Math. Control Signals Systems (1989) 2: 47–69

Mathematics of Control, Signals, and Systems

On Supremal Languages of Classes of Sublanguages that Arise in Supervisor Synthesis Problems with Partial Observation*

Hangju Cho† and Steven I. Marcus†

Abstract. This paper characterizes the class of closed and (M, N)-recognizable languages in terms of certain structural aspects of relevant automata. This characterization leads to algorithms that effectively compute the supremal (M, N)-recognizable sublanguage of a given language. One of these algorithms is used, in an alternating manner with an algorithm which yields the supremal (Σ_u, N)-invariant sublanguage, to compute the supremal sublanguage of a given language that is both (Σ_u, N)-invariant and (M, N)-recognizable. Finite convergence of the resulting algorithm is proved. An example illustrates the use of these algorithms.

Key words. Discrete-event systems, Supervisor synthesis, Languages, automata, Partial observations.

1. Introduction

Many complex man-made dynamical systems evolve not according to differential equations, but according to the intricate interaction of discrete events. In this paper we consider such discrete-event dynamical systems, examples of which are flexible manufacturing systems and computer/communication networks. In these examples the discrete events are the completion of a task or the arrival of a message. The state of the system, which changes only at asynchronous discrete instants of time, consists of numbers and discrete variables, such as the number of parts waiting at each station and the readiness status of each station. Models and control algorithms for such systems have been developed by Ramadge and Wonham [RW], [WR1], [WR2] (see also Smedinga [S]); in this work, a dynamical system (or plant) is modeled by a finite automaton with certain controllable events, the occurrence of which can be disabled by means of control action. The control task is formulated as that of synthesizing a controller (called a supervisor), which can observe each occurrence of events in the plant, in such a way that the behavior of the closed-loop

* Date received: June 1, 1987. Date revised: December 3, 1987. This research was supported in part by the Air Force Office of Scientific Research under Grant No. AFOSR-86-0029, in part by the National Science Foundation under Grant No. ECS-8412100, and in part by the DoD Joint Services Electronics Program through the Air Force Office of Scientific Research (AFSC) Contract No. F49620-86-C-0045.

† Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, Texas 78712-1084, U.S.A.

system is minimally restrictive while being confined to a prespecified language (i.e., prespecified set of strings of events). The reader is assumed to be familiar with the work of Ramadge and Wonham.

Recently, two papers [CDFV], [LW] have considered, in the framework of Ramadge and Wonham, supervisor synthesis problems of discrete-event processes in which the observation of events is assumed to be imperfect. They gave necessary and sufficient conditions for the language to be realized by means of constructing suitable supervisors.

It turns out, however, that the classes of sublanguages of a given language L which satisfy the necessary and sufficient conditions do not, in general, possess supremal elements, and therefore the minimally restrictive solutions may not exist. On the other hand, the classes of sublanguages of L where the closure of each element is (Σ_u, N)-invariant and (M, N)-recognizable [CDFV] (or equivalently, controllable and normal with respect to N and M in [LW], if we restrict ourselves to the case where M is a projection) are not only subsets of the classes of languages mentioned above, but are algebraically well-behaved so that they possess supremal elements. Thus the consideration of the supremal elements of the latter classes of sublanguages has been appreciated in [CDFV] and [LW] as a convenient way to obtain minimally restrictive solutions in smaller classes or to investigate the existence of the solution to the problems. The computational aspects of the aforementioned supremal sublanguages, however, have not yet been fully developed: an algorithm to compute the supremal sublanguage that solves the (restricted) supervisor synthesis problem of [CDFV] has been presented in [CDFV] under the assumption that $M^{-1}[M(\Sigma_u)] \subset \Sigma_u \cup \{\varepsilon\}$.

We present in this paper an algorithm that is shown to compute effectively the supremal sublanguage considered in [CDFV] without the above restriction. In addition, our algorithm possesses a simpler and more graphical structure than that of [CDFV]; this structure enhances understanding of the problem and is well-suited to computer implementation. Moreover, our results are extended to solve the (restricted) Supervisory Marking Problem [RW] where L is not necessarily closed. In the following we give definitions of various objects that are necessary for the subsequent development, and specifically state the problem studied in the paper.

Let Σ be a (nonempty) finite set of events, called an alphabet, and let $\Sigma_u \subset \Sigma$ (Σ_u represents the set of uncontrollable events). The set of all finite strings of events in Σ is denoted by Σ^*. The empty string is denoted by ε. A subset $O \subset \Sigma^*$ is called a language over Σ. The closure of O, denoted by $\bar{O}$, is the set of strings that are prefixes of strings in O. If $O = \bar{O}$, O is said to be closed. Also, for any $s \in \Sigma^*$, we denote by $|s|$ the length of string and by $\bar{s}$ the set of strings that are prefixes of s. O is regular if O is the language $L_m(G)$ marked by some finite automaton G (see the definitions in Section 2). Let Δ be a finite set, and let $M: \Sigma \to \Delta \cup \{\varepsilon\}$ be a function; M is a *mask* or observation function, which will represent partial observations by the supervisor of events in Σ. Then M can be extended to Σ^* in a natural way; i.e., $M(\varepsilon) = \varepsilon$ and $M(s\sigma) = M(s)M(\sigma)$ for all $s \in \Sigma^*$, $\sigma \in \Sigma$ (this function was introduced in [CDFV] to model imperfect observations, and was specialized to a projection in [LW]). For $O \subset N \subset \Sigma^*$, O is (M, N)-recognizable if $N \cap M^{-1}[M(O)] = O$. Also O is (Σ_u, N)-invariant if $O\Sigma_u \cap N \subset O$.

Now let $L \subset N \subset \Sigma^*$. We assume that L and N are regular and closed. The assumption that L is closed will be dropped later in Section 5 when we discuss an application to Supervisory Marking Problems. Consider the classes of sublanguages of L defined as follows:

$$\underline{R}(L) = \{O \subset L | O \text{ is closed and } (M, N)\text{-recognizable}\},$$

$$\underline{D}(L) = \{O \subset L | O \text{ is closed, } (\Sigma_u, N)\text{-invariant, and } (M, N)\text{-recognizable}\}.$$

It was shown in [CDFV] that the supremal elements of $\underline{R}(L)$ and $\underline{D}(L)$ (denoted by $Sup\ \underline{R}(L)$ and $Sup\ \underline{D}(L)$, respectively) exist and that $Sup\ \underline{D}(L)$ solves the (restricted) supervisor synthesis problem posed in [CDFV]. That is, if G denotes a finite automaton (or plant), $N = L(G)$ denotes the language generated by G, G_c denotes the corresponding controlled discrete-event process [RW], $\underline{S}$ denotes a supervisor, $L(\underline{S}/G_c)$ denotes the set of sequences of events that can occur when $\underline{S}$ is coupled to G_c, then the (restricted) supervisor synthesis problem of [CDFV] is that of finding, for $L \subset L(G)$, a complete supervisor $\underline{S}$ such that $L(\underline{S}/G_c)$ is the largest (M, N)-recognizable language contained in L.

Our objective is to find an effective way to compute $Sup\ \underline{D}(L)$. For this purpose, we give a characterization of (M, N)-recognizable languages in Section 2, suggest two algorithms that compute $Sup\ \underline{R}(L)$ in Section 3, and present an algorithm to compute $Sup\ \underline{D}(L)$ in Section 4. Also, in Section 5, we discuss a method that computes $Sup\ \underline{R}'(L)$ and $Sup\ \underline{D}'(L)$ using these algorithms, where L is not necessarily closed and

$$\underline{R}'(L) := \{O \subset L | \bar{O} \text{ is } (M, N)\text{-recognizable}\},$$

$$\underline{D}'(L) := \{O \subset L | \bar{O} \text{ is } (\Sigma_u, N)\text{-invariant and } (M, N)\text{-recognizable}\}.$$

We note that if L is closed, then $Sup\ \underline{R}'(L) = Sup\ \underline{R}(L)$ and $Sup\ \underline{D}'(L) = Sup\ \underline{D}(L)$.

2. Strict-Subautomata and a Characterization of (M, N)-Recognizable Languages

A Nondeterministic Finite Automaton (NFA) G is a 5-tuple $G = (Q, \Sigma, f, q_0, Q_m)$, where Q is a finite set of states, Σ is an alphabet, $q_0 \in Q$ is the initial state, $Q_m \subset Q$ is the set of final (marked) states, and $f: \Sigma \times Q \to P(Q)$ is a transition function (where $P(Q)$ is the power set of Q). f is extended to $\Sigma^* \times Q$ in the standard way (see, e.g., [HU]), and the extension is denoted by the same symbol f. For $\sigma \in \Sigma$ and $q \in Q$, we say that $f(\sigma, q)$ is defined and write $f(\sigma, q)!$, if $f(\sigma, q)$ is nonempty. G is said to be a Deterministic Finite Automaton (DFA) if for any $\sigma \in \Sigma$, $q \in Q$, $f(\sigma, q)$ is a singleton set $\{r\}$, $r \in Q$ whenever $f(\sigma, q)!$. In this case we write $f(\sigma, q) = r$ to mean that $f(\sigma, q) = \{r\}$. We denote by Φ the empty automaton which has as its set of states the empty set. $L(G)$, the language generated by G, and $L_m(G)$, the language marked by G, are defined by

$$L(G) := \{w \in \Sigma^* | f(w, q_0)!\},$$

$$L_m(G) := \{w \in \Sigma^* | \text{there exists } q \in Q_m \text{ such that } q \in f(w, q_0)\}.$$

Clearly, $L(\Phi) = \varnothing$. We often write $L_m(G)$ as $|G|$. Also, we note that $L(G)$ is closed. By $Tr\ G$, we mean the trim component of G defined by

$$Tr\ G := \begin{cases} (Q_{tr}, \Sigma, f_{tr}, q_0, Q_m \cap Q_{tr}) & \text{if } Q_{tr} \neq \varnothing, \\ \Phi & \text{otherwise,} \end{cases}$$

where $Q_{tr} := \{q \in Q | \text{there exist } s, t \in \Sigma^*, r \in Q_m \text{ such that } q \in f(s, q_0) \text{ and } r \in f(t, q)\}$ and $f_{tr} := f|_{\Sigma \times Q_{tr}}$. Clearly, $|Tr\ G| = |G|$ and $Tr\ G$ is effectively constructible [E]. G is said to be trim if $G = Tr\ G$. In this case, $\overline{|G|} = L(G)$. In this paper all automata are assumed to be trim unless otherwise stated.

Consider two NFAs $A = (Q_A, \Sigma, f_A, q_{A0}, Q_{Am})$ and $B = (Q_B, \Sigma, f_B, q_{B0}, Q_{Bm})$ with $|B| \subset |A|$. We say that B is a *subautomaton* of A if

(i) $f_B(s, q_{B0}) = f_A(s, q_{A0})$ for all $s \in \overline{|B|}$.

Also, we say that B is a *strict-subautomaton* of A if, in addition to (i),

(ii) if $s \in \overline{|A|}$ and $s \notin \overline{|B|}$, there exist $\tilde{s} \in \bar{s}$ such that $f_A(\tilde{s}, q_{A0}) \notin Q_B$.

Note that Φ is a strict-subautomaton of any NFA. If A and B are deterministic, a detailed description of a subautomaton can be given as follows. Consider the conditions

(iii) $Q_B \subset Q_A$, $q_{A0} = q_{B0}$ and
(iv) for all $q \in Q_B$, $\sigma \in \Sigma$, $f_B(\sigma, q) = f_A(\sigma, q)$ whenever $f_B(\sigma, q)!$.

