Control of Stochastic Discrete Event Systems Modeled by Probabilistic Languages \(^1\)

Ratnesh Kumar  
Dept. of Electrical Engineering, Univ. of Kentucky  
Lexington, KY 40506

Vijay K. Garg  
Dept. of Elec. and Comp. Eng., Univ. of Texas at Austin  
Austin, TX 78712

\(^1\)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.
Abstract

In earlier papers [7, 6, 5] we introduced the formalism of probabilistic languages for modeling the stochastic qualitative behavior of discrete event systems (DESs). In this paper we study their supervisory control where the 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. This is a special case of “probabilistic supervision” introduced in [15]. 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 pre-specified occurrence probabilities. In other words, the probabilistic language of the controlled system lies within a pre-specified range, where the upper bound is a “non-probabilistic language” representing a legality constraint. 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). Next we give a technique to compute a maximally permissive supervisor on-line.

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

Discrete event systems (DESs) are systems with discrete states that are event-driven, i.e., that change states in response to the occurrence of events at random instants of time. Supervisory control of DESs pioneered by Ramadge-Wonham [22] and subsequently extended by other researchers (see for example [12]) provides a framework for designing controllers for controlling the non-stochastic untimed or qualitative behavior of DESs. The non-stochastic qualitative behavior of DESs is generally modeled by the set of all possible sequences of events, a formal language [10], that can occur in the system. Such non-stochastic behavior of a discrete event system can alternatively 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 non-probabilistic language.

For control purposes the set of events is partitioned into the sets of controllable and uncontrollable events. A supervisor controls the given system, called plant, by dynamically disabling certain controllable events based on its observations of the executed traces. The control objective is to design a supervisor so that the non-probabilistic language of the controlled plant is bounded within a certain range specified as a pair of non-probabilistic languages. The upper bound specifies the set of legal traces, whereas the lower bound gives the minimally adequate set of traces. A supervisor exists if and only if the infimal controllable superlanguage [14] of the lower bound is bounded above by the supremal controllable sublanguage of the upper bound [23]. (A language is said to be controllable if it is closed with respect to the executions of feasible uncontrollable events.)

In [7, 6, 5] we introduced a more general map over the set of all traces that take 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. The first constraint follows from the fact that the execution of the zero length trace is always possible, whereas the second constraint follows from the fact that if the execution of a trace is possible, then the execution of any of its prefix is also possible. We call such maps probabilistic languages and use them for modeling the stochastic qualitative behavior of DESs.

A probabilistic language can be viewed as a formal power series [24], 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 [18], Rabin’s probabilistic automata [21, 20, 4] and their supervisory control as studied by Mortzavian [19], fuzzy set theory [17], 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 [25, 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 resulting problem is one of infinite horizon optimal control of Markov processes [11].

In contrast, in this paper we study the problem of supervisory control of stochastic behavior of DESs modeled as probabilistic languages. As above, 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. A notion of probabilistic supervision, which is more general than the “non-probabilistic supervision” considered here, was first introduced in [15].

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. Summarizing, the upper bound constraint is given as a non-probabilistic language, whereas the lower bound constraint is 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. A special case of the control objective, where the upper bound constraint is the same as the support of the lower bound constraint, was first studied in [15]; however, a more general probabilistic supervisor was used to achieve that control objective.

We obtain a necessary and sufficient condition for the existence of the supervisor for the above control problem as: 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 we need to (i) compute infimal controllable superlanguage
of a non-probabilistic language, which is known \([14, 13]\), 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.

Next we study the problem of finding a \textit{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 \([22]\), and the set of probabilistic languages is a complete partial order \([7]\), non-unique maximally permissive supervisors do exist. 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 \textit{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 \(O(n^3)\).

The rest of the paper is organized as follows. Section 2 gives the notations and preliminaries. Section 3 formulates the supervisory control problem, and gives the existence results, whereas Section 4 presents an algorithm for checking existence. Section 5 shows the existence of a non-unique maximally permissive and a unique minimally permissive supervisor, and gives effective algorithms for their on-line and off-line computation, respectively. Section 6 concludes the work presented and identifies some future research directions.

## 2 Notation and Preliminaries

We use \(\Sigma\) to denote the universe of events over which a given DES evolves. The set \(\Sigma^*\) is the set of all finite length event sequences, called traces, including the zero length trace, denoted \(\epsilon\). A subset of \(\Sigma^*\) is called a language. Given traces \(s\) and \(t\), we use \(s \leq t\) to denote that \(s\) is a \textit{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., \(t = ss^{-1}t\). The notation \(|t|\) denotes the length of trace \(t \in \Sigma^*\).

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 \textit{probabilistic map}—over \(\Sigma^*\), \(L : \Sigma^* \rightarrow [0, 1]\). For a probabilistic map \(L\), its \textit{support}, denoted \(\text{supp}(L) \subseteq \Sigma^*\), is the set of traces such that \(L(s) > 0\). \(L\) is said to be a \textit{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
\[
L(\varepsilon) = 1; \quad \forall s \in \Sigma^*, \sigma \in \Sigma : L(s\sigma) = 1 \Rightarrow L(s) = 1.
\]
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 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]. As is discussed in [7] a p-language \( L \) can alternatively be viewed as a probabilistic map satisfying the following constraints:

\[ P1: \quad L(\varepsilon) = 1 \]
\[ P2: \quad \forall s \in \Sigma^* : \sum_{\sigma \in \Sigma} L(s\sigma) \leq L(s) \]

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. Note although we view \( L \) a probabilistic map over the set of traces, it actually is a probability measure over the set of \( \sigma \)-algebra \( \mathcal{F} \) defined in [7], and we also use it in that manner sometimes.

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 \( s \). It was shown in [7] that \( \sum_s \Delta(L)(s) \leq 1 \) with the equality holding if and only if \( \lim_{k \to \infty} \sum_{|t|=k} L(t) = 0 \). In other words, the probability that a system terminates following the execution of arbitrary traces is bounded above by one (a system is not necessarily guaranteed to terminate), and it equals one if and only if the probability of traces of arbitrary length converges to zero. We say that a p-language \( L \) is a terminating system if
\[
\sum_s \Delta(L)(s) = 1.
\]

Qualitative behavior of DESs can alternatively be represented by automata. An automaton \( G \) over the event set \( \Sigma \) is a quadruple, \( G := (X, \Sigma, x_{\text{init}}, P) \), where \( X \) is the set of states of \( G \), \( x_{\text{init}} \in X \) 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 \((x, \sigma', x)\); 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 a deterministic np-automaton or dnp-automaton (resp., deterministic p-automaton or dp-automaton) if

$$\forall x \in X, \sigma \in \Sigma : |\{x' \in X \text{ s.t. } P(x, \sigma, x') > 0\}| \leq 1.$$  

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 $\pi = x_0\sigma_1x_1 \ldots \sigma_n x_n \in X(\Sigma X)^*$, we use $|\pi| = n$ to denote its length; for each $k \leq |\pi|$, $\pi^k := x_0\sigma_1x_1 \ldots \sigma_k x_k$ 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:

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

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 [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 $L_G = L$. It is known that the class of deterministic regular np-languages is the same as the class of regular np-languages [10], whereas whether or not the class of deterministic regular p-languages is a strict subclass of the class of regular p-languages is an open problem.

