Predicates and Predicate Transformers for Supervisory Control of Discrete Event Dynamical Systems [1]

Ratnesh Kumar
Department of Electrical Engineering
University of Kentucky
Lexington, KY 40506-0046

Vijay Garg
Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, TX 78712-1084

Steven I. Marcus
Department of EE and System Research Center
University of Maryland
College Park, MD 20742

February 2, 1995

[1] This research was supported in part by the Center for Robotics and Manufacturing, University of Kentucky, in part by the National Science Foundation under Grant NSFD-CDR-8803012 and NSF-CCR-9110605, in part by the Air Force Office of Scientific Research (AFOSR) under Contract F49620-92-J-0045, in part by a University Research Institute Grant and in part by a Bureau of Engineering Research Grant.

Abstract

Most discrete event system models are based on defining the alphabet set or the set of events as a fundamental concept. In this paper, we take an alternative view of treating the state space as the fundamental concept. We approach the problem of controlling discrete event systems by using predicates and predicate transformers. Predicates have the advantage that they can concisely characterize an infinite state space. The notion of controllability of a predicate is defined, and the supervisory predicate control problem introduced in this paper is solved. A closed form expression for the weakest controllable predicate is obtained. The problem of controlling discrete event systems under incomplete state observation is also considered and observability of predicates is defined. Techniques for finding extremal solutions of boolean equations is used to derive minimally restrictive supervisors.

1 Introduction

Many discrete event system models [24, 23, 25, 9, 11, 12] are based on defining the alphabet set or the set of events as a fundamental concept. The language of a deterministic system characterizes its behavior, and two systems are considered equivalent if they have the same alphabet and language [9]. In this paper, we take an alternative view of treating the state space as the fundamental concept. Problems treated in [22, 19, 3, 1, 18, 14, 7] are also formalized with a similar point of view.

We approach the problem of controlling the behavior of a discrete event system described in terms of its state trajectories by using predicates and predicate transformers. Predicates have the advantage that they can concisely characterize an infinite state space. Petri net based models have also been used for describing infinite state discrete event dynamical systems [26, 17]. The notions of two types of predicate transformers, namely *strongest post-condition* (sp) and *weakest liberal precondition* (wlp) [5, 6, 8] are very useful in characterizing the dynamics of discrete event dynamical systems. In this paper, we study the system dynamics in the framework of these predicate transformers. We use the notion of *duality* of predicate transformers and show that sp and wlp are duals of each other. Thus one of the predicate transformers - either sp or wlp - can be treated as fundamental and the other as a derived notion. In this paper we treat sp to be the fundamental predicate transformer, as it describes the forward evolution of the system behavior, and develop the supervisory control theory using it.

We describe a few basic properties - strictness, monotonicity, disjunctivity, conjunctivity etc. - of predicate transformers. sp is a strict, monotone and disjunctive predicate transformer, while its dual wlp is a strict, monotone and conjunctive. One or more of these properties of sp and wlp are used to obtain all the results in this paper. It is known [6] that a predicate equation in the variable predicate Q of the type $Q : f(Q) \preceq g(Q)$ has unique extremal solutions, provided the predicate transformers f, g satisfy certain basic properties. We use the extremal solutions of such predicate equations to demonstrate the existence and uniqueness of *minimally restrictive* [23, 2] supervisors.

We introduce the supervisory predicate control problem as the problem of synthesizing a supervisor for a given system so that the state trajectories of the system remain confined to a set of "legal" states, and also visit all the states in the set of legal states. Thus the set of legal predicate corresponds to the *weakest predicate that remains invariant* under control. A special case of this problem where the latter constraint is relaxed was considered in [22]. The notion of *controllability* is defined and it is shown that it serves as a necessary and sufficient condition for the existence of a supervisor that solves the supervisory predicate control problem. A different definition of controllability is presented in [19], which can be shown to be equivalent to our definition. Our definition of controllability is purely in terms of the predicate transformer sp, which results in a more compact definition, simplicity of the proofs (as demonstrated by the proof of Theorem 4.5) as well as the synthesis techniques of the supervisors.

In this paper, we also address the problem of synthesizing supervisors for the case when the required predicate is not controllable. This problem is quite important and is not addressed in [19, 22]. We show that if the given "legal" predicate is not controllable, then the minimally restrictive supervisor can be constructed so that the state trajectories of the controlled system remain confined to and also visit all the states in the set of states where the weakest controllable legal predicate holds. We prove that the *weakest controllable* predicate stronger than the required predicate exists and present an algorithm for computing it. This algorithm is then used for constructing the minimally restrictive supervisor. Thus our work extends the earlier works on supervisory predicate control [22, 19]. In [22] a method is presented for computing the weakest "invariant" predicate stronger than the legal predicate; which can also be computed as a special case using the method of computing the weakest controllable predicate presented in this paper.

Next we consider the supervisory predicate control problem under partial state observations. The behavior of the system, i.e. the state trajectories, are observed under a "mask" which maps the state space of the system to the "observation space", and is not necessarily injective. This problem is much harder, as the controller also consists of a state estimator, and based on its state estimates takes the appropriate control actions. This problem was first addressed in [19] and a solution was obtained under very restrictive assumptions on the desired reachable predicate. This is because the observability condition obtained in [19] is based only on the current observations and ignores all the past observations. We obtain the *observability* condition based on the entire available information: control as well as observation; present as well as past. Thus the notion of observability introduced in this paper is quite general. The notion of dynamical observers, which use the entire history for estimating states, is presented in [3, 21]. However, in [3] it is assumed that the transition events are completely known at all the transition steps, which is not the case in this paper. All that is known is at any transition step one of the several events that are enabled by the supervisor will occur and cause a system transition. In [21] the issue of synthesizing dynamical state estimator for a partially observed system was addressed and no control was exercised. In this paper we address the issue of simultaneously estimating the state and controlling the system so that the set of reachable states equals the required legal set of states.

The notion of observability of a predicate described in this paper also leads to a synthesis technique for the minimally restrictive supervisor. Unlike the supervisory control of system behavior described in terms of event trajectories under partial observation where the minimally restrictive supervisor does not exist [20], we show in this paper that it is possible to construct the minimally restrictive supervisor for the supervisory predicate control problem under partial state observation, where the system behavior is described as the weakest invariant predicate.

The advantage of using predicates and predicate transformers to represent a DEDS is that we can concisely characterize systems with a very large, possibly infinite, number of states. This is illustrated by the *Readers-Writers* example considered in this paper, in which case the state space is infinite. We provide a technique for synthesizing a supervisor so as to ensure *mutual exclusion* of readers and writers. Also, a technique is provided for the synthesis of the minimally restrictive supervisor for a modified *Readers-Writers* problem, again with infinite state space. Thus we have successfully developed a technique for supervisory control of infinite state systems. The computational complexity of our approach depends on the number of variables and conditional assignment statements representing the system rather than the actual number of states and transitions. Computationally more efficient algorithms for supervisory control of infinite state systems need to be developed based on the theory presented in this paper. Some such techniques that involve mathematical induction as an analysis tool are reported in [7], and further research on this issue is currently under investigation.

2 Notation and Terminology

The discrete event dynamical system (DEDS) to be controlled - called the *plant* - is modeled as a state machine (SM) [10] following the framework of [23]. Let the quadruple $G \stackrel{\text{def}}{=} (X, \Sigma, \delta, x_0)$ denote a SM representing a plant; where X denotes the state set; Σ denotes the finite event or alphabet set; $\delta : X \times \Sigma \to X$ denotes the partial state transition function; and $x_0 \in X$ denotes the initial state.

A supervisor or controller for the given DEDS is designed so that the behavior of the closed loop system satisfies certain qualitative constraints described in terms of the state set of the plant. The event set is partitioned into $\Sigma = \Sigma_u \cup (\Sigma - \Sigma_u)$, the sets of *uncontrollable* and *controllable* events. A supervisor for the given plant G is characterized by a (static) control law $S : X \to 2^{\Sigma}$. A control law is said to be static if the control action at each step depends only on the observation at that step (for a more detailed and formal definition refer to [13, 16]). A dynamic control law will be considered in section 5, where the supervisory control problem under partial observation is addressed.

Thus if $\sigma \in \Sigma$ is such that $\sigma \in S(x)$ for some $x \in X$, then σ is said to be enabled by the supervisor in state x. Since a supervisor can disallow only controllable events from occurring, we also have for each $x \in X$, $\Sigma_u(x) \subseteq S(x)$, where $\Sigma_u(x)$ is the set of uncontrollable events defined at state x. The controlled system is then described by the state machine $G_S \stackrel{\text{def}}{=} (X, \Sigma, \delta_S, x_0)$, where for $x \in X$ and $\sigma \in \Sigma$, $\delta_S(x, \sigma) = \delta(x, \sigma)$ if $\sigma \in S(x)$, and undefined otherwise.

Remark 2.1 One way to implement a static control law as described above is to let the *subautomaton* of G corresponding to the control law S run in *synchrony* with G, as described in [13, 16].

2.1 Predicates and Predicate Transformers

Next we introduce a few definitions from the theory of *predicate* calculus [5, 6, 8] that we use in this paper to study and formulate the supervisory predicate control problem. Let $\mathcal{P}$ denote the collection of predicates defined on the state set X, i.e. if $P \in \mathcal{P}$, then it is a boolean valued map $P : X \to \{0, 1\}$. With every $P \in \mathcal{P}$, we associate a set $X_P \subseteq X$ on which P takes the value one, i.e. $x \in X_P$ if and only if $P(x) = 1$. We say that the predicate P holds on $x \in X$ if $P(x) = 1$. Conversely, given a set $X' \subseteq X$, it can be associated with a predicate $P_{X'} \in \mathcal{P}$ such that $P_{X'}(x) = 1$ if and only if $x \in X'$. Thus the collection of predicates $\mathcal{P}$ can be associated with the power set 2^X using the association described above. In what follows next, we use the names predicates and subsets interchangeably. The symbols P, Q, R etc. are used for denoting predicates.

Definition 2.2 Given $P \in \mathcal{P}$, its *negation*, denoted $\neg P$, is another predicate defined to be: for every $x \in X, \neg P(x) = 1 \Leftrightarrow P(x) = 0$. Given an indexing set Λ, let $P_\lambda \in \mathcal{P}$ for each $\lambda \in \Lambda$. Then the *conjunction* $\bigwedge_{\lambda \in \Lambda} P_\lambda$ is defined to be: for every $x \in X, \bigwedge_{\lambda \in \Lambda} P_\lambda(x) = 1 \Leftrightarrow \forall \lambda \in$

$\Lambda, P_\lambda(x) = 1$; and the *disjunction* $\bigvee_{\lambda\in\Lambda} P_\lambda$ is defined to be: for every $x \in X, \bigvee_{\lambda\in\Lambda} P_\lambda(x) = 1 \Leftrightarrow \exists\lambda \in \Lambda$ s.t. $P_\lambda(x) = 1$.

Definition 2.3 The symbols $true$ and $false$ are used for denoting predicates that hold on all and none of the states respectively, i.e. $true(x) = 1, \forall x \in X$ and $false(x) = 0, \forall x \in X$. Thus $true = \neg false$. Also, the predicate $true$ can be associated with the entire state space X, and the predicate $false$ can be associated with the empty set $\emptyset$.

Example 2.4 Let $X = \mathcal{R}^2$, i.e. the state space equals the real plane. Let x, y be state variables taking values in $\mathcal{R}$. Then the predicate $(x \le y)$ holds in those states in $\mathcal{R}^2$ where the value of the variable x is not greater than that of variable y. Notice that predicates can concisely describe an infinite state space.

The quadruple $(\mathcal{P}, \neg, \wedge, \vee)$ forms a boolean algebra which is isomorphic to the algebra of subsets of X under the operations of complementation, intersection and union. For $P_1, P_2 \in \mathcal{P}$, we say that $P_1 \preceq P_2$ if and only if $P_1 \wedge P_2 = P_1$, or $P_1 \vee P_2 = P_2$. P_1 is said to be *stronger* than P_2, equivalently, P_2 is said to be *weaker* than P_1 if $P_1 \preceq P_2$. Note that $\preceq$ induces a partial order on $\mathcal{P}$, i.e. $\preceq$ is a reflexive, transitive and antisymmetric relation on $\mathcal{P}$. Since $\preceq$ is antisymmetric, if $P_1, P_2 \in \mathcal{P}$ are such that $P_1 \preceq P_2$ and $P_2 \preceq P_1$, then $P_1 = P_2$, i.e. they are the same predicate. It can be shown that the partial order $(\mathcal{P}, \preceq)$ is also *complete* [9].