Then it is easy to check

Lemma 2.1. *Let A and B be deterministic. Then B is a subautomaton of A if and only if the conditions* (iii) *and* (iv) *hold.*

Remark 2.1. Given two NFAs C_1 and C_2 with $|C_2| \subset |C_1|$, we can always obtain two DFAs A and B with $|A| = |C_1|$, $|B| = |C_2|$ such that B is a strict-subautomaton of A. For example, first construct two DFAs $D_1 = (Q_1, \Sigma, f_1, q_1^0, Q_{1m})$ and $D_2 = (Q_2, \Sigma, f_2, q_2^0, Q_{2m})$ such that $|D_1| = |C_1|$ and $|D_2| = |C_2|$ (the existence of such DFAs and the procedure for obtaining them are well known; see, e.g., [HU]); then define $A := Tr\ (Q_A, \Sigma, f_A, q_0, Q_{Am})$ and $B := Tr\ (Q_B, \Sigma, f_B, q_0, Q_{Bm})$ by

$$Q_A := Q_1 \times (Q_2 \cup \{q_s\}), \quad q_s \notin Q_2, \qquad Q_{Am} := Q_{1m} \times (Q_2 \cup \{q_s\}), \quad q_0 := (q_1^0, q_2^0),$$

$$Q_B := Q_1 \times Q_2, \qquad Q_{Bm} := Q_{1m} \times Q_{2m},$$

$$f_A(\sigma, (q_1, q_2)) := \begin{cases} \{(r_1, r_2) | r_1 \in f_1(\sigma, q_1), r_2 \in f_2(\sigma, q_2)\} & \text{if } q_2 \in Q_2 \text{ and } f_2(\sigma, q_2)!, \\ \{(r_1, q_s) | r_1 \in f_1(\sigma, q_1)\} & \text{otherwise,} \end{cases}$$

$$f_B(\sigma, (q_1, q_2)) := \{(r_1, r_2) | r_1 \in f_1(\sigma, q_1), r_2 \in f_2(\sigma, q_2)\}.$$

Now let $L \subset N \subset \Sigma^*$ be regular and closed, and let $G := (Q, \Sigma, f, q_0, Q)$ and $G_s := (Q_s, \Sigma, f_s, q_0, Q_s)$ be NFAs such that $|G| = N$, $|G_s| = L$, and G_s is a strict-subautomaton of G. The following equivalent statements are convenient and suggestive for the subsequent development.

Lemma 2.2.

$$
\begin{aligned}
&L \text{ is } (M, N)\text{-recognizable} \\
&\Leftrightarrow \text{ for all } s \in N, t \in L, \quad M(s) = M(t) \text{ implies } s \in L, \\
&\Leftrightarrow \text{ for all } d \in M(L), \quad \{s \in N \mid M(s) = d\} \subset L, \\
&\Leftrightarrow \text{ for all } d \in M(L), \quad \{s \in N \mid M(s) = d\} = \{s \in L \mid M(s) = d\}.
\end{aligned}
$$

Thus, if there were a way of obtaining two sets of strings $\{s \in N \mid M(s) = d\}$ and $\{s \in L \mid M(s) = d\}$ for each $d \in M(L)$, then it would become an easy task to check if L is (M, N)-recognizable. This observation motivates the following (standard) construction of two DFAs T and T_s that generate $M(N)$ and $M(L)$, respectively.

Let $T := (X, \Delta, h, x_0, X)$ and $T_s := (X_s, \Delta, h_s, x_{s0}, X_s)$ be defined as follows:

$$
\begin{aligned}
X &:= P(Q) - \{\varnothing\}, \qquad X_s := P(Q_s) - \{\varnothing\}, \\
x_0 &:= \{q \in f(s, q_0) \mid M(s) = \varepsilon\}, \qquad x_{s0} := \{q \in f_s(s, q_0) \mid M(s) = \varepsilon\}, \\
h(\delta, x) &:= \{q \in f(s, q') \mid q' \in x \text{ and } M(s) = \delta\}, \\
h_s(\delta, x_s) &:= \{q \in f_s(s, q') \mid q' \in x_s \text{ and } M(s) = \delta\}.
\end{aligned}
$$

Then it follows from a simple induction argument that, for all $d \in \Delta^*$,

$$
\begin{aligned}
h(d, x_0) &= \{q \in f(s, q_0) \mid M(s) = d\}, \\
h_s(d, x_{s0}) &= \{q \in f_s(s, q_0) \mid M(s) = d\}.
\end{aligned}
$$

Since G_s is a subautomaton of G, it is clear from the construction that $h_s(d, x_{s0}) \subset h(d, x_0)$ for all $d \in |T_s|$. From now on we assume that T and T_s are trim; i.e., we identify T and T_s with their trim components.

Lemma 2.3. *Let $d \in |T|$. Consider the following statements:*

(1) $h_s(d, x_{s0}) \neq h(d, x_0)$.
(2) $\{s \in |G| \mid M(s) = d\} \not\subset |G_s|$.
(3) *There exists $\tilde{d} \in \bar{d}$ such that $h(\tilde{d}, x_0) \notin X_s$.*

Then (1) $\Rightarrow$ (2) $\Rightarrow$ (3).

Proof. (1) $\Rightarrow$ (2) Suppose that $\{s \in |G| \mid M(s) = d\} \subset |G_s|$. Since G_s is a subautomaton of G, this implies that $f(s, q_0) = f_s(s, q_0)$ for all $s \in |G|$ whenever $M(s) = d$. Thus $h(d, x_0) = \{q \in f(s, q_0) \mid M(s) = d\} = \{q \in f_s(s, q_0) \mid M(s) = d\} = h_s(d, x_{s0})$, which contradicts the assumption.

(2) $\Rightarrow$ (3) Let $s \in |G|$ be such that $M(s) = d$ and $s \notin |G_s|$. By the definition of strict-subautomaton, there is $\tilde{s} \in \bar{s}$ such that $f(\tilde{s}, q_0) \notin Q_s$. Let $\tilde{d} = M(\tilde{s})$, then $\tilde{d} \in \bar{d}$ and $f(\tilde{s}, q_0) \subset h(\tilde{d}, x_0)$. Noting that $X_s \subset P(Q_s)$, we have that $h(\tilde{d}, x_0) \notin X_s$. ■

Corollary 2.1. *Let $d \in |T|$. If $h_s(\tilde{d}, x_{s0}) = h(\tilde{d}, x_0)$ for all $\tilde{d} \in \bar{d}$, then $\{s \in |G| \mid M(s) = d\} \subset |G_s|$.*

Proof. Suppose that $\{s \in |G| | M(s) = d\} \not\subset |G_s|$. By Lemma 2.3, there exists $\tilde{d} \in \bar{d}$ such that $h(\tilde{d}, x_0) \notin X_s$. Note that $\tilde{d} \in |T|$ since $|T|$ is closed. Thus $h(\tilde{d}, x_0)$ is non-empty. Hence $h(\tilde{d}, x_0) \neq h_s(\tilde{d}, x_{s0})$, which contradicts the assumption. ■

Now we are ready to establish a characterization of (M, N)-recognizable languages.

Theorem 2.1. *L is (M, N)-recognizable if and only if T_s is a subautomaton of T. In this case, T_s is also a strict-subautomaton of T.*

Proof. Note that $|T_s|$ is closed, so that if $d \in |T_s|$, then $\tilde{d} \in |T_s|$ for all $\tilde{d} \in \bar{d}$. Thus

T_s is a subautomaton of T

$\Leftrightarrow$ for all $d \in |T_s|$, $h_s(d, x_{s0}) = h(d, x_0)$ (definition of subautomaton),

$\Leftrightarrow$ for all $d \in |T_s|$, $\{s \in |G| | M(s) = d\} \subset |G_s|$ (Corollary 2.1 and Lemma 2.3),

$\Leftrightarrow$ L is (M, N)-recognizable (Lemma 2.2).

Also, it is obvious from the Lemma 2.3 and the definition of strict-subautomaton that if T_s is a subautomaton of T, then it is also a strict-subautomaton of T. ■

3. Algorithms for *Sup* $\underline{R}(L)$

In this section two algorithms which compute *Sup* $\underline{R}(L)$ are presented. We assume that $L \subset N \subset \Sigma^*$, and L, N are regular and closed. We first give an equivalent description of *Sup* $\underline{R}(L)$.

Let $K := \{s \in L | N \cap M^{-1}[M(\bar{s})] \subset L\}$. Then we have

Lemma 3.1. $K = Sup\ \underline{R}(L)$.

Proof. First we show that $K \in \underline{R}(L)$. Clearly, $K \subset L$ and $K = \bar{K}$. Thus it remains to prove that K is (M, N)-recognizable. Suppose now that $s \in K$, $t \in N$, and $M(s) = M(t)$. Then $N \cap M^{-1}[M(\bar{t})] = N \cap M^{-1}[M(\bar{s})] \subset L$. Note that $t \in N \cap M^{-1}[M(\bar{t})]$, so $t \in L$. Hence, from the definition of K, $t \in K$. By Lemma 2.2, K is (M, N)-recognizable.

Next, let $O \in \underline{R}(L)$ and let $s \in O$. Then $s \in L$. Also, $\bar{s} \subset O$ since O is closed. Thus $N \cap M^{-1}[M(\bar{s})] \subset N \cap M^{-1}[M(O)] = O \subset L$. Hence $s \in K$, and therefore $O \subset K$. ■

Theorem 2.1 motivates the consideration of the following algorithm to obtain *Sup* $\underline{R}(L)$.

Algorithm A (*Sup* $\underline{R}(L)$). Given NFAs G and G_s, G_s a strict-subautomaton of G, with $|G| = N$ and $|G_s| = L$:

Step 1. Construct two DFAs $T := (X, \Delta, h, x_0, X)$ and $T_s := (X_s, \Delta, h_s, x_{s0}, X_s)$ as illustrated in Section 2.

Step 2. Obtain a DFA $\tilde{T}_s$ by eliminating states and/or edges of T_s so that $\tilde{T}_s$ is the largest subautomaton of T that is also a subautomaton of T_s; more specifi-

cally, if $x_0 \neq x_{s0}$, then $\tilde{T}_s := \Phi$. If $x_0 = x_{s0}$, $\tilde{T}_s := Tr(\tilde{X}_s, \Delta, \tilde{h}_s, x_0, \tilde{X}_s)$ is defined by

(A) $\tilde{X}_s := X_s \cap X$.

(B) For all $x \in \tilde{X}_s$, $\delta \in \Delta$, $\tilde{h}_s(\delta, x)!$ if and only if $h_s(\delta, x)!$ and $h_s(\delta, x) = h(\delta, x)$. In this case, $\tilde{h}_s(\delta, x) := h_s(\delta, x)$.

Step 3. Compute $\tilde{K} = L \cap M^{-1}[|\tilde{T}_s|]$. ($\tilde{K}$ can also be effectively computed; the detailed procedure and some related material will be discussed later.)

From the definition of $\tilde{T}_s$, it is clear that $|\tilde{T}_s|$ is closed and that $\tilde{T}_s$ is a subautomaton of T and of T_s as claimed. The following lemma can be proved by a simple induction argument.