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

$$\forall x^1, x^2 \in X^1, x^2, \bar{x}^2 \in X^2, \sigma \in \Sigma : P((x^1, x^2), \sigma, (x^1, \bar{x}^2)) := P^1(x^1, \sigma, x^1)P^2(x^2, \sigma, \bar{x}^2).$$

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, denoted by \( x \cap y \) and \( x \cup y \), respectively. A poset \( (X, \leq) \) is said to be a upper (resp., lower) semi-lattice if the 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 a (complete) upper and lower semi-lattice. The infimum (resp., supremum) of \( X \) whenever defined is called the bottom (resp., top) element, and is denoted \( \bot \) (resp., \( \top \)).

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 \( x_i \leq x_j \) whenever \( i \leq j \). A poset \( (X, \leq) \) 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 \( \Sigma^* \) 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. The bottom element for the ordering, called the nil p-language, denoted \( \mathcal{I} : \Sigma^* \rightarrow [0, 1] \), is given by

\[
I(\epsilon) = 1; \quad \forall s \neq \epsilon : I(s) = 0.
\]

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 \( S : \text{supp}(L) \rightarrow 2^{\Sigma - \Sigma_u} \) that, following the occurrence of a trace \( s \in \text{supp}(L) \), 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:

\[
L^S(\epsilon) := 1; \quad \forall s \in \Sigma^*, \sigma \in \Sigma : [L^S(s\sigma) := 1] \Leftrightarrow [L^S(s) = 1, L(s\sigma) = 1, \sigma \notin S(s)].
\]

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 \( \text{pr}(K)\Sigma_u \cap \text{supp}(L) \subseteq \text{pr}(K) \). It is known that there exists a supervisor \( S \) for a plant with np-language \( L \) such that \( \text{supp}(L^S) = K \) if and only if \( K \) is nonempty, prefix closed, and controllable [22]. 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 \( \text{infPC}(K) \) [14], and its supremal prefix closed and controllable sublanguage, denoted \( \text{supPC}(K) \) [22], exist and are effectively computable.
3 Existence of Supervisor for Stochastic DESs

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 \( (sL) \) to denote the probability of trace \( t \) given that trace \( s \) has already occurred:

\[
\forall s, t \in \Sigma^* : (s^{-1}L)(t) := L(st|s) = \begin{cases} 
\frac{L(st \land s)}{L(s)} = \frac{L(st)}{L(s)} & \text{if } L(s) \neq 0 \\
I(t) & \text{otherwise,}
\end{cases}
\]

where we have used 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

\[
L(st) = L(st \land s) = L(st|s)L(s) = (s^{-1}L)(t)L(s).
\]

From now on we will assume that \( L(s) \neq 0 \) whenever we refer to \( s^{-1}L \). 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.

**Proof:** We begin by proving the first part. Pick \( t \in \Sigma^* \). Then

\[
\Delta(s^{-1}L)(t) = (s^{-1}L)(t) - \sum_{\sigma}(s^{-1}L)(t\sigma)
\]

\[
= \frac{L(st)}{L(s)} - \sum_{\sigma} \frac{L(st\sigma)}{L(s)}
\]

\[
= \frac{L(st) - \sum_{\sigma} L(st\sigma)}{L(s)}
\]

\[
= \frac{\Delta(L)(st)}{L(s)},
\]

as desired.

To prove the second part we need to establish P1 and P2 for \( s^{-1}L \). By definition, \( s^{-1}L(\epsilon) = \frac{L(s)}{L(s)} = 1 \), i.e., P1 holds. To show P2, it suffices to show that \( \Delta(s^{-1}L) \geq 0 \). This follows from the first part since for each \( t \in \Sigma^* \), we have

\[
\Delta(s^{-1}L)(t) = \frac{\Delta(L)(st)}{L(s)} \geq 0,
\]

\[
\Delta(s^{-1}L)(t) = \frac{\Delta(L)(st)}{L(s)} \geq 0,
\]
Remark 1. Note that $\Delta(s^{-1}L)(t) = \frac{\Delta(L(s))_{L(s)}}{L(s)}$ is the conditional probability of termination following the execution of trace $t$ given that trace $s$ has already occurred.

Also, 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 $tr(\pi_s) = s$, and $L(s) = P(\pi_s)$. So for a trace $s$ and an event $\sigma$ we have:

$$L(s\sigma) = P(\pi_{s\sigma}) = P(\pi_s)P(x_{|\pi_s|}, \sigma, x_{|\pi_{s\sigma}|}) = L(s)P(x_{|\pi_s|}, \sigma, x_{|\pi_{s\sigma}|}).$$

This implies

$$P(x_{|\pi_s|}, \sigma, x_{|\pi_{s\sigma}|}) = \frac{L(s\sigma)}{L(s)} = (s^{-1}L)(\sigma)$$

Similarly it can be shown that

$$\Delta(G)(x_{|\pi_s|}) = \Delta(s^{-1}L)(\epsilon).$$

As described above, a supervisor $S : supp(L) \to 2^{\Sigma^*}$ 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 $\sigma$ 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 notational convenience, given a plant with p-language $L$ and a supervisor $S : supp(L) \to 2^{\Sigma^*}$, 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. The following lemma can be viewed as a special case of [15, Equation (1)] when [15, Section IV] is taken into consideration.

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

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

Proof: Pick $s \in \Sigma^*$ and $\sigma \in \Sigma$. If $\sigma \in S(s)$, then it is disabled following the occurrence of $s$. So, clearly, $(s^{-1}L^S)(\sigma) = 0$. Otherwise, when $\sigma \in \Sigma - S(s)$, we have:

$$(s^{-1}L^S)(\sigma)$$

{by definition of supervisor $S$}

$$= (s^{-1}L)(\sigma \mid \hat{\sigma} \in S(s) \text{ does not occur following } s)$$
rewriting \( \hat{s} \in S(s) \) does not occur following \( s \)}
\[
REW\ 2 \: (s^{-1}L)(\sigma \mid \text{termination or } \hat{s} \in \Sigma - S(s) \text{ occurs following } s) \\
\{\text{definition of conditional: } \text{Prob}[A|B] = \frac{\text{Prob}[A \cap B]}{\text{Prob}[B]} \}
\]
\[
= (s^{-1}L)(\sigma \wedge [\text{termination or } \hat{s} \in \Sigma - S(s) \text{ occurs following } s])
\]
\[
\{\text{since } \sigma \in \Sigma - S(s)\}
\]
\[
= \frac{(s^{-1}L)(\sigma)}{(s^{-1}L)(\text{termination or } \hat{s} \in \Sigma - S(s) \text{ occurs following } s)}
\]
\[
\{\text{rewriting } (s^{-1}L)(\sigma) \text{ occurs following } s\}
\]
\[
= \frac{(s^{-1}L)(\sigma)}{\Phi^S(L)(s)}
\]

This completes the proof. 