Let $\mathcal{F}$ denote the collection of all *predicate transformers*, i.e. if $f \in \mathcal{F}$, then $f : \mathcal{P} \to \mathcal{P}$. We use the symbols f, g, h etc. to denote predicate transformers.

Definition 2.5 The *negation* $\neg f$ for some $f \in \mathcal{F}$ is defined to be $(\neg f)(P) = \neg(f(P))$ for each $P \in \mathcal{P}$. The *conjunction* and *disjunction* of an arbitrary set of predicate transformers are defined in an analogous way and are obtained by taking the conjunction and disjunction, respectively, over the set of image predicates, i.e. given an arbitrary indexing set Λ, $(\bigwedge_{\lambda\in\Lambda} f_\lambda)(P) = \bigwedge_{\lambda\in\Lambda}(f_\lambda(P))$, and $(\bigvee_{\lambda\in\Lambda} f_\lambda)(P) = \bigvee_{\lambda\in\Lambda}(f_\lambda(P))$ for each $P \in \mathcal{P}$.

Definition 2.6 Consider $G \stackrel{\text{def}}{=} (X, \Sigma, \delta, x_0)$. For each $\sigma \in \Sigma$, $sp_\sigma : \mathcal{P} \to \mathcal{P}$ is defined to be: $sp_\sigma(P) \stackrel{\text{def}}{=} Q$, where $X_Q = \{x \in X \mid \exists y \in X_P \text{ s.t. } \delta(y, \sigma) = x\}$. We use sp, sp_u to denote $\bigvee_{\sigma\in\Sigma} sp_\sigma, \bigvee_{\sigma\in\Sigma_u} sp_\sigma$ respectively.

Thus $sp_\sigma(P)$ is the predicate which holds on the set of states that are reached by the transition σ from a state where P holds.

Definition 2.7 For the system G, the predicate transformer $wlp_\sigma \in \mathcal{F}$ for each $\sigma \in \Sigma$ is: $wlp_\sigma(P) \stackrel{\text{def}}{=} Q$, where $X_Q = \{x \in X \mid \text{ either } \delta(x, \sigma) \in X_P \text{ or } \delta(x, \sigma) \text{ is undefined}\}$. The predicate transformers wlp, wlp_u are defined to be $\bigwedge_{\sigma\in\Sigma} wlp_\sigma$, $\bigwedge_{\sigma\in\Sigma_u} wlp_\sigma$ respectively.

Thus $wlp_\sigma(P)$ is the predicate which holds in those states where either the transition σ is not defined or a σ transition from them leads to a state where P holds.

Example 2.8 Let the state space be $\mathcal{R}^2$. We will use the pair (x, y) to denote an arbitrary element of $\mathcal{R}^2$. Consider a program G that assigns (x, y) to $(x + y, x - y)$, i.e.

$$G : (x, y) := (x + y, x - y).$$

Then given any set $R \subseteq \mathcal{R}^2$, it gets "transformed" to the set $\{(x', y') \mid (x' = x + y) \wedge (y' = x - y), (x, y) \in R\}$ whenever G is executed. Let the predicate $(xy \geq 10)$ be true upon execution of G. Then the predicate $(x^2 - y^2) \geq 10$ must be true before execution of G. In other words, $wlp_{(x,y):=(x+y,x-y)}((xy \geq 10)) = (x^2 - y^2 \geq 10)$.

The predicate transformers sp and wlp as defined above are called *strongest postcondition* and the *weakest liberal precondition* [5, 6, 8]. We use the notation $sp_S(wlp_S)$ to denote the strongest postcondition (weakest liberal precondition) operator induced by the transition function δ_S of the controlled system G_S, i.e. $sp_S = \bigvee_{\sigma \in \Sigma}(sp_S)_\sigma$ and $wlp_S = \bigwedge_{\sigma \in \Sigma}(wlp_S)_\sigma$. With the above introduction on predicates and predicate transformers it is clear that a DEDS can also be represented in terms of predicates and predicate transformers.

Definition 2.9 $G \stackrel{\text{def}}{=} (\mathcal{P}_X, \Sigma, sp, I)$; where $\mathcal{P}_X$ corresponds to the set of predicates defined on the state space X of G; Σ denotes the event set; $sp \in \mathcal{F}$ is the predicate transformer corresponding to the state transition function of G; and $I \in \mathcal{P}_X$ corresponds to the initial states (also called initial condition) of G.

The behavior of a DEDS is essentially described by $sp \in \mathcal{F}$ which is specified using the state transition function. It may also be specified using a finite set of conditional assignment statements of the type:

$$x := F(x) \text{ if } C(x) \qquad : \sigma$$

The above assignment statement, labeled by the event σ, is said to be enabled if the the condition specified as the predicate $C(x)$ holds, and if executed, value of variable x becomes $F(x)$. Thus given any predicate $P(x)$, $sp_\sigma(P(x))$ is the predicate reached after the execution of σ and can be readily calculated to be $[P(F^{-1}(x)) \wedge C(F^{-1}(x))]$, where $F^{-1}(x) = \{x' \mid F(x') = x\}$. Similarly, the predicate $wlp_\sigma(P(x))$ can be easily computed to be $[[P(F(x)) \wedge C(x)] \vee \neg C(x)]$.

Consider for example a conditional assignment statement:

$$x, y := x + y, x - y \text{ if } x > y \qquad : \sigma$$

Let $P(x, y) = x + y \geq 10$. Then $sp_\sigma(P(x, y)) = [\{\frac{x+y}{2} + \frac{x-y}{2} \geq 10\} \wedge \{\frac{x+y}{2} > \frac{x-y}{2}\}] = [(x \geq 10) \wedge (y > 0)]$; and $wlp_\sigma(P(x, y)) = [\{(x + y) + (x - y) \geq 10\} \wedge \{x > y\}] \vee [x \leq y] = [\{(x \geq 5) \wedge (x > y)\} \vee \{x \leq 5\}]$.

Note that the DEDS representation as described in Definition 2.9 describes a wide range of DEDS's, such as system with an infinite state space, a nondeterministic [10] system, or a system that could initially be in a set of states where the predicate I holds. Henceforth we use the representation of G introduced in Definition 2.9 for describing a DEDS.

Example 2.10 Consider the following program which corresponds to the *Readers–Writers* problem written in a programming logic adapted from UNITY framework [4]. Informally stated, the Readers–Writers problem can be expressed as a DEDS which has a distributed database, access to which is sought by an infinite numbers of readers and writers.

Program	Rd_Wr				
declare	nr, nw	:	integer		
	$st\_rd, st\_wt, end\_rd, end\_wt$	:	event		
initially	nr, nw	=	0,0		
assign	nr	:=	$nr+1$		$:st\_rd$
			$nr-1$	if $(nr>0)$	$:end\_rd$
	nw	:=	$nw+1$		$:st\_wt$
			$nw-1$	if $(nw>0)$	$:end\_wt$
end	{Rd_Wr}				

The **declare** section contains the description of the program variables. The *state variables* nr, nw are integer type and denote the number of readers, number of writers respectively accessing the database. Both nr, nw are bounded below by zero. Thus the state space of the system is $\mathcal{N}^2$ which is infinite. The symbols $st\_rd, st\_wt, end\_rd, end\_wt$ correspond to the events start-read, start-write, end-read, end-write respectively. The program starts executing with the system being in the initial state $((nr, nw) = (0, 0))$, which is described in the **initial** section of the program. The system evolves according to the execution of the "enabled" assignment statements of the **assign** section. An assignment statement is said to be enabled whenever the condition under the "if" part of the assignment is satisfied. One of the enabled assignment statements is nondeterministically picked for execution and upon execution the state variables accordingly change their values. The entries in the last column of the **assign** section are the event names for the corresponding assignment statements.

The program Rd_Wr describes a DEDS of the type: $G \stackrel{\text{def}}{=} (\mathcal{P}_X, \Sigma, sp, I)$; where $\mathcal{P}_X$ denotes the set of predicates corresponding to the subsets of the state set $X = \mathcal{N}^2$; $\Sigma = \{st\_rd, st\_wt, end\_rd, end\_wt\}$; sp corresponds to the assignments of the **assign** section as aforementioned; and $I = ((nr, nw) = (0, 0))$.

3 On Solving Predicate Equations

So far we have defined the notions of predicates and predicate transformers. Next we consider some of their properties and describe a few methods for solving some predicate equations that we use later to design supervisors for a given DEDS. Some of the results presented in this section can also be found in [6], however, we present their proofs here mainly to illustrate the proof style that we follow throughout the paper.

Definition 3.1 [6] Consider $f \in \mathcal{F}$. f is said to be *strict* if $f(false) = false$; *monotone* if $P \preceq Q \Rightarrow f(P) \preceq f(Q)$; *disjunctive* if $f(\bigvee_{\lambda\in\Lambda} P_\lambda) = \bigvee_{\lambda\in\Lambda} f(P_\lambda)$; and *conjunctive* if $f(\bigwedge_{\lambda\in\Lambda} P_\lambda) = \bigwedge_{\lambda\in\Lambda} f(P_\lambda)$ (Λ denotes an arbitrary indexing set, and we adopt the convention that the disjunction as well as conjunction over the empty set is predicate $false$, i.e. $\bigvee_{\lambda\in\Lambda} P_\lambda = \bigwedge_{\lambda\in\Lambda} P_\lambda = false$ if $\Lambda = \emptyset$).

It is easily shown that $sp_\sigma(\bigvee_{\lambda\in\Lambda} P_\lambda) = \bigvee_{\lambda\in\Lambda} sp_\sigma(P_\lambda)$, as $\{x \in X \mid \exists\lambda \in \Lambda : \exists y \in X_{P_\lambda} \text{ s.t. } \delta(y, \sigma) = x\} = \exists\lambda \in \Lambda : \{x \in X \mid \exists y \in X_{P_\lambda} \text{ s.t. } \delta(y, \sigma) = x\}$. Thus sp is disjunctive, and similarly it is easily verified that wlp is conjunctive.

Example 3.2 To illustrate that sp is disjunctive, consider the program of Example 2.8, and let $[x + y \leq 1]$ and $[(x, y) = (0, 1)]$ be two predicates on the state space $\mathcal{R}^2$. Then under program G, $sp([x+y \leq 1]) = ([x \leq 1])$, and $sp([(x,y)=(0,1)]) = [(x,y)=(1,-1)]$. Since $[x+y \leq 1] \vee [(x,y)=(0,1)] = [x+y \leq 1]$, $sp([x+y \leq 1] \vee [(x,y)=(0,1)]) = sp([x+y \leq 1]) = [x \leq 1]$. Notice that $[x \leq 1] \vee [(x,y)=(1,-1)] = [x \leq 1]$ as expected, for sp is disjunctive.

wlp is conjunctive; to illustrate this consider two predicates $xy \leq 10$ and $x=2$ on the state space $\mathcal{R}^2$. Then $xy \leq 10 \wedge (x=2) = y \leq 5$. Consider the program G of Example 2.8. Then under G, $wlp([xy \leq 10]) = [x^2 - y^2 \leq 10]$, $wlp([x=2]) = [x+y=2]$, and $wlp([y \leq 5]) = [x-y \leq 5]$. Note that $[x^2-y^2 \leq 10] \wedge [x+y=2] = [x-y \leq 5]$ as expected, for wlp is conjunctive.

Lemma 3.3 [6] Consider $f \in \mathcal{F}$.

1. f disjunctive implies that f is strict and monotone.

2. f conjunctive implies that f is strict and monotone.

Proof: We prove the first part of Lemma 3.3; the proof of the second part is obtained in a similar manner. Let $P, Q \in \mathcal{P}$ be arbitrary.