Lemma 3.2. *Let $d \in |T|$. Then $d \in |\tilde{T}_s|$ if and only if $h_s(\tilde{d}, x_{s0}) = h(\tilde{d}, x_0)$ for all $\tilde{d} \in \bar{d}$. In this case, $\tilde{h}_s(\tilde{d}, x_0) = h_s(\tilde{d}, x_{s0})$ for all $\tilde{d} \in \bar{d}$.*

Corollary 3.1. *$\tilde{T}_s$ is a strict-subautomaton of T.*

Proof. It suffices to show that the condition (ii) holds. Let $d \in |T|$ and $d \notin |\tilde{T}_s|$. Then, by Lemma 3.2, there is $d' \in \bar{d}$ such that $h_s(d', x_{s0}) \neq h(d', x_0)$. Since $\tilde{d}' \in \bar{d}$ for all $\tilde{d}' \in \bar{d}'$, (ii) holds by Lemma 2.3. ∎

Also, we have, from Lemma 3.2 and Corollary 2.1,

Corollary 3.2. *If $d \in |\tilde{T}_s|$, then $\{s \in N | M(s) = d\} \subset L$.*

The following theorem states that the above algorithm indeed yields $Sup\ \underline{R}(L)$.

Theorem 3.1. *$\tilde{K} = Sup\ \underline{R}(L)$.*

Proof. It suffices to show that $\tilde{K} = K$. Let $s \in \tilde{K}$. Then $s \in L$ and $M(s) \in |\tilde{T}_s|$. Now suppose that $t \in N \cap M^{-1}[M(\bar{s})]$. Then $t \in N$ and $M(t) = d$ for some $d \in M(\bar{s}) = \overline{M(s)}$. Since $|\tilde{T}_s|$ is closed, $d \in |\tilde{T}_s|$. By Corollary 3.2, $t \in L$. Thus $s \in K$, and therefore $\tilde{K} \subset K$.

Conversely, if $s \in K$, then $s \in L$ and $N \cap M^{-1}[M(\bar{s})] \subset L$. Let $d = M(s)$. Then $d \in |T_s|$. Note that

$$N \cap M^{-1}[M(\bar{s})] \subset L \text{ if and only if } \{w \in |G| | M(w) = \tilde{d}\} \subset |G_s| \quad \text{for all } \tilde{d} \in \bar{d}.$$

Thus if $f(w, q_0) \subset h(\tilde{d}, x_0)$, $\tilde{d} \in \bar{d}$, then $M(w) = \tilde{d}$ and $w \in |G|$, and therefore $w \in |G_s|$ from the above observation. Since G_s is a subautomaton of G, this in turn implies that $f(w, q_0) = f_s(w, q_0) \subset h_s(\tilde{d}, x_{s0})$. Hence $h(\tilde{d}, x_0) \subset h_s(\tilde{d}, x_{s0})$ for all $\tilde{d} \in \bar{d}$. Since the reverse inclusion is always true, it follows that $h(\tilde{d}, x_0) = h_s(\tilde{d}, x_{s0})$ for all $\tilde{d} \in \bar{d}$. By Lemma 3.2, $d \in |\tilde{T}_s|$. Thus we have shown that $s \in \tilde{K}$, and therefore that $K \subset \tilde{K}$. ∎

Now we give a method of constructing an NFA V_s that generates the language $L \cap M^{-1}[|\tilde{T}_s|]$. Assume that $\tilde{T}_s \neq \Phi$ (i.e., $x_0 = x_{s0}$). For later purposes, we also

construct an NFA V that generates the language N. Thus we let NFAs $V := (Y, \Sigma, g, y_0, Y)$ and $V_s := Tr(Y_s, \Sigma, g_s, y_0, Y_s)$ be defined by:

$$Y := X \times Q, \qquad Y_s := \tilde{X}_s \times Q_s, \qquad y_0 := (x_0, q_0),$$

$$g(\sigma, (x, q)) := \{(x', q') | x' = h(M(\sigma), x), q' \in f(\sigma, q)\},$$

$$g_s(\sigma, (x_s, q)) := \{(x', q') | x' = \tilde{h}_s(M(\sigma), x_s), q' \in f_s(\sigma, q)\}.$$

Then it is straightforward to check

Lemma 3.3.

(i) $g(s, y_0) = \{(x, q) | x = h(M(s), x_0), q \in f(s, q_0)\}$,
$g_s(s, y_0) = \{(x, q) | x = \tilde{h}_s(M(s), x_0), q \in f_s(s, q_0)\}$.
(ii) $|V| = |G|$ *and* $|V_s| = L \cap M^{-1}[|\tilde{T}_s|]$.
(iii) V_s *is a strict-subautomaton of* V.

Remark 3.1. Since all procedures involved in each step are clearly effectively computable, we have, in fact, given an effectively computable algorithm to obtain $Sup\ \underline{R}(L)$.

A more efficient algorithm can be obtained when a pair of NFAs equipped with stronger structural properties are available from the beginning. Let $G = (Q, \Sigma, f, q_0, Q)$ be an NFA. For $d \in M(|G|)$, we define a set $Q_G(d)$ by

$$Q_G(d) := \{q \in Q | q \in f(s, q_0), M(s) = d\}.$$

Thus, if we consider $T = (X, \Delta, h, x_0, X)$ to be a DFA constructed from G as in Section 2, then $Q_G(d)$ is precisely equal to a state $h(d, x_0)$ of T whenever $Q_G(d)$ is nonempty. Now we say that G is *M-recognizable* if for all $d_1, d_2 \in M(|G|)$, $Q_G(d_1) \cap Q_G(d_2) = \varnothing$ whenever $Q_G(d_1) \neq Q_G(d_2)$. Thus the $Q_G(d)$'s, in this case, form equivalence classes in Q. Also note that it is always possible to get an M-recognizable NFA from a given NFA, even though the condition looks difficult to satisfy. For example, the NFA V constructed in the previous section is M-recognizable, which will be proved in the following.

Lemma 3.4. *V is M-recognizable.*

Proof. Using Lemma 3.3, we have that, for $d \in M(|V|)$,

$$Q_V(d) = \{(x, q) | x = h(d, x_0), q \in f(s, q_0) \text{ where } M(s) = d\}.$$

Now suppose that $d_1, d_2 \in M(|V|)$ and $Q_V(d_1) \neq Q_V(d_2)$. Then either $h(d_1, x_0) \neq h(d_2, x_0)$ or $Q_G(d_1) = \{q \in Q | q \in f(s, q_0), M(s) = d_1\} \neq \{q \in Q | q \in f(t, q_0), M(t) = d_2\} = Q_G(d_2)$. Note from the definition of h that $Q_G(d) = h(d, x_0)$ for all $d \in M(|G|) = M(|V|)$. Thus either case implies $h(d_1, x_0) \neq h(d_2, x_0)$. Hence $Q_V(d_1) \cap Q_V(d_2) = \varnothing$. ∎

Consider again the finite automata G, G_s, T, T_s, and $\tilde{T}_s$ discussed before. Here, we make an additional assumption that G is M-recognizable. We demonstrate a method

for obtaining an NFA $\tilde{G}_s$ which is constructible directly from G, G_s, and T, and which generates the language $\tilde{K} = L \cap M^{-1}[|\tilde{T}_s|]$. We begin with the following lemma.

Lemma 3.5. *Let $d \in |\tilde{T}_s|$ and $q \in Q_G(d)$. If there exists $d' \in |\tilde{T}_s|$ and $\delta \in \Delta$ such that $q \in h_s(d'\delta, x_{s0})$, then $d'\delta \in |\tilde{T}_s|$.*

Proof. Note that $q \in Q_G(d'\delta)$ since $h_s(d'\delta, x_{s0}) \subset h(d'\delta, x_0)$. Thus $Q_G(d) = Q_G(d'\delta)$ by the assumption that G is M-recognizable. Now suppose to the contrary that $d'\delta \notin |\tilde{T}_s|$. It follows from Lemma 3.2 and Lemma 2.3 that there is $\tilde{d} \in \overline{d'\delta}$ such that $h(\tilde{d}, x_0) \notin X_s$. But $d' \in |\tilde{T}_s|$. Thus $\tilde{d} = d'\delta$. Consequently, $h(d'\delta, x_0) \notin X_s$, and therefore $h(d, x_0) \notin X_s$ since $h(d, x_0) = Q_G(d) = Q_G(d'\delta) = h(d'\delta, x_0)$. Hence $h(d, x_0) \neq h_s(d, x_{s0})$. By Lemma 3.2, $d \notin |\tilde{T}_s|$ which contradicts the assumption. Thus it must be the case that $d'\delta \in |\tilde{T}_s|$. ■

Now we define an NFA $\tilde{G}_s := Tr(\tilde{Q}_s, \Sigma, \tilde{f}_s, q_0, \tilde{Q}_s)$, $\tilde{Q}_s \subset Q_s$, by:

(a) $q \in \tilde{Q}_s$ if and only if there exists $d \in |\tilde{T}_s|$ such that $q \in Q_G(d)$.
(b) For all $q \in \tilde{Q}_s$ and $\sigma \in \Sigma$, $\tilde{f}_s(\sigma, q)!$ if and only if $f_s(\sigma, q)!$ and $f_s(\sigma, q) \cap \tilde{Q}_s \neq \varnothing$. In this case, $\tilde{f}_s(\sigma, q) := f_s(\sigma, q)$.

If $q_0 \notin \tilde{Q}_s$ by rule (a), then $\tilde{G}_s := \Phi$. Note that $q_0 \in \tilde{Q}_s$ if and only if $\tilde{T}_s \neq \Phi$.

Lemma 3.6. *For all $w \in \Sigma^*$, $\tilde{f}_s(w, q_0)!$ if and only if $f_s(w, q_0)!$ and $M(w) \in |\tilde{T}_s|$. In this case, $\tilde{f}_s(w, q_0) = f_s(w, q_0)$.*

Proof. We use induction on $|w|$. If $|w| = 0$, the assertion trivially holds.

Let $w \in \Sigma^*$ and $\sigma \in \Sigma$. Suppose that $\tilde{f}_s(w\sigma, q_0)!$. Then $\tilde{f}_s(w, q_0)!$ and there exists $q' \in \tilde{f}_s(w, q_0)$ such that $\tilde{f}_s(\sigma, q')!$. By induction hypothesis, $f_s(w, q_0)!$ and $M(w) \in |\tilde{T}_s|$. Also, $\tilde{f}_s(w, q_0) = f_s(w, q_0)$ so that $q' \in f_s(w, q_0)$. By (b), $f_s(\sigma, q')!$ and there exists $\tilde{q} \in \tilde{Q}_s$ such that $\tilde{q} \in f_s(\sigma, q')$. Thus $f_s(w\sigma, q_0)!$. Note that $\tilde{q} \in f_s(w\sigma, q_0)$, so $\tilde{q} \in h_s(M(w\sigma), x_{s0})$. Moreover, by (a), there exists $d \in |\tilde{T}_s|$ such that $\tilde{q} \in Q_G(d)$. Using Lemma 3.5, we have $M(w\sigma) \in |\tilde{T}_s|$.

