semi-lattice [19], and the set of probabilistic languages is a complete partial order [7], non-unique maximally permissive supervisors do exist. An off-line computation of a maximally permissive supervisor need not be feasible. We present an effective algorithm for on-line computation of a maximally permissive supervisor (refer [9, 2, 8] for other on-line supervisory computation algorithms). The computational complexity of each step is again $O(n^3)$, where n is the product of the number of states in the automata representations of the plant and the lower bound specification.

We also show that a unique *minimally permissive* supervisor does exist. This is because the set of controllable non-probabilistic languages as well as the set of probabilistic languages is a complete lower semi-lattice. We present an effective algorithm for off-line computation of the unique minimally permissive supervisor. The computational complexity of this algorithm is again $O(n^3)$.

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 [7] to model the stochastic qualitative behavior of DESs. A definition of p-languages based on their underlying probability measure space was presented in [7]. 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 [7] 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 $|\pi|$ to denote its length. The state transition function is extended inductively to the set of paths as follows:

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

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 [7] 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 [7].

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 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 $f : X \rightarrow X$ 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 [7]:

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

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 [7] that the set of all p-languages forms a cpo as well as 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 $S : supp(L) \rightarrow 2^{\Sigma - \Sigma_u}$ 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 [19]. 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 $inf\overline{PC}(K)$ [13], and its *supremal prefix closed and controllable sublanguage*, denoted $sup\underline{PC}(K)$ [19], exist and are effectively computable.

3 Existence of Supervisor

In this section we review our earlier work from [11] that extended 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.

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 an appropriate normalization factor. For the notational convenience, given a plant with p-language L and a supervisor $S : supp(L) \to 2^{\Sigma - \Sigma_u}$, we define a probabilistic map $\Phi^S(L) : \Sigma^* \to [0,1]$:

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

$\Phi^S(L)(s)$ computes 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 $S : supp(L) \to 2^{\Sigma - \Sigma_u}$ be a supervisor. Then

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

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

$$L^S(\epsilon) := 1; \quad \forall s \in \Sigma^*, \sigma \in \Sigma : L^S(s\sigma) := L^S(s)(s^{-1}L^S)(\sigma).$$

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 $S : supp(L) \to 2^{\Sigma - \Sigma_u}$ 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 $S : supp(L) \to 2^{\Sigma - \Sigma_u}$ 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.

Remark 1 Since from the first part of Lemma 1, $\Delta(s^{-1}L^S)(\epsilon) = \frac{\Delta(L^S)(s)}{L^S(s)}$, using the result of the first part of Theorem 1 it follows

$$\Delta(s^{-1}L^S)(\epsilon) = \frac{\Delta(s^{-1}L)(\epsilon)}{\Phi^S(L)(s)}.$$

This equates the probability of termination given that trace s has already occurred in the controlled plant to various parameters of the uncontrolled plant.

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., $D : \Sigma^* \to \{0,1\}$, 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 are 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 [20]. 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 $S : supp(L) \to 2^{\Sigma - \Sigma_u}$ 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.

An algorithm for verifying the existence condition of Theorem 2 was presented in [11] when languages K, L, D are all regular (i.e., they each admit a finite automaton representation), the complexity of which was polynomial.

4 Maximally Permissive Supervisor

Theorem 2 provides a condition for the existence of a supervisor S for a plant with p-language L with control specifications $K \preceq D$, that ensures $K \preceq L^S \preceq D$. It also mentions that a supervisor $S^{\downarrow}$ that restricts the support of the plant to $inf\overline{PC}(supp(K))$ can be used as a supervisor whenever the existence condition is satisfied. However, $S^{\downarrow}$ need not be a maximally permissive supervisor. In fact, we show below that it is the minimally permissive supervisor. In this section we present a technique to construct supervisors that are maximally permissive.

Definition 2 Given a plant with p-language L, and supervisors $S_1, S_2 : supp(L) \to 2^{\Sigma - \Sigma_u}$, S_1 is said to be less permissive than S_2 (or equivalently, S_2 is said to be more permissive than S_1), denoted $S_1 \preceq S_2$, if for each $s \in supp(L)$,

$S_1(s) \supseteq S_2(s)$, i.e., S_1 disables more events than S_2 following the execution of any trace s. The infimum and supremum of S_1 and S_2 are defined as follows:

$$\forall s \in supp(L) : \begin{cases} S_1 \sqcap S_2(s) := S_1(s) \cup S_2(s) \\ S_1 \sqcup S_2 := S_1(s) \cap S_2(s) \end{cases}$$

It is clear that the set of all supervisors together with the partial order of Definition 2 forms a complete lattice. Also,

$$\begin{aligned} supp(L^{S_1 \sqcap S_2}) &= supp(L^{S_1}) \cap supp(L^{S_2}) \\ supp(L^{S_1 \sqcup S_2}) &= supp(L^{S_1}) \cup supp(L^{S_2}). \end{aligned}$$

Given a plant with p-language L and specifications $K \preceq D$, define the following class of supervisors:

$$\mathcal{S} := \{S : supp(L) \to 2^{\Sigma - \Sigma_u} \mid K \preceq L^S \preceq D\}. \qquad (1)$$

$\mathcal{S}$ is the class of supervisors which can control the plant to meet the given specifications. The following theorem states a few properties of this class of supervisors.