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

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

**Example 1** Consider a plant with \( \Sigma = \{a,b\} \) and \( \Sigma_u = \{b\} \) shown in Figure 1(a). The plant p-language \( L \) is deterministic regular represented by a two state dp-automaton. The transitions are labeled by events along with their occurrence probabilities shown in parenthesis. Consider a supervisor \( S \) that disables \( a \) after occurrences of \( a, bb, abb \), i.e.,

\[
S(a) = S(bb) = S(abb) = \{a\}, \quad \text{and } S(s) = \emptyset, \text{ otherwise.}
\]

The resulting controlled plant is depicted in Figure 1(b). Note from Lemma 2, the transition probability of \( b \) given that \( a \) has already occurred is:

\[
(a^{-1}L^S)(b) = \frac{(a^{-1}L)(b)}{\Phi^S(L)(a)} = \frac{q}{(1 - p - q) + q} = \frac{q}{1 - p}.
\]
Similarly, the transition probability of $b$ given that $bb$ has already occurred is:

$$(bb)^{-1}L^S(b) = \frac{((bb)^{-1}L)(b)}{\Phi^S(L)(bb)} = \frac{e}{(1 - r - e) + e} = \frac{e}{1 - r}.$$ 

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. $\forall s \in supp(L^S) : L(s) \leq L^S(s)$.

**Proof:** The first part is obvious since by definition $S$ restricts the support of the plant behavior by dynamically disabling events. The second part follows from the fact that $supp(L^S)$ is the language of the controlled plant, which must be nonempty, prefix closed, and controllable [22]. Finally, the third part follows from induction on the length of traces, and the fact that for each $s \in supp(L^S)$ and $\sigma \in \Sigma - S(s)$, $(s^{-1}L)(\sigma) \leq (s^{-1}L^S)(\sigma)$, where this final inequality follows from Lemma 2 and the fact that

$$\Phi^S(L)(s) \leq 1.$$ 

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.

**Proof:** We begin with the proof for the first part. Pick $s \in \Sigma^*$. Then

$$\Delta(L^S)(s) \{\text{by definition of } \Delta\} = L^S(s) - \sum_{\sigma \in \Sigma} L^S(s\sigma) \{\text{by definition of } L^S\} = L^S(s) - \sum_{\sigma \in \Sigma} L^S(s)(s^{-1}L^S)(\sigma) \{\text{rewriting}\}$$
\[ L^S(s)[1 - \sum_{\sigma \in \Sigma} (s^{-1}L^S)(\sigma)] = L^S(s)[1 - \sum_{\sigma \in \Sigma - S(s)} (s^{-1}L^S)(\sigma)] \]

\{from Lemma 2, \((s^{-1}L^S)(\sigma) = 0, \forall \sigma \in S(s)\}\}

\[ = L^S(s)[1 - \sum_{\sigma \in \Sigma - S(s)} (s^{-1}L^S)(\sigma)] \]

\{applying Lemma 2\}

\[ = L^S(s) \left[ 1 - \sum_{\sigma \in \Sigma - S(s)} \frac{(s^{-1}L)(\sigma)}{\Phi^S(L)(s)} \right] \]

\{using definition of \(\Phi^S(L)(s)\) and simplifying\}

\[ = L^S(s) \left[ \frac{\Delta(s^{-1}L)(\epsilon)}{\Phi^S(L)(s)} \right] \]

This proves the first assertion.

To prove the second part, it suffices to show that \(L^S\) satisfies P1 and P2. P1 holds from definition since \(L^S(\epsilon) := 1\). To show P2, it suffices to show that \(\Delta(L^S) \geq 0\), which follows from the first part. \(\blacksquare\)

**Remark 2** Since from the first part of Lemma 1, \(\Delta(s^{-1}L^S)(\epsilon) = \frac{\Delta(L^S(s))}{E^S(s)}\), 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 \(L^S\). The control objective is to ensure that the controlled plant p-language lies within a certain pre-specified 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 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 \(\text{supp}(K) \subseteq \text{supp}(L^S) \subseteq \text{supp}(D)\), which is the supervisory control problem studied in the non-stochastic setting [23]. 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 \( S : \text{supp}(L) \rightarrow 2^{\Sigma^-} \) such that \( K \preceq L^S \preceq D \) if and only if

\[
K \preceq L^{S^\dagger} \preceq D,
\]

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

Proof: We prove the sufficiency first. Suppose the given condition is satisfied. Select the supervisor \( S = S^\dagger \). Then from the hypothesis it is a desired supervisor.

Next to see the necessity, suppose there exists a supervisor \( S \) such that \( K \preceq L^S \preceq D \). We first show that \( K \preceq L^{S^\dagger} \). From hypothesis that \( K \preceq L^S \), we have \( \text{supp}(K) \subseteq \text{supp}(L^S) \), i.e., \( \text{supp}(L^S) \) is a superlanguage of \( \text{supp}(K) \). By the second part of Corollary 1, \( \text{supp}(L^S) \) is prefix closed and controllable. So it must also be a superlanguage of \( \inf \overline{PC}(\text{supp}(K)) \), which is the infimal such language, i.e.,

\[
\inf \overline{PC}(\text{supp}(K)) = \text{supp}(L^{S^\dagger}) \subseteq \text{supp}(L^S).
\]  

We can view \( S^\dagger \) to be a supervisor for the “plant” with p-language \( L^S \). So from the third part of Corollary 1

\[
\forall s \in \text{supp}(L^{S^\dagger}) : L^S(s) \leq L^{S^\dagger}(s).
\]

From hypothesis \( K \preceq L^S \), so we have from Equation (4)

\[
\forall s \in \text{supp}(L^{S^\dagger}) : K(s) \leq L^S(s) \leq L^{S^\dagger}(s).
\]

From Equation (3) we have \( \text{supp}(K) \subseteq \inf \overline{PC}(\text{supp}(K)) = \text{supp}(L^{S^\dagger}) \). This together with Equation (5) implies that

\[
\forall s \in \text{supp}(K) : K(s) \leq L^{S^\dagger}(s).
\]

On the other hand, we have

\[
\forall s \notin \text{supp}(K) : 0 = K(s) \leq L^{S^\dagger}(s).
\]

Combining Equations (6) and (7) we obtained the first part of necessity condition: \( K \preceq L^{S^\dagger} \).