Conversely, suppose that $f_s(w\sigma, q_0)!$ and $M(w\sigma) \in |\tilde{T}_s|$. Then $f_s(w, q_0)!$ and $M(w) \in |\tilde{T}_s|$. By induction hypothesis, $\tilde{f}_s(w, q_0)!$ and $\tilde{f}_s(w, q_0) = f_s(w, q_0)$. Let $q' \in f_s(w, q_0)$, and note that $f_s(\sigma, q') \subset f_s(w\sigma, q_0) \subset h_s(M(w\sigma), x_{s0}) \subset Q_G(M(w\sigma))$. By (a), $f_s(\sigma, q') \subset \tilde{Q}_s$. It follows from (b) that for all $q' \in f_s(w, q_0)$, $f_s(\sigma, q') = \tilde{f}_s(\sigma, q')$ whenever $f_s(\sigma, q')!$. Thus it is clear that $\tilde{f}_s(w\sigma, q_0)!$. Moreover, $f_s(w\sigma, q_0) = \{q \in f_s(\sigma, q') | q' \in f_s(w, q_0)\} = \{q \in \tilde{f}_s(\sigma, q') | q' \in \tilde{f}_s(w, q_0)\} = \tilde{f}_s(w\sigma, q_0)$. ■

We now have the following theorem.

Theorem 3.2. $|\tilde{G}_s| = Sup\ \underline{R}(L)$.

Proof.

$$
\begin{aligned}
w \in |\tilde{G}_s| &\Leftrightarrow w \in |G_s| \text{ and } M(w) \in |\tilde{T}_s| \qquad \text{(Lemma 3.6)} \\
&\Leftrightarrow w \in |G_s| \cap M^{-1}[|\tilde{T}_s|] = \tilde{K} \\
&\Leftrightarrow w \in Sup\ \underline{R}(L) \qquad \text{(Theorem 3.1).}
\end{aligned}
$$

■

The previous construction of the NFA $\tilde{G}_s$ requires the DFA $\tilde{T}_s$ to be available. However, it is possible to obtain $\tilde{G}_s$ without the knowledge of $\tilde{T}_s$. Note that, for $d \in |T|$,

$$d \in |\tilde{T}_s| \Leftrightarrow h_s(\tilde{d}, x_{s0}) = Q_G(\tilde{d}) \quad \text{for all} \quad \tilde{d} \in \bar{d} \qquad \text{(Lemma 3.2)}$$

$$\Leftrightarrow Q_G(\tilde{d}) \subset Q_s \quad \text{for all} \quad \tilde{d} \in \bar{d}$$

("$\Rightarrow$" is trivial. "$\Leftarrow$" is immediate from the proof of Lemma 2.3 if we note that $Q_G(\tilde{d}) = h(\tilde{d}, x_0)$).

Thus, (a) in the definition of $\tilde{G}_s$ can be replaced by:

(a)′ $q \in \tilde{Q}_s$ if and only if there exists $d \in |T|$ such that $q \in Q_G(d)$ and $Q_G(\tilde{d}) \subset Q_s$ for all $\tilde{d} \in \bar{d}$.

This condition can be further simplified since we take as $\tilde{G}_s$ the trim component of the automaton constructed by (a) and (b). Consider:

(a)″ $q \in \tilde{Q}_s$ if and only if $q \in Q_G(d)$ implies $Q_G(d) \subset Q_s$.

Recall that the $Q_G(d)$'s form equivalence classes in Q. Thus it is clear that the states of $\tilde{G}_s$ generated by (a)′ are contained in those generated by (a)″. It is also easy to verify that the additional states generated by (a)″ are precisely those which are not accessible from the initial state q_0, and therefore are excluded from the state space of the resulting trim automaton. Note that the condition in (a)″ can be checked simply by inspecting the state $h(d, x_0)$ of T which contains q. Thus the following algorithm effectively computes $Sup\ \underline{R}(L)$.

Algorithm B ($Sup\ \underline{R}(L)$). Given NFAs G and G_s where $|G| = N$, $|G_s| = L$, G_s is a strict-subautomaton of G, and G is M-recognizable:

Step 1. Construct a DFA $T := (X, \Delta, h, x_0, X)$ as illustrated in Section 2.

Step 2. Construct an NFA $\tilde{G}_s := Tr\,(\tilde{Q}_s, \Sigma, \tilde{f}_s, q_0, \tilde{Q}_s)$, $\tilde{Q}_s \subset Q_s$, according to (a)″ and (b). If $q_0 \notin \tilde{Q}_s$ by rule (a)″, then $\tilde{G}_s := \Phi$. Since a nonempty $Q_G(d)$ is a state of the DFA T, the determination of $\tilde{Q}_s$ might be done in the following way: check each state $x \in X$ to see if x contains any $q \notin Q_s$. If so, remove all $q \in x \cap Q_s$ from Q_s.

Our introductory comments concerning the simplicity of the structure of our algorithm relative to that of [CDFV] clearly refer to our Algorithm B.

4. An Algorithm for $Sup\ \underline{D}(L)$

In this section, an algorithm that effectively computes $Sup\ \underline{D}(L)$ is presented. Recall that

$$\underline{R}(L) = \{O \subset L | O \text{ is closed and } (M, N)\text{-recognizable}\},$$

$$\underline{D}(L) = \{O \subset L | O \text{ is closed, } (\Sigma_u, N)\text{-invariant, and } (M, N)\text{-recognizable}\}.$$

We define two more classes of sublanguages of L as follows:

$$\underline{C}(L) = \{O \subset L | O \text{ is closed and } (\Sigma_u, N)\text{-invariant}\},$$

$$\underline{C}'(L) = \{O \subset L | \bar{O} \text{ is } (\Sigma_u, N)\text{-invariant}\}.$$

So far, we have presented two algorithms that compute $Sup\ \underline{R}(L)$. Also, we note that Wonham and Ramadge [WR1] have presented an efficient algorithm which computes $Sup\ \underline{C}'(L)$ effectively when L, N are regular and N is closed. Moreover, it is not difficult to see that if L is closed, then $Sup\ \underline{C}'(L)$ is the same as $Sup\ \underline{C}(L)$. Thus we are naturally led to the consideration of the strategy where the algorithm of Wonham and Ramadge and one of the algorithms developed in Section 3 are applied in an alternating manner to compute $Sup\ \underline{D}(L)$. In the following we explore this idea in detail.

We again assume that L, N are regular and closed. Also, all automata are assumed to be deterministic in this section, which makes the algorithm of Wonham and Ramadge [WR1, Section 6] readily applicable. Note that this additional assumption is not a restriction, since for each NFA A, there is a DFA B such that $|A| = |B|$. A standard procedure that converts an NFA into a DFA can be found easily (e.g., see [HU]).

Now we summarize Wonham and Ramadge's algorithm as follows.

Consider DFAs $A = (Q_A, \Sigma, f_A, q_A^0, Q_{Am})$ and $B = (Q_B, \Sigma, f_B, q_B^0, Q_{Bm})$ with $|B| \subset |A|$. We say that B refines A if for all s, $t \in \overline{|B|}$, $f_B(s, q_B^0) = f_B(t, q_B^0)$ implies $f_A(s, q_A^0) = f_A(t, q_A^0)$. If B refines A, then there is a unique function $\psi: Q_B \to Q_A$ satisfying $\psi(f_B(s, q_B^0)) = f_A(s, q_A^0)$ for all $s \in \overline{|B|}$.

Algorithm C ($Sup\ \underline{C}'(L)$) (Wonham and Ramadge). Let $G = (Q, \Sigma, f, q_0, Q)$ and $C_0 = (Z_0, \Sigma, g_0, z_0^0, Z_{0,m})$ be DFAs such that $|G| = N$, $|C_0| \subset N$, and C_0 refines G. For each $j \geq 0$, define a DFA $C_{j+1} := Tr\,(Z_{j+1}, \Sigma, g_{j+1}, z_{j+1}^0, Z_{j+1,m})$ by

$$Z_{j+1} := \{z \in Z_j | \Sigma(\psi_j(z)) \cap \Sigma_u \subset \Sigma(z)\}, \qquad Z_{j+1,m} = Z_{j,m} \cap Z_{j+1},$$

$$g_{j+1}(\sigma, z) := \begin{cases} g_j(\sigma, z) & \text{if } g_j(\sigma, z) \in Z_{j+1}, \\ \text{undefined} & \text{otherwise,} \end{cases}$$

where $\Sigma(q)$ and $\Sigma(z)$ are the sets of events for which the transition functions are defined at the states $q \in Q$ and $z \in Z_j$, respectively, and $\psi_j: Z_j \to Q$ is the unique map defined as above.

It was shown in [WR1] that each C_{j+1} refines G and that there exists $k \in \mathbb{N}$ such that $|C_j| = |C_k|$ for all $j \geq k$ and $|C_k| = Sup\ \underline{C}(|C_0|)$.

Remark 4.1. If C_0 is a strict-subautomaton of G, then C_0 refines G. It is also easy to check that C_j is a strict-subautomaton of G for all j.

Now we present an algorithm that computes $Sup\ \underline{D}(L)$.

Algorithm D ($Sup\ \underline{D}(L)$). Let DFAs G and G_s^0 be such that $|G| = N$, $|G_s^0| = L$, and G_s^0 is a strict-subautomaton of G. Let $j = 0$.

Step 1. Compute $Sup\ \underline{C}(|G_s^j|)$ (use Algorithm C). Denote the resulting DFA by G_{sc}^j.

Step 2. Compute $Sup\ \underline{R}(|G_{sc}^j|)$ (use Algorithm A or Algorithm B). Denote the resulting DFA by G_s^{j+1}. Also, if G is not M-recognizable, construct a DFA G' such that $|G'| = |G|$ and G' is M-recognizable. Otherwise, let $G' := G$.

Step 3. Set $j := j + 1$. Go to step 1 with G replaced by G'.

Remark 4.2. The DFA G' in step 2 can be identified with the automaton V in Section 3.

Note that we can always use Algorithm B in step 2 after the first iteration. In this case, the procedures after the first iteration will be simply those of removing states from the set of states of the automaton constructed in the previous step. Hence it is clear that the algorithm stops in a finite number of iterations. Since the algorithm converges, we must now also show that it does indeed converge to $Sup\ \underline{D}(L)$.

Let $K_j = |G_s^j|$, $K_j^c = |G_{sc}^j|$, and let $S = Sup\ \underline{D}(L)$. Then $K_0 = L$, $K_j^c = Sup\ \underline{C}(K_j)$ and $K_{j+1} = Sup\ \underline{R}(K_j^c)$. Note that, for all j, $K_{j+1} \subset K_j^c \subset K_j$, so the sequence $\{K_j\}$ is monotone decreasing.

Lemma 4.1.

(1) $S \subset K_j$ *for all* $j \geq 0$.
(2) *If* $K_i = K_{i+1}$, *then* $K_i \subset S$.

Proof. (1) Clearly, $S \subset K_0 = L$. Suppose that $S \subset K_j, j \geq 0$. Then $S = Sup\ \underline{D}(S) \subset Sup\ \underline{D}(K_j) \subset Sup\ \underline{C}(K_j) = K_j^c$ and therefore $S = Sup\ \underline{D}(S) \subset Sup\ \underline{R}(S) \subset Sup\ \underline{R}(K_j^c) = K_{j+1}$. Thus $S \subset K_{j+1}$. Therefore we have proved that $S \subset K_j$ for all $j \geq 0$.

(2) Suppose that $K_i = K_{i+1}$, then $K_i = K_i^c = K_{i+1}$. Thus $K_i = K_i^c = Sup\ \underline{C}(K_i)$ and $K_i = K_{i+1} = Sup\ \underline{R}(K_i^c) = Sup\ \underline{R}(K_i)$. Hence, $K_i = Sup\ \underline{D}(K_i) \in \underline{D}(L)$. Therefore $K_i \subset Sup\ \underline{D}(L) = S$. ■

In view of Lemma 4.1, we conclude that if the algorithm stops after a finite number of iterations, or, equivalently, if there is $i \geq 0$ such that $K_j = K_i$ for all $j \geq i$, then $K_i = Sup\ \underline{D}(L)$. Thus we have

Theorem 4.1. *The sequence* $\{K_j\}$ *finitely converges to* $Sup\ \underline{D}(L)$. *Thus Algorithm D effectively computes* $Sup\ \underline{D}(L)$.