Theorem 3 Let L be the p-language of a plant with control specifications $K \preceq D$. Consider the class of supervisors $\mathcal{S}$ as defined in Equation (1). Then

1. $(\mathcal{S}, \preceq)$ is a complete lower semi-lattice.
2. If $\mathcal{S} \neq \emptyset$, then $S^{\downarrow}$ is the bottom element of $(\mathcal{S}, \preceq)$, where $S^{\downarrow}$ is the supervisor that restricts the support of the plant to $inf\overline{PC}(supp(K))$.
3. $(\mathcal{S}, \preceq)$ is not a upper semi-lattice.
4. $(\mathcal{S}, \preceq)$ is a complete partial order.

The first part of Theorem 3 shows that a unique minimally permissive supervisor exists, whereas the second part of the theorem shows that the supervisor given by Theorem 2 is actually the unique minimally permissive supervisor. The third part of Theorem 3 shows that no unique maximally permissive supervisor exists, but the fourth part of the theorem shows that non-unique maximally permissive supervisors do exist. So we next present an algorithm for the on-line computation of such a supervisor assuming that all maps L, K, D are deterministic regular. (Note that an algorithm for the off-line computation of a maximally permissive supervisor may not exist since the controlled plant p-language under such a supervisor may not be regular.) For each trace $t \in supp(L)$, we define $S_t^{\downarrow} : supp(t^{-1}L) \to 2^{\Sigma - \Sigma_u}$ to be the supervisor for plant with p-language $(t^{-1}L)$ that restricts its support to $inf\overline{PC}(supp(t^{-1}K))$.

Algorithm 1 Consider a plant with deterministic regular p-language L, a lower bound specification K which is also a deterministic regular p-language, and an upper bound specification D which is a deterministic regular np-language. Assume that $K \preceq L^{S^{\downarrow}} \preceq D$, i.e., a supervisor ensuring the given specifications exists. Obtain a maximal supervisor $S^{max} : supp(L) \to 2^{\Sigma - \Sigma_u}$ as follows.

For each $s \in supp(L)$, define $S^{max}(s) := \Sigma - \Sigma^{max}(s)$, where $\Sigma^{max}(s) \subseteq \Sigma$ is a *maximal* set such that for any $t = s\sigma \in s\Sigma^{max}(s)$:

$$\frac{K(t)}{L^{S^{max}}(t)}(t^{-1}K) \preceq (t^{-1}L)^{S_t^{\downarrow}} \preceq (t^{-1}D), \qquad (2)$$

where

$$\begin{aligned} L^{S^{max}}(t) &:= L^{S^{max}}(s)(s^{-1}L^{S^{max}})(\sigma) \\ &= L^{S^{max}}(s)\frac{(s^{-1}L)(\sigma)}{\Phi^{S^{max}}(L)(s)}. \end{aligned}$$

Algorithm 1 computes the set of events to be enabled by S^{max} following the execution of each trace s as follows. S^{max} enables events in $\Sigma^{max}(s)$ if (i) after the occurrence of any event $\sigma \in \Sigma^{max}(s)$ it is possible to control the plant in future so that the given specifications are satisfied, which is captured by Equation 2, and (ii) $\Sigma^{max}(s)$ is a maximal set of events having such property.

The next theorem states the correctness of Algorithm 1. We first prove a lemma.

Lemma 3 Given a plant with p-language L and specifications $K \preceq D$, let S be a supervisor such that $K \preceq L^S \preceq D$, i.e., $S \in \mathcal{S}$. Then

$$\forall t = s\sigma \in s(\Sigma - S(s)) : \frac{K(t)}{L^S(t)}(t^{-1}K) \preceq (t^{-1}L)^{S_t^{\downarrow}} \preceq (t^{-1}D) \qquad (3)$$

Theorem 4 Let L, K, D, S^{max} be as in Algorithm 1. Then S^{max} is a maximal supervisor such that $K \preceq L^{S^{max}} \preceq D$.

Remark 2 In Algorithm 1 the computation of $S^{max}(s)$ at each trace s requires testing the condition of Equation (2) for all possible set of events $\Sigma^{max}(s)$. Ignoring the dependence of computational complexity on the size of the set Σ, it can be shown (refer [11] for an algorithm to test ordering of deterministic regular p-languages) that the computational complexity at each step of Algorithm 1 is $O(n^3)$, where n is the product of the number of states in the automata representations of the plant and the lower bound specification.

In the following remark we compare the maximally permissive supervisor of Algorithm 1 with the maximally permissive supervisor in the non-stochastic setting.