1.	$f(\bigvee_{\lambda\in\Lambda} P_\lambda) = \bigvee_{\lambda\in\Lambda} f(P_\lambda)$	;f is disjunctive
2.	$f(\bigvee_{\lambda\in\emptyset} P_\lambda) = \bigvee_{\lambda\in\emptyset} f(P_\lambda)$	;replace Λ by $\emptyset$ in 1
3.	$f(false) = false$	;from 2 and using $\bigvee_{\lambda\in\Lambda} P_\lambda = false$
4.	f is strict	;from 3
5.	$P \preceq Q \Rightarrow P \vee Q = Q$	;by definition of $\preceq$
6.	$P \preceq Q \Rightarrow f(P \vee Q) = f(Q)$	;apply f on 5
7.	$P \preceq Q \Rightarrow f(P) \vee f(Q) = f(Q)$	;f is disjunctive
8.	$P \preceq Q \Rightarrow f(P) \preceq f(Q)$	;follows from 7 and definition of $\preceq$
9.	f is monotone	;from 8

This completes the proof. □

We define the following operations on $f \in \mathcal{F}$ (we have already defined negation, disjunction and conjunction above):

Definition 3.4 The *conjugate* of f, written as $\overline{f}$, is defined to be $\neg f \neg$, i.e. for $P \in \mathcal{P}$, $\overline{f}(P) = \neg(f(\neg P))$; the *disjunctive closure* of f, written as $f^\star$, is defined to be $\bigvee_{n\geq 0} f^n$; and the *conjunctive closure* of f, written as $f_\star$, is defined to be $\bigwedge_{n\geq 0} f^n$, where f^0 is defined to be the identity predicate transformer.

Thus $\overline{sp}(P)$ characterizes the predicate which holds in states that cannot be reached by a single transition from a state where $\neg P$ holds, i.e. in states that either have no transitions leading into them or which can be reached by a single transition only from those states where P holds. $sp^\star(P)$ denotes the predicate which holds in those states that can be reached by any number of transitions from a state where P holds. Thus $sp^\star$ is useful in characterizing the set of *reachable* states.

$\overline{wlp}(P)$ characterizes the predicate which holds in states from which only a state where P holds can be reached by a single transition. $wlp_\star(P)$ characterizes the the weakest predicate stronger than P that is closed under the executions of G, i.e. $X_{wlp_\star(P)}$ is the supremal Σ-*invariant* [13, 16] subset of X_P.

Lemma 3.5 Let $f \in \mathcal{F}$ be monotone; then

1. $f(P) \preceq P \Leftrightarrow f^\star(P) \preceq P$
2. $P \preceq f(P) \Leftrightarrow P \preceq f_\star(P)$.

Proof: We only prove the first part of Lemma 3.5; the proof for the second part is derived in an analogous way. Note that the reverse implication is obvious; we use induction on the exponent n in the definition of $f^\star$ to prove the forward implication.

1.	$f^\star(P) \preceq P \Rightarrow f(P) \preceq P$	;by definition of $f^\star$
2.	$f^0(P) \preceq P \Leftrightarrow true$	;by definition of f^0
3.	$f(P) \preceq P \Rightarrow f^0(P) \preceq P$	;from 2 (base case for induction)
4.	$f(P) \preceq P \Rightarrow f^n(P) \preceq P$	;induction hypothesis
5.	$f(P) \preceq P \Rightarrow f(f^n(P)) \preceq f(P)$	;apply f on RHS of 4, and f monotone
6.	$f(P) \preceq P \Rightarrow f^{n+1}(P) \preceq P$	;simplifying 5
7.	$\forall i \geq 0 : f(P) \preceq P \Rightarrow f^i(P) \preceq P$	;from 3, 6 and induction
8.	$f(P) \preceq P \Rightarrow f^\star(P) \preceq P$	;taking disjunct wrt i in 7
9.	$f(P) \preceq P \Leftrightarrow f^\star(P) \preceq P$	;from 1 and 8

This completes the proof. $\square$

Since sp is disjunctive, it is also monotone (Lemma 3.3). Thus Lemma 3.5 applies to sp as well; the implication of part 1 is that if the set of states reached by a single transition is contained in the set of starting states, then so is the entire set of reachable states. The implication of the second part of Lemma 3.5 applied to wlp (wlp conjunctive implies wlp monotone from Lemma 3.3) is that if the set of states from which only the states in a target state set can be reached in a single transition contains the set of target states, then so does the set of states from which only the target state set is reached in all numbers of transitions.

Lemma 3.6 [6] Consider $f \in \mathcal{F}$.

1. If f is disjunctive, then so is $f^\star$.
2. If f is conjunctive, then so is $f_\star$.

Proof: As above we omit the proof of the second part, which can be obtained analogously to the proof of part 1 that we present next. It suffices to show that f^n is disjunctive for each $n \in \mathcal{N}$, for if f^n is disjunctive for each $n \in \mathcal{N}$, then $f^\star(\bigvee_{\lambda\in\Lambda} P_\lambda) = \bigvee_{n\geq 0} f^n(\bigvee_{\lambda\in\Lambda} P_\lambda) = \bigvee_{n\geq 0}\bigvee_{\lambda\in\Lambda} f^n(P_\lambda) = \bigvee_{\lambda\in\Lambda}\bigvee_{n\geq 0} f^n(P_\lambda) = \bigvee_{\lambda\in\Lambda} f^\star(P_\lambda)$. Hence we prove disjunctivity of f^n for each $n \in \mathcal{N}$ by induction on the exponent n in the definition of $f^\star$.

1.	$f^1 = f$ is disjunctive	;by assumption (base case for induction)
2.	$f^{n+1} = f(f^n)$	;definition of exponent
3.	$f^n(\bigvee_{\lambda\in\Lambda} P_\lambda) = \bigvee_{\lambda\in\Lambda} f^n(P_\lambda)$	;induction hypothesis
4.	$f^{n+1}(\bigvee_{\lambda\in\Lambda} P_\lambda) = f(\bigvee_{\lambda\in\Lambda} f^n(P_\lambda))$	;using 2 and 3
5.	$f^{n+1}(\bigvee_{\lambda\in\Lambda} P_\lambda) = \bigvee_{\lambda\in\Lambda} f^{n+1}(P_\lambda)$	;from 4 and f is disjunctive

Thus the proof is completed. $\square$

Lemma 3.7 [6] If f is disjunctive (conjunctive), then $\overline{f}$ is conjunctive (disjunctive).

Proof: The proof is simple and based on application of De Morgan's law. $\square$

It thus follows from Lemma 3.6 that $sp^\star$ is disjunctive, and from Lemma 3.7 that $\overline{sp}$ and $\overline{sp^\star}$ are conjunctive. Similarly, since wlp is conjunctive, so is $wlp_\star$ (Lemma 3.6) and $\overline{wlp}$ and $\overline{wlp_\star}$ are disjunctive (Lemma 3.7). Next we quote a result regarding existence of the extremal solution of a predicate equation, the proof of which can be found in [6]. A notation of the type $Q : f(Q) \preceq h(Q)$, where $f, h \in \mathcal{F}$ and $Q \in \mathcal{P}$, is used to denote a predicate equation in the variable predicate Q such that it staisfies $f(Q) \preceq h(Q)$.

Theorem 3.8 [6] Consider the predicate equation $Q : f(Q) \preceq h(Q)$.

1. If f is disjunctive and h is monotone, then the weakest solution of the above equation exists (and is unique).
2. If f is monotone and h is conjunctive, then the strongest solution of the above equation exists (and is unique).

An immediate consequence of Theorem 3.8 is the following corollary:

Corollary 3.9 Let f be disjunctive and $P \in \mathcal{P}$ be arbitrary; then

1. the weakest solution of the equation $Q : f(Q) \preceq P$ exists.
2. the weakest solution of the equation $Q : f(Q) \preceq Q$ exists.

Proof: 1. Follows from the fact that P, treated as a constant predicate transformer, is monotone.
2. Follows from the fact that the *identity* function is monotone. $\square$

Next we show that there exists a strong relationship between the set of disjunctive and the set of conjunctive predicate transformers. A result of similar nature is obtained in [6, p. 202, Theorem 1] regarding *converses* of predicates.

Theorem 3.10 Consider $f, g \in \mathcal{F}$. Let f be disjunctive and g be conjunctive; then the following are equivalent:

C-1. $g(P)$ is the weakest solution of $Q : f(Q) \preceq P$ for all $P \in \mathcal{P}$.

C-2. $(f(g(P)) \preceq P) \wedge (P \preceq g(f(P)))$ for all $P \in \mathcal{P}$

C-3. $f(P) \preceq Q \Leftrightarrow P \preceq g(Q)$ for all $P, Q \in \mathcal{P}$.

C-4. $f(P)$ is the strongest solution of $Q : P \preceq g(Q)$ for all $P \in \mathcal{P}$.

Proof: Refer to Appendix A. $\square$

The weakest solution of $Q : f(Q) \preceq P$ depends both on f and P. Let it be denoted by $f^\perp(P)$. Note that $f^\perp \in \mathcal{F}$. We define it to be the *dual* of f. Formally,

Definition 3.11 Let $f^\perp(\cdot) \in \mathcal{F}$ be the weakest solution of $Q : f(Q) \preceq (\cdot)$, where $f \in \mathcal{F}$ is disjunctive. Then $f^\perp$ is called the *dual* of f.

Lemma 3.12 If f is disjunctive, then its dual $f^\perp$ is conjunctive.

Proof: Consider the equation $Q : f(Q) \preceq \bigwedge_{\lambda\in\Lambda} P_\lambda$. Then by definition $f^\perp(\bigwedge_{\lambda\in\Lambda} P_\lambda)$ is the weakest solution of this equation. We show that $\bigwedge_{\lambda\in\Lambda}(f^\perp(P_\lambda))$ is also the weakest solution; hence from the uniqueness of the solution it follows that $f^\perp$ is conjunctive. First we show that $\bigwedge_{\lambda\in\Lambda}(f^\perp(P_\lambda))$ is a solution.

1.	$\forall\lambda\in\Lambda : f(f^\perp(P_\lambda)) \preceq P_\lambda$	;$f^\perp(P_\lambda)$ is a solution of $Q : f(Q) \preceq P_\lambda$
2.	$\forall\lambda\in\Lambda : \bigwedge_{\lambda\in\Lambda}(f^\perp(P_\lambda)) \preceq f^\perp(P_\lambda)$	;definition of conjunction
3.	$\forall\lambda\in\Lambda : f(\bigwedge_{\lambda\in\Lambda}(f^\perp(P_\lambda))) \preceq f(f^\perp(P_\lambda))$	;apply f on 2, f monotone (Lemma 3.3)
4.	$\forall\lambda\in\Lambda : f(\bigwedge_{\lambda\in\Lambda}(f^\perp(P_\lambda))) \preceq P_\lambda$	;from 1 and 3
5.	$f(\bigwedge_{\lambda\in\Lambda}(f^\perp(P_\lambda))) \preceq \bigwedge_{\lambda\in\Lambda} P_\lambda$	;by taking conjunct wrt λ in 4
6.	$\bigwedge_{\lambda\in\Lambda}(f^\perp(P_\lambda))$ is a solution	;from 5

Next we show that $\bigwedge_{\lambda\in\Lambda}(f^\perp(P_\lambda))$ is the weakest solution also. Let R be another solution of $Q : f(Q) \preceq \bigwedge_{\lambda\in\Lambda}(P_\lambda)$. Then we need to show that $R \preceq \bigwedge_{\lambda\in\Lambda}(f^\perp(P_\lambda))$.

1.	$f(R) \preceq \bigwedge_{\lambda\in\Lambda} P_\lambda$	;by assumption
2.	$\forall\lambda\in\Lambda : f(R) \preceq P_\lambda$	;from 1
3.	$\forall\lambda\in\Lambda : R$ is a solution of $Q : f(Q) \preceq P_\lambda$	;from 2
4.	$\forall\lambda\in\Lambda : f^\perp(P_\lambda)$ is weakest solution of $Q : f(Q) \preceq P_\lambda$	;by definition of $f^\perp$
5.	$\forall\lambda\in\Lambda : R \preceq f^\perp(P_\lambda)$	;from 3 and 4
6.	$R \preceq \bigwedge_{\lambda\in\Lambda}(f^\perp(P_\lambda))$	;taking conjunct wrt λ in 5
7.	$\bigwedge_{\lambda\in\Lambda}(f^\perp(P_\lambda))$ is the weakest solution	;from 5

This proves that $f^\perp$ is conjunctive. $\square$

An immediate consequence of Theorem 3.10 and Lemma 3.12 is the following corollary:

Corollary 3.13 Let $f \in \mathcal{F}$ be disjunctive. Then the strongest solution of the equation $Q : P \preceq f^\perp(Q)$ exists and is given by $f(P)$.