Remark 4.3. Algorithm D is more general than the algorithm of [CDFV], since the latter is only valid under condition

(i) $M^{-1}[M(\Sigma_u)] \subset \Sigma_u \cup \{\varepsilon\}$.

However, in this special case, the algorithm of [CDFV] yields $Sup\ \underline{D}(L)$ without an iterative procedure. It is interesting in this context to note that if the mask M satisfies an additional condition

(ii) $M(\sigma) \neq \varepsilon$ for all $\sigma \in \Sigma - \Sigma_u$,

then Algorithm D can be shown to yield $Sup\ \underline{D}(L)$ after a single iteration of the procedure.

An alternative proof of convergence of the algorithm, which provides additional insights, is given in the Appendix A. Now we give an example which illustrates the use of Algorithm D.

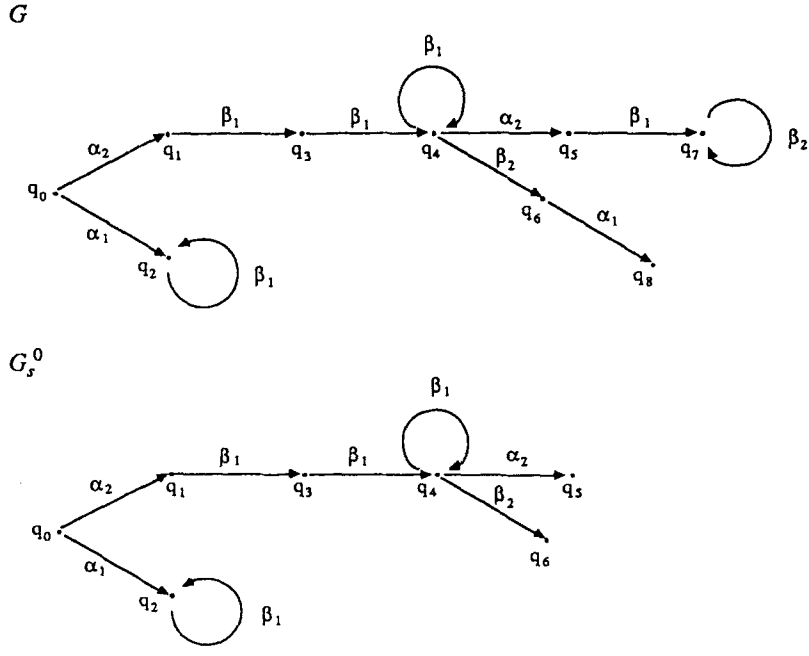

Fig. 1. G and G_s^0 (Example).

Example. Let $\Sigma = \{\alpha_1, \alpha_2, \beta_1, \beta_2\}$, $\Sigma_u = \{\alpha_2, \beta_2\}$, and let $M(\alpha_1) = M(\alpha_2) = \delta_1$ and $M(\beta_1) = M(\beta_2) = \delta_2$. Notice that in this case $M^{-1}[M(\Sigma_u)] \not\subset \Sigma_u \cup \{\varepsilon\}$ (i.e., the condition of [CDFV] is not satisfied). Two regular languages N and L over Σ, with $L \subset N$, are given as follows:

$$N = \overline{\alpha_2\beta_1^2\beta_1^*(\beta_2\alpha_1 + \alpha_2\beta_1\beta_2^*)} + \alpha_1\beta_1^*,$$
$$L = \overline{\alpha_2\beta_1^2\beta_1^*(\beta_2 + \alpha_2)} + \alpha_1\beta_1^*.$$

Figure 1 displays two DFAs G and G_s^0 which generate languages N and L, respectively. Note that G_s^0 is a strict-subautomaton of G. Also, it can be easily checked (Algorithm C) that $G_s^0 = G_{sc}^0$. In other words, L is (Σ_u, N)-invariant.

We use Algorithm A to compute $Sup\ \underline{R}(|G_{sc}^0|)$. Two DFAs T and T_s with $|T| = M(N)$ and $|T_s| = M(|G_{sc}^0|)$ are displayed in Fig. 2. We note here that the $Q_G(d)$'s, $d \in M(N)$ (which are states of T when they are nonempty), do not form equivalence classes on the set of states of G, since they are not disjoint. Thus G is not M-recognizable. Also, it is clear that T_s is not a subautomaton of T. We remove the edge $x_4 \xrightarrow{\delta_1} x_5$ from T_s so that the resulting automaton $\bar{T}_s$ is the largest subautomaton of T that is also a subautomaton of T_s. The DFA $\bar{T}_s$ is also displayed in Fig. 2. From $\bar{T}_s$, we obtain the DFA G_s^1, which generates the language $|G_{sc}^0| \cap M^{-1}[|\bar{T}_s|] = Sup\ \underline{R}(|G_{sc}^0|)$, according to the method given in Section 3. Since G is not M-recognizable, we also construct a DFA G' (in the same way as for G_s^1) such that $|G'| = N$. The corresponding DFA T' which generates the language

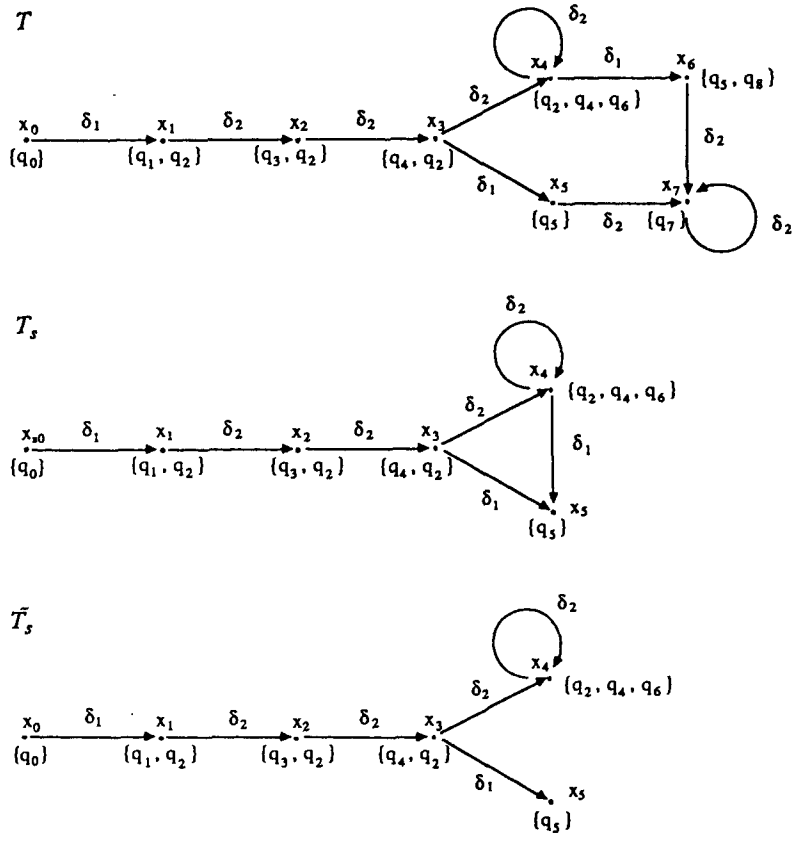

Fig. 2. T, T_s, and $\tilde{T}_s$ (Example—first iteration).

$M(|G'|) = M(N)$ is constructed accordingly. The DFAs G_s^1, G', and T' are displayed in Fig. 3. This completes the first iteration of the algorithm. Note that T and T' have the same structure, but different labels for states. Note also that states of T' now represent equivalence classes on the set of states of G'. Thus we see that G' is M-recognizable. We use Algorithm B hereafter for obtaining $Sup\ \underline{R}(G_{sc}^j)$, $j \geq 1$.

We remove the state 7 of G_s^1 using Algorithm C to get G_{sc}^1, which is displayed in Fig. 4. Observe that the state x_4' $(= \{7, 8, 9\})$ of T contains the state 7 of G' which is not in the set of states of G_{sc}^1. Thus we remove states 8 and 9 of G_{sc}^1 as well (Algorithm B). The resulting automaton G_s^2 is also displayed in Fig. 4. This completes the second iteration.

The same process is repeated in the third iteration, and the resulting automata G_{sc}^2 and G_s^3 are shown in Fig. 5. Now, Algorithm C gives $G_{sc}^3 = G_s^3$; i.e., $|G_s^3|$ is (Σ_u, N)-invariant. Since $|G_s^3| = Sup\ \underline{R}(|G_{sc}^2|)$ is (M, N)-recognizable, it is clear that $G_s^j = G_s^3$ for $j \geq 4$. Hence, $Sup\ \underline{D}(L) = |G_s^3| = \overline{(\alpha_1 + \alpha_2)\beta_1}$.

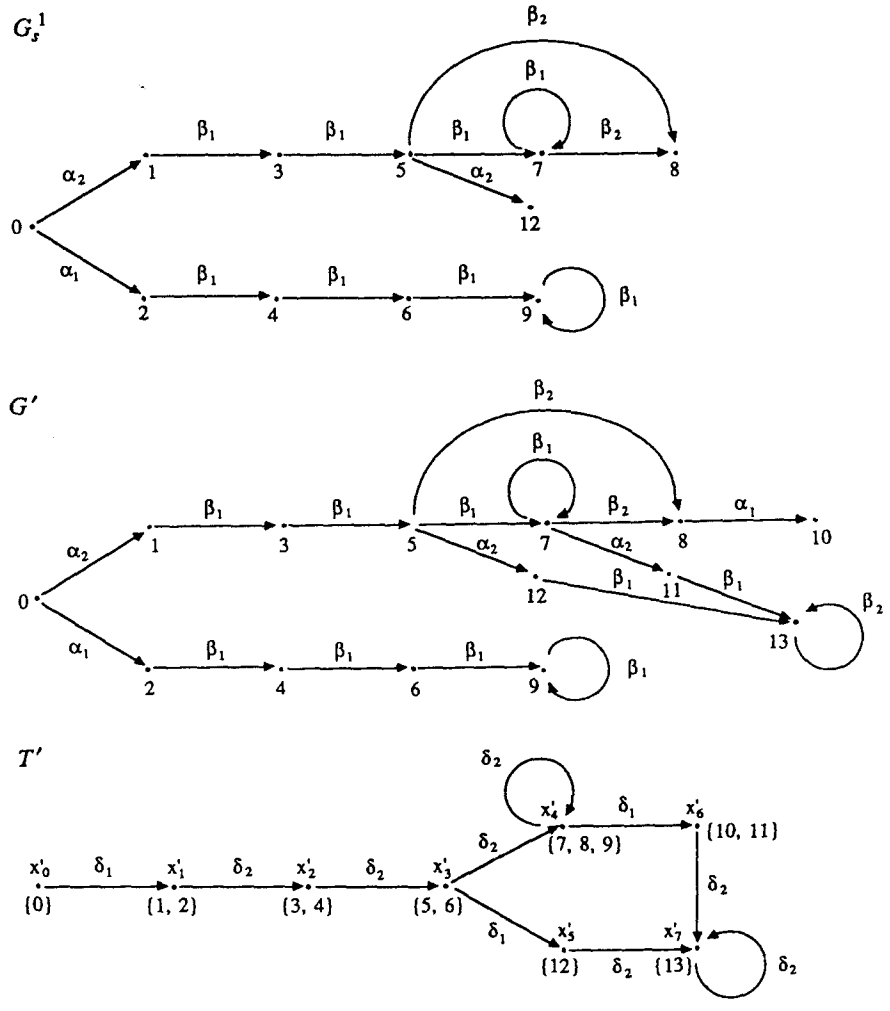

Fig. 3. G_s^1, G', and T' (Example—first iteration).