It remains to show that \( L^{S^\dagger} \preceq D \). Since \( D \) is a non-probabilistic map this is equivalent to showing \( \text{supp}(L^{S^\dagger}) \subseteq \text{supp}(D) \). From hypothesis we have \( L^S \preceq D \), which is equivalent to \( \text{supp}(L^S) \subseteq \text{supp}(D) \). This together with Equation (3) yields the desired result that \( \text{supp}(L^{S^\dagger}) \subseteq \text{supp}(L^S) \subseteq \text{supp}(D) \).

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

Remark 3 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 \( \text{supp}(K) \subseteq \text{supp}(L) \)

\[
K \preceq L \preceq D
\]
It can be shown using known results that such a supervisor exists if and only if

\[ \text{supp}(K) \subseteq \text{supp}(L^S) = \inf PC(\text{supp}(K)) \subseteq \text{supp}(D). \]

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 \( \Sigma_u = \emptyset \):

- \( L(\epsilon) = 1, L(a) = L(b) = L(c) = \frac{1}{3} \), and \( L(s) = 0 \), otherwise
- \( K(a) = \frac{3}{4}, K(b) = \frac{1}{4}, \) and \( K(s) = 0 \), otherwise
- \( D(c) = 0, \) and \( D(d) = 1 \), otherwise

Then

\[ \text{supp}(K) = \{a, b\} \subseteq \text{supp}(D) = \{\epsilon, a, b\} \subseteq \text{supp}(L) = \{\epsilon, a, b, c\}. \]

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}, \) and \( L^S(s) = 0, \) otherwise.\]

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

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 \( K \preceq L^S \preceq D \), where \( S^1 \) is the supervisor that restricts the support of the plant to \( \inf PC(\text{supp}(K)) \). 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 \( L_G = L^{S^1} \). 

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

1. Obtain dnp-automata \( G^L, G^K \) 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 $\inf PC(supp(K))$, i.e.,

$$supp(L_{G^1}) = supp(L^S_i) = \inf PC(supp(K)) = pr(supp(K))\Sigma_u \cap supp(L).$$

This is done in two steps: first by adding a dump state in $G^K$ with self-loops on uncontrollable events and also adding a transition from each state of $G^K$ 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 $G^L$.

3. Next obtain the dp-automaton $G$ such that $L_G = L^{S_i}$. 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 $G^L$, and $G^K$ 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 to be

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

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

The following example illustrates the steps of Algorithm 1.

**Example 2** Consider the plant of Example 1 with p-language $L$ generated by the dp-automaton depicted in Figure 1(a). Suppose the lower bound specification $K$ is the deterministic regular p-language generated by the dp-automaton depicted in Figure 2(a). Then

![Figure 2: Lower bound specification $K$, and $L^{S_i}$](image)

$G^L$ and $G^K$ are the automata depicted in Figure 1(a) and Figure 2(a), respectively. Using Algorithm 1 we obtain the automaton $G$ with $L_G = L^{S_i}$.
1. The dnp-automata $G_L$ and $G_K$ are the automata $G^L$ and $G^K$, respectively, with the probability labels removed from the transitions.

2. The dnp-automaton $G^1$ that generates $\inf PC(supp(K))$ is the automaton shown in Figure 2(b), but without the probability labels on the transitions.

3. The dp-automaton $G$ with p-language $L^{S^1}$ is shown in Figure 2(b).

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 $L_G = L^{S^1}$, where $S^1$ is a supervisor that restricts the support of the plant with p-language $L$ to $\inf PC(supp(K))$.

**Proof:** First note that from construction

$$supp(L_G) = supp(L) = pr(supp(K)) \Sigma_u \cap supp(L) = \inf PC(supp(K)) = supp(L^{S^1}). \quad (9)$$

This also implies $supp(L_G) \subseteq supp(L) = supp(L_G^1)$.

In light of Equation (9), we only need to show that

$$\forall s \in supp(L_G) : L_G(s) = L^{S^1}(s).$$

This we prove by induction on length of $s$. Since both $L_G$ and $L^{S^1}$ are p-languages, $L_G(\epsilon) = L^{S^1} = 1$, which establishes the base step. For the induction step, set $s = t \sigma \in supp(L_G)$. Then $L_G(s) = L_G(t)(t^{-1}L_G)(\sigma)$ and $L^{S^1}(s) = L^{S^1}(t)(t^{-1}L^{S^1})(\sigma)$. From induction hypothesis, $L_G(t) = L^{S^1}(t)$. So it suffices to show that

$$(t^{-1}L_G)(\sigma) = (t^{-1}L^{S^1})(\sigma).$$

Since $\sigma \in \Sigma - S^1(t)$, from Lemma 2 we have:

$$(t^{-1}L^{S^1})(\sigma) = \frac{(t^{-1}L)(\sigma)}{\Delta(t^{-1}L)(\epsilon) + \sum_{\hat{\sigma} \in \Sigma - S^1(t)}(t^{-1}L)(\hat{\sigma})} \quad (10)$$

Let $x^L$ be the state reached by execution of $t$ in $G^L$. Then

$$\Sigma - S^1(t) = \Sigma(x^L, x^K); \quad \Delta(t^{-1}L) = \Delta(G^L)(x^L); \quad \forall \sigma \in \Sigma : (t^{-1}L)(\sigma) = P^L(x^L, \sigma,*), \quad (11)$$

where the last two equalities follow from Equations (1) and (2). So combining Equations (10) and (11) we get:

$$(t^{-1}L^{S^1})(\sigma) = \frac{P^L(x^L, \sigma,*)}{\Delta(G^L)(x^L) + \sum_{\hat{\sigma} \in \Sigma(x^L,x^K)}P^L(x^L, \hat{\sigma},*)}.$$  

This together with Equation (8) of construction yields

$$(t^{-1}L^{S^1})(\sigma) = P((x^L, x^K), \sigma,*).$$
This gives the desired result since \((x^L, x^K)\) is the state reached by execution of \(t\) in \(G\) implying from Equation (1) that \(P((x^L, x^K), \sigma, *) = (t^{-1}L_G)(\sigma)\).

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 \(s \in \text{supp}(K), \quad L_G(s) = \frac{L(s)}{K(s)}\).

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

1. Define \(X := X^L \times X^K\)
2. Define \(x_{\text{init}} := (x^L_{\text{init}}, x^K_{\text{init}})\)
3. For each \(x^L, \pi^L \in X^L, x^K, \pi^K \in X^K, \sigma \in \Sigma\) define
   \[
   P((x^L, x^K), \sigma, (\pi^L, \pi^K)) := \begin{cases} 
   \frac{P^L(x^L, \sigma, \pi^L)}{P^K(x^K, \sigma, \pi^K)} & \text{if } P^K(x^K, \sigma, \pi^K) \neq 0 \\
   \infty & \text{otherwise}
   \end{cases}
   \]

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 \(s \in \text{supp}(K), \quad L_G(s) = \frac{L(s)}{K(s)}\)
2. \(K \preceq L\) if and only if \(\min_{s \in \text{supp}(K)} L_G(s) \geq 1\).