Proof: From Lemma 3.12 we have that $f^\perp$ is conjunctive. Since P as a constant predicate transformer is monotone, the strongest solution of $Q : P \preceq f^\perp(Q)$ exists (Theorem 3.8). That $f(P)$ is the strongest solution follows by substituting $f^\perp$ for g in Theorem 3.10. $\square$

Remark 3.14 The result of Corollary 3.13 justifies the term dual for the functions f and $f^\perp$. Note that C-1 is used to define the dual of a disjunctive predicate transformer. Since in Theorem 3.10 we showed the equivalence of C-1 and C-2 and C-3 and C-4, any one of them can be used to equivalently define the dual predicate transformer.

Next we prove a result that is interesting from the control perspective.

Theorem 3.15 *wlp* and *sp* are duals of each other.

Proof: Note that *sp* is disjunctive and *wlp* is conjunctive; hence the duality is well defined in this context. In order to show duality we need to show that *wlp* and *sp* satisfy any of the conditions C-1 through C-4 (refer to Theorem 3.10 and Remark 3.14). We show that C-2 holds.

Firstly, it follows from the definitions of *sp* and *wlp* that $sp(wlp(P)) \preceq P$ for any $P \in \mathcal{P}$. This is true because $sp(wlp(P))$ holds only in those states of X_P which have at least one transition leading into them. Secondly, it again follows from the definitions of *sp* and *wlp* that $P \preceq wlp(sp(P))$ for any $P \in \mathcal{P}$. This is true because $wlp(sp(P))$ holds in those states where either P holds or which have no transitions leading out of them. Thus both the conjuncts of C-2 hold, which proves the duality of *wlp* and *sp*. $\square$

4 Predicate Transformers and Supervisory Control

In the previous section we described the conditions under which extremal solutions of various boolean equations exist, and introduced the notion of duality of predicate transformers, which is one of the key concepts relating the extremal solutions of the above boolean equations. We now show how these concepts can be useful in synthesizing static supervisors [16, 13] for a given DEDS.

Let $G \stackrel{\text{def}}{=} (\mathcal{P}_X, \Sigma, sp, I)$ be a plant as described in Definition 2.9. Let $R \in \mathcal{P}$ denote the required behavioral constraint on G. In other words, the control task is to design a static controller $S : X \rightarrow 2^{\Sigma}$ such that as the closed loop system evolves, it visits only and all those states where R holds. Formally,

Supervisory Predicate Control Problem (SPCP): The control task is to construct a static controller $S : X \rightarrow 2^{\Sigma}$ for the plant $G \stackrel{\text{def}}{=} (\mathcal{P}_X, \Sigma, sp, I)$ such that $sp_S^{\star}(I) = R$.

The SPCP requires that the state trajectories (in the controlled system G_S) starting from a state where the initial predicate I holds, remain confined to the set of states where the required predicate R holds, and visit all the states where R holds. A special case of this problem was first treated in [22] where the control task was to synthesize a static supervisor S such that $sp_S^{\star}(I) \preceq R$, i.e. the states visited under closed loop control be confined to R. Thus R in [22] represents *a* predicate that remains invariant under control. The required predicate R considered in this paper represents the *weakest* predicate that remains invariant under control (i.e. if S solves SPCP, then no other predicate weaker than R is invariant under S).

It is clear that SPCP is solvable only if the set of initial states is contained in the set of states where R holds. Hence in order to allow nontrivial solution of the SPCP we assume that the above condition is satisfied, which we state as assumption A-1 below:

A-1. $I \preceq R$.

Next we define a few notions that play a central role in supervisory control of DEDS.

Definition 4.1 R is said to be *invariant* if $sp(R) \preceq R$. R is said to be Σ_u*-invariant* if $sp_u(R) \preceq R$. R is said to be *control-invariant* if there exists a static controller $S : X \rightarrow 2^{\Sigma}$ such that $sp_S(R) \preceq R$.

Thus if R is an invariant predicate and if the system starts from a state where R holds, then as it evolves it visits only those states where R holds. If R is Σ_u-invariant, and if the system starts in a state where R holds, then under the execution of any uncontrollable event it remains in a state where R holds. If R is control-invariant, then there exists a static supervisor S such that R is invariant in the controlled system G_S. Note that all the above notions are defined with respect to the plant G, for they depend on the plant transition function sp. It also follows in view of C-3 of Theorem 3.10 that R being invariant, Σ_u-invariant, control-invariant is equivalent to $R \preceq wlp(R), R \preceq wlp_u(R), R \preceq wlp_S(R)$ respectively. These are the same as the definitions given in [22]. Hence it follows in view of Proposition 7.1 of [22] that R is control-invariant if and only if R is Σ_u-invariant.

These notions of invariance introduced in [22] are useful in characterizing those sets of states which are closed under the system execution, i.e. if the system starts in a state of an invariant set, then under all executions the state of the system remains in that invariant set. However, it may be quite possible that the system may never visit some states in that invariant set. Thus the notion of invariance alone is not enough for addressing the SPCP. The notion of *controllability* of predicates (a notion stronger than that of invariance) was introduced in [19] for addressing the above mentioned problem. We present an equivalent but slightly different definition of *controllability* and show that controllability is a necessary and sufficient condition for a solution of SPCP to exist. In our opinion, our definition is much more compact and uses a more convenient notation that results in simplicity of proofs and supervisory synthesis techniques.

Definition 4.2 Given $f \in \mathcal{F}$ and $P \in \mathcal{P}$, the *restriction* of f to P, denoted $f\,|_P$, is the predicate transformer defined as: $(f\,|_P)(Q) = f(P \wedge Q) \wedge P$ for each $Q \in \mathcal{P}$.

Next we prove a useful property of the restriction operator. We say $f \in \mathcal{F}$ is *weakening* if $P \preceq f(P)$ for any $P \in \mathcal{P}$.

Lemma 4.3 Let $f \in \mathcal{F}$ be monotone and weakening, and $P, Q \in \mathcal{P}$ be arbitrary. Then $f\,|_{(f|_P(Q))}\,(Q) = f\,|_P\,(Q)$.

Proof: Let $R \stackrel{\text{def}}{=} f\,|_P\,(Q)$. Then we need to show that $f\,|_R\,(Q) = R$.

1.	$R = f(P \wedge Q) \wedge P$	;definition of restriction
2.	$f\,|_R\,(Q) = f(f(P \wedge Q) \wedge P \wedge Q) \wedge f(P \wedge Q) \wedge P$	;definition of $f\,|_{(\cdot)}$ and 1
3.	$P \wedge Q \preceq f(P \wedge Q)$	;f is weakening
4.	$P \wedge Q \preceq f(P \wedge Q) \wedge P \wedge Q$	;conjunct with $P \wedge Q$ in 3
5.	$f(P \wedge Q) \preceq f(f(P \wedge Q) \wedge P \wedge Q)$	;apply f in 4, f monotone
6.	$f\,|_R\,(Q) = f(P \wedge Q) \wedge P$	;from 2 and 5
7.	$f\,|_R\,(Q) = R$	;from 6 and definition of R

This completes the proof. $\square$

Thus it follows from Lemma 4.3 that the restriction of a monotone predicate transformer, the application of which results in an image predicate weaker than its preimage predicate, exhibits a nice "invariance" property. For example, the disjunctive closure of any predicate transformer is weakening as well as monotone (Lemma 3.3 and 3.5), and thus exhibits such a property.

Definition 4.4 R is said to be *controllable* with respect to G if

1. $sp_u(R) \preceq R$, and
2. $R = (sp\,|_R)^\star(I)$

Note that it follows from the definition of restriction and disjunctive closure that the following ordering always holds: $(sp\,|_R)^\star(I) \preceq R$. Thus the second condition in the definition of controllability is equivalent to $R \preceq (sp\,|_R)^\star(I)$. We use either of these equivalent definitions of controllability interchangeably. In the next theorem we present a solution to the SPCP. The simplicity of the proof obtained by using the theory of predicate transformers and a more compact definition of controllability is easily seen.

Theorem 4.5 The solution to SPCP exists if and only if R is controllable with respect to G.

Proof: First assume that R is controllable; we show that there exists a controller $S : X \to 2^{\Sigma}$ such that $sp_S^\star(I) = R$. Consider the controller defined as: for each $x \in X_R, \sigma \in S(x) \Leftrightarrow \delta(x, \sigma) \in X_R$. Since R is Σ_u-invariant, it follows that the events disabled by S are all controllable (S never disables any uncontrollable events). Also, note that the strongest postcondition predicate transformer sp_u corresponds to the maximally restrictive control law - the control law that disables all the controllable events from occurring.

1.	$sp_u(R) \preceq R$	;by assumption (R is controllable)
2.	$sp \mid_R = sp_S$	;from 1, definitions of S and $sp \mid_R$ and A-1
3.	$R = (sp \mid_R)^\star(I)$	;by assumption (R is controllable)
4.	$R = sp_S^\star(I)$	;from 2 and 3

Next we show that if there exists a controller S such that $sp_S^\star(I) = R$, then R is controllable.

1.	$sp_S^\star(I) = R$	;by assumption
2.	$sp_S^\star(I) = sp_S^\star(R)$	;apply $sp_S^\star$ on 1
3.	$sp_S^\star(R) = R$	;from 1 and 2
4.	$sp_u^\star(R) \preceq sp_S^\star(R)$	;sp_u : most restrictive control
5.	$sp_u^\star(R) \preceq R$	;from 3 and 4
6.	$sp_u(R) \preceq R$	;from 5 and Lemma 3.5
7.	$(sp_S \mid_R)^\star(I) = R$	;from 1 and Lemma 4.3
8.	$(sp_S \mid_R)^\star(I) \preceq (sp \mid_R)^\star(I)$	;S restricts behavior
9.	$R \preceq (sp \mid_R)^\star(I)$	;from 7 and 8
10.	R is controllable	;from 6 and 9

This completes the proof of Theorem 4.5. □

Example 4.6 Consider the problem of *mutual exclusion* for the Readers-Writers program of Example 2.10. The mutual exclusion constraint requires that the number of writers accessing the database should never be more than one, and a reader can access the database only when no writer is accessing it. Thus the mutual exclusion constraint can be written as the following required predicate: $R = ((nw = 0) \vee (nw = 1 \wedge nr = 0))$. Let $\Sigma_u = \{end\_rd, end\_wt\}$. Then it is easily verified that R is controllable, namely, $sp_u(R) \preceq R$ and $(sp \mid_R)^\star(I) = R$ as described below. First consider the event *end_rd*:

$$nr := nr - 1 \text{ if } nr > 0 \qquad : end\_rd$$

Comparing the above statement with the standard form:

$$x := F(x) \text{ if } C(x) \qquad : \sigma$$

we obtain $F(x) = x - 1$, i.e. $F^{-1}(x) = x + 1$; and $C(x) = [x > 0]$. Thus

$sp_{end\_rd}(R(nr))$
$= R(F^{-1}(nr)) \wedge C(F^{-1}(nr))$
$= [(nw = 0) \vee (nw = 1 \wedge nr + 1 = 0)] \wedge [nr + 1 > 0]$
$= [(nw = 0) \wedge (nr + 1 > 0)] \vee [(nw = 1) \wedge (nr + 1 = 0) \wedge (nr + 1 > 0)]$
$= [(nw = 0) \wedge true] \vee false$

$= (nw = 0)$
$\preceq R.$

Next consider the event end_wt:

$$nw := nw - 1 \text{ if } nw > 0 \qquad :$$

Thus $F(x) = x - 1$ and $C(x) = [x > 0]$ as before. Hence
$sp_{end\_wt}(R(nw))$
$= R(F^{-1}(nw)) \wedge C(F^{-1}(nw))$
$= [(nw + 1 = 0) \vee (nw + 1 = 1 \wedge nr = 0)] \wedge [nw + 1 > 0]$
$= [nw + 1 = 0 \wedge nw + 1 > 0] \vee [nw + 1 = 1 \wedge nr = 0 \wedge nw + 1 > 0]$
$= false \vee [nw = 0 \wedge nr = 0]$
$= [nw = 0 \wedge nr = 0]$
$\preceq R.$

Combining the results of the above two derivations, we thus obtain:
$sp_u(R)$
$= (nw = 0) \vee (nw = 0 \wedge nr = 0) = (nw = 0)$
$\preceq R.$

Next, in order to verify $(sp \mid R)^\star(I) = R$, it can be easily shown by induction on $n \in \mathcal{N}$ that $(sp\,|_R)^n(I) = (nw = 0 \wedge nr \leq n) \vee (nw = 1 \wedge nr = 0)$. Hence
$(sp\,|_R)^\star(I)$
$= \bigvee_{n \geq 0}(nw = 0 \wedge nr \leq n) \vee (nw = 1 \wedge nr = 0)$
$= (nw = 0 \wedge true) \vee (nw = 1 \wedge nr = 0)$
$= (nw = 0) \vee (nw = 1 \wedge nr = 0)$
$= R.$

It then follows that R is controllable and hence the SPCP is solvable. The supervisor S can be computed as follows. For each controllable event $\sigma \in (\Sigma - \Sigma_u)$ of the type:

$$x := F(x) \text{ if } C(x) \qquad : \sigma,$$

the predicate on which the event σ is disable by S is computed as

$$C(x) \wedge R(x) \wedge wlp_\sigma(\neg R(x)).$$

This is the weakest predicate stronger than R, where σ is enabled ($C(x)$ holds), and from which a state in $X_{\neg R}$ (states where R does not hold) is reachable by a single execution of σ. First consider the event st_rd; then S disables st_rd on the predicate:
$true \wedge [(nw = 0) \vee (nw = 1 \wedge nr = 0)] \wedge [wlp_{st\_rd}([nw > 1] \vee [nw = 1 \wedge nr > 0])]$
$= [(nw = 0) \vee (nw = 1 \wedge nr = 0)] \wedge [(nw > 1) \vee (nw = 1 \wedge nr + 1 > 0)]$
$= [(nw = 0) \vee (nw = 1 \wedge nr = 0)] \wedge (nw \geq 1)$
$= (nw = 1 \wedge nr = 0).$
Thus S disables st_rd on $[nw = 1 \wedge nr = 0]$.