5. Application to the Supervisory Marking Problem

In this section we seek a generalization of the algorithms developed so far so that the resulting algorithm provides minimally restrictive solutions within a smaller class to the following Supervisory Marking Problems (SMP) [RW] with partial observation: given $L \subset L_m(G)$, $\overline{L_m(G)} = L(G)$, and a mask M, find a *proper* supervisor $\underline{S}$ (i.e., $\underline{S}$ is complete and $\overline{L_m(\underline{S}/G_c)} = \overline{L(\underline{S}/G_c) \cap L_m(G)} = L(\underline{S}/G_c)$) such that $L_m(\underline{S}/G_c) \subset L$. It turns out, as is the case for the supervisor synthesis problem of [CDFV], that the class $\underline{DD}(L)$ of sublanguages O of L which satisfy the necessary and sufficient condition for the existence of proper supervisors $\underline{S}$ with $L_m(\underline{S}/G_c) = O$

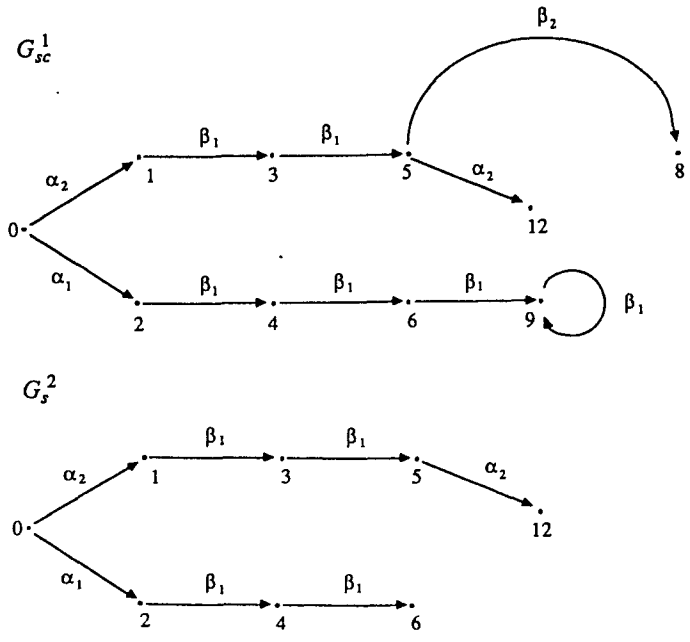

Fig. 4. G_{sc}^1 and G_s^2 (Example—second iteration).

does not, in general, possess a supremal element [CDFV], [LW] (see also Appendix B). In other words, the minimally restrictive solution does not necessarily exist for SMP. Thus it is again natural and convenient to restrict ourselves to a subclass of $\underline{DD}(L)$ that has a unique supremal element [LW].

Let $L \subset L_m \subset N \subset \Sigma^*$, $\bar{L}_m = N$. We assume that L and N are regular and N is closed. Here, L is not necessarily closed. Let $O \subset L_m$. We say that O is L_m-*closed* if $\bar{O} \cap L_m = O$. Consider the following classes of sublanguages of L:

$$\underline{R}'(L) := \{O \subset L | \bar{O} \text{ is } (M, N)\text{-recognizable}\},$$

$$\underline{DD}'(L) := \{O \subset L | O \text{ is } L_m\text{-closed}, \bar{O} \text{ is } (\Sigma_u, N)\text{-invariant and } (M, N)\text{-recognizable}\},$$

$$\underline{D}'(L) := \{O \subset L | \bar{O} \text{ is } (\Sigma_u, N)\text{-invariant and } (M, N)\text{-recognizable}\}.$$

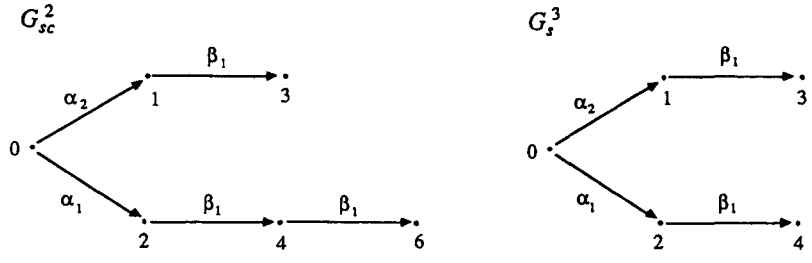

Fig. 5. G_{sc}^2 and G_s^3 (Example—third iteration).

These classes of sublanguages of L are closed under arbitrary union and possess supremal elements. Also, if we identify L_m and N with $L_m(G)$ and $L(G)$, each language of the class $\underline{DD}'(L)$ is a solution to SMP; i.e., $\underline{DD}'(L) \subset \underline{DD}(L)$ [LW] (see also Appendix B). Thus we consider $Sup\ \underline{DD}'(L)$ as a minimally restrictive solution to the "restricted" SMP. Consider now the class $\underline{D}'(L)$ of sublanguages of L. Clearly, $\underline{DD}'(L) \subset \underline{D}'(L)$. However, the requirement for a sublanguage O of L to be in $\underline{D}'(L)$ is not sufficient to guarantee the existence of a supervisor $\underline{S}$ such that $L_m(\underline{S}/G_c) = O$. Nevertheless, it is shown in Appendix B that if L is L_m-closed, then $Sup\ \underline{DD}'(L) = Sup\ \underline{D}'(L)$, and that we may assume without loss of generality that L is L_m-closed. Since the problem of obtaining $Sup\ \underline{D}'(L)$ is easier than that of obtaining $Sup\ \underline{DD}'(L)$ (there are less constraints in the former problem), we discuss hereafter an algorithm that effectively computes $Sup\ \underline{D}'(L)$.

The algorithm that is developed in this section is essentially parallel to the one presented in the previous sections for $Sup\ \underline{D}(L)$. The only major difference is that Algorithm A or the new version of Algorithm B does not, in general, yield $Sup\ \underline{R}'(L)$ with a single iteration of the procedure any more. Consider an operator Ω: $P(\Sigma^*) \to P(\Sigma^*)$ defined by

$$\Omega(R) := \{s \in R \mid N \cap M^{-1}[M(\bar{s})] \subset \bar{R}\}.$$

Clearly, Ω is monotone: i.e., if $R_1 \subset R_2$, then $\Omega(R_1) \subset \Omega(R_2)$. Let $S_R := Sup\ \underline{R}'(L)$. Then the following lemma is immediate.

Lemma 5.1. $S_R = \Omega(S_R)$.

Proof. Trivially, $\Omega(S_R) \subset S_R$. Let $s \in S_R$. Then $N \cap M^{-1}[M(\bar{s})] \subset N \cap M^{-1}[M(\overline{S_R})] = \overline{S_R}$, where the equality follows from the fact that $\overline{S_R}$ is (M, N)-recognizable. Thus $s \in \Omega(S_R)$. This proves that $S_R \subset \Omega(S_R)$. ■

Define a sequence of sublanguages $\{R_j\}$ of L by

$$R_o := L,$$

$$R_{j+1} := \Omega(R_j).$$

Clearly, $\{R_j\}$ is monotone decreasing; i.e., $R_{j+1} \subset R_j$ for all $j \geq 0$.

Lemma 5.2.

(1) $S_R \subset R_j$ *for all* $j \geq 0$.
(2) *If* $R_j = R_{j+1}$, *then* $R_j \subset S_R$.

Proof. (1) Clearly, $S_R \subset L = R_0$. Suppose that $S_R \subset R_i$, $i \geq 0$. Using Lemma 5.1, we have $S_R = \Omega(S_R) \subset \Omega(R_i) = R_{i+1}$. Thus $S_R \subset R_j$ for all $j \geq 0$.

(2) Assume that $R_j = R_{j+1}$, or equivalently, $R_j = \Omega(R_j)$. From the definition of Ω, $N \cap M^{-1}[M(\bar{s})] \subset \bar{R}_j$ for all $s \in R_j$. Thus $N \cap M^{-1}[M(\bar{R}_j)] \subset \bar{R}_j$. Hence $\bar{R}_j$ is (M, N)-recognizable. Since $R_j \subset L$, we have that $R_j \in \underline{R}'(L)$. Therefore $R_j \subset S_R$. ■

By Lemma 5.2, it is clear that if the sequence $\{R_j\}$ converges after a finite number of terms, then the limit is $Sup\ \underline{R}'(L)$.

Observe now that $\Omega(\overline{R_j}) = Sup\ \underline{R}(\overline{R_j})$ by Lemma 3.1 and the definition of Ω. Also, it is easy to see from the definition that $\Omega(R_j) = R_j \cap \Omega(\overline{R_j})$. Thus we have that $\Omega(R_j) = R_j \cap Sup\ \underline{R}(\overline{R_j})$. This observation therefore suggests that we could use the algorithms developed in Section 3 to compute $\Omega(R_j)$ for each j.

In fact, Algorithm A in Section 3 computes $\Omega(L)$ without any modification. This can be easily seen by noting in Algorithm A that steps 1 and 2 yield the same $\tilde{T}_s$ regardless of whether L is closed or not (recall that the construction of DFAs T and T_s is such that $|T| = M(|\overline{G}|)$ and $|T_s| = M(|\overline{G_s}|)$, and that $\tilde{T}_s$ is determined entirely by the structural properties between T and T_s), and that the last step computes $L \cap M^{-1}[|\tilde{T}_s|] = L \cap (\bar{L} \cap M^{-1}[|\tilde{T}_s|]) = L \cap Sup\ \underline{R}(\bar{L})$. A similar observation can be made concerning Algorithm B with a minor modification; that is, we replace the set of marked states $\tilde{Q}_s$ of $\tilde{G}_s$ by $\tilde{Q}_{sm} := \tilde{Q}_s \cap Q_{sm}$ where Q_{sm} is the set of marked states of G_s. Then this modified version, called Algorithm B′, effectively computes $\Omega(L)$. We summarize the above argument in

Lemma 5.3. *Let $R \subset N$ be a regular language over Σ^*. Then Algorithm A (or Algorithm B′) effectively computes $\Omega(R)$.*

Thus the sequence $\{R_j\}$ can be computed by repeatedly applying Algorithm A or Algorithm B′. The convergence of this sequence is now obvious in view of procedures involved in these algorithms, and can be argued as in Section 4; that is, after the first iteration, we could use Algorithm B′ exclusively so that at each iteration, the procedure is simply to remove some states from a subautomaton of a fixed M-recognizable NFA. Therefore the whole procedure must stop after a finite number of iterations. A proof of convergence which is similar to the one presented for Algorithm D in Appendix A can also be constructed.