**Proof:** The first part is straightforward by induction on the length of traces in \(\text{supp}(K)\) and using the fact that both \(G^L\) and \(G^K\) are deterministic.

The second part follows from the fact that \(K \preceq L\) if and only if for each \(s \in \text{supp}(K), \quad K(s) \leq L(s)\), or equivalently, \(\frac{L(s)}{K(s)} \geq 1\). So the result follows from the first part.

**Example 3** Consider the plant of Example 1 with p-language \(L\) generated by the dp-automaton of Figure 1(a), and the lower bound specification \(K\) of Example 2 generated by the dp-automaton of Figure 2(a). Suppose the upper bound specification is \(D = 1\), i.e., all traces are legal. Then from Theorem 2, a supervisor exists if and only if \(K \preceq L^{S1}\). Using Algorithm 2 we obtain the automaton \(G\) with \(L_G(s) = \frac{L^{S1}(s)}{K(s)}\) for each trace \(s\). \(G\) is depicted in Figure 3. Since all transitions of \(G\) are labeled by numbers at least one, it follows that \(K \preceq L^{S1}\).
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. **Initiation step:**
   
   
   $k := 0; \quad \forall i, j \leq n : d^0[i, j] := \min_{\sigma \in \Sigma} P(i, \sigma, j)$. 

2. **Iteration step:**
   
   
   $k := k + 1; \quad \forall i, j \leq n : d^k[i, j] := \begin{cases} \min\{d^{k-1}[i, j], d^{k-1}[i, k]d^{k-1}[k, j]\} & \text{if } d^{k-1}[k, k] \geq 1 \\ 0 & \text{otherwise} \end{cases}$

3. **Termination step:**

   If $\forall i, j \leq n : d^{k-1}[i, j] = d^k[i, j]$, then stop; else go to step 2.

   Note the iteration step is obtained using the following standard observation: The set of paths from $i$ to $j$ visiting states with label at most $k$ is the union of the set of such paths visiting states with label at most $k - 1$, and the set of such paths visiting state $k$. So the least probable path may be obtained by taking the minimum over the least probable paths of the two sets. Now consider those paths that visit state $k$. Then a segment of each such path forms a cycle at state $k$, and $d^{k-1}[k, k]$ is the least probable cyclic path at $k$ visiting states with label at most $k - 1$. Hence if $d^{k-1}[k, k] < 1$, by executing the cycle an arbitrary number of times $d^k[i, j]$ can be made arbitrarily close to zero. On the other hand if $d^{k-1}[k, k] \geq 1$, then the cycle must never be executed in computing $d^k[i, j]$. 

Figure 3: Automaton $G$ with $L_G(s) = \frac{L_{S^1}(s)}{K(s)}$
Remark 4 Using a proof similar to the proof of the correctness of shortest path computation [16], 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 Maximally Permissive Supervisor

In Theorem 2 we obtained a condition for the existence of a supervisor \( S \) for a plant with p-language \( L \) with control specifications \( K \preceq D \), that ensures \( K \preceq L^S \preceq D \). We also mentioned that a supervisor \( S^\dagger \) that restricts the support of the plant to \( \inf PC(supp(K)) \) can be used as a supervisor whenever the existence condition is satisfied. However, \( S^\dagger \) 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 \( S_1, S_2 : supp(L) \to 2^{\Sigma_u} \), \( S_1 \) 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:

\[
\forall s \in supp(L) : S_1 \cap S_2(s) := S_1(s) \cup S_2(s); \quad S_1 \cup S_2(s) := S_1(s) \cap S_2(s).
\]

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

\[
supp(L^{S_1 \cap S_2}) = supp(L^{S_1}) \cap supp(L^{S_2}); \quad supp(L^{S_1 \cup S_2}) = supp(L^{S_1}) \cup supp(L^{S_2}).
\]

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

\[
S := \{ S : supp(L) \to 2^{\Sigma_u} \mid K \preceq L^S \preceq D \}. \tag{12}
\]

\( S \) 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 5 Let \( L \) be the p-language of a plant with control specifications \( K \preceq D \). Consider the class of supervisors \( S \) as defined in Equation (12). Then

1. \((S, \preceq)\) is a complete lower semi-lattice.

2. If \( S \neq \emptyset \), then \( S^\dagger \) is the bottom element of \((S, \preceq)\), where \( S^\dagger \) is the supervisor that restricts the support of the plant to \( \inf PC(supp(K)) \).
3. \((S, \leq)\) is not a upper semi-lattice.

4. \((S, \leq)\) is a complete partial order.

**Proof:** 1. To see the first part, let \(\Lambda\) be an index set such that for each \(\lambda \in \Lambda\), \(S_\lambda \in S\). Then
\[
\forall \lambda \in \Lambda : K \preceq L^{S_\lambda} \preceq D. \tag{13}
\]
We need to show that \(K \preceq L^{\bigcap \Lambda S_\lambda} \preceq D\).
First note from Equation (13):
\[
\forall \lambda \in \Lambda : supp(K) \subseteq supp(L^{S_\lambda}) \subseteq supp(D),
\]
which implies
\[
supp(K) \subseteq \bigcap_\lambda supp(L^{S_\lambda}) = supp(L^{\bigcap \Lambda S_\lambda}) \subseteq supp(D). \tag{14}
\]
Since \(D\) is a np-language, this gives us one of the desired inequalities:
\[
L^{\bigcap \Lambda S_\lambda} \preceq D.
\]
So it suffices to show that
\[
\forall s \in supp(K) : K(s) \leq L^{\bigcap \Lambda S_\lambda}(s). \tag{15}
\]
From Equation (13),
\[
\forall \lambda \in \Lambda, s \in supp(K) : K(s) \leq L^{S_\lambda}(s). \tag{16}
\]
Since for each \(\lambda \in \Lambda\), \(\bigcap \Lambda S_\lambda\) is more restrictive than \(S_\lambda\), we can view \(\bigcap \Lambda S_\lambda\) to be a supervisor for the “plant” with p-language \(L^{S_\lambda}\). So from the third part of Corollary 1,
\[
\forall \lambda \in \Lambda, s \in supp(L^{\bigcap \Lambda S_\lambda}) : L^{S_\lambda}(s) \leq L^{\bigcap \Lambda S_\lambda}(s). \tag{17}
\]
Since, from Equation (14), \(supp(K) \subseteq supp(L^{\bigcap \Lambda S_\lambda})\), Equation (17) gives:
\[
\forall \lambda \in \Lambda, s \in supp(K) : L^{S_\lambda}(s) \leq L^{\bigcap \Lambda S_\lambda}(s).
\]
This together with Equation (16) gives the desired inequality of Equation (15).