The predicate where S disables the other controllable event st_wt can be computed as:
$true \wedge [(nw = 0) \vee (nw = 1 \wedge nr = 0)] \wedge [wlp_{st\_wr}((nw > 1) \vee (nw = 1 \wedge nr > 0))]$
$= [(nw = 0) \vee (nw = 1 \wedge nr = 0)] \wedge [(nw + 1 > 1) \vee (nw + 1 = 1 \wedge nr > 0)]$
$= [(nw = 0) \vee (nw = 1 \wedge nr = 0)] \wedge [(nw > 0) \vee (nw = 0 \wedge nr > 0)]$

$= [nw = 0 \wedge nw > 0] \vee [nw = 0 \wedge (nw = 0 \wedge nr > 0)] \vee [(nw = 1 \wedge nr = 0) \wedge nw > 0] \vee [(nw = 1 \wedge nr = 0) \wedge (nw = 0 \wedge nr > 0)]$
$= false \vee (nw = 0 \wedge nr > 0) \vee (nw = 1 \wedge nr = 0) \vee (nw = 1 \wedge nr = 0)$
$= (nw = 0 \wedge nr > 0) \vee (nw = 1 \wedge nr = 0)$.
Thus S disables st_wt in $(nw = 0 \wedge nr > 0) \vee (nw = 1 \wedge nr = 0)$.

Thus it is clear that at states where $(nw = 1 \wedge nr = 0)$ holds both the controllable events are disabled by S, and at states where $(nw = 0 \wedge nr > 0)$ holds only st_wr is disabled. Thus $S : X \to 2^{\Sigma}$ is given by: (x is used to denote an arbitrary element of $X = \mathcal{N}^2$, i.e. $x = (nr, nw)$):

$$S(x) \stackrel{\text{def}}{=} \begin{cases} \Sigma(x) - \{st\_wt, st\_rd\} & \text{if } (nw = 1 \wedge nr = 0) \\ \Sigma(x) - \{st\_wt\} & \text{if } (nw = 0 \wedge nr > 0) \\ \Sigma(x) & \text{otherwise} \end{cases}$$

where $\Sigma(x) = \{\sigma \in \Sigma \mid \delta(x, \sigma) \text{ is defined}\}$.

Remark 4.7 This example illustrates that techniques based on predicates and predicate transformers can be used for solving the supervisory control problem in an infinite state space setting. The computational complexity of computing the supervisor is linear in the number of variables used and the number of conditional assignment statements in the program description of the plant, and does not depend on the actual number of states and transitions in the system which may be very large, possibly infinite.

Another advantage of using the theory based on predicate transformers is that it provides an automated technique for synthesizing supervisors as illustrated by the above example.

4.1 Minimally Restrictive Supervisors for Predicate Control

It follows from Theorem 4.5 that if the required predicate R is controllable, i.e. satisfies the Σ_u-invariance and the reachability constraint $R \preceq (sp\mid_R)^*(I)$, then the following control law can be used for solving the SPCP: for each $x \in X_R$, $\sigma \in S(x) \Leftrightarrow \delta(x, \sigma) \in X_R$. In this subsection, we address the problem of supervisory synthesis when the required predicate R is not controllable. This problem was not addressed in [19]. If the required predicate R is not controllable, then a supervisor cannot be constructed which solves the SPCP. In the next Theorem we prove that in such a situation, the minimally restrictive supervisor can be constructed.

Theorem 4.8 The weakest controllable predicate stronger than a given predicate exists (and is unique).

Proof: Let $R \in \mathcal{P}$ be the required predicate. Assume that R is not controllable; then at least one of the following two conditions is not satisfied:

1. $sp_u(R) \preceq R$
2. $R \preceq (sp\mid_R)^*(I)$

We show that the weakest solution to the following set of equations exists:

E-1. $Q : sp_u(Q) \preceq Q$

E-2. $Q : Q \preceq (sp\mid_Q)^\star(I)$

E-3. $Q : Q \preceq R$

Equations E-1 and E-2 correspond to conditions 1 and 2 respectively. Equation E-3 requires that the weakest solution be stronger than R. Consider the first equation; since sp_u is disjunctive and Q as an identity function is monotone, it follows from Theorem 3.8 that the weakest solution of it exists. Similarly consider the second equation; since Q as an identity function is disjunctive and $(sp\mid_Q)^\star(I)$ as a function of Q is monotone, it follows that the second equation also has a weakest solution. Also, since R as a constant function is monotone and Q as an identity function is disjunctive, the third equation possess a weakest solution as well. Since the weakest solutions of all the equations E-1 through E-3 exist, it follows that the weakest solution of the above set of equation exists and by its definition it is unique. More formally, let Λ be an indexing set such that for each $\lambda \in \Lambda, Q_\lambda$ satisfies the above set of equations. Then as explained above $\bigvee_{\lambda\in\Lambda} Q_\lambda$ satisfies all the equations E-1 through E-3 individually. Thus $\bigvee_{\lambda\in\Lambda} Q_\lambda$ is the weakest solution of the above set of equations. $\Box$

We will use $R^\uparrow \preceq R$ to denote the weakest solution of the above set of equations, then $R^\uparrow$ denotes the weakest controllable predicate stronger than R. If $R^\uparrow$ also satisfies A-1, i.e. if $I \preceq R^\uparrow$, then since $R^\uparrow$ is controllable, it follows from Theorem 4.5 that there exists a static control law S such that $sp_S^\star(I) = R^\uparrow$. A supervisor exercising such a control law is called the *minimally restrictive* supervisor. However, $R^\uparrow$ may not satisfy A-1, i.e. it is possible that $I \not\preceq R^\uparrow$, in which case the minimally restrictive supervisor does not exist.

4.2 Computation of $R^\uparrow$

In Theorem 4.8 we proved the existence of the weakest controllable predicate $R^\uparrow$, which is the weakest solution of equations E-1, E-2 and E-3, stronger than R. Now we present a method for computing it. We proceed by first proving a few lemmas.

First we note that the weakest solution of E-2 stronger than any predicate $P \in \mathcal{P}$ exists. This follows easily from 1 and 2 below:

1. E-2 has a weakest solution (follows from Theorem 4.8), and

2. Equation $Q : Q \preceq P$ has a weakest solution, for Q as an identity predicate transformer is monotone and P as a constant predicate transformer is disjunctive.

Lemma 4.9 If $P \in \mathcal{P}$ is a solution of E-1 and E-3 and P' is the weakest solution of E-2 stronger than P, then P' is also a solution of E-1 and E-3.

Proof: We first show that P' is a solution of E-3.

1.	$P' \preceq P$	;by assumption
2.	$P \preceq R$	;P is a solution of E-3
3.	$P' \preceq R$	;from 1 and 2
4.	P' is a solution of E-3	;from 3

Next we show that P' is also a solution of E-1.

1.	$sp_u(P) \preceq P$	;P is a solution of E-1
2.	$P' \preceq P$	;by assumption
3.	$sp_u(P') \preceq sp_u(P)$	;apply sp_u on 2, sp_u monotone
4.	$sp_u(P') \preceq P$	;from 1 and 3
5.	$sp_u(P') \vee P' \preceq P$	;from 2 and 4
6.	$P' \preceq (sp\mid_{P'})^\star(I)$	;P' is a solution of E-2
7.	$P' \preceq (sp\mid_{sp_u(P') \vee P'})^\star(I)$	;by weakening the RHS of 6
8.	$sp_u(P') \preceq (sp\mid_{sp_u(P') \vee P'})^\star(I)$	;from 6 and definition of $sp\mid_{(\cdot)}$
9.	$sp_u(P') \vee P' \preceq (sp\mid_{sp_u(P') \vee P'})^\star(I)$	;taking disjunct of 7 and 8
10.	$sp_u(P') \vee P'$ is a solution of E-2 stronger than P	;from 9 and 5
11.	$sp_u(P') \vee P' \preceq P'$	;from 10 and definition of P'
12.	$sp_u(P') \preceq P'$	;simplifying 11
13.	P' is a solution of E-1	;from 12

This completes the proof. □

Thus the weakest solution of E-2 stronger than any solution of E-1 and E-3 is also a solution of E-1 and E-3 and hence a solution of the all the three equations.

Lemma 4.10 Let P be the weakest solution of E-1 and E-3 and P' be the weakest solution of E-2 stronger than P. Then P' is the weakest solution of E-1 through E-3.

Proof: It follows from Lemma 4.9 that P' is a solution of E-1 through E-3. We need to show that it is the weakest solution also, i.e. $P' = R^\uparrow$.

1.	$P' \preceq P$	;by assumption
2.	$P' \preceq R^\uparrow$	;P' a solution of E-1, E-2, E-3 and $R^\uparrow$ weakest solution of E-1, E-2, E-3
3.	$R^\uparrow \preceq P$	;$R^\uparrow$ weakest solution of E-1, E-2, E-3 and P weakest solution of E-1, E-3
4.	$R^\uparrow \preceq P'$	;$R^\uparrow$ weakest solution of E-1, E-2, E-3 stronger than P (from 3) and P' weakest solution of E-2 stronger than P (from 1)
5.	$R^\uparrow = P'$	;from 2 and 4

Thus the proof is completed. □

It follows from Lemma 4.10 that one way to compute $R^\uparrow$ is by computing the weakest solution of E-2 stronger than the weakest solution of E-1 and E-3.

Theorem 4.11 The weakest solution of E-1 and E-3 is $(wlp_u)_\star(R)$.

Proof: Note that by definition, $(wlp_u)_\star(R) = \bigwedge_{n \geq 0} wlp_u(R)$. We first show that $(wlp_u)_\star(R)$ is a solution of E-1.

1.	$\bigwedge_{n \geq 0} wlp_u(R) \preceq \bigwedge_{n \geq 1} wlp_u(R)$	;trivially
2.	$(wlp_u)_\star(R) \preceq wlp_u(\bigwedge_{n \geq 0} wlp_u(R))$	;rewriting LHS and RHS of 1
3.	$(wlp_u)_\star(R) \preceq wlp_u((wlp_u)_\star(R))$	;rewriting RHS of 2
4.	$sp_u((wlp_u)_\star(R)) \preceq (wlp_u)_\star(R)$	;from 3 and C-3
5.	$(wlp_u)_\star(R)$ is a solution of E-1	;from 4

Next we show that $(wlp_u)_\star(R)$ is a solution of E-3.

1.	$\bigwedge_{n \geq 0} wlp_u(R) \preceq \bigwedge_{n=0} wlp_u(R)$	;trivially
2.	$(wlp_u)_\star(R) \preceq R$	;rewriting LHS and simplifying RHS of 1
3.	$(wlp_u)_\star(R)$ is a solution of E-3	;from 2

Next we prove that $(wlp_u)_\star(R)$ is the weakest solution of E-1 and E-3. Assume P is also a solution of E-1 and E-3.