Now we consider Algorithm D with $Sup\ \underline{C}(\cdot)$ and $Sup\ \underline{R}(\cdot)$ replaced by $Sup\ \underline{C}'(\cdot)$ and $Sup\ \underline{R}'(\cdot)$. We call this modified version Algorithm D′. It is easy to verify that Lemma 4.1 still holds with S and $\{K_{i+1}\}$ interpreted as $Sup\ \underline{D}'(L)$ and $\{Sup\ \underline{R}'(Sup\ \underline{C}'(K_i))\}$, respectively. Again, the convergence of Algorithm D′ can be shown in the same manner as in Section 4. Thus we have

Theorem 5.1. *Algorithm D′ effectively computes $Sup\ \underline{D}'(L)$.*

6. Concluding Remarks

A graphical characterization of (M, N)-recognizable sublanguages of a given language L has been presented and has led to the development of an algorithm which effectively computes $Sup\ \underline{D}(L)$ under the assumption that N and L are regular. No restrictions on the mask M have been made, although some conditions, one being the same condition imposed in the algorithm of [CDFV], guarantee that Algorithm D yields $Sup\ \underline{D}(L)$ after a single iteration of the procedure. Moreover, the results have been extended to the case where we desire to obtain $Sup\ \underline{D}'(L)$ and L is not necessarily closed. Thus Algorithm D′ effectively computes $Sup\ \underline{D}'(L)$ which is a minimally restrictive solution within a smaller class to SMP. Also, the notion of M-recognizable automata has been introduced in order to construct

an algorithm (Algorithm B or B′) with a simpler structure than that of Algorithm A or the corresponding algorithm in [CDFV] to compute $Sup\ \underline{R}(L)$ or $Sup\ \underline{R}'(L)$. This simply structured algorithm not only makes the entire procedure of extracting $Sup\ \underline{D}(L)$ or $Sup\ \underline{D}'(L)$ much more straightforward so that the procedure consists of simply removing some states from the automaton obtained in the previous step, but also simplifies the proofs of convergence for various algorithms. It is expected that the concept of M-recognizable automata will play a key role in the development of algorithms which compute other sublanguages of L that solve supervisor synthesis problems with partial observation.

It is clear from their structures that all of the algorithms for $Sup\ \underline{R}(L)$ (Algorithm A, B, B′, and that of [CDFV]) are of exponential worst-case complexity; thus so are Algorithms D and D′. In Algorithm A the exponential time complexity results from the construction of the DFAs T, T_s, and $\tilde{T}_s$; i.e., the state spaces of these DFAs have cardinality that is, in general, exponential in the size of the state spaces of G and G_s. The algorithm of [CDFV] includes the construction of DFAs similar to T, T_s, and $\tilde{T}_s$. The state space of T in Algorithm B (B′) has cardinality less than or equal to that of G, since G is M-recognizable. However, if G is not M-recognizable in the first place, the M-recognizable version of G has to be obtained in order for Algorithm B to be applied. The procedure of obtaining an M-recognizable automaton will be of exponential-time complexity.

The fact that all algorithms in the paper are of exponential-time complexity is not surprising. Indeed, the same argument as in [T] can be shown to lead to the conclusion that unless P = NP, there is no polynomial-time algorithm for the decision problem corresponding to the problem of obtaining $Sup\ \underline{R}(L)$. In other words, all the problems considered in this paper are "intractable" from the point of view of computational complexity theory. Note, however, that this view comes from the worst-case analysis of the problems. Also, there are exponential-time algorithms that are successfully used in practice. The practicality of the algorithms presented in this paper is yet to be determined in applications to real problems.

Appendix A. An Alternative Proof of the Convergence of Algorithm D

In this appendix we give an alternative proof of the convergence of Algorithm D. We begin with definitions of some equivalence relations on Σ^*.

For any $K \subset N \subset \Sigma^*$, we define equivalence relations $\equiv_K$ and $\sim_K$ on Σ^* as follows:

(i) $s \equiv_K t$ if for all $w \in \Sigma^*$, $sw \in K$ if and only if $tw \in K$.
(ii) $s \sim_K t$ if $s \equiv_N t$ and $s \equiv_K t$.

Also, we define an equivalence relation $\sim_K$ on $P(\Sigma^*)$ according to

(iii) $O_1 \sim_K O_2$ if for all $s \in O_1$, there exists $t \in O_2$ such that $s \sim_K t$, and vice versa.

Recall that the sequence of regular languages $\{K_j\}$ generated by Algorithm D are defined recursively by $K_0 = L$, $K_j^c = Sup\ \underline{C}(K_j)$, and $K_{j+1} = Sup\ \underline{R}(K_j^c)$. We note again that for all j, $K_{j+1} \subset K_j^c \subset K_j$, so the sequence $\{K_j\}$ is monotone decreasing. Now, it has been shown in [WR1] that if a sequence of regular languages $\{K_j\}$ is

decreasing and the sequence $\{\text{Card}(\Sigma^*/\equiv_{K_j})\}$ is bounded, then there is $i \in \mathbb{N}$ such that $K_j = K_i$ for all $j \geq i$. To prove the convergence of Algorithm D, it is therefore enough to show that the sequence $\{\text{Card}(\Sigma^*/\equiv_{K_j})\}$ is bounded. For this purpose, we proceed as follows.

We first state a lemma which is a direct consequence of Theorem 3.1 of [WR1] and the proof of Lemma 3.4 of [WR1].

Lemma A.1. *Let $s, t \in \overline{K_j^c}, j \geq 0$. If $s \sim_{K_j} t$, then $s \sim_{K_j^c} t$.*

Note that $\overline{K_j^c} = K_j^c$ in our case. We recall from Lemma 3.1 that

$$K_{j+1} = Sup\, \underline{R}(K_j^c) = \{s \in K_j^c | N \cap M^{-1}[M(\bar{s})] \subset K_j^c\}. \qquad (*)$$

Lemma A.2. *Let $s \in K_{j+1}$. If $w \in N \cap M^{-1}[M(s)]$, then $w \in K_{j+1}$.*

Proof. Let $w \in N \cap M^{-1}[M(s)]$. Since $s \in K_{j+1}$, $w \in K_j^c$ by $(*)$. Also, $M(w) = M(s)$, so $M(\bar{w}) = M(\bar{s})$. Thus $N \cap M^{-1}[M(\bar{w})] = N \cap M^{-1}[M(\bar{s})] \subset K_j^c$. Therefore $w \in K_{j+1}$. ■

Lemma A.3. *Let $s, t \in K_{j+1}, j \geq 0$. If $s \equiv_{K_j^c} t$ and $N \cap M^{-1}[M(s)] \sim_{K_j^c} N \cap M^{-1}[M(t)]$, then $s \equiv_{K_{j+1}} t$ and $N \cap M^{-1}[M(s)] \sim_{K_{j+1}} N \cap M^{-1}[M(t)]$.*

Proof. (Part A) It will be shown that if $s \equiv_{K_j^c} t$ and $N \cap M^{-1}[M(s)] \sim_{K_j^c} N \cap M^{-1}[M(t)]$, then $s \equiv_{K_{j+1}} t$. Let $su \in K_{j+1}$. Then $su \in K_j^c$ and therefore $tu \in K_j^c$. Suppose that $w \in N \cap M^{-1}[M(\overline{tu})]$. We show that $w \in K_j^c$. If $w \in M^{-1}[M(\bar{t})]$, then $w \in K_j^c$ by $(*)$ since $t \in K_{j+1}$. So let $w \in M^{-1}[M(t\bar{u})]$, $\bar{u} \in \bar{u}$. Thus $w = w_1 w_2$ where $w_1 \in M^{-1}[M(t)]$ and $w_2 \in M^{-1}[M(\bar{u})]$. Note that $w_1 \in N$ since N is closed. By the assumption, there exists $v \in N \cap M^{-1}[M(s)]$ such that $v \sim_{K_j^c} w_1$. Thus $vw_2 \in N$. Also $vw_2 \in M^{-1}[M(s\bar{u})]$. Hence, $vw_2 \in K_j^c$ by $(*)$ and by the assumption that $su \in K_{j+1}$. But this implies that $w = w_1 w_2 \in K_j^c$ since $v \sim_{K_j^c} w_1$. Thus we have proved that $tu \in K_{j+1}$, and therefore that $s \equiv_{K_{j+1}} t$.

(Part B) We show that $N \cap M^{-1}[M(s)] \sim_{K_{j+1}} N \cap M^{-1}[M(t)]$. Let $x \in N \cap M^{-1}[M(s)]$. By the assumption, there exists $y \in N \cap M^{-1}[M(t)]$ such that $x \sim_{K_j^c} y$. By Lemma A.2, $x, y \in K_{j+1}$. Note that $N \cap M^{-1}[M(x)] \sim_{K_j^c} N \cap M^{-1}[M(y)]$ since $M(x) = M(s)$ and $M(y) = M(t)$. Thus, by part A, $x \equiv_{K_{j+1}} y$, and therefore $x \sim_{K_{j+1}} y$. Hence we have shown that $N \cap M^{-1}[M(s)] \sim_{K_{j+1}} N \cap M^{-1}[M(t)]$. ■

Lemma A.4. *Let $s, t \in K_j, j \geq 0$. If $s \sim_L t$ and $N \cap M^{-1}[M(s)] \sim_L N \cap M^{-1}[M(t)]$, then $s \sim_{K_j} t$ and $N \cap M^{-1}[M(s)] \sim_{K_j} N \cap M^{-1}[M(t)]$.*

Proof. We use induction on j. If $j = 0$, the assertion trivially holds. Suppose that $s, t \in K_{i+1}$, $i \geq 0$, such that $s \sim_L t$ and $N \cap M^{-1}[M(s)] \sim_L N \cap M^{-1}[M(t)]$. Then $s, t \in K_i$ since $K_{i+1} \subset K_i$. By induction hypothesis, $s \sim_{K_i} t$ and $N \cap M^{-1}[M(s)] \sim_{K_i} N \cap M^{-1}[M(t)]$. Note also that $s, t \in K_i^c$ since $K_{i+1} \subset K_i^c$. By Lemma A.1, $s \sim_{K_i^c} t$. Suppose now that $x \in N \cap M^{-1}[M(s)]$. Then there exists $y \in N \cap M^{-1}[M(t)]$ such that $x \sim_{K_i} y$. By Lemma A.2, $x, y \in K_{i+1}$. Thus $x, y \in K_i^c$, and it follows from Lemma A.1 that $x \sim_{K_i^c} y$. Hence $N \cap M^{-1}[M(s)] \sim_{K_j^c} N \cap M^{-1}[M(t)]$. Now we

apply Lemma A.3 to get $s \sim_{K_{i+1}} t$ and $N \cap M^{-1}[M(s)] \sim_{K_{i+1}} N \cap M^{-1}[M(t)]$. This completes the induction step. ■

We define another equivalence relation $\approx$ on Σ^* according to

(iv) $s \approx_K t$ if $s \sim_K t$ and $N \cap M^{-1}[M(s)] \sim_K N \cap M^{-1}[M(t)]$.

Then we have the following corollary.

Corollary A.1. *For all* $j \geq 0$, $\mathrm{Card}(\Sigma^*/\equiv_{K_j}) \leq \mathrm{Card}(K_j/\approx_L) + 1 \leq \mathrm{Card}(\Sigma^*/\approx_L) + 1$. *In particular,* $\mathrm{Card}(\Sigma^*/\equiv_{K_j}) \leq \mathrm{Card}(K_1/\approx_L) + 1$ *for all* $j \geq 1$.

Proof. Note that the subset $\Sigma^* - K_j$ of Σ^* forms at most one equivalence class under $\equiv_{K_j}$. Thus the assertion follows from Lemma A.4. ■

Recall that $L = K_0$ is regular. To show that the sequence $\{\mathrm{Card}(\Sigma^*/\equiv_{K_j})\}$ is bounded, it therefore suffices to prove that $\mathrm{Card}(K_1/\approx_L) < \infty$.