2. For the second part we need to show that whenever \(S \neq \emptyset\), \(S^1 = \bigcap_{\lambda \in \Lambda} S_\lambda\), where the indexing set \(\Lambda\) is as introduced in the proof of the first part. From Theorem 2, \(S \neq \emptyset\) if and only if \(S^1 \in S\). So from hypothesis \(S^1 \in S\). Since from the first part, \(\bigcap \Lambda S_\lambda\) is the bottom element of \((S, \leq)\), it follows that \(\bigcap \Lambda S_\lambda \preceq S^1\).

Thus it suffices to show that \(S^1 \preceq \bigcap \Lambda S_\lambda\). Since \(supp(L^{S^1}) = \inf \overline{PC}(supp(K))\), this is equivalent to showing that
\[
inf \overline{PC}(supp(K)) \subseteq supp(L^{\bigcap \Lambda S_\lambda}). \tag{18}
\]
From the first part we have \(K \preceq L^{\bigcap \Lambda S_\lambda} \preceq D\). This implies \(supp(L^{\bigcap \Lambda S_\lambda})\) is a prefix closed and controllable superlanguage of \(supp(K)\), which in turn implies that the desired containment of Equation (18) holds, and completes the proof.

3. In order to see that \((S, \leq)\) is not a upper semi-lattice consider the following example with \(\Sigma = \{a, b, c\}\) and \(\Sigma_u = \emptyset\):
Let supervisor $S_1$ be such that $S_1(\epsilon) = \{b\}$, i.e., it disables $b$ initially. Then
\[ L^{S_1}(\epsilon) = 1, L^{S_1}(a) = L^{S_1}(c) = \frac{1}{2}, \text{ and } L^{S_1}(s) = 0, \text{ otherwise.} \]

This implies $K \preceq L^{S_1} \preceq D$. Let supervisor $S_2$ be such that $S_2(\epsilon) = \{c\}$, i.e., it disables $c$ initially. Then
\[ L^{S_2}(\epsilon) = 1, L^{S_2}(a) = L^{S_2}(b) = \frac{1}{2}, \text{ and } L^{S_2}(s) = 0, \text{ otherwise.} \]

This also implies $K \preceq L^{S_2} \preceq D$. Thus we have $S_1, S_2 \in \mathcal{S}$. Now consider the supervisor $S_1 \sqcup S_2$. Then
\[ S_1 \sqcup S_2(\epsilon) = S_1(\epsilon) \cap S_2(\epsilon) = \{b\} \cap \{c\} = \emptyset, \]

i.e., this supervisor does not disable any event. So $L^{S_1 \sqcup S_2} = L$. Since $K(a) = \frac{1}{2} > \frac{2}{3} = L(a)$, $K \not\preceq L = L^{S_1 \sqcup S_2}$, which shows that $S_1 \sqcup S_2 \not\in \mathcal{S}$.

4. We proved in the second part that $\mathcal{S}$ possesses the bottom element namely, $S_1$. So we only need to prove that given a chain $\{S_i \mid \forall i \geq 0 : S_i \in \mathcal{S}\}$, $\sqcup_i S_i \in \mathcal{S}$. By hypothesis
\[ \forall i \geq 0 : K \preceq L^{S_i} \preceq D, \]

which implies
\[ \forall i \geq 0 : \text{supp}(K) \subseteq \text{supp}(L^{S_i}) \subseteq \text{supp}(D), \]

which in turn implies
\[ \text{supp}(K) \subseteq \sqcup_i \text{supp}(L^{S_i}) = \text{supp}(L^{\sqcup_i S_i}) \subseteq \text{supp}(D). \]

Since $D$ is a np-language, this gives us one of the desired inequalities: $L^{\sqcup_i S_i} \preceq D$.

It remains to be shown that $K \preceq L^{\sqcup_i S_i}$. From hypothesis, for each $i \geq 0$, $K \preceq L^{S_i} \preceq D$. Pick a trace $s \in \Sigma^*$. We will show that $L^{\sqcup_i S_i}(s) := \lim_{i \to \infty} L^{S_i}(s)$ exists and satisfies $K(s) \preceq L^{\sqcup_i S_i}(s)$. First consider the case when there exists an integer $i_s \geq 0$ such that $L^{S_{i_s}}(s) > 0$ (such an integer exists for only $s \in \text{supp}(D)$). Then since $\{S_i\}_{i \geq 0}$ is a chain with monotonically increasing permissiveness, it follows from the part 3 of Corollary 1 that $\{L^{S_i}(s)\}_{i \geq i_s}$ is a monotonically decreasing sequence. By hypothesis, the numbers in this sequence are bounded below by $K(s)$, and hence their limit $L^{\sqcup_i S_i}(s)$ exists and satisfies $K(s) \preceq L^{\sqcup_i S_i}(s)$. Note that $L^{\sqcup_i S_i}(s)$ is also the limit of the sequence $\{L^{S_i}(s)\}_{i \geq 0}$ obtained by adding a finite number of numbers to the sequence $\{L^{S_i}(s)\}_{i \geq i_s}$. On the other hand, if there exists no $i_s \geq 0$ such that $L^{S_{i_s}}(s) > 0$ (note that this can only hold for $s \not\in \text{supp}(K)$, since for $s \in \text{supp}(K)$ we have $i_s = 0$), then $L^{\sqcup_i S_i}(s) = 0$. Thus $L^{\sqcup_i S_i}(s)$ exists for each $s \in \Sigma^*$ and satisfies $K(s) \preceq L^{\sqcup_i S_i}(s)$. 

The first part of Theorem 5 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 5 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. For each trace $t \in \text{supp}(L)$, we define $S^t_\# : \text{supp}(t^{-1}L) \to 2^{\Sigma - \Sigma_u}$ to be the supervisor for plant with p-language $(t^{-1}L)$ that restricts its support to $\inf \mathcal{F}(\text{supp}(t^{-1}K))$.

**Algorithm 4** 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 $K \preceq L \preceq D$, i.e., a supervisor ensuring the given specifications exists. Obtain a maximal supervisor $S^\text{max} : \text{supp}(L) \to 2^{\Sigma - \Sigma_u}$ as follows.