1.	$sp_u(P) \preceq P$	;P is a solution of E-1
2.	$P \preceq wlp_u(P)$	;from 1 and C-3
3.	$P \preceq (wlp_u)_\star(P)$	;from 2 and Lemma 3.5
4.	$P \preceq R$	;P is a solution of E-3
5.	$(wlp_u)_\star(P) \preceq (wlp_u)_\star(R)$	;apply $(wlp_u)_\star$ on 4, $(wlp_u)_\star$ monotone
6.	$P \preceq (wlp_u)_\star(R)$	;from 3 and 5
7.	$(wlp_u)_\star(R)$ is the weakest solution	;from 6

This completes the proof of Theorem 4.11. □

Thus it follows from Theorem 4.11 that the weakest Σ_u-invariant predicate stronger than R (i.e. the weakest solution of E-1 and E-3) is given by $(wlp_u)_\star(R)$. Finally we have the following Theorem for computing $R^\uparrow$. This is one of the main results of this paper.

Theorem 4.12 $R^\uparrow = (sp\ |_{(wlp_u)_\star(R)})^\star(I)$.

Proof: Let $Q \stackrel{\text{def}}{=} (sp\ |_{(wlp_u)_\star(R)})^\star(I)$. In view of Lemma 4.10 it suffices to show that Q is the weakest solution of E-2 stronger than the weakest solution of E-1 and E-3. Since the weakest solution of E-1 and E-3 is $(wlp_u)_\star(R)$ (Theorem 4.11), we need to show that Q is the weakest solution of E-2 stronger than $(wlp_u)_\star(R)$, i.e. satisfies $Q \preceq (wlp_u)_\star(R)$. First we show that Q is a solution of E-2 and $Q \preceq (wlp_u)_\star(R)$.

1.	$(sp\ |_{(wlp_u)_\star(R)})^\star(I) \preceq (wlp_u)_\star(R)$	;by definition of restriction
2.	$Q \preceq (wlp_u)_\star(R)$	;from 1 and definition of Q
3.	$Q = (sp\ |_Q)^\star(I)$	;from Lemma 4.3
4.	$Q \preceq (sp\ |_Q)^\star(I)$	;from 3
5.	Q is a solution of E-2	;from 4

Next we show that Q is the weakest solution of E-2 stronger than $(wlp_u)_\star(R)$. Let P be another solution of E-2 stronger than $(wlp_u)_\star(R)$. Since we have already shown above that Q is stronger than $(wlp_u)_\star(R)$, it suffices to show that $P \preceq Q$.

1.	$P \preceq (wlp_u)_\star(R)$	;by assumption (P stronger than $(wlp_u)_\star(R)$)
2.	$P \preceq (sp\ |_P)^\star(I)$	;by assumption (P solution of E-2)
3.	$P \preceq (sp\ |_{(wlp_u)_\star(R)})^\star(I)$	;by weakening RHS of 2 using 1
4.	$P \preceq Q$	;from 3 and definition of Q

Hence the proof is completed. □

Remark 4.13 The set of states $X_{R^\uparrow}$ corresponding to the weakest controllable predicate stronger than R can be easily computed in two steps:

1. Compute $R_u \stackrel{\text{def}}{=} (wlp_u)_\star(R)$
2. Compute $(sp\ |_{R_u})^\star(I)$

The first step corresponds to the computation of the supremal Σ_u-invariant subset of X_R. This we denote by X_{R_u}. The second step consists of computing the set of states reachable from the initial state set X_I in the state space restricted to X_{R_u}. If G is represented as a finite state machine, then the set X_{R_u} as well as the states reachable from X_I in the state space restricted to X_{R_u} can be computed (in the worst case) in time linear in the number of transitions present in G (refer to [11, 15] for more elaborate discussions on computationally optimal algorithmic techniques for similar computations). If G has infinite states, then the computation of $R^{\uparrow}$ based on the state machine approach as described above will not terminate, as it involves the computation of the "$\star$" operator - disjunctive and conjunctive closure. However, computation based on predicates and predicate transformers can be used to automatically construct the minimally restrictive supervisor in an infinite state space setting using efficient techniques such as those reported in [7]. Further research on this issue is currently under investigation.

Example 4.14 Consider the following refinement of the Readers-Writers program of Example 2.10. We use variables ar, aw to denote the number of *active* readers, writers, respectively, and the variables wr, ww to denote the number of *waiting* readers, writers, respectively. The event set consists of: $\Sigma = \{st\_rd, st\_wt, end\_rd, end\_wt, req\_rd, req\_wt, ovflo\}$. Informally described, readers and writers are first buffered in separate queues of finite capacity, and whenever req_rd or req_wt occurs, the size of the corresponding queue increases. The number of active readers/writers is increased (decreased) according to the occurrence of $st\_rd/st\_wt(end\_rd/end\_wt)$. If the number of waiting readers is more than a positive number B, then the number of active readers increases by one whenever the event $ovflo$ occurs. Formally,

Program	Rd_Wr 1				
declare	wr, ww, ar, aw	:	integer		
	$st\_rd, st\_wt, ovflo,$				
	$end\_rd, end\_wt,$				
	req_rd, req_wt	:	event		
initially	wr, ww, ar, aw	$=$	0,0,0,0		
assign	wr, ar	$:=$	$wr+1, ar$		$: req\_rd$
			$wr-1, ar+1$	if $wr > 0$	$: st\_rd$
			$wr, ar+1$	if $wr \geq B$	$: ovflo$
			$wr, ar-1$	if $ar > 0$	$: end\_rd$
	ww, aw	$:=$	$ww+1, aw$		$: req\_wt$
			$ww-1, aw+1$	if $ww > 0$	$: st\_wt$
			$ww, aw-1$	if $aw > 0$	$: end\_wt$
end	{Rd_Wr 1}				

Let the uncontrollable event set be given by $\Sigma_u = \{ovflo\}$. As in Example 4.6, the mutual exclusion constraint for the above program is given as the required predicate $R = (aw = 0) \vee (aw = 1 \wedge ar = 0)$. It can be readily verified that R is not a controllable predicate, for $sp_u(R) \not\preceq R$. In order to show that R is not controllable consider the uncontrollable event $ovflo$. Note that $F(ar) = ar + 1$ and $C(wr) = (wr \geq B)$ for the event $ovflo$. Then

$sp_{ovflo}(R(ar, aw))$
$= (aw = 0 \vee (aw = 1 \wedge ar - 1 = 0)) \wedge (wr \geq B)$
$= (aw = 0 \vee (aw = 1 \wedge ar = 1)) \wedge (wr \geq B)$
$\not\preceq R.$

In order to compute $R^{\uparrow}$, we first compute $(wlp_{ovflo})_{\star}(R) = \bigwedge_{n \geq 0}(wlp_{ovflo})^n(R)$. We need to compute $(wlp_{ovflo})^n(R)$ for each $n \in \mathcal{N}$. First we compute $wlp_{ovflo}(R)$:
$wlp_{ovflo}(R)$

$= wlp_{ar:=ar+1 \text{ if } wr \geq B}((aw = 0) \vee (aw = 1 \wedge ar = 0))$
$= [((aw = 0) \vee (aw = 1 \wedge ar + 1 = 0)) \wedge (wr \geq B)] \vee [wr < B]$
$= [(aw = 0) \wedge (wr \geq B)] \vee [wr < B]$
$= [(aw = 0) \vee (wr < B)] \wedge [(wr \geq B) \vee (wr < B)]$
$= [(aw = 0) \vee (wr < B)]$

Similarly we compute $(wlp_{ovflo})^2(R)$:
$(wlp_{ovflo})^2(R)$
$= wlp_{ovflo}(aw = 0 \vee wr < B)$
$= [(aw = 0 \vee wr < B) \wedge (wr \geq B)] \vee [wr < B]$
$= (aw = 0 \vee wr < B)$
$= wlp_{ovflo}(R)$

Hence $(wlp_{ovflo})^n(R) = wlp_{ovflo}(R)$ for each $n \geq 1$. Thus
$(wlp_{ovflo})_\star(R)$
$= R \wedge wlp_{ovflo}(R)$
$= [(aw = 0) \vee (aw = 1 \wedge ar = 0)] \wedge [(aw = 0) \vee (wr < B)]$
$= [(aw = 0) \wedge (aw = 0)] \vee [(aw = 0) \wedge (wr < B)] \vee [(aw = 1 \wedge ar = 0) \wedge (aw = 0)] \vee [(aw = 1 \wedge ar = 0) \wedge (wr < B)]$
$= (aw = 0) \vee (aw = 0 \wedge wr < B) \vee false \vee (aw = 1 \wedge ar = 0 \wedge wr < B)$
$= (aw = 0) \vee (aw = 1 \wedge ar = 0 \wedge wr < B)$

Thus $R_u \stackrel{\text{def}}{=} (wlp_u)_\star(R) = (aw = 0) \vee (aw = 1 \wedge ar = 0 \wedge wr < B)$. In order to compute $R^\uparrow$, we need to compute $(sp\,|_{R_u})^\star(I)$. This can be easily shown to equal R_u. Thus

$$R^\uparrow = (aw = 0) \vee (aw = 1 \wedge ar = 0 \wedge wr < B).$$

Using the technique described in Example 4.6, the predicate on which the minimally restrictive supervisor disables a given controllable event can be computed. Essentially, if $\sigma \in \Sigma - \Sigma_u$ is a controllable event of the type:

$$x := F(x) \text{ if } C(x) \qquad : \sigma,$$

then σ is disabled by S in the predicate $C(x) \wedge R^\uparrow(x) \wedge wlp_\sigma(\neg R^\uparrow(x)) = C(x) \wedge R^\uparrow(x) \wedge \neg R^\uparrow(F(x))$. The following minimally restrictive control $S : X \to 2^\Sigma$ can be used for achieving the mutual exclusion constraint (we use x to denote an arbitrary element of $X = \mathcal{N}^4$:

$$S(x) \stackrel{\text{def}}{=} \begin{cases} \Sigma(x) - \{st\_wt, st\_rd, req\_rd\} & \text{if } (aw = 1 \wedge ar = 0 \wedge wr = B - 1) \\ \Sigma(x) - \{st\_wt, st\_rd\} & \text{if } (aw = 1 \wedge ar = 0 \wedge wr < B - 1) \\ \Sigma(x) - \{st\_wt\} & \text{if } (aw = 0 \wedge ar > 0) \\ \Sigma(x) & \text{otherwise} \end{cases}$$

5 Observability of Predicates

So far we have considered the supervisory predicate control problem assuming that complete information about the system states is available. Next we generalize the theory of supervisory predicate control developed above to the case where the system states are not necessarily completely observed. In order to formulate the problem of supervisory predicate control under partial state observation, consider a mask M, which is a map from the system state space X to the observation space Y, i.e. $M : X \to Y$. Note that the mask M is not necessarily injective, and it is possible that two different states may yield an identical observation under the mask M.

The supervisory predicate control problem under partial observation was first studied in [19]. However, the conditions of observability of a predicate were obtained under very restrictive assumptions on the mask M. It was assumed in [19] that given any pair of states $x_1, x_2 \in X$ and any event $\sigma \in \Sigma$ such that $\delta(x_1, \sigma), \delta(x_2, \sigma)$ are both defined, the mask M is such that $M(x_1) = M(x_2) \Leftrightarrow M(\delta(x_1, \sigma)) = M(\delta(x_2, \sigma))$. Note that this assumption may be violated even when the mask M equals the identity function, which corresponds to the case of complete observation. Thus the observability theory developed in [19] is applicable only to a very small class of systems. We extend the condition of observability of predicates without assuming any restriction on the mask M.

Supervisory Predicate Control and Observation Problem (SPCOP): Consider the plant $G \stackrel{\text{def}}{=} (\mathcal{P}_X, \Sigma, sp, I)$ and the observation mask $M : X \to Y$. Let $R \in \mathcal{P}_X$ denote the required predicate. The control task is to obtain a dynamic control law $D : Y^\star \times (2^\Sigma)^\star \to 2^\Sigma$ such that $(sp_D)^\star(I) = R$.

The notation $Y^\star, (2^\Sigma)^\star$ is used to denote the set of finite sequences of observations in Y, the set of finite sequences of control actions, respectively. The supervisor uses all the information available corresponding to the entire past to determine the current control action (the set of events to be enabled). Thus the supervisor is *dynamic*. The supervisor considered for the SPCOP in [19] is static (current control actions are determined by the current observation only) and can be obtained as a special case of the supervisor considered in this section.