We consider again DFAs $G = (Q, \Sigma, f, q_0, Q)$, $G_s = (Q_s, \Sigma, f_s, q_0, Q_s)$, and $\tilde{G}_s = (\tilde{Q}_s, \Sigma, \tilde{f}_s, q_0, \tilde{Q}_s)$ which were defined in Section 3. Thus $|G| = N$, $|G_s| = L$, and $|\tilde{G}_s| = Sup\ \underline{R}(L)$. Also, G is M-recognizable and G_s is a strict-subautomaton of G. We recall here that such DFAs can always be effectively constructed. We also note that $K_1 \subset |\tilde{G}_s|$ since $K_1 = Sup\ \underline{R}(K_0^c) \subset Sup\ \underline{R}(L)$.

Define equivalence relations $\equiv_G$ and $\equiv_{G_s}$ on Σ^* as follows:

(v) $s \equiv_G t$ if either $s, t \in |G|$ and $f(s, q_0) = f(t, q_0)$, or $s, t \notin |G|$.

(vi) $s \equiv_{G_s} t$ if either $s, t \in |G_s|$ and $f_s(s, q_0) = f_s(t, q_0)$, or $s, t \notin |G_s|$.

Then it is easy to check that $s \equiv_G t$ implies $s \equiv_N t$ and $s \equiv_{G_s} t$ implies $s \equiv_L t$. Also, we have

Lemma A.5. $\mathrm{Card}(K_1/\approx_L) \leq \mathrm{Card}(L/\equiv_{G_s}) < \infty$.

Proof. Let $s, t \in |\tilde{G}_s|$. Then $s, t \in |G_s|$. We first show that $s \equiv_{G_s} t$ implies $s \approx_L t$. Let $s \equiv_{G_s} t$. Since G_s is a subautomaton of G, $s \equiv_G t$ so that $s \sim_L t$. Thus $f(s, q_0) = f(t, q_0)$. Note that $f(s, q_0) \in Q_G(M(s))$ and $f(t, q_0) \in Q_G(M(t))$. Thus $Q_G(M(s)) = Q_G(M(t))$ since G is M-recognizable. Moreover, we have, from Lemmas 3.6 and 3.2, that $\{f_s(u, q_0) | M(u) = M(s)\} = Q_G(M(s))$ and $\{f_s(v, q_0) | M(v) = M(t)\} = Q_G(M(t))$. Hence

$$\{f_s(u, q_0) | M(u) = M(s)\} = \{f_s(v, q_0) | M(v) = M(t)\}. \qquad (**)$$

Now let $w \in N \cap M^{-1}[M(s)]$. Then $M(w) = M(s)$. Since $s \in |\tilde{G}_s| = Sup\ \underline{R}(L)$, it follows from Lemma A.2 (with K_{j+1} and K_j^c replaced by $|\tilde{G}_s|$ and $|G_s|$, respectively) that $w \in |\tilde{G}_s|$. Thus $f_s(w, q_0)!$. By (**), there exists $w' \in \Sigma^*$ such that $f_s(w', q_0) = f_s(w, q_0)$ and $M(w') = M(t)$. Thus $w' \equiv_{G_s} w$ and $w' \in N \cap M^{-1}[M(t)]$. Note that $w' \sim_L w$ by the same argument as in the beginning of the proof. Thus $N \cap M^{-1}[M(s)] \sim_L N \cap M^{-1}[M(t)]$.

Hence, we have shown that for $s, t \in |\tilde{G}_s|$, $s \equiv_{G_s} t$ implies $s \approx_L t$. Therefore $\mathrm{Card}(|\tilde{G}_s|/\approx_L) \leq \mathrm{Card}(|\tilde{G}_s|/\equiv_{G_s})$. Since $K_1 \subset |\tilde{G}_s|$, it follows that $\mathrm{Card}(K_1/\approx_L) \leq \mathrm{Card}(|\tilde{G}_s|/\approx_L) \leq \mathrm{Card}(|\tilde{G}_s|/\equiv_{G_s}) \leq \mathrm{Card}(L/\equiv_{G_s}) < \infty$. ■

Thus we have established that the sequence $\{\text{Card}(\Sigma^*/\equiv_{K_j})\}$ is bounded. Hence

Theorem A.1. *The sequence $\{K_j\}$ of regular languages finitely converges. Thus Algorithm D converges after a finite number of iterations.*

Appendix B. Minimally Restrictive Solution of SMP

In this appendix we discuss a minimally restrictive solution within a smaller class to SMP. We first note the following result which can be easily verified using the results in [CDFV] (the same result can be found in [LW], where the mask is assumed to be a projection).

Lemma B.1. *Given a language K, $\varnothing \neq K \subset L_m(G)$, there exists a proper supervisor $\underline{S}$ such that $L_m(\underline{S}/G_c) = K$ if and only if:*

(i) *K is $(M, \bar{K} \cap L_m(G))$-recognizable.*
(ii) *$\bar{K}$ is $(\Sigma_u, L(G))$-invariant and $(M, \Sigma_c, L(G))$-controllable.*[1]

Let $\underline{DD}(L)$ be a class of sublanguages of L which satisfy conditions (i) and (ii) above. Then $\underline{DD}(L)$ is the solution set for SMP. However, $\underline{DD}(L)$ is not closed under union and does not have a supremal element; the same example as in [CDFV] illustrates this fact.

Consider now the class $\underline{DD}'(L)$ of sublanguages of L introduced in Section 5. If we identify L_m and N in $\underline{DD}'(L)$ with $L_m(G)$ and $L(G)$, then it is straightforward to check that $\underline{DD}'(L) \subset \underline{DD}(L)$ (see, e.g., Proposition 4.1 of [LW]). Since $Sup\ \underline{DD}'(L)$ always exists, we regard it as a minimally restrictive solution to the *restricted* SMP. Let

$$\underline{F}(L, L_m) := \{O \subset L | O \text{ is } L_m\text{-closed}\}.$$

Then $\underline{F}(L, L_m)$ is closed under arbitrary union and possesses a unique supremal element [RW]. Note that $Sup\ \underline{DD}'(L) \in \underline{F}(L, L_m)$, and therefore $Sup\ \underline{DD}'(L) \subset Sup\ \underline{F}(L, L_m)$. It follows that

$$Sup\ \underline{DD}'(L) = Sup\ \underline{DD}'(Sup\ \underline{F}(L, L_m)).$$

Thus we may assume that the given language $L \subset L_m$ is L_m-closed. If not, we first obtain $Sup\ \underline{F}(L, L_m)$ and consider it as a given sublanguage of L_m. The procedure which effectively computes the language $Sup\ \underline{F}(L, L_m)$ can be given in a simple manner: for example, let $A := (Q, \Sigma, f, q_0, Q_m)$ and $A_s := (Q_s, \Sigma, f_s, q_0, Q_{sm})$ be DFAs such that $|A| = L_m$, $|A_s| = L \subset L_m$ and A_s is a subautomaton of A. Define a DFA $\tilde{A}_s := Tr\,(\tilde{Q}_s, \Sigma, \tilde{f}_s, q_0, \tilde{Q}_{sm})$ by:

(A) $q \in \tilde{Q}_s$ if and only if $q \in Q_{sm}$ whenever $q \in Q_m$.
(B) $\tilde{Q}_{sm} = \tilde{Q}_s \cap Q_{sm}\ (= Q_{sm})$.
(C) $\tilde{f}_s(\sigma, q)!$ if and only if $f_s(\sigma, q)!$ and $f_s(\sigma, q) \in \tilde{Q}_s$. In this case, $\tilde{f}_s(\sigma, q) = f_s(\sigma, q)$.

Then it is not hard to check that $|\tilde{A}_s| = Sup\ \underline{F}(L, L_m)$.

[1] Σ_c is the set of controllable events; i.e., $\Sigma_c = \Sigma - \Sigma_u$. A language $O \subset L(G)$ is said to be $(M, \Sigma_c, L(G))$-controllable if $s, t \in O$, $\sigma \in \Sigma_c$, $s\sigma \in O$, $t\sigma \in L(G)$, and $M(s) = M(t)$ implies $t\sigma \in O$.

Consider the class $\underline{D}'(L)$ of sublanguages of L defined in Section 5. We prove in the following a result similar to the one in [WR2]; our proof is, however, more straightforward.

Lemma B.2. *If L is L_m-closed, then Sup $\underline{D}'(L)$ is L_m-closed.*

Proof. Let $S := Sup\ \underline{D}'(L)$. Then it suffices to show that $\bar{S} \cap L_m \subset S$, since the reverse inclusion trivially holds. Thus if we show that $\overline{\bar{S} \cap L_m} \in \underline{D}'(L)$, we are done. By hypothesis, $\bar{S} \cap L_m \subset \bar{L} \cap L_m = L$. Note also that $\overline{\bar{S} \cap L_m} = \bar{S}$ (clearly, $\overline{\bar{S} \cap L_m} \subset \bar{S}$; also, $\bar{S} \subset \overline{\bar{S} \cap L_m}$ since $S \in \bar{S}$ and $S \subset L \subset L_m$), and that $\bar{S}$ is (Σ_u, N)-invariant and (M, N)-recognizable since $S \in \underline{D}'(L)$. Thus we have proved that $\bar{S} \cap L_m \in \underline{D}'(L)$. ■

Now we prove the claim made in Section 5.

Theorem B.1. *Let L be L_m-closed. Then Sup $\underline{D}\underline{D}'(L) = Sup\ \underline{D}'(L)$.*

Proof. Clearly, $\underline{D}\underline{D}'(L) \subset \underline{D}'(L)$. Therefore $Sup\ \underline{D}\underline{D}'(L) \subset Sup\ \underline{D}'(L)$. To prove the reverse inclusion, we note that $Sup\ \underline{D}'(L)$ is L_m-closed by the hypothesis and Lemma B.2. Thus $Sup\ \underline{D}'(L) \in \underline{D}\underline{D}'(L)$. Hence $Sup\ \underline{D}'(L) \subset Sup\ \underline{D}\underline{D}'(L)$. ■

References

[CDFV] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya, Supervisory control of discrete-event processes with partial observations, *IEEE Trans. Automat. Control*, **33** (1988), 249–260.

[E] S. Eilenberg, *Automata, Languages, and Machines*, Academic Press, New York, 1974.

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

[LW] F. Lin and W. M. Wonham, On Observability of Discrete-Event Systems, Systems Control Group Report No. 8701, Department of Electrical Engineering University of Toronto, 1987.

[RW] P. J. Ramadge and W. M. Wonham, Supervisory control of a class of discrete-event processes, *SIAM J. Control Optim.*, **25** (1987), 206–230.

[S] R. Smedinga, Control of Discrete Events, Preprint, University of Groningen, Groningen, 1987.

[T] J. N. Tsitsiklis, On the control of discrete-event dynamical systems, *Proceedings of the 26th IEEE Conference on Decision and Control*, Los Angeles, 1987, pp. 419–422.

[WR1] W. M. Wonham and P. J. Ramadge, On the supremal controllable sublanguage of a given language, *SIAM J. Control Optim.*, **25** (1987), 637–659.

[WR2] W. M. Wonham and P. J. Ramadge, Modular supervisory control of discrete-event systems, *Math. Control Signals Systems*, **1** (1988), 13–30.