For each $s \in \text{supp}(L)$, define $S^\text{max}(s) := \Sigma - \Sigma^\text{max}(s)$, where $\Sigma^\text{max}(s) \subseteq \text{supp}(s^{-1}D) \cap \Sigma$ is a maximal set such that

$$\forall t = s\sigma \in s\Sigma^\text{max}(s) : \frac{K(t)}{L^\text{max}(t)}(t^{-1}K) \preceq (t^{-1}L)^s_1 \preceq (t^{-1}D),$$

where

$$L^\text{max}(t) := L^\text{max}(s)(s^{-1}L^\text{max})^{(\sigma)} = L^\text{max}(s)(s^{-1}L)(\sigma)_{\Phi^\text{max}(L)(s)}.$$

Algorithm 4 computes $\Sigma^\text{max}(s)$, the set of events to be enabled by $S^\text{max}$ following the execution of each trace $s$ as follows. $S^\text{max}$ enables events in $\Sigma^\text{max}(s)$ if (i) after the occurrence of any event $\sigma \in \Sigma^\text{max}(s)$ it is possible to control the plant in future so that the given specifications are satisfied, which is captured by Equation (19), and (ii) $\Sigma^\text{max}(s)$ is a maximal set of events satisfying the upper bound constraint (recall $\Sigma^\text{max}(s) \subseteq \text{supp}(s^{-1}(D)) \cap \Sigma$) having such a property.

It should be noted that $\Sigma^\text{max}(s)$ is defined as a “fixed point”, and its definition should not be taken to be a circular one. To determine $\Sigma^\text{max}(s)$, one guesses an initial value for this event set and verifies whether Equation (19) holds for it. If it does hold (resp., does not hold), then initial guess is replaced by a larger (resp., smaller) event set, and the verification step is repeated till no larger (resp., smaller) event set can be found.

Note that in Equation (19) the scaling factor $\frac{K(t)}{L^\text{max}(t)}$ has been used to take into account the effect of the probability of the trace $t$ when determining control beyond $t$. As an example, given that $t$ has already occurred, an event $\sigma$ can occur with a probability lower than $t^{-1}K(\sigma)$, since this single step may be outweighed in the overall probability of $t\sigma$ due to a greater probability of $t$ occurring in $L^\text{max}$ (this is precisely what the scaling factor $\frac{K(t)}{L^\text{max}(t)}$ determines).

The next theorem states the correctness of Algorithm 4. 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

$$\forall t = s\sigma \in s(\Sigma - S(s)) : \frac{K(t)}{L^S(t)}(t^{-1}K) \preceq (t^{-1}L)^{S^1} \preceq (t^{-1}D) \quad (20)$$

Proof: Pick $t = s\sigma \in s(\Sigma - S(s))$ and $u \in \Sigma^*$. Then from hypothesis that $S \in \mathcal{S}$ we have:

$$K(tu) = K(t)(t^{-1}K)(u) \preceq L^S(tu) = L^S(t)(t^{-1}L^S)(u) \preceq D(tu) = D(t)(t^{-1}D)(u).$$

Since $u \in \Sigma^*$ is arbitrary, this implies

$$\frac{K(t)}{L^S(t)}(t^{-1}K) \preceq (t^{-1}L^S) \preceq \frac{D(t)}{L^S(t)}(t^{-1}D).$$

Since $\sigma$ is enabled after $s$ by $S$, it follows that $L^S(s\sigma) = L^S(t) \leq 1$. By hypothesis, $L^S(t) \leq D(t) \in \{0, 1\}$. So we must have $D(t) = 1$, and hence $\frac{D(t)}{L^S(t)} \geq 1$. So the last inequality is equivalent to

$$\frac{K(t)}{L^S(t)}(t^{-1}K) \preceq (t^{-1}L^S) \preceq (t^{-1}D). \quad (21)$$

Note $\frac{K(t)}{L^S(t)}(t^{-1}K)$ is a p-map (by hypothesis $K(t) \leq L^S(t)$), $(t^{-1}D)$ is a np-language, and $\text{supp}(\frac{K(t)}{L^S(t)}(t^{-1}K)) = \text{supp}(t^{-1}K)$. Thus Equation (21) implies the existence of a supervisor for the plant $(t^{-1}L)$ such that the controlled plant satisfies the specifications $\frac{K(t)}{L^S(t)}(t^{-1}K) \preceq (t^{-1}D)$. So it follows from the necessity part of Theorem 2 that

$$\frac{K(t)}{L^S(t)}(t^{-1}K) \preceq (t^{-1}L)^{S^1} \preceq (t^{-1}D),$$

as desired.  

Theorem 6 Let $L, K, D, S^{\text{max}}$ be as in Algorithm 4. Then $S^{\text{max}}$ is a maximal supervisor such that $K \preceq L^{S^{\text{max}}} \preceq D$.

Proof: From the hypothesis that $K \preceq L^{S^1} \preceq D$ (refer to Algorithm 4) we know that a supervisor exists. So from the fourth part of Theorem 5, a maximal supervisor also exists. We need to show that $S^{\text{max}}$ is such a supervisor.

We first show using induction on the length of traces that $S^{\text{max}} \in \mathcal{S}$, i.e., $K \preceq L^{S^{\text{max}}} \preceq D$. The base step follows from the hypothesis that $K \preceq L^{S^1} \preceq D$, which implies

$$K(\epsilon) \preceq L^{S^1}(\epsilon) = 1 = L^{S^{\text{max}}}(\epsilon) \preceq D(\epsilon).$$

For the induction step consider a trace $s$ and an event $\sigma$ such that $t := s\sigma \in \text{supp}(L)$. We need to show that

$$K(t) \preceq L^{S^{\text{max}}}(t) \preceq D(t),$$

22
or equivalently,
\[
\frac{K(t)}{L^{S_{\text{max}}}(t)} \leq 1, \text{ and } L^{S_{\text{max}}}(t) \leq D(t).
\]

The first part follows from construction by evaluating Equation (19) at \( \epsilon \), since
\[
(t^{-1}K)(\epsilon) = (t^{-1}L)^{S_{i}}(\epsilon) = 1.
\]

Using \( t = s\sigma \), the second part can be rewritten as
\[
L^{S_{\text{max}}}(s)[(s^{-1}L^{S_{\text{max}}})(\sigma)] \leq D(s)[(s^{-1}D)(\sigma)].
\]

This follows from the induction hypothesis, which gives \( L^{S_{\text{max}}}(s) \leq D(s) \), and the construction, which gives \( \text{supp}(s^{-1}L^{S_{\text{max}}}) \cap \Sigma = S^{\text{max}}(s) \subseteq \text{supp}(s^{-1}D) \cap \Sigma \).

Next suppose for contradiction that \( S^{\text{max}} \) as obtained in Algorithm 4 is not maximal. Then there exists another supervisor \( S \in \mathcal{S} \) such that \( S^{\text{max}} \leq S \), or equivalently, \( \text{supp}(L^{S_{\text{max}}}) \subseteq \text{supp}(L^{S}) \). This implies that there exists a trace in \( \text{supp}(L^{S}) - \text{supp}(L^{S_{\text{max}}}) \).