We propose the following algorithm for dynamically estimating the current state of the system using the information available from the entire past. The notation $y_k \in Y$ for each $k \in \mathcal{N}$ is used to denote the observation at the kth step.

Algorithm 5.1

Initiation step: $P_0 = true_X$

Recursion step: $P_{k+1} = sp(P_k) \wedge M^{-1}(y_{k+1})$; $k \geq 0$

where $M^{-1}(y_{k+1})$ corresponds to the predicate which holds in those states which have the same mask value y_{k+1}, and P_k for each $k \geq 0$ denotes the predicate corresponding to the state estimate at the kth step.

Thus initially when no observation is made the set of states corresponding to the initial state estimate equals the entire state space; hence P_0 is set equal to $true_X$. The set of states corresponding to the state estimate at the $(k+1)$th step, where the predicate P_{k+1} holds, equals the set of states that correspond to the observation y_{k+1} and are reachable from a state where P_k holds. Algorithm 5.1 can be used to define the following *dynamic observer* for the system G.

Definition 5.2 Consider the plant $G \stackrel{\text{def}}{=} (\mathcal{P}_X, \Sigma, sp, I)$ and the mask $M : X \to Y$. The *dynamic observer* for estimating the current state of G is a DEDS $O \stackrel{\text{def}}{=} (\mathcal{P}_X, Y, sp_O, true_X)$, where $\mathcal{P}_X$ corresponds to the state set of the observer O; Y corresponds to the event set of O; $true_X$ corresponds to the initial condition of O; and sp_O, the strongest postcondition predicate transformer of O, is defined to be $(sp_O)_y(P) = sp(P) \wedge M^{-1}(y)$ for each $P \in \mathcal{P}_X$ and $y \in Y$.

A similar definition of dynamic observer is presented in [3], which uses the past sequence of observations as well as the past sequence of control inputs for estimating the current state.

5.1 Static and Dynamic Control Laws

An algorithm similar to Algorithm 5.1 can be used to simultaneously observe the evolution of the plant and control its behavior. Since the goal of the SPCOP is to obtain a dynamic control law $D : Y^\star \times (2^\Sigma)^\star \to 2^\Sigma$ such that the required predicate R remains invariant under the evolution of the controlled system, we assume that the system never starts in a state where R does not hold. This is stated as assumption A-1 in the previous section. Keeping assumption A-1 in mind, a dynamic control law D° is obtained in the manner described below. The notations $y_k, P_k^{D^\circ}$ are used to denote the observation at the kth step, and the predicate corresponding to the state estimate at the kth step under the control law D° respectively. The control action at the kth step, $k \geq 1$, depends on the observation sequence up to the kth step and the control sequence up to the $(k-1)$th step; using this available information at the kth step, first the predicate $P_k^{D^\circ}$ corresponding to the state estimate at the kth step is obtained and then an identical control action is defined for each of the states in the set $X_{P_k^{D^\circ}}$. Thus a dynamic controller $D^\circ : Y^\star \times (2^\Sigma)^\star \to 2^\Sigma$ can equivalently be viewed as a map $D^\circ : \mathcal{P}_X \to 2^\Sigma$.

Algorithm 5.3

Initiation step: $P_1^{D^\circ} = M^{-1}(y_1) \wedge I$
$\sigma \in D^\circ(P_1^{D^\circ}) \Leftrightarrow sp_\sigma(P_1^{D^\circ}) \preceq R$

Recursion step: $P_{k+1}^{D^\circ} = sp_S(P_k^{D^\circ}) \wedge M^{-1}(y_{k+1})$
$\sigma \in D^\circ(P_{k+1}^{D^\circ}) \Leftrightarrow sp_\sigma(P_{k+1}^{D^\circ}) \preceq R; k \geq 1$

where $\sigma \in D^\circ(P)$ for any $\sigma \in \Sigma$ and $P \in \mathcal{P}$ means that σ is enabled by the control law D° in every state in the set X_P.

Since the system is assumed to start in $I \preceq R$ (Assumption A-1), the initial state estimate after the first observation $y_1 \in Y$ is given by the predicate $M^{-1}(y_1) \wedge I$. For every state $x \in X_{P_1^{D^\circ}}$, a transition $\sigma \in \Sigma$ is enabled by the control law if and only if the states reached by executing σ in $X_{P_1^{D^\circ}}$ are all contained in X_R. The predicate $P_{k+1}^{D^\circ}$ corresponding to the state estimate at the $(k+1)$th step is obtained by determining the states that correspond to the $(k+1)$th observation y_{k+1}, and that are reachable in G under the control law D° from the states where $P_k^{D^\circ}$ holds. Since all the states in the set $X_{P_{k+1}^{D^\circ}}$ (at step $(k+1)$) correspond to the same past observation and control sequence up to step $(k+1)$, the same control action is applied at all of them; and the controller enables an event $\sigma \in \Sigma$ at all the states $x \in X_{P_{k+1}^{D^\circ}}$ if and only if the states reached by executing σ in the states $X_{P_{k+1}^{D^\circ}}$ are all contained in X_R.

Remark 5.4 It is clear that if the mask M is the identity map (or is injective), then the dynamic control law D° defined in Algorithm 5.3 is the same as the static control law S defined in section 4. In other words, if complete state observation is possible, then static and dynamic control laws are identical. This, as we will see, is not the case when incomplete state observations are made.

Lemma 5.5 Consider the plant G and the mask M. Let G be controlled by the dynamic control law D° described in Algorithm 5.3. Then $\bigvee_{\{y_k\}\in Y^*}\bigvee_{k\geq 1} P_k^{D^\circ} \preceq R$.

Proof: It suffices to show that for any $y_k \in Y^*$, $\bigvee_{k\geq 1} P_k^{D^\circ} \preceq R$. The controller D° is defined recursively in Algorithm 5.3. We use induction on k for obtaining the desired result.

1.	$I \preceq R$	;Assumption A-1
2.	$P_1^{D^\circ} \preceq I$	;initiation step of Algorithm 5.3
3.	$P_1^{D^\circ} \preceq R$	;from 1 and 2 (base case)
4.	$P_k^{D^\circ} \preceq R$	;induction hypothesis
5.	$P_{k+1}^{D^\circ} \preceq R$	;recursion step of Algorithm 5.3 and 3
6.	$\forall k \geq 1, P_k^{D^\circ} \preceq R$	;from 3, 4, 5 and induction
7.	$\bigvee_{k\geq 1} P_k^{D^\circ} \preceq R$	;taking disjunct wrt k in 6

This completes the proof. $\square$

Thus under the dynamic control law described in Algorithm 5.3, the state trajectories of the system remain confined to the states where R holds. In fact, it follows from Lemma 5.5 that $\bigvee_{\{y_k\}\in Y^*}\bigvee_{k\geq 1} P_k^{D^\circ}$ is the weakest predicate stronger than R to which the state trajectories of the partially observed controlled system G are confined. The property, that this weakest predicate stronger than R equals R, is termed *observability* of R.

Definition 5.6 The required predicate R of the partially observed system G under the mask M is said to be *observable* if and only if $\bigvee_{\{y_k\}\in Y^*}\bigvee_{k\geq 1} P_k^{D^\circ} = R$, where $P_k^{D^\circ}$ for each $k \geq 1$ is recursively defined in Algorithm 5.3.

Algorithm 5.3 can be specialized to define a static control law $S : Y \to 2^\Sigma$ in which the control action at any step depends only on the observation at that step:

Algorithm 5.7

Initiation step: $P_1^S = M^{-1}(y_1) \wedge I$

$\sigma \in S(M^{-1}(y_1) \wedge I) \Leftrightarrow sp_\sigma(M^{-1}(y_1) \wedge I) \preceq R$

Recursion step: $P_{k+1}^S = sp_S(P_k^S) \wedge M^{-1}(y_{k+1})$

$\sigma \in S(M^{-1}(y_{k+1})) \Leftrightarrow sp_\sigma(M^{-1}(y_{k+1})) \preceq R; k \geq 1$

Note that the control law S at each step $k \geq 1$ depends only on the kth observation $y_k \in Y$, and an identical control action is applied at each of the states in the set where $M^{-1}(y_1)$ holds (except for the case $k = 1$, where an identical control is applied at each of the states in the set where $M^{-1}(y_1) \wedge I$ holds). Since by definition, $P_k^S \preceq M^{-1}(y_k)$, $S(P_k^S) = S(M^{-1}(y_k))$. Thus the term $sp_S(P_k^S)$ in the recursion step is well defined.

Remark 5.8 It can be proved, similar to Lemma 5.5, that under the control of the static controller S defined in Algorithm 5.7, the state trajectories of the system remain confined to R, i.e. $\bigvee_{\{y_k\}\in Y^*} \bigvee_{k\geq 1} P_k^S \preceq R$. However, the static control law S is more restrictive than the dynamic control law D°, for at every step $k \geq 1$, S depends on $M^{-1}(y_k)$, whereas D° depends on $P_k^{D^\circ}$, and $P_k^{D^\circ} \preceq M^{-1}(y_k)$; as a result, $\bigvee_{k\geq 1} P_k^S \preceq \bigvee_{k\geq 1} P_k^{D^\circ}$. Thus in case of incomplete state observations, a static control law is of course more restrictive than a dynamic control law. In fact, the dynamic control law D° defined in Algorithm 5.3 is the minimally restrictive control law, i.e. if $D' : Y^\star \times (2^\Sigma)^\star \to 2^\Sigma$ is any other control law, then $\bigvee_{k\geq 1} P_k^{D'} \preceq \bigvee_{k\geq 1} P_k^{D^\circ}$.

The dynamic control law D° defined in Algorithm 5.3 and the static control law defined in Algorithm 5.7 can both be implemented by controllers of the type:

Definition 5.9 Consider the plant $G \stackrel{\text{def}}{=} (\mathcal{P}_X, \Sigma, sp, I)$ and the mask $M : X \to Y$. The controller that implements the dynamic control law of Algorithm 5.3 is another DEDS $C \stackrel{\text{def}}{=} (\mathcal{P}_X, Y \times 2^\Sigma, sp_C, I_C)$, where $\mathcal{P}_X$ is the state set of the controller C; $Y \times 2^\Sigma$ is the event set of C; $I_C = M^{-1}(y_1) \wedge I$ is initial condition of C; and sp_C, the strongest postcondition predicate transformer of C, is defined to be $(sp_C)_{(y,\Sigma')}(P) = sp_{\Sigma'}(P) \wedge M^{-1}(y)$ for each $P \in \mathcal{P}_X$, $y \in Y$ and $\Sigma' \subseteq \Sigma$ ($sp_{\Sigma'} = \bigvee_{\sigma\in\Sigma'} sp_\sigma$).

Finally, we present a necessary and sufficient condition under which a solution to SPCOP exists, the proof of which is constructive so that a dynamic control law D that solves SPCOP is automatically obtained.

Theorem 5.10 Consider the partially observed plant G under the mask M. Let R be the required predicate. Then a solution to SPCOP exists if and only if R is controllable and observable.

Proof: Assume first that R is controllable and observable. We will show that there exists a dynamic control law D such that $(sp_D)^\star(I) = R$. Let $D = D^\circ$, where D° is as defined in Algorithm 5.3. Since R is Σ_u-invariant (R is controllable) and $P_k \preceq R$ for each $k \geq 1$ (Lemma 5.5), $D^\circ(= D)$ in Algorithm 5.3 can be rewritten to yield the same dynamic control law:

$$\forall \sigma \in (\Sigma - \Sigma_u), \sigma \in D^\circ(P_k) \Leftrightarrow sp_\sigma(P_k) \preceq R; k \geq 1$$

In other words, $D^\circ(= D)$ never disables any uncontrollable events.

1. $(sp_{D^\circ})^\star(I) = \bigvee_{\{y_k\}\in Y^*} \bigvee_{k\geq 1} P_k^{D^\circ}$;by definition of D°
2. $\bigvee_{\{y_k\}\in Y^*} \bigvee_{k\geq 1} P_k^{D^\circ} = R$;R is observable
3. $(sp_{D^\circ})^\star(I) = R$;from 1 and 2
4. $(sp_D)^\star(I) = R$;from 3 and $D = D^\circ$

Next we show that if there exists a dynamic control law D such that $(sp_D)^\star(I) = R$, then R is controllable and observable.