Remark 3 In the non-stochastic setting we are interested in designing a supervisor S such that for a plant with p-language L, a probabilistic map lower bound specification K, and a non-probabilistic map upper bound specification D, satisfying $K \preceq D$, we obtain $supp(K) \subseteq supp(L^S) \subseteq supp(D)$.

It can be shown that whenever such a supervisor exists, there exists a unique maximally permissive supervisor S^{sup}, called *supremely permissive* supervisor, which can be computed as follows. For each $s \in supp(L)$, define $S^{sup}(s) := \Sigma - \Sigma^{sup}(s)$, where $\Sigma^{sup}(s) \subseteq \Sigma$ is the *supremal* set such that for any $t = s\sigma \in s\Sigma^{sup}(s)$:

$$supp(t^{-1}K) \subseteq supp((t^{-1}L)^{S_t^{\downarrow}}) \subseteq supp(t^{-1}D). \qquad (4)$$

In other words, S^{sup} enables an event following the occurrence of s if it is possible to control the plant in the future so that the specifications of the non-stochastic setting are satisfied. Since $supp[\frac{K(t)}{L^{S^{max}(s)}}(t^{-1}K)] = supp(t^{-1}K)$, it follows from Equation (2) that for any $t = s\sigma \in s\Sigma^{max}(s)$:

$$supp(t^{-1}K) \subseteq supp((t^{-1}L)^{S_t^{\downarrow}}) \subseteq supp(t^{-1}D).$$

This together with the definition of S^{sup} given in Equation (4) implies that $\Sigma^{max}(s) \subseteq \Sigma^{sup}(s)$, i.e., any maximally permissive supervisor in the stochastic setting is less permissive than the maximally permissive supervisor in the non-stochastic setting, as expected.

The following example with $\Sigma = \{a, b, c\}$ and $\Sigma_u = \emptyset$ illustrates that the converse does not hold in general.

$L(\epsilon) = 1, L(a) = L(b) = L(c) = \frac{1}{3}$, and $L(s) = 0$, otherwise
$K(a) = K(b) = \frac{1}{2}$, and $K(s) = 0$, otherwise
$D(s) = 1, \forall s$

Then it is clear that the supremely permissive supervisor in the non-stochastic setting disables no events initially, whereas the maximally permissive supervisor in the stochastic setting disables the event c initially.

5 Conclusion

The supervisory control of stochastic DESs naturally generalizes the supervisory control formalism of the non-stochastic DESs. 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). The computational complexity of the test for the existence of a supervisor, and also that for the on-line computation of a maximally permissive supervisor (in the case when the languages involved are deterministic regular) are both $O(n^3)$.

References

[1] C. G. Cassandras. *Discrete Event Systems: Modeling and Performance Analysis.* Aksen Associates, Boston, MA, 1993.

[2] S. L. Chung, S. Lafortune, and F. Lin. Supervisory control using variable lookahead policies. *Discrete Event Dynamic Systems: Theory and Applications*, 4(3):237–268, July 1994.

[3] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order.* Cambridge University Press, Cambridge, UK, 1990.

[4] E. Doberkat. *Stochastic Automata: Stability, Nondeterminism and Prediction*, volume 113 of *Lecture Notes in Computer Science.* Springer-Verlag, New York, 1981.

[5] 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.

[6] 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.

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

[8] N. B. Hadj-Alouane, S. Lafortune, and F. Lin. Centralized and distributed algorithm for on-line synthesis of maximal control policies under partial observation. *Discrete Event Dynamical Systems: Theory and Applications*, 6(41):379–427, 1996.

[9] M. Heymann and F. Lin. On-line control of partially observed discrete event systems. *Discrete Event Dynamical Systems: Theory and Applications*, 4(3):221–236, July 1994.

[10] P. R. Kumar and P. Varaiya. *Stochastic Systems: Estimation, identification and adaptive control.* Prentice Hall, 1986.

[11] R. Kumar and V. K. Garg. Control of stochastic discrete event systems: Existence. In *Proceedings of 1998 International Workshop on Discrete Event Systems*, Cagliari, Italy, August 1998.

[12] R. Kumar and M. A. Shayman. Formulae relating controllability, observability, and co-observability. *Automatica*, 34(1), 1998. To appear.

[13] 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.

[14] E. T. Lee and L. A. Zadeh. Note on fuzzy languages. *Information Sciences*, pages 421–434, 1969.

[15] M. K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers*, C-31(9):913–917, September 1982.

[16] H. Mortzavian. Controlled stochastic languages. In *Proceedings of 1993 Allerton Conference*, pages 938–947, Urbana, IL, 1993.

[17] A. Paz. *Introduction to Probabilistic Automata.* Academic Press, New York, 1971.

[18] M. O. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.

[19] 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.

[20] 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.

[21] A. Salomaa. Formal languages and power series. In J. v. Leeuwen, editor, *Handbook of theoretical computer science.* MIT Press, Cambridge, MA, 1994.

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