Let \( t \) be such a minimal length trace. Since \( \epsilon \in \text{supp}(L^{S_{\text{max}}}) \), \( t \neq \epsilon \). So \( t = s\sigma \), where \( s \in \text{supp}(L^{S_{\text{max}}}) \) and \( \sigma \in \Sigma \). Since \( t \) is a minimal trace in \( \text{supp}(L^{S}) - \text{supp}(L^{S_{\text{max}}}) \), we have
\[
\forall u < s : S(u) = S^{\text{max}}(u); \quad S(s) \subseteq S^{\text{max}}(s).
\]

This implies
\[
\forall t = s\sigma \in s(\Sigma - S(s)), \forall u < t : L^{S}(u) = L^{S_{\text{max}}}(u); \quad L^{S}(t) \leq L^{S_{\text{max}}}(t).
\]

By multiplying both sides of the last inequality by \( K(t)(t^{-1}K) \) and rearranging we get
\[
\forall t = s\sigma \in s(\Sigma - S(s)) : \frac{K(t)}{L^{S_{\text{max}}}(t)}(t^{-1}K) \leq \frac{K(t)}{L^{S}(t)}(t^{-1}K).
\]

(22)

Since \( S \in \mathcal{S} \), from Lemma 3, Equation (20) holds. This together with Equation (22) gives:
\[
\forall t = s\sigma \in s(\Sigma - S(s)) : \frac{K(t)}{L^{S_{\text{max}}}(t)}(t^{-1}K) \leq \frac{K(t)}{L^{S}(t)}(t^{-1}K) \leq (t^{-1}L)^{S_{i}} \leq (t^{-1}D).
\]

By construction \( S^{\text{max}}(s) \) is a maximal set such that Equation (19) holds. So we must have \( \Sigma - S(s) \subseteq S^{\text{max}}(s) \) for at least one such maximal sets, implying \( S^{\text{max}}(s) \subseteq S(s) \), a contradiction.

Remark 5 In Algorithm 4 the computation of \( S^{\text{max}}(s) \) at each trace \( s \) requires testing the condition of Equation (19) for all possible set of events \( \Sigma^{\text{max}}(s) \). Ignoring the dependence of computational complexity on the size of the set \( \Sigma \), we see that the computational complexity at each step of Algorithm 4 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.
Remark 6 Algorithm 4 can be used to derive some intuition regarding the off-line computation of a maximally permissive supervisor. It follows that given two traces $s$ and $t$, a maximally permissive supervisor $S_{\text{max}}$ exists that takes identical control actions in future whenever

1. $\forall u : [D(su) = D(tu)]$,
2. $\forall u : [\frac{L(su)}{L(s)} = \frac{L(tu)}{L(t)}]$,
3. $\forall u : \frac{K(su)}{K(s)} = \frac{K(tu)}{K(t)}$, and
4. $\frac{K(s)}{L_{S_{\text{max}}}(s)} = \frac{K(t)}{L_{S_{\text{max}}}(t)}$.

In other words, the pair of traces $s$ and $t$ are equivalent in the maximally permissive supervised behavior whenever they are equivalent according to all the above four relations. The first three relations have finite cardinality whenever $D, L, K$ are all regular. Determining the cardinality of the final relation remains open.

In the following remark we compare the maximally permissive supervisor of Algorithm 4 with the maximally permissive supervisor in the non-stochastic setting.

**Remark 7** 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 $K \leq D$, we obtain $\text{supp}(K) \subseteq \text{supp}(L^S) \subseteq \text{supp}(D)$.

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 $s \in \text{supp}(L)$, define $S_{\sup}(s) := \Sigma - \Sigma_{\sup}(s)$, where $\Sigma_{\sup}(s) \subseteq \Sigma$ is the supremal set such that

$$\forall t = s\sigma \in s\Sigma_{\sup}(s) : \text{supp}(t^{-1}K) \subseteq \text{supp}(t^{-1}L) \subseteq \text{supp}(t^{-1}D). \quad (23)$$

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 $\text{supp}[\frac{K(t)}{L_{S_{\sup}}(t)}(t^{-1}K)] = \text{supp}(t^{-1}K)$, it follows from Equation (19) that

$$\forall t = s\sigma \in s\Sigma_{\max}(s) : \text{supp}(t^{-1}K) \subseteq \text{supp}(t^{-1}L) \subseteq \text{supp}(t^{-1}D).$$

This together with the definition of $S_{\sup}$ given in Equation (23) 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.
\[ L(e) = 1, L(a) = L(b) = L(c) = \frac{1}{3}, \text{ and } L(s) = 0, \text{ otherwise} \]
\[ K(a) = K(b) = \frac{1}{2}, \text{ and } K(s) = 0, \text{ otherwise} \]
\[ D(s) = 1, \forall s \]

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.

**Example 4** Let \( L, K, D \) be as in Example 3. Then since \( \text{supp}(K) \subset \text{supp}(L) \), and since all traces are legal under the specification \( D \), in the non-stochastic setting the supremely permissive supervisor \( S^{sup} \) disables no events, i.e., \( L^{S^{sup}} = L \).

However, in the stochastic setting, it may not hold that

\[ S^{sup} \in \mathcal{S} = \{ S : \text{supp}(L) \rightarrow 2^{\Sigma - \Sigma_u} | K \preceq L^S \preceq D \}. \]

To see this suppose \( r > \frac{1}{2} \), which implies \( 2(1 - r) < 1 \). Then

\[ K(bb) = \frac{qe}{2(1 - r)} > qe = L(bb) = L^{S^{sup}}(bb), \]

which implies \( S^{sup} \not\in \mathcal{S} \). From Example 3 we know that \( \mathcal{S} \neq \emptyset \). Any supervisor \( S \in \mathcal{S} \) must be such that \( K(bb) \leq L^S(bb) \), which is possible only if \( S \) disables \( a \) following the occurrence of \( b \), i.e., \( S(b) = \{ a \} \). (Note that if \( S(b) = \{ a \} \), then

\[ L^S(bb) = L^S(b)^{(b^{-1}L(b))} = q \frac{e}{1 - r - e} + e = \frac{qe}{1 - r} > \frac{qe}{2(1 - r)} = K(bb). \]

From Example 2 we know that the minimally permissive supervisor \( S^1 \) disables \( a \) following the occurrence of \( b \), implying that \( S^1 \) is the only supervisor in \( \mathcal{S} \), i.e., it is also the maximally permissive supervisor.

**6 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, 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 \( O(n^3) \), where \( n \) is the product of the number of states in the automata representations of the plant and the lower bound specification.
The supervisory control formalization presented here can be extended in many ways. Firstly, the upper bound constraint may be more general, specified as a probabilistic map. Secondly, the control objective may be different, such as: in the controlled plant only legal states can be visited, and that the probability of visiting each legal state exceeds a certain minimum probability. Finally, an off-line computation of a maximally permissive supervisor remains an interesting open problem.

References