1.	$(sp_D)^\star(I) = R$	;by assumption
2.	$(sp_D)^\star(I) = \bigvee_{\{y_k\}\in Y^*} \bigvee_{k\geq 1} P_k^D$	; definition of D
3.	$\bigvee_{\{y_k\}\in Y^*} \bigvee_{k\geq 1} P_k^D = R$	;from 1 and 2
4.	$\bigvee_{\{y_k\}\in Y^*} \bigvee_{k\geq 1} P_k^D \preceq \bigvee_{\{y_k\}\in Y^*} \bigvee_{k\geq 1} P_k^{D^\circ}$	;D° is minimally restrictive (Remark 5.8)
5.	$R \preceq \bigvee_{\{y_k\}\in Y^*} \bigvee_{k\geq 1} P_k^{D^\circ}$	;from 3 and 4
6.	$\bigvee_{\{y_k\}\in Y^*} \bigvee_{k\geq 1} P_k^{D^\circ} \preceq R$	;from Lemma 5.5
7.	$\bigvee_{\{y_k\}\in Y^*} \bigvee_{k\geq 1} P_k^{D^\circ} = R$	;from 5 and 6
8.	R is observable	;from 7
9.	$(sp_D)^\star(I) = (sp_D)^\star(R)$	;apply $sp_D^\star$ on 1
10.	$(sp_D)^\star(R) = R$	;from 1 and 9
11.	$sp_u^\star(R) \preceq (sp_D)^\star(R)$	;sp_u is maximally restrictive control
12.	$sp_u^\star(R) \preceq R$	;from 10 and 11
13.	$sp_u(R) \preceq R$	;using Lemma 3.5
14.	$(sp_D \mid_R)^\star(I) = R$	;from 1 and Lemma 4.3
15.	$(sp_D \mid_R)^\star(I) \preceq (sp \mid_R)^\star(I)$	;D restricts behavior
16.	$R \preceq (sp \mid_R)^\star(I)$	;from 14 and 15
17.	R is controllable	;from 13 and 16

This completes the proof. □

Example 5.11 Consider the problem of mutual exclusion discussed in Example 4.6 for the Readers-Writers program of Example 2.10. Assume that the mask $M : X \to Y$ is such that the number of writers always appears to be the same, namely, zero; however, the number of readers can be observed completely, i.e. $M((nr = p, nw = q)) = (nr = p, nw = 0)$ for all $(p, q) \in \mathcal{N}^2$. As discussed in Example 4.6, the mutual exclusion constraint is written as the required predicate $R = ((nw = 0) \vee (nw = 1 \wedge nr = 0))$, and it is controllable. However, R may or may not be observable, depending on the initial condition I.

Case 1: $I = ((nr, nw) = (0, 0))$ as in Example 2.10. Then R is observable, as in this case the number of writers is completely determined by observing the occurrences of events *st_wt* and *end_wt*. Algorithm 5.3 yields the required dynamic supervisor.

Case 2: $I = (nr = 0) \wedge ((nw = 0) \vee (nw = 1))$. In this case the number of writers cannot be fully determined by the past observations, and R is not observable. Hence a dynamic supervisor cannot constructed to solve SPCOP. Under the control of the dynamic supervisor $D^\circ : Y^\star \times (2^\Sigma)^\star \to 2^\Sigma$ as described in Algorithm 5.3, the closed-loop system can only achieve the predicate $(nw = 0)$, i.e. $(sp_{D^\circ})^\star(I) = (nw = 0) \preceq R$.

6 Conclusion

We have presented in this paper a methodology for designing controllers for a wide variety of systems described in terms of a set of predicates and a set of predicate transformers. Predicates can concisely represent an infinite state space, hence many of the discrete event systems including *clocks*, *queues* with unbounded buffers etc. can be modeled in this framework. The above theory can also be useful in synthesizing controller programs for programs describing possibly complex DEDS's. Thus the framework is quite general.

The strongest postcondition transformer has been presented as a fundamental concept for describing the state space evolution of a DEDS. We have presented the notion of duality of predicates and shown that sp and wlp are duals of each other. Many of the basic properties of a predicate transformer have been highlighted, and the relation of these properties to the existence of extremal solutions of some predicate equations has been pointed out. We have shown how these properties and extremal solutions of boolean equations can be applied for supervisory synthesis purposes. The notion of controllability of a required predicate describing the set of legal states has been defined, and it has been shown that controllability is a necessary and sufficient condition for the existence of a supervisor that guarantees the invariance of the required predicate under system evolution. The supervisory predicate control problem has been presented and solved using the notion of controllability. It has further been shown that the weakest controllable predicate stronger than the given predicate exists, and hence the construction of minimally restrictive supervisors is possible in case the required predicate is not controllable. We have presented a method for computing the weakest controllable predicate; this is one of the main results in this paper.

We also address the problem of designing supervisors for a partially observed plant. We introduce the notion of observability which, together with controllability, is a necessary and sufficient condition for the existence of a supervisor that solves the supervisory predicate control and observation problem introduced in this paper.

A Proof of Theorem

Proof: We prove Theorem 3.10 by way of cyclic implication. First we show that C-1 $\Rightarrow$ C-2. Note that since f is disjunctive, the weakest solution in C-1 exists.

1.	$f(g(P)) \preceq P$	;from C-1
2.	conjunct 1 of C-2 holds	;from 1
3.	P is a solution of $Q : f(Q) \preceq f(P)$	;$f(P) \preceq f(P)$
4.	$g(f(P))$ is weakest solution of $Q : f(Q) \preceq f(P)$	;from C-1
5.	$P \preceq g(f(P))$	;from 3 and 4
6.	conjunct 2 of C-2 holds	;from 5

Next we show that C-2 $\Rightarrow$ C-3. This we show by showing that under C-2 the LHS of C-3 implies RHS of C-3 and vice versa.

1.	$f(P) \preceq Q$	;assume LHS of C-3
2.	$g(f(P)) \preceq g(Q)$	;apply g on 1, g is monotone (Lemma 3.3)
3.	$P \preceq g(f(P))$	;conjunct 2 of C-2
4.	$P \preceq g(Q)$	;from 2 and 3
5.	RHS of C-3 holds	;from 4
6.	$P \preceq g(Q)$	;assume RHS of C-3
7.	$f(P) \preceq f(g(Q))$	;apply f on 6, f monotone (Lemma 3.3)
8.	$f(g(Q)) \preceq Q$	;conjunct 1 of C-2 with P replaced by Q
9.	$f(P) \preceq Q$	;from 7 and 8
10.	LHS of C-3 holds	;from 9

Next we show that C-3 $\Rightarrow$ C-4. Since g is conjunctive and P, treated as a constant predicate transformer, is monotone, it follows from Theorem 3.8 that the strongest solution in C-4 exists.

1.	$f(P) \preceq f(P) \Leftrightarrow P \preceq g(f(P))$	;replace Q by $f(P)$ in C-3
2.	$true \Leftrightarrow P \preceq g(f(P))$	;from 1
3.	$f(P)$ a solution in C-4	;from 2
4.	$P \preceq g(Q) \Leftrightarrow f(P) \preceq Q$	;rewriting C-3
5.	$f(P)$ strongest solution in C-4	;from 5

Next we show that C-4 $\Rightarrow$ C-1.

1.	$P \preceq g(Q) \Leftrightarrow f(P) \preceq Q$	;$f(P)$ is the strongest solution in C-4
2.	$g(P) \preceq g(P) \Leftrightarrow f(g(P)) \preceq P$	;replace P, Q by $g(P), P$ respectively in 1
3.	$true \Leftrightarrow f(g(P)) \preceq P$	;simplifying 2
4.	$f(P)$ a solution in C-1	;from 3
5.	$f(P) \preceq Q \Leftrightarrow P \preceq g(Q)$	;rewritting 1
6.	$f(Q) \preceq P \Leftrightarrow Q \preceq g(P)$	;replace P, Q by Q, P respectively in 5
7.	$g(P)$ weakest solution in C-1	;from 6

This completes the proof of Theorem 3.10. □

B Acknowledgement

Authors would like to thank Shigemasa Takai, Department of Electronics Engineering, Osaka University, Japan for pointing out an omission in the definition of observability.

References

[1] A. Arnold and M. Nivat. Controlling behaviors of systems: Some basic concepts and some applications. In *MFSC 1980 (Lecture Notes in Computer Science, 88)*, pages 113–122. Springer-Verlag, New York, 1980.

[2] R. D. Brandt, V. K. Garg, R. Kumar, F. Lin, S. I. Marcus, and W. M. Wonham. Formulas for calculating supremal controllable and normal sublanguages. *Systems and Control Letters*, 15(8):111–117, 1990.

[3] P. E. Caines, R. Greiner, and S. Wang. Dynamical logic observers for finite automaton. In *Proceedings of the 1988 Conference on Decision and Control*, pages 226–233, Austin, Texas, December 1988.

[4] K. M. Chandy and J. Misra. *Parallel Program Design: A Foundation.* Addison-Wesley, Reading, MA, 1988.

[5] E. W. Dijkstra. *A Discipline of Programming.* Prentice Hall, Inc., Englewood Cliffs, NJ, 1976.

[6] E. W. Dijkstra and C. S. Scholten. *Predicate Calculus and Program Semantics*. Springer-Verlag, New York, 1990.

[7] V. K. Garg and R. Kumar. State-variable approach for controlling discrete event systems with infinite states. In *Proceedings of 1992 American Control Conference*, pages 2809–2813, Chicago, IL, July 1992.

[8] D. Gries. *The Science Of Programming.* Springer-Verlag, New York, NY, 1985.

[9] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice Hall, Inc., Englewood Cliffs, NJ, 1985.

[10] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, Reading, MA, 1979.

[11] R. Kumar, V. K. Garg, and S. I. Marcus. On controllability and normality of discrete event dynamical systems. *Systems and Control Letters*, 17(3):157–168, 1991.

[12] R. Kumar, V. K. Garg, and S. I. Marcus. On ω-controllability and ω-normality of deds. In *Proceedings of 1991 ACC*, pages 2905–2910, Boston, MA, June 1991.

[13] R. Kumar, V. K. Garg, and S. I. Marcus. Stability of discrete event system behavior. In *Proceedings of 1991 IFAC Symposium on Distributed Intelligent Systems*, pages 13–18, August 1991.

[14] R. Kumar, V. K. Garg, and S. I. Marcus. Using predicate transformers for supervisory control. In *Proceedings of 1991 IEEE Conference on Decision and Control*, pages 98–103, Brighton, UK, December 1991.

[15] R. Kumar, V. K. Garg, and S. I. Marcus. On supervisory control of sequential behaviors. *IEEE Transactions on Automatic Control*, 37(12):1978–1985, December 1992.

[16] R. Kumar, V. K. Garg, and S. I. Marcus. Language stability and stabilizability of discrete event dynamical systems. *SIAM Journal of Control and Optimization*, 31(5):1294–1320, September 1993.

[17] R. Kumar and L. E. Holloway. Supervisory control of Petri net languages. In *Proceedings of 1992 IEEE Conference on Decision and Control*, pages 1190–1195, Tucson, AZ, December 1992.

[18] S. Lam and A. U. Shankar. Refinement and projection of relational specifications. In *Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness*, New York, NY, May 1989. REX Workshop, Springer-Verlag.

[19] Y. Li and W. M. Wonham. Controllability and observability in the state feedback control of discrete event systems. In *Proceedings of the 27th CDC*, pages 203–208, Austin, Texas, December 1988.

[20] F. Lin and W. M. Wonham. On observability of discrete-event systems. *Information Sciences*, 44(3):173–198, 1988.

[21] P. J. Ramadge. Observability of discrete event systems. In *Proceedings of 1986 IEEE Conference on Decision and Control*, pages 1108–1112, Athens, Greece, December 1986.

[22] P. J. Ramadge and W. M. Wonham. Modular feedback logic for discrete event systems. *SIAM Journal of Control and Optimization*, 25(5):1202–1218, 1987.

[23] 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.

[24] 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.

[25] R. Smedinga. Using trace theory to model discrete events. In P. Varaiya and A. B. Kurzhanski, editors, *Discrete Event Systems: Models and Applications*, pages 81–99. Springer-Verlag, 1987.

[26] R. S. Sreenivas and B. H. Krogh. On Petri net models of infinite state supervisors. *IEEE Transactions of Automatic Control*, 37(2):274–277, February 1